



AV/C Digital Interface Command Set General Specification

Version 2.0.1
January 5, 1998

Sponsored by:
Audio/Video Working Group of the 1394 Trade Association

Approved for Release by:
1394 Trade Association Steering Committee

Abstract: This specification defines a command set for consumer and professional audio/video equipment over IEEE Std. 1394-1995. The command set makes use of the Function Control Protocol (FCP) defined by IEC-1883, proposed standard for Digital Interface for Consumer Electronic Audio/Video Equipment, for the transport of audio/video command requests and responses. The audio/video devices are implemented as a common unit architecture within IEEE Std. 1394-1995.

Keywords: Audio, Video, VCR, 1394, Digital, Interface

1394 Trade Association
3925 W. Braker Lane, Austin, TX 78759 USA
<http://www.1394TA.org>

Copyright 1996-1997 by the 1394 Trade Association. Permission is granted to members of the 1394 Trade Association to reproduce this document for their own use or the use of other 1394 Trade Association members only, provided this notice is included. All other rights reserved. Duplication for sale, or for commercial or for-profit use is strictly prohibited without the prior written consent of the 1394 Trade Association.

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 approved 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
3925 W. Braker Lane
Austin, TX 78759
USA

1394 Trade Association Specifications are adopted by the 1394 Trade Association without regard to patents which may exist on articles, materials or processes, or to other proprietary intellectual property which may exist within a specification. Adoption of a specification by the 1394 Trade Association does not assume any liability to any patent owner or any obligation whatsoever to those parties who rely on the specification documents. Readers of this document are advised to make an independent determination regarding the existence of intellectual property rights which may be infringed by conformance to this specification.

Table of Contents

1. REFERENCES.....	2
2. CHANGES FROM PREVIOUS VERSION.....	3
3. DEFINITIONS AND ABBREVIATIONS.....	4
3.1 Conformance glossary.....	4
3.2 Technical glossary.....	4
4. FUNCTION CONTROL PROTOCOL (INFORMATIVE).....	6
5. AV/C FRAMES.....	7
5.1 AV/C command frame.....	7
5.2 AV/C response frame.....	7
5.3 AV/C frame components.....	8
5.3.1 Command type (<i>ctype</i>).....	8
5.3.2 Response code (<i>response</i>).....	8
5.3.3 AV/C address (<i>subunit_type</i> , <i>subunit_ID</i>).....	9
5.3.4 Operation (<i>opcode</i>).....	11
5.3.5 Operands.....	11
5.3.6 Subunit classification process.....	11
6. AV/C OPERATIONS.....	13
7. AV/C COMMANDS.....	16
7.1 Support levels.....	16
7.2 Control commands.....	16
7.3 Status commands.....	17
7.4 SPECIFIC INQUIRY commands.....	18
7.5 Notify commands.....	18
7.6 GENERAL INQUIRY commands.....	19
8. UNIT COMMANDS.....	20
8.1 CHANNEL USAGE command.....	20
8.2 CONNECT command.....	22
8.3 CONNECT AV command.....	25
8.4 CONNECTIONS command.....	27
8.5 DIGITAL INPUT command.....	27
8.6 DIGITAL OUTPUT command.....	28
8.7 DISCONNECT command.....	28
8.8 DISCONNECT AV command.....	29
8.9 INPUT PLUG SIGNAL FORMAT command.....	29
8.10 OUTPUT PLUG SIGNAL FORMAT command.....	30
8.11 SUBUNIT INFO command.....	31
8.12 UNIT INFO command.....	32
9. COMMON UNIT AND SUBUNIT COMMANDS.....	34
9.1 POWER command.....	34
9.2 RESERVE command.....	35
9.3 PLUG INFO command.....	37
9.4 VENDOR-DEPENDENT commands.....	38
A. AV/C COMMANDS IN NUMERICAL ORDER (NORMATIVE).....	39
B. UNRESOLVED ISSUES (INFORMATIVE).....	40

List of Figures

FIGURE 4-1 — FCP FRAME WITHIN A SERIAL BUS PACKET	6
FIGURE 5-1 — AV/C COMMAND FRAME	7
FIGURE 5-2 — AV/C RESPONSE FRAME	7
FIGURE 5-3 — AV/C COMMAND FRAME WITH EXTENDED TYPE AND ID ADDRESSES	10
FIGURE 6-1 — AV/C IMMEDIATE TRANSACTION	13
FIGURE 6-2 — AV/C DEFERRED TRANSACTION	14
FIGURE 8-1 — CHANNEL USAGE STATUS COMMAND FORMAT	21
FIGURE 8-2 — CHANNEL USAGE STATUS RESPONSE FORMAT	21
FIGURE 8-3 — CONNECT CONTROL COMMAND	22
FIGURE 8-4 — CONNECT CONTROL COMMAND WITH EXTENDED SUBUNIT_TYPE AND EXTENDED SUBUNIT_ID	23
FIGURE 8-5 — CONNECT STATUS COMMAND FORMAT FOR A SOURCE PLUG	24
FIGURE 8-6 — CONNECT STATUS COMMAND FORMAT FOR A DESTINATION PLUG	24
FIGURE 8-7 — CONNECT AV CONTROL COMMAND FORMAT FOR AUDIO/VIDEO STREAM	25
FIGURE 8-8 — CONNECT AV STATUS COMMAND FORMAT FOR AUDIO/VIDEO STREAM	26
FIGURE 8-9 — CONNECTIONS STATUS COMMAND FORMAT	27
FIGURE 8-10 — CONNECTIONS RESPONSE FORMAT	27
FIGURE 8-11 — DIGITAL INPUT COMMAND FORMAT	28
FIGURE 8-12 — DIGITAL OUTPUT COMMAND FORMAT	28
FIGURE 8-13 — DISCONNECT COMMAND FORMAT	29
FIGURE 8-14 — DISCONNECT AV COMMAND FORMAT	29
FIGURE 8-15 — INPUT PLUG SIGNAL FORMAT CONTROL COMMAND FORMAT	29
FIGURE 8-16 — INPUT PLUG SIGNAL FORMAT STATUS COMMAND FORMAT	30
FIGURE 8-17 — OUTPUT PLUG SIGNAL FORMAT CONTROL COMMAND FORMAT	30
FIGURE 8-18 — OUTPUT PLUG SIGNAL FORMAT STATUS COMMAND FORMAT	31
FIGURE 8-19 — SUBUNIT INFO STATUS COMMAND FORMAT	31
FIGURE 8-20 — SUBUNIT INFO RESPONSE FORMAT	32
FIGURE 8-21 — SUBUNIT TABLE ENTRY	32
FIGURE 8-22 — UNIT INFO STATUS COMMAND FORMAT	32
FIGURE 8-23 — UNIT INFO RESPONSE FORMAT	33
FIGURE 9-1 — POWER COMMAND FORMAT	34
FIGURE 9-2 — RESERVE CONTROL COMMAND FORMAT	35
FIGURE 9-3 — RESERVE STATUS COMMAND FORMAT	36
FIGURE 9-4 — PLUG INFO STATUS COMMAND FORMAT	37
FIGURE 9-5 — PLUG INFO STATUS RESPONSE FORMAT FROM AN AV SUBUNIT	37
FIGURE 9-6 — PLUG INFO RESPONSE FORMAT FROM AN AV UNIT	37
FIGURE 9-7 — VENDOR-DEPENDENT COMMAND FORMAT	38

List of Tables

TABLE 5.3-1 — SUBUNIT TYPE ENCODING	9
TABLE 5.3-2 — SUBUNIT ID ENCODING	9
TABLE 7.6-1 — UNIT COMMANDS	20
TABLE 8.2-1 — SERIAL BUS AND EXTERNAL PLUG NUMBERS	23
TABLE 8.12-1 — COMMON UNIT AND SUBUNIT COMMANDS	34

Preface

This document specifies a command set used to control consumer electronic audio/video equipment. The command set builds upon an extensive body of standards work, underway and completed, as referenced in section 1. Serial Bus, an IEEE standard, is the digital interface used to transport commands from controllers to AV devices (targets) and to return responses to the controllers. The unit architectures of these AV devices are defined within the scope of the configuration ROM and CSR architecture standardized by ISO/IEC. The commands themselves are encapsulated within a generic Function Control Protocol (FCP) developed by the HD Digital VCR Conference and now proposed as an IEC standard. Similarly, the format of the isochronous data itself has been developed by the HD Digital VCR Conference.

This specification concerns itself narrowly with the syntax and semantics of a general set of commands transmitted by controllers to AV devices and the resultant actions that occur at the AV device. The suite of AV/C documentation has been separated into this general AV/C specification document and separate documents for each type of subunit (VCR, Tuner, Disc, etc.). The reader is strongly encouraged to read this document in conjunction with the references given below, as well as with any AV/C-related documents which may be created in the future.

1. References

AV/C Digital Interface Command Set for VCR Subunit Specification, version 2.0.1,
January 5, 1998

IEEE Std 1394–1995, Standard for a High Performance Serial Bus

IEC-1883, proposed standard for Digital Interface for Consumer Electronic Audio/Video
Equipment

ISO/IEC 13123:1994, Control and Status Register (CSR) Architecture for Microcomputer
Buses

HD Digital VCR Conference, Specifications of Consumer-Use Digital VCR's using 6.3 mm
magnetic tape (December 1995)

2. Changes from previous version

This document differs from version 2.0 in the following ways:

- The AV/C Digital Interface Command Set 2.0 manual was separated into two books: General Specification and the VCR Subunit Specification

Version 2.0 of this document differs from version 1.0 in the following ways:

- Table 5.3-1 — Subunit type encoding - the disc recorder/player type has been added, and the subunit type 05 has been changed to “Tuner” from “TV Tuner”.
- A model for extended subunit addressing has been defined in section 5.3.3. As a result, item C3 in Annex C (extended subunit addressing) has been removed (what was item C4 - Notification Support - is now item C3).
- A process for defining new device types and command sets has been defined in section 5.3.6.
- The ctype GENERAL INQUIRY (value 4) was added. This allows a controller to ask a target “do you support this opcode?” without passing any specific operands.
- The original ctype INQUIRY (value 2) was renamed SPECIFIC INQUIRY, to indicate that a set of operands must be supplied along with the opcode when issuing the command.

3. Definitions and abbreviations

3.1 Conformance glossary

Several keywords are used to differentiate between different levels of requirements and optionality, as follows:

- expected:** A keyword 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.
- may:** A keyword that indicates flexibility of choice with no implied preference.
- shall:** A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products conforming to this specification.
- should:** A keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “is recommended.”

3.2 Technical glossary

- ATN:** A sequential reference number recorded as part of each track of a DVCR cassette. Within the context of a single, uninterrupted recording session, these reference numbers are monotonically increasing and, in that sense, *absolute track numbers*. However, if the medium has been recorded at different times there may be gaps between different recorded areas and there is no guarantee of relationship between the absolute track numbers in one area and those in another.
- 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.
- AV subunit:** an instantiation of a virtual entity that can be identified uniquely within an AV unit and offers a set of coherent functions.
- AV/C:** Audio/video control, as in the AV/C Digital Interface Command Set specified by this document.
- byte:** Eight bits of data.
- CSR:** A node or unit Control and Status Register, as defined by IEEE Std 1394–1995.
- DVCR:** Digital video cassette recorder as defined by the HD Digital VCR Conference, Specifications of Consumer-Use Digital VCR's using 6.3 mm magnetic tape.
- EUI-64:** Extended Unique Identifier, 64-bits, as defined by the IEEE. The EUI-64 is a concatenation of the 24-bit company_ID obtained from the IEEE Registration Authority Committee (RAC) and a 40-bit number (typically a silicon serial number) that the vendor identified by company_ID guarantees to be unique for all of its products. The EUI-64 is also known as the node unique ID and is redundantly present in a node's configuration ROM in both the Bus_Info_Block and the Node_Unique_Id leaf.
- FCP:** Function Control Protocol, as defined by IEC-1883, proposed standard for Digital Interface for Consumer Electronic Audio/Video Equipment.
- IEEE:** The Institute of Electrical and Electronics Engineers, Inc.
- 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 durations 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.”
MIC:	An acronym for memory in cassette, a feature of DVCR cassettes that provides a limited amount of nonvolatile memory that may be used for any purpose. Standard MIC formats have been specified by the HD Digital VCR Conference.
module:	The smallest component of physical management, <i>i.e.</i> , a replaceable device.
nibble:	Four bits of data. A byte is composed of two nibbles.
node:	An addressable device attached to Serial Bus with at least the minimum set of control registers defined by IEEE Std 1394–1995.
node ID:	A 16-bit number, unique within the context of an interconnected group of Serial Buses. The node ID is used to identify both the source and destination of Serial Bus asynchronous data packets. It can identify one single device within the addressable group of Serial Buses (unicast), or it can identify all devices (broadcast).
PCR:	Plug Control Register, as defined by IEC-1883, proposed standard for Digital Interface for Consumer Electronic Audio/Video Equipment.
iPCR:	Input plug PCR, as defined by IEC-1883.
oPCR:	Output plug PCR, as defined by IEC-1883.
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.
quadlet:	Four bytes of data.
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.
SMPTE/EBU time code:	Time code format for professional use.
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.
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.

4. Function Control Protocol (informative)

The AV/C commands and responses are transported by the Function Control Protocol (FCP) defined by IEC-1883, proposed standard for Digital Interface for Consumer Electronic Audio/Video Equipment. FCP provides a simple means to encapsulate device commands and responses within IEEE Std 1394–1995 asynchronous block write transactions. The format of an FCP frame, encapsulated within a Serial Bus block write packet, is illustrated below in Figure 4-1.

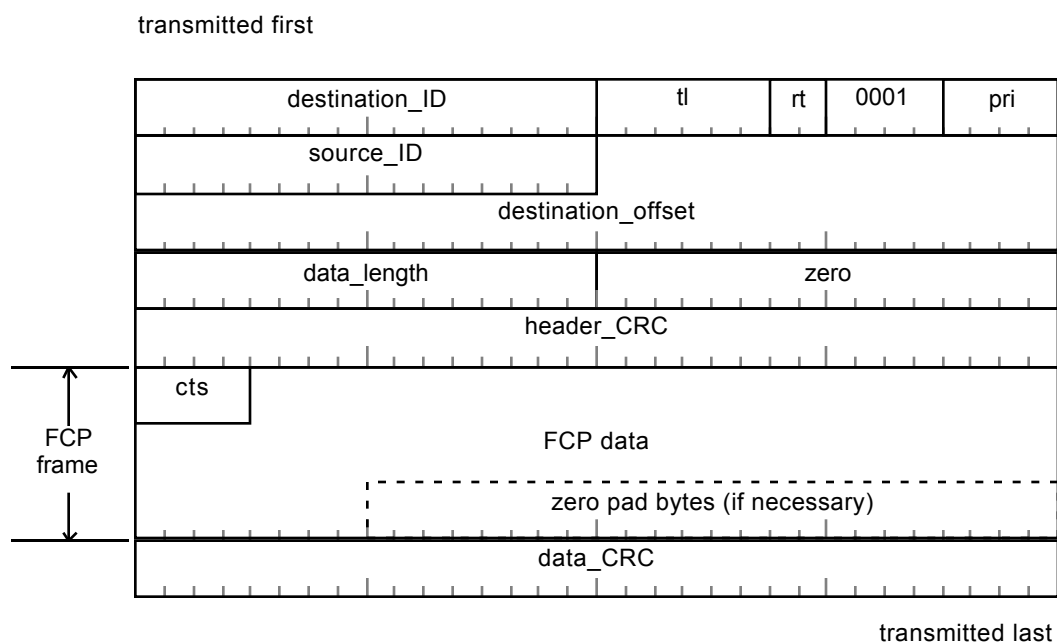


Figure 4-1 — FCP frame within a Serial Bus packet

The ***destination_ID***, ***tl***, ***rt***, ***tcode*** (write request for data block, 0001₂), ***pri***, ***source_ID***, ***data_length*** and CRC fields are as defined by IEEE Std 1394–1995.

The ***cts*** field defines the command transaction format used by the FCP frame. For the AV/C commands defined by this document, the ***cts*** field shall be zero.

Commands originated by a device at a Serial Bus node, the controller, are addressed to the FCP_COMMAND register, ***destination_offset*** FFFF F000 0B00₁₆ at the Serial Bus node that contains the device to be controlled, the target. The remotely controlled device in turn returns its response(s) to the FCP_RESPONSE register, ***destination_offset*** FFFF F000 0D00₁₆, at the controller.

The data payload of both FCP request and response packets, specified by ***data_length***, is limited to a maximum of 512 bytes.

NOTE: If the size of an FCP frame is exactly four bytes, a Serial Bus quadlet write transaction shall be used to transmit the data instead of the block write packet illustrated above.

5. AV/C frames

AV/C commands and responses are encapsulated within FCP frames, as described above, and are transmitted between the controller and target FCP_COMMAND and FCP_RESPONSE registers. The format of both the AV/C command and the AV/C response frames are similar, as described in the clauses that follow.

5.1 AV/C command frame

An AV/C command frame is up to 512 bytes of command payload with the structure shown in Figure 5-1 below:

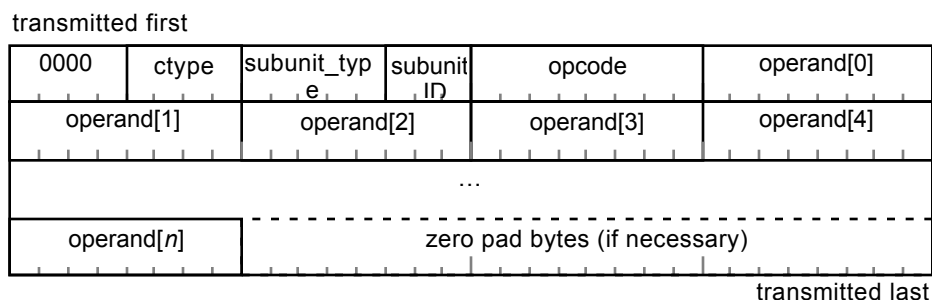


Figure 5-1 — AV/C command frame

All of the operands, up to a maximum permitted by the overall payload limit of 512 bytes, are optional and are defined by *ctype*, *subunit_type* and *opcode*.

NOTE: If an AV/C command frame exceeds the maximum capacity of an AV unit or subunit to which it is addressed, it may be ignored.

The *subunit_type* and *subunit_ID* fields form an AV/C address which identifies the destination of the AV/C command frame and the source of the AV/C response frame. If either the subunit type or subunit ID values have been extended, then there will be additional bytes used before the opcode byte. Please refer to the section titled “AV/C address (*subunit_type*, *subunit_ID*)” for details.

5.2 AV/C response frame

An AV/C response frame is up to 512 bytes of response payload with the structure shown in Figure 5-2 below:

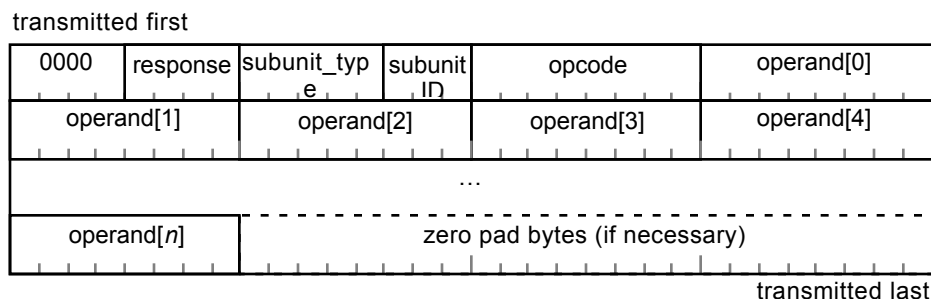


Figure 5-2 — AV/C response frame

All of the operands, up to a maximum permitted by the overall payload limit of 512 bytes, are optional and are defined by *response*, *subunit_type* and *opcode*.

The *subunit_type* and *subunit_ID* fields form an AV/C address which identifies the source of the responding AV/C entity and equals the destination to which the corresponding AV/C command frame was sent. As with the command frame, it is possible that the subunit type and/or subunit ID have been extended, thus requiring more bytes before the opcode field.

5.3 AV/C frame components

The component fields and code values for AV/C command and response frames are defined in this clause.

Except as otherwise indicated, reserved codes and fields within an AV/C frame are reserved for future specification. All reserved fields shall be set to zero by the sender of the AV/C frame. The sender shall not use reserved or invalid values for any components of an AV/C frame.

Responses to reserved or invalid codes and fields are defined in section 6.

5.3.1 Command type (*ctype*)

The 4-bit command type, *ctype*, defines one of five types of commands, as defined by the table below:

Value	Command type
0	CONTROL
1	STATUS
2	SPECIFIC INQUIRY
3	NOTIFY
4	GENERAL INQUIRY
5 – 7	Reserved for future specification
8 – F ₁₆	Reserved for response codes

5.3.2 Response code (*response*)

The 4-bit response code, *response*, defines one of seven types of response, as defined by the following table:

Value	Response
0 – 7	Reserved for command types
8	NOT IMPLEMENTED
9	ACCEPTED
A ₁₆	REJECTED
B ₁₆	IN TRANSITION
C ₁₆	IMPLEMENTED / STABLE
D ₁₆	CHANGED
E ₁₆	Reserved for future specification
F ₁₆	INTERIM

5.3.3 AV/C address (*subunit_type*, *subunit_ID*)

Taken together, the *subunit_type* and *subunit_ID* fields define the address of the recipient of the command or the source of the response. Version 1.0 of the AV/C specification limited subunit addressing to 32 subunit types and 5 subunits of a given type within a unit (refer to Table 5.3-1 and Table 5.3-2 of the original 1.0 specification for details). To allow for growth beyond these limitations, a backward compatible model for an *extended subunit address* has been devised using previously reserved *subunit_type* and *subunit_ID* values. The following tables illustrate the new definitions:

Subunit type	Meaning
0	Video monitor
1 – 2	Reserved for future specification
3	disc recorder/player (audio or video)
4	tape recorder/player (audio or video)
5	Tuner
6	Reserved for future specification
7	Video camera
8 – 1B ₁₆	Reserved for future specification
1C ₁₆	Vendor unique
1D ₁₆	Reserved for all subunit types
1E ₁₆	<i>subunit_type</i> extended to next byte
1F ₁₆	Unit

Table 5.3-1 — Subunit type encoding

Subunit ID	Meaning
0 - 4	Instance number
5	<i>subunit_ID</i> extended to next byte
6	Reserved for all instances
7	Ignore

Table 5.3-2 — Subunit ID encoding

An AV/C address with *subunit_type* value 1F₁₆ and *subunit_ID* value 7 addresses the complete AV unit instead of one of its subunits. The combinations of *subunit_type* value 1F₁₆ and *subunit_ID* values 0 through 6 are reserved.

If the *subunit_type* value is not equal to 1F₁₆, the *subunit_ID* indicates the ordinal of the subunit as indicated by *subunit_type*. In this case, the *subunit_ID* commences at zero and is consecutively numbered up to the total instances less one.

IMPORTANT: The *subunit_ID* value specified in *extended* fields shall be equal to the exact instance number, NOT less one. This is required because there are restrictions on the value

0 in extended address fields. Please refer to the tables below, which define the meaning of extended subunit_type and subunit_ID fields.

The subunit_ID field can be used as a normal instance number in the case where an extended subunit_type is specified. This gives an identical numbering scheme for extended subunit_types and normal subunit_types.

If extended addressing is used, each extra byte is either completely used for an extended subunit_type or an extended subunit_ID. This differs from the normal subunit address byte, in which both the type and ID are specified within a single byte.

If byte n in an AV/C frame contains an AV/C address value that indicates that both the subunit-type and subunit-ID are extended to the next byte, the subunit_type is extended using byte n+1 of the AV/C frame and the subunit-ID is extended using byte n+2. Compare Figure 5-1 to the following diagram of an AV/C command frame with extended type and ID addresses:

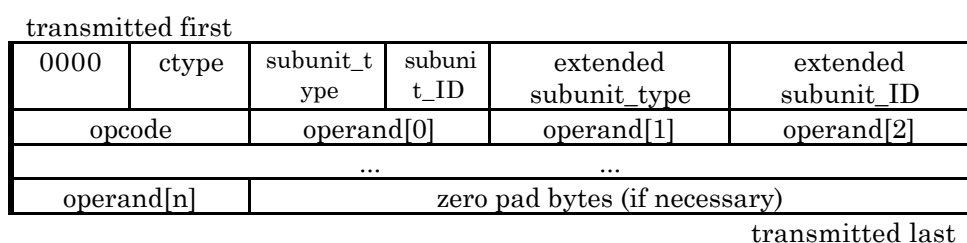


Figure 5-3 — AV/C command frame with extended type and ID addresses

The following tables illustrate the definitions for extended subunit_type and subunit_ID values:

Extended subunit_type value	Meaning
0	reserved for future specification
1..FE ₁₆	extended subunit_type
FF ₁₆	extended subunit_type extended to next byte

Extended subunit_ID value	Meaning
0	reserved for future specification
1..FE ₁₆	extended subunit_ID
FF ₁₆	extended subunit_ID extended to next byte

Note that by using the value FF₁₆ in the extended type or ID fields, it is possible to specify an unlimited number of subunit types or ID's of a given type.

Extended subunit_ID values continue counting where the previous (extended) subunit_ID stopped. For example, if 3 bytes (one normal and two extended) are used for the subunit_ID, the value 1 in the third byte addresses instance number: 5 + 254 + 1 = the 260th instance. Note that the actual entries in the fields would be 5, FF₁₆, 1.

In this example, the instance number 5 is derived from the normal subunit_ID field entry of 5 which indicates that this is an extended ID. The instance number is indicated by the highest legitimate ID for that field. In the case of the normal address field, the highest ID value is 5. For extended ID fields, it is FE₁₆, or 254. Remember that the ID entries in the

normal subunit_ID field are equal to the “instance number - 1” and that the entries in all of the extended fields are equal to the “instance number”.

5.3.4 Operation (*opcode*)

Within the five types of AV/C commands, CONTROL, STATUS, SPECIFIC INQUIRY, NOTIFY and GENERAL INQUIRY, the *opcode* field defines the operation to be performed or the status to be returned. The permissible values of *opcode* are divided into ranges valid for commands addressed to AV subunits, AV units or both, as follows.

Value	Addressing mode
0 – F ₁₆	Unit and subunit commands
10 ₁₆ – 3F ₁₆	Unit commands
40 ₁₆ – 7F ₁₆	Subunit commands
80 ₁₆ – 9F ₁₆	Reserved for future specification
A0 ₁₆ – BF ₁₆	Unit and subunit commands
C0 ₁₆ – DF ₁₆	Subunit commands
EE ₁₆ – FF ₁₆	Reserved for future specification

5.3.5 Operands

The number and meaning of the *operand[n]* fields are determined by the *ctype*, *subunit_type* and *opcode* fields, as defined in clauses that follow.

5.3.6 Subunit classification process

The AV/C protocol has been designed for future growth, to accommodate the creation of new types of products that were not envisioned when the protocol was originally developed. When a manufacturer is designing such a new piece of equipment, the following guidelines should be used for determining if the device falls into an existing category (as defined in Table 5.3-1), or if a new type needs to be defined. Note that new device types may require modifications to existing commands or the creation of entirely new commands.

The basic approach to type classification is a two step process:

1) Examine the MAIN functionality of the subunit, in terms of the following:

- transport mechanism - does it have one?
- signal input - is the usefulness of this subunit defined mostly by the fact that a signal ends up here (regardless of the fact that it may be propagated without changes)?
- signal output - is this subunit a signal source?
- signal processing - accepts input, performs some sort of processing, and then outputs modified data
- no signal input or output - a utility of some kind

2) If a set of commands do not apply equally to audio or video data, then decompose the subunit type into separate audio and video categories.

While many subunits may have both input and output signals, the important item to consider is the MAIN functionality - in other words, what is the purpose of this subunit? The main purpose of a video camera subunit is to capture data through its lens and send that signal somewhere - it's a signal source. The main purpose of a television monitor is for viewing the input signal - it's a signal input or destination.

A utility such as a timer or a mechanism that can pan/tilt a camera does not deal with signal input or output, but it may be part of a controllable subunit.

When selecting a new type value and the appropriate command set, the following guidelines should be followed:

- 1) Find an unused subunit type value from the table of pre-defined types (Table 5.3-1).
- 2) Select only the specified new subunit type from the table; other codes for unused types must remain reserved.
- 3) Define a (relatively) complete set of commands for this new type. This step includes the definition of new commands that are unique to this type, as well as the verification that existing commands (where applicable) will work as defined. New devices that have similar functionality to existing devices should map their control features to the existing commands.

6. AV/C operations

AV/C commands transmitted by a controller and the associated response(s) returned by the target are called an AV/C transaction. An AV/C transaction consists of one AV/C command frame addressed to the target's FCP_COMMAND register and zero or more AV/C response frames addressed to the controller's FCP_RESPONSE register. Unless stated otherwise within individual command descriptions, it is assumed that at least one response will be returned.

The target's node_ID identifies either a specific AV unit or it identifies all AV units (broadcast). Unless stated otherwise within individual command descriptions, it is assumed that a single AV unit is addressed by the command. An example of a simple AV/C transaction, in which the target is able to complete the request before responding, is shown below in Figure 6-1.

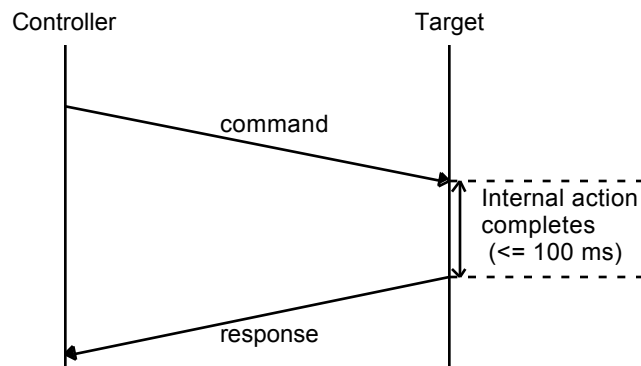


Figure 6-1 — AV/C immediate transaction

In an immediate transaction any response code, except INTERIM or CHANGED, is permitted. The transaction is complete when the target writes the AV/C response frame to the controller's FCP_RESPONSE register.

For some transactions the target may not be able to complete the request (or determine if it is possible to complete the request) within the 100 milliseconds allowed. In this case, the target shall return an initial response of INTERIM with the expectation that a final response will follow later. Figure 6-2 below illustrates an AV/C transaction with an intermediate response.

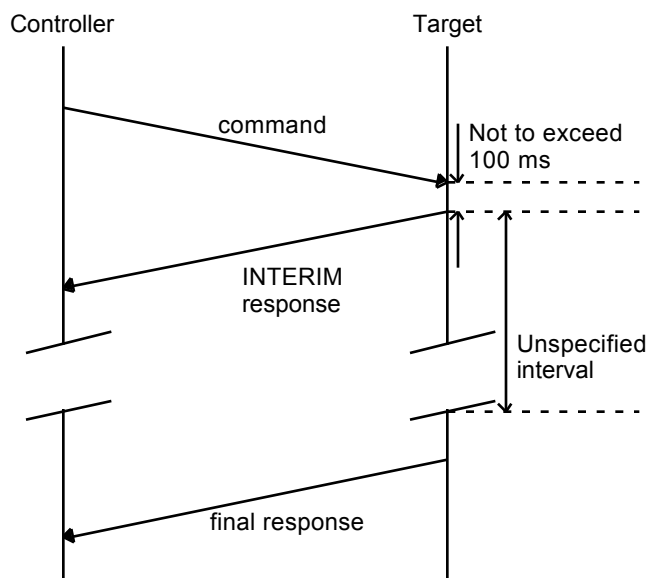


Figure 6-2 — AV/C deferred transaction

A target shall follow the following procedures when AV/C response frames are returned to the controller:

The target shall generate a response frame within 100 milliseconds of the receipt of the AV/C command frame. Targets should respond as quickly as possible.

If the AV/C command frame contains a reserved value in the *ctype* field, the target shall ignore the command and shall not generate a response frame.

If the target is already occupied with a previous command, it may ignore any AV/C command frames received. Note that the receipt of an AV/C command frame shall always be acknowledged by a target. The target ignores a command frame by a failure to return a response frame.

NOTE: A controller that does not receive a response frame for an AV/C command frame within 100 milliseconds may retry the command by resending the same command frame.

If the target is not occupied with a previous command, it shall create an AV/C response frame from the command frame by first copying all of the bytes of the command frame that precede *opcode* and then inserting the correct response code. The remainder of the response frame, *opcode* and operands, is dependent upon the *ctype*, *subunit_type* and *opcode* of the original command.

NOTE: Response frames returned after control commands are usually identical to the original command frame except for the *response* field. A response to a status or notify command typically has different *response* and *operand* fields and, in some cases, a different *opcode* field.

If the target receives a command frame whose *subunit_type* and *subunit_ID* fields address the command to a nonexistent subunit, the target shall return a response code of NOT IMPLEMENTED.

If the target is able to initiate the requested command in less than 100 milliseconds, it shall return a response code other than INTERIM. This includes those cases where the target determines that it cannot execute the command, such as a REJECTED, response. The return of any response code other than INTERIM marks the transaction completed and the target is normally ready to accept other transactions at its FCP_COMMAND register.

If the target is unable to complete the command within 100 milliseconds, it shall promptly return an intermediate response code of INTERIM. Subsequent to an initial response of INTERIM, the target shall not send any additional INTERIM responses for this command. There is no time limit on command completion once an INTERIM response has been sent. The target shall ultimately send a final response when the command completes.

If the target detects a Serial Bus reset, it shall reset its state to be able to accept AV/C command frames at its FCP_COMMAND register. Any in progress transactions shall be discarded without the return of a response frame.

If a target receives an AV/C command frame using the broadcasting node_ID, and a response of NOT IMPLEMENTED would be required, no response shall be returned.

In order to correlate a response frame with an outstanding AV/C command, a controller shall examine certain fields in the response frame. The *subunit_type* and *subunit_ID* fields are never modified by the target. The *ctype* field is overwritten with the response code returned. The *opcode* and *operand[n]* fields may or may not be altered, dependent upon the command type, *subunit_type* and *opcode*. For any particular *opcode*, consult the clauses that follow for the details of the response frame returned by the target.

7. AV/C commands

AV/C commands are variable-length strings of bytes that are embedded within a command frame and addressed to a particular AV unit or subunit. A command consists of a command type (*ctype*), a unit or subunit to which the command is addressed (*subunit_type*), an operation code (*opcode*) and one or more operands. Commands are described in the clauses that follow according to their command type, specified by *ctype* values of CONTROL, STATUS, SPECIFIC INQUIRY, NOTIFY or GENERAL INQUIRY.

7.1 Support levels

Each AV unit or subunit may implement a subset of the AV/C command set. An unsupported command shall be rejected with a response of NOT IMPLEMENTED. Support for the different commands is characterized as mandatory, recommended, optional and vendor-dependent, as defined below:

Mandatory	The command shall be supported by any audio/video device that claims compliance with this specification and that implements the AV unit or subunit type(s) for which the command is defined. AV/C compliant devices are identified by configuration ROM entries.
Recommended	For an AV/C compliant device, the command is optional but it represents a basic functionality, <i>e.g.</i> , video and audio insert modes for a VCR subunit's RECORD command. If the device supports unit or subunit type(s) that have the functionality corresponding to the command, it is recommended that the command be implemented.
Optional	The command is optional for an AV/C compliant device.
Vendor-dependent	Support for and interpretation of the command are defined by the device vendor.

Support levels for the different commands vary according to *ctype*.

7.2 Control commands

A control command is sent by a controller to another AV device, the target, to instruct the target to perform an operation. Either the AV unit or a subunit may be the recipient of the command, as determined by the *subunit_type* and *subunit_ID* fields in the command frame. The remaining fields, *opcode* and *operand[n]*, specify the command.

Subject to the procedures described in section 6, a target that receives a control command shall return an AV/C response frame with one of the four response codes described below.

NOT IMPLEMENTED	The target does not support the control command specified by <i>opcode</i> and <i>operand[n]</i> or the command is addressed to a subunit not implemented by the target. Target state is not modified.
ACCEPTED	The target implements the control command specified by <i>opcode</i> and <i>operand[n]</i> and the target state permits execution of the command. Note that command execution may not be complete at the time a response of ACCEPTED is returned. For example, a PLAY control command sent to a VCR may be acknowledged as accepted before the head mechanisms have engaged and the tape has started to move. The return of a response of ACCEPTED does not distinguish between a

	command that has completed immediately and one that is deferred but expected to complete without error.
REJECTED	The target implements the control command specified by <i>opcode</i> and <i>operand[n]</i> but the target state does not permit execution of the command. For example, a PLAY control command sent to a VCR that has no cassette inserted would be rejected. The target state may be modified as a result of the control command.
INTERIM	If the control command specified by <i>opcode</i> and <i>operand[n]</i> is implemented but the target is unable to respond with either ACCEPTED or REJECTED within 100 milliseconds, it shall return a response frame that indicates INTERIM. Unless a subsequent bus reset causes the AV/C transaction to be aborted, the target shall ultimately return a response frame with a response code of ACCEPTED or REJECTED.

7.3 Status commands

A status command is sent by a controller to an AV device to request the device's current status. Status commands may be sent to either AV units or subunits. No target state is altered by the receipt of a status command.

NOTE: With some notable exceptions, for example the status commands that deal with a VCR's transport states, the status commands bear a family resemblance to the control commands. The same *opcode* that is used to issue a control command to a target is generally used to request corresponding status.

A target that receives a status command shall return an AV/C response frame with one of the four response codes described below:

NOT IMPLEMENTED	The target does not support the status command specified by <i>opcode</i> and <i>operand[n]</i> or the command is addressed to a subunit not implemented by the target.
REJECTED	The target implements the status command specified by <i>opcode</i> and <i>operand[n]</i> but the target state does not permit the return of status for the command.
IN TRANSITION	The target implements the status command specified by <i>opcode</i> and <i>operand[n]</i> but the target state is in transition, possibly because of an already acknowledged command or a manual operation. A subsequent status command, at an unspecified future time, may result in the return of STABLE status.
STABLE	The target implements the status command specified by <i>opcode</i> and <i>operand[n]</i> and the information requested is reported in the <i>opcode</i> and <i>operand[n]</i> values in the AV/C response frame.

NOTE: Stable information may be returned for target information that is changing because of command execution. For example, the tape position reported by a VCR may be an accurate snapshot at the time the status command was accepted, but a subsequent status command could yield a different result.

Except for the STABLE and IN TRANSITION responses, the AV/C response frame data contains the same *opcode*, operands and addressing fields as the command frame. When status information is available, both the *opcode* field and one or more of the *operand[n]* fields may be updated with the status information.

7.4 SPECIFIC INQUIRY commands

Inquiry commands may be used by a controller to determine whether or not a target AV device supports a particular control command. Except for the *ctype* field, the AV/C command frame for an inquiry command is identical to the corresponding control command.

A controller may reliably use inquiry commands to probe the capabilities of a target, since the target shall not modify any state nor initiate any command execution in response to an inquiry command.

Only two response codes, IMPLEMENTED or NOT IMPLEMENTED are permitted in the response frame returned by the target. All other fields in the response frame are exact copies of the command frame. A response of IMPLEMENTED specifies that the corresponding control command specified by *opcode* and *operand[n]* is implemented by the target AV device. An AV device implementation may validate all of the operands or it may validate only *opcode* and enough of the operands to uniquely identify the control command and determine its support level.

NOTE: If a controller wishes to determine whether or not a particular status command is supported, it should issue the command. This is safe because status commands, whether or not implemented by a target, shall not cause state changes in the target.

Unlike the other command types, inquiry commands do not have a support level since they return information about the support level of the corresponding control command. However, the ability of an AV device to provide a response to an inquiry command for any *opcode* is mandatory. This insures that a controller shall always receive a response to a support level inquiry command.

The broadcasting node_ID shall not be used for inquiry commands.

7.5 Notify commands

A controller that desires to receive notification of future changes in an AV device's state may use the notify command. Responses to a notify command shall indicate the current state of the target and then, at some indeterminate time in the future, indicate the changed state of the target.

A target that receives a notify command shall not modify its state but shall generate an immediate response frame with one of the three response codes described below:

NOT IMPLEMENTED	The target does not support the notify command specified by <i>opcode</i> and <i>operand[n]</i> or the command is addressed to a subunit not implemented by the target.
REJECTED	The target implements event notification for the condition specified by <i>opcode</i> and <i>operand[n]</i> but is not able to supply the requested information.
INTERIM	The target supports the requested event notification and has accepted the notify command for any future change of state. The current state is indicated by the <i>opcode</i> and <i>operand[n]</i> data returned in the response frame. At a some future time, the target shall return an AV/C response frame with either a REJECTED or CHANGED response code.

Once a target has accepted a notify command by the return of an INTERIM response frame, the target is primed to return a subsequent response frame upon the first change in target

state. The future change of target state could be the result of an operation in progress when the notify command was received or it could be the result of a control command not yet received by the target.

CHANGED

The target supports the event notification specified by *opcode* and *operand[n]* and the target state differs from the target state at the time the INTERIM response was returned. The altered target state is indicated by the *opcode* and *operand[n]* data returned in the response frame.

A typical example of the use of a notify command might involve a VCR whose cassette is being rewound. The initial response to a TRANSPORT STATE notify command is an indication of INTERIM and a “rewinding” state. When the cassette’s beginning of medium is reached, the target generates a final response frame of CHANGED and a state that indicates “stopped”.

Note that notification is a one-shot operation. If the controller wishes to be notified of additional changes in a target, the controller must issue a notify command after each CHANGED response.

7.6 GENERAL INQUIRY commands

General inquiry commands may be used by a controller to determine whether or not a target AV device supports a particular control command WITHOUT being required to specify a particular set of parameters for that command. The format of the GENERAL INQUIRY command frame consists of only the opcode of the command which is being queried.

As with the SPECIFIC INQUIRY command, the target shall not modify any state nor initiate any command execution in response to a general inquiry command.

Only two response codes, IMPLEMENTED or NOT IMPLEMENTED are permitted in the response frame returned by the target. The response frame shall also contain the opcode that was originally passed in. A response of IMPLEMENTED specifies that at least one of the corresponding control command variations specified by *opcode* is implemented by the target AV device. For example, a VCR which supports the BACKWARD control command with the *video scene* operand, but not the *video frame* or *index* operands, shall return IMPLEMENTED for the BACKWARD general inquiry command.

Unlike the other command types, general inquiry commands do not have a support level since they return information about the support level of the corresponding control command. However, the ability of an AV device to provide a response to a general inquiry command for any *opcode* is mandatory. This insures that a controller shall always receive a response to a support level inquiry command.

The general inquiry command type was defined after the original AV/C specification, and some products, were created. Hence, there will be some devices which do not respond to this command type. A controller which does not receive a response should try the SPECIFIC INQUIRY command as a fallback measure.

The broadcasting node_ID shall not be used for general inquiry commands.

8. Unit commands

Unit commands are those that are addressed to an AV device implemented as a unit architecture at a Serial Bus node. Unit commands are identified by a *subunit_type* value of $1F_{16}$ and a *subunit_ID* value of seven. Table 7.6-1 below summarizes the AV/C unit commands.

Table 7.6-1 — Unit commands

Opcode	Value	Support level (by <i>ctype</i>)			Comments
		C	S	N	
CHANNEL USAGE	12_{16}	–	R	R	Report information on IEEE 1394 isochronous channel usage
CONNECT	24_{16}	O	O	R	Establish connections for unspecified streams between plugs and subunits
CONNECT AV	20_{16}	O	O	O	Establish AV connections between plugs and subunits
CONNECTIONS	22_{16}	–	O	–	Report connection status
DIGITAL INPUT	11_{16}	O	O	–	Make or break broadcast Serial Bus connections
DIGITAL OUTPUT	10_{16}	O	O	–	
DISCONNECT	25_{16}	O	–	–	Break unspecified stream connections between plugs and subunits
DISCONNECT AV	21_{16}	O	–	–	Break AV connections between plugs and subunits
INPUT PLUG SIGNAL FORMAT	19_{16}	O	R	O	Set or report signal formats for Serial Bus plugs
OUTPUT PLUG SIGNAL FORMAT	18_{16}	O	R	O	
SUBUNIT INFO	31_{16}	–	M	–	Report subunit information
UNIT INFO	30_{16}	–	M	–	Report unit information

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 operand formats and corresponding response frame formats are described for each of the commands in the clauses that follow.

8.1 CHANNEL USAGE command

The CHANNEL USAGE status command can be used to find out which AV unit, if any, is using a particular IEEE 1394 isochronous channel.

Using a channel means that one of the AV unit's oPCRs indicates that there exists a connection which uses this channel.

For the CHANNEL USAGE status command, it is permissible to use the broadcasting node_ID.

NOTE: When using the broadcasting node-ID, this command shall only generate a broadcast on one particular bus. Pending the definition of the addressing scheme in a bridged environment, a controller shall use the enumerated bus-ID value of the bus for which the command is intended as part of the

broadcasting node-ID. This also holds in case the command is intended for the bus to which the controller is attached. Only in case no bus-ID has been assigned, it is allowed to use the bus-ID value $3FF_{16}$ as part of the broadcasting node-ID.

The CHANNEL USAGE status command has the format as illustrated in Figure 8-1 below.

	msb							lsb
opcode	CHANNEL USAGE (12_{16})							
operand[0]	IEEE 1394 isochronous channel							
operand[1]	FF_{16}							
operand[2]								
operand[3]								

Figure 8-1 — CHANNEL USAGE status command format

Operand[0] denotes the isochronous channel which the target must check, to see if it is using that channel.

The CHANNEL USAGE status response has the format as illustrated in Figure 8-2 below.

	msb							lsb
opcode	CHANNEL USAGE (12_{16})							
operand[0]	IEEE 1394 isochronous channel							
operand[1]	node_ID							
operand[2]								
operand[3]	oPCR number							

Figure 8-2 — CHANNEL USAGE status response format

If in the STATUS response frame operand[1] through operand[3] are NOT all FF_{16} , it indicates that the target identified by operand[1] and operand[2] is using the channel indicated in operand[0], through the oPCR identified by operand[3]. Operand[1] contains the most significant byte of the node_ID.

If in the STATUS response frame operand[1] through operand[3] ARE all FF_{16} , it indicates that the target that returned the response is not using the channel indicated in operand[0].

In case the CHANNEL USAGE status command was broadcast (as opposed to unicast), the response obligation on this command exists only for those targets that use the channel. Because at most one target can meet this condition, at most one response frame will be returned and that response shall have operand[1] through operand[3] NOT all equal to FF_{16} .

In case the CHANNEL USAGE status command was unicast, the target shall return a response with operand[1] through operand[3] indicating whether or not the target is using the channel.

The CHANNEL USAGE command may also be used as a notify command. The notify command has the same syntax and meaning for its operands as the CHANNEL USAGE status command.

For the CHANNEL USAGE notify command, it is permissible to use the broadcasting node_ID.

In case the CHANNEL USAGE notify command was unicast and the target is not using the channel, it shall return a REJECTED response. Otherwise, it shall return an INTERIM

response with operand[1] through operand[3] NOT all equal to FF₁₆. If an INTERIM response has been returned, a CHANGED response shall be returned with operand[1] through operand[3] all equal to FF₁₆ once the target stops using the specified channel.

In case the CHANNEL USAGE notify command was broadcast, the response obligation on this command exists only for those targets that use the channel. Because at most one target can meet this condition, at most one INTERIM response frame will be returned with operand[1] through operand[3] NOT all equal to FF₁₆. If an INTERIM response has been returned, a CHANGED response shall be returned with operand[1] through operand[3] all equal to FF₁₆ once the target stops using the specified channel.

8.2 CONNECT command

An AV *subunit* has *source* and *destination* plugs. A source plug outputs one stream from the AV subunit and a destination plug inputs one stream into the AV subunit.

An AV *unit* has *Serial Bus input* and *output* plugs to model the Serial Bus interface of the AV unit. An AV unit can have at most one Serial Bus interface and thereby at most one node ID on the Serial Bus. A Serial Bus input plug inputs one stream from the Serial Bus interface into the AV unit and a Serial Bus output plug outputs one stream from the AV unit to the Serial Bus interface.

An AV unit also has *external input* and *output* plugs to model external interfaces of the AV unit other than Serial Bus. An external input plug inputs one stream from an external interface into the AV unit and an external output plug outputs one stream from the AV unit to one external interface.

The CONNECT control command establishes a connection within an AV Unit between:

a source plug of an AV subunit and a destination plug of an AV subunit to carry a stream that flows inside the AV unit.

a source plug of an AV subunit and a Serial Bus or external output plug to carry a stream that flows from the AV subunit to the Serial Bus or external interface.

a Serial Bus or external input plug and a destination plug of an AV subunit to carry a stream that flows from the Serial Bus or external interface to the AV subunit.

These connections are independent from the type of data (audio, video, data, ...) inside the stream which they carry. These streams are named “unspecified streams.”

	msb							lsb
opcode	CONNECT (24 ₁₆)							
operand[0]	3F ₁₆						lock	perm
operand[1]	source_subunit_type					source_subunit_ID		
operand[2]	source_plug							
operand[3]	destination_subunit_type					destination_subunit_ID		
operand[4]	destination_plug							

Figure 8-3 — CONNECT control command

The subunit_type and subunit_ID fields for both the source and destination plugs have the same syntax and meaning as an AV/C address (see section 5.3.3).

In case the value of source and destination *subunit_type* and *subunit_ID* are extended in the above control command, the frame will look as follows:

	msb							lsb
opcode	CONNECT (24 ₁₆)							
operand[0]	3F ₁₆						lock	perm
operand[1]	source_subunit_type = 1E ₁₆					source_subunit_ID = 5 ₁₆		
operand[2]	extended_source_subunit_type							
operand[3]	extended_source_subunit_ID							
operand[4]	source_plug							
operand[5]	destination_subunit_type = 1E ₁₆					destination_subunit_ID = 5 ₁₆		
operand[6]	extended_destination_subunit_type							
operand[7]	extended_destination_subunit_ID							
operand[8]	destination_plug							

Figure 8-4 — CONNECT control command with extended subunit_type and extended subunit_ID

For the example above, the source and destination *subunit_type* and *subunit_ID* values have been extended only once.

The source_plug and destination_plug fields are defined by the table below:

value	source plug	destination plug
0 - 1E ₁₆	Source plug 0 - 30	Destination plug 0 - 30
1F ₁₆ - FC ₁₆	Reserved for future specification	Reserved for future specification
FD ₁₆	Reserved for future specification	Multiple plugs
FE ₁₆	Invalid	Invalid
FF ₁₆	Any available source plug	Any available destination plug

When the stream flows from or to one of the AV unit's Serial Bus or external plugs, the *subunit_type* field shall have a value of 1F₁₆ (AV unit) and the *subunit_ID* field shall have a value of 7. In this case, the *source_plug* and *destination_plug* fields identify either a Serial Bus or an external plug according to Table 8.2-1 below:

Table 8.2-1 — Serial Bus and external plug numbers

value	source plug	destination plug
0 - 1E ₁₆	Serial Bus iPCR[0] - iPCR[30]	Serial Bus oPCR[0] - oPCR[30]
1F ₁₆ - 7E ₁₆	Reserved for future specification	Reserved for future specification
7F ₁₆	Any available Serial Bus plug iPCR[x]	Any available Serial Bus plug oPCR[x]
80 ₁₆ - 9E ₁₆	External input plug 0 - 30	External output plug 0 - 30
9F ₁₆ - FC ₁₆	Reserved for future specification	Reserved for future specification
FD ₁₆	Reserved for future specification	Multiple plugs
FE ₁₆	Invalid	Invalid
FF ₁₆	Any available External input plug	Any available External output plug

The PLUG INFO status command may be used to determine the number of Serial Bus and external plugs of an AV unit.

Note that overlaying a connection with another connection between the same source plug and another destination plug resulting in a one-to-many flow of the same stream may or may not be allowed, depending on the capabilities of the target.

The *lock* bit pertains to the connection between the source and destination plugs as indicated in the CONNECT command. If the lock bit in the CONNECT command is set to one to establish a connection between a source and a destination plug, any subsequent CONNECT command that would result in a disruption of the stream flowing between these plugs shall return a REJECTED response. This rule shall remain valid until a subsequent DISCONNECT command has been received by the target for that source plug.

The *perm* bit is ignored in a CONNECT control command.

The CONNECT command may also be used as a status command to determine the current state of the connections within an AV unit. The CONNECT status command is used to request the identity of the source plug that is connected to a given destination plug, or the identity of the destination plug for a given source plug. The two formats for the corresponding CONNECT status commands are shown in Figure 8-5 and Figure 8-6 below, and have the same meaning as the corresponding fields of the CONNECT control command.

	msb							lsb
opcode	CONNECT (24 ₁₆)							
operand[0]	FF ₁₆							
operand[1]	source_subunit_type					source_subunit_ID		
operand[2]	source_plug							
operand[3]	FF ₁₆							
operand[4]	FE ₁₆							

Figure 8-5 — CONNECT status command format for a source plug

	msb							lsb
opcode	CONNECT (24 ₁₆)							
operand[0]	FF ₁₆							
operand[1]	FF ₁₆							
operand[2]	FE ₁₆							
operand[3]	destination_subunit_type					destination_subunit_ID		
operand[4]	destination_plug							

Figure 8-6 — CONNECT status command format for a destination plug

The CONNECT status response frame has the same format and the same meaning for all fields as the CONNECT control command except for the *perm* field.

Except for the *perm* bit, the CONNECT status response frame contains exact copies of the CONNECT operands that were used to establish the connection. This includes the extended source and destination *subunit_type* and *subunit_ID* if they were used.

The *perm* bit in a CONNECT status response frame indicates whether a connection is permanent (value 1) or not (value 0). Permanent connections within an AV unit are connections that cannot be altered by the CONNECT control command or deleted by the DISCONNECT command, in which case a REJECTED response shall be returned.

In case there is no source plug connected to a destination plug, the *source_plug* field of the CONNECT status response frame shall indicate FE₁₆ (invalid).

In case there is no destination plug connected to a source plug, the *destination_plug* field of the CONNECT status response frame shall indicate FE₁₆ (invalid).

In case there are multiple destination plugs connected to a source plug, the *destination_plug* field of the CONNECT status response frame shall indicate FD₁₆ (multiple plugs).

The CONNECT command may also be used as a notify command. The notify command has the same syntax as the CONNECT status command. A notification shall be returned by the target to the controller that issued the notify command in case a connection involving the plug, as indicated in the notify command, changes. These changes shall include establishing a connection to the plug, deleting a connection from the plug, and connecting the plug to another plug.

The notify responses (INTERIM and CHANGED) have the same format as the CONNECT status response frame and indicate the current status of the plug for which the notification was requested. If the plug is still connected, the plug to which it is connected shall be indicated. If the plug is no longer connected, the source or destination plug field shall be indicated as invalid (plug field value FE₁₆).

8.3 CONNECT AV command

The CONNECT AV control command is used to establish audio/video connections between subunits and plugs.

	msb						lsb
opcode	CONNECT AV (20 ₁₆)						
operand[0]	video_source_type	audio_source_type	video_dest_type	audio_dest_type			
operand[1]	video_source						
operand[2]	audio_source						
operand[3]	video_destination						
operand[4]	audio_destination						

Figure 8-7 — CONNECT AV control command format for audio/video stream

The four fields of operand[0], video_source_type, audio_source_type, video_dest_type and audio_dest_type, encode the meaning of the four following source and destination identifying fields, as described in the table below.

Value	Source or destination type
0	Subunit
1	Ignore
2	Serial Bus or external plug
3	Reserved

If the source or destination type is zero, the corresponding source or destination operand is a subunit address, encoded as described in 5.3.3. The value of the source or destination type may be extended, and one or more bytes will be added accordingly. For an example, refer to the CONNECT control command. A source or destination value of FF₁₆ is a special case and indicates that the AV device may select any appropriate, available subunit.

If the source or destination type is one, the corresponding source or destination operand is ignored. This value may be used to leave existing connections unchanged or it may be used if the AV unit does not implement the connection type. For example, in a CONNECT AV control command sent to an AV unit that had only audio recording capabilities, it would be appropriate to specify a value of one for both *video_source_type* and *video_dest_type*.

If the source or destination type is two, the corresponding source or destination operand represents a Serial Bus or an external plug, as encoded by the table below:

Value	Plug
0 — 1E ₁₆	Serial Bus plug zero — 30
1F ₁₆ — 7E ₁₆	Reserved for future specification
7F ₁₆	Any available Serial Bus plug
80 ₁₆ — 9E ₁₆	External plug zero — 30
9F ₁₆ — FE ₁₆	Reserved for future specification
FF ₁₆	Any available external plug

NOTE: In the preceding, some of the encoded values permit the AV device to select, at its option, an available subunit, Serial Bus or external plug. The set of plugs from which the device may choose is further limited by what is appropriate. For example, a dual-deck VCR might have one deck capable of recording SD signals and another capable of recording both HD and SD signals. If a Serial Bus input plug is active and configured for HD signals, a CONNECT AV control command for an audio/video stream that specified “any available” subunit would result in the natural connection to the deck capable of recording HD signals. On the other hand, if a Serial Bus input plug is active and configured for SD signals, an arbitrary connection could be established with either deck. In cases where more than one choice is possible, it is expected that the determination will be vendor-dependent.

In addition to its use as a control command, the CONNECT AV command may also be used as a status command to determine the current state of internal A/V connections for a unit or subunit. The form is shown in Figure 8-8 below.

	msb						lsb
opcode	CONNECT AV (20 ₁₆)						
operand[0]	F ₁₆			video_dest_type		audio_dest_type	
operand[1]	FF ₁₆						
operand[2]	FF ₁₆						
operand[3]	video_destination						
operand[4]	audio_destination						

Figure 8-8 — CONNECT AV status command format for audio/video stream

The fields *video_dest_type*, *audio_dest_type*, *video_destination* and *audio_destination* are used as previously described for the CONNECT AV command.

The response frame returned for the CONNECT AV status command has the same format as described in Figure 8-7. This includes fields for extended subunit type if used.

In case there is no source plug connected to the destination plug indicated in the CONNECT AV status command, the *video_source* and *audio_source* fields shall have the value FF₁₆ (invalid), and the *video_source_type* and *audio_source_type* fields shall both have the value 1 (ignore).

The CONNECT AV command may also be used as a notify command. The notify command has the same syntax as the CONNECT AV status command. A notification shall be returned by the target to the controller that issued the notify command in case a connection

involving the destination as indicated in the notify command changes. These changes shall include establishing a connection to the destination, deleting a connection from the destination, and connecting the destination to another source. The notify response has the same format as the CONNECT AV response frame.

8.4 CONNECTIONS command

The CONNECTIONS status command is used to inquire the state of all connections for unspecified streams. The format of the CONNECTIONS status command is illustrated by Figure 8-9 below.

	msb						lsb
opcode	CONNECTIONS (22 ₁₆)						
operand[0]	FF ₁₆						

Figure 8-9 — CONNECTIONS status command format

The response frame returned after a CONNECTIONS status command is variable in length and depends upon the number of connections established. The response frame has the format defined by Figure 8-10 below.

	msb						lsb
opcode	CONNECTIONS (22 ₁₆)						
operand[0]	total_connections						
operand[1]	3F ₁₆					lock	perm
operand[2]	connection[0].source						
operand[3]							
operand[4]	connection[0].destination						
operand[5]							
...	connection[1] — connection[total_connections - 2]						
operand[n-4]	3F ₁₆					lock	perm
operand[n-3]	connection[total_connections - 1].source						
operand[n-2]							
operand[n-1]	connection[total_connections - 1].destination						
operand[n]							

Figure 8-10 — CONNECTIONS response format

The *total_connections* field specifies the number of five-byte connection descriptors returned in the operands that follow. The value of *n* is determined by $5 * total_connections$.

The format of each connection descriptor is identical to operand[1] through operand[4] of the CONNECT control command. For a connection that includes an extended subunit_type or subunit_ID, these addresses may change depending on the number of extended fields.

8.5 DIGITAL INPUT command

The DIGITAL INPUT control command permits an AV unit to establish a broadcast input connection according to its own preferences. Figure 8-11 below illustrates the format of the command.

	msb							lsb
opcode	DIGITAL INPUT (11 ₁₆)							
operand[0]	connection_state							

Figure 8-11 — DIGITAL INPUT command format

When the DIGITAL INPUT command is issued with a *ctype* value of CONTROL, the *connection_state* field specifies whether the AV unit is expected to establish (70₁₆) or break (60₁₆) a broadcast input connection.

The DIGITAL INPUT command, with a *ctype* value of STATUS, may also be used to determine the current input broadcast connection state of the unit. In this case, *operand[0]* is set to FF₁₆ when the status command is issued and is updated to the current *connection_state* when the STABLE response frame is returned.

8.6 DIGITAL OUTPUT command

The DIGITAL OUTPUT control command permits an AV unit to establish a broadcast output connection according to its own preferences. Figure 8-12 below illustrates the format of the command.

	msb							lsb
opcode	DIGITAL OUTPUT (10 ₁₆)							
operand[0]	connection_state							

Figure 8-12 — DIGITAL OUTPUT command format

When the DIGITAL OUTPUT command is issued with a *ctype* value of CONTROL, the *connection_state* field specifies whether the AV unit is expected to establish (70₁₆) or break (60₁₆) a broadcast output connection. The AV unit shall be responsible to allocate or deallocate the necessary isochronous resources, *e.g.*, bandwidth and channel number, and to program an output PCR as appropriate.

The DIGITAL OUTPUT command, with a *ctype* value of STATUS, may also be used to determine the current output broadcast connection state of the unit. In this case, *operand[0]* is set to FF₁₆ when the status command is issued and is updated to the current *connection_state* when the STABLE response frame is returned.

8.7 DISCONNECT command

The DISCONNECT control command removes a connection between a destination and a source plug for an unspecified stream as described in the CONNECT control command, even if the connection was established with the lock bit set to one. In the case where multiple connections are overlaid on the same source plug, all connections will be deleted.

The format of the DISCONNECT control command is illustrated by Figure 8-13 below.

	msb							lsb
opcode	DISCONNECT (25 ₁₆)							
operand[0]	FF ₁₆							
operand[1]	source_subunit_type					source_subunit_ID		
operand[2]	source_plug							
operand[3]	destination_subunit_type					destination_subunit_ID		
operand[4]	destination_plug							

Figure 8-13 — DISCONNECT command format

The meaning of all fields are identical to the fields as described in the CONNECT control command. This includes the extended source and destination *subunit_type* and *subunit_ID* if they are used.

8.8 DISCONNECT AV command

The DISCONNECT AV control command is used to remove audio/video connections between subunits and plugs. The value of *operand[0]* is other than FF₁₆ and the syntax is shown in Figure 8-14 below.

	msb							lsb
opcode	DISCONNECT AV (21 ₁₆)							
operand[0]	video_source_type	audio_source_type		video_dest_type		audio_dest_type		
operand[1]	video_source							
operand[2]	audio_source							
operand[3]	video_destination							
operand[4]	audio destination							

Figure 8-14 — DISCONNECT AV command format

The field definitions and their uses for DISCONNECT AV are identical to the field definitions given in Figure 8-7 for the CONNECT AV command. This includes the extended source and destination *subunit_type* and *subunit_ID* if they are used.

8.9 INPUT PLUG SIGNAL FORMAT command

The INPUT PLUG SIGNAL FORMAT control command is used to configure a specified Serial Bus input plug to receive data in the designated signal format. The syntax of the INPUT PLUG SIGNAL FORMAT control command is shown in Figure 8-15 below.

	msb							lsb
opcode	INPUT PLUG SIGNAL FORMAT (19 ₁₆)							
operand[0]	plug							
operand[1]	2		fmt					
operand[2]	(most significant byte) fdf (least significant byte)							
operand[3]								
operand[4]								

Figure 8-15 — INPUT PLUG SIGNAL FORMAT control command format

The fields *fmt* and *fdf* are as defined in IEC-1883, proposed standard for Digital Interface for Consumer Electronic Audio/Video Equipment. Together they specify the desired signal format for the Serial Bus input plug identified by *plug*.

The INPUT PLUG SIGNAL FORMAT status command is used to inquire which signal format a specified Serial Bus input plug is configured to receive. The syntax of the INPUT PLUG SIGNAL FORMAT status command is shown in Figure 8-16 below.

	msb							lsb
opcode	INPUT PLUG SIGNAL FORMAT (19 ₁₆)							
operand[0]	plug							
operand[1]	FF ₁₆							
...								
operand[4]								

Figure 8-16 — INPUT PLUG SIGNAL FORMAT status command format

The *plug* field specifies which one of the 31 Serial Bus input plugs, zero through 1E₁₆, is referenced.

If the status command is accepted, the response frame has the same format as the INPUT PLUG SIGNAL FORMAT control command illustrated by Figure 8-15 above. The fields *fmt* and *fdf* are as previously defined and together they specify the signal format that the Serial Bus input plug identified by *plug* is configured to receive.

The INPUT PLUG SIGNAL FORMAT command may also be used as a 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 format of the data that the Serial Bus input plug is receiving changes. The notify response has the same format as the status response frame.

8.10 OUTPUT PLUG SIGNAL FORMAT command

The OUTPUT PLUG SIGNAL FORMAT control command is used to configure a specified Serial Bus output plug to transmit data in the designated signal format. The syntax of the OUTPUT PLUG SIGNAL FORMAT control command is shown in Figure 8-17 below.

	msb							lsb
opcode	OUTPUT PLUG SIGNAL FORMAT (18 ₁₆)							
operand[0]	plug							
operand[1]	2		fmt					
operand[2]	(most significant byte)							
operand[3]	fdf							
operand[4]	(least significant byte)							

Figure 8-17 — OUTPUT PLUG SIGNAL FORMAT control command format

The fields *fmt* and *fdf* are as defined in IEC-1883, proposed standard for Digital Interface for Consumer Electronic Audio/Video Equipment. Together they specify the desired signal format for the Serial Bus output plug identified by *plug*.

The OUTPUT PLUG SIGNAL FORMAT status command is used to inquire which signal format a specified Serial Bus output plug is configured to transmit. The format of the OUTPUT PLUG SIGNAL FORMAT command is illustrated by Figure 8-18 below.

	msb							lsb
opcode	OUTPUT PLUG SIGNAL FORMAT (18 ₁₆)							
operand[0]	plug							
operand[1]	FF ₁₆							
...								
operand[4]								

Figure 8-18 — OUTPUT PLUG SIGNAL FORMAT status command format

The *plug* field specifies which of the 31 Serial Bus output plugs, zero through 1E₁₆, is referenced.

If the status command is accepted, the response frame has the same format as the OUTPUT PLUG SIGNAL FORMAT control command illustrated by Figure 8-17 above. The fields *fmt* and *fd* are as previously defined and together they specify the signal format that the Serial Bus output plug identified by *plug* is configured to transmit.

The OUTPUT PLUG SIGNAL FORMAT command may also be used as a 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 format of the data that the Serial Bus output plug is transmitting changes. The notify response has the same format as the status response frame.

8.11 SUBUNIT INFO command

The SUBUNIT INFO status command is used to obtain information about the subunit(s) of an AV unit. The format of the SUBUNIT INFO status command is illustrated by Figure 8-19 below.

	msb						lsb
opcode	SUBUNIT INFO (31 ₁₆)						
operand[0]	0	page			0	extension_code	
operand[1]	FF ₁₆						
...							
operand[4]							

Figure 8-19 — SUBUNIT INFO status command format

The *page* field value specifies which part of the subunit table is to be returned. An AV unit may implement up to 32 bytes of information in eight pages.

The *extension_code* field shall have a value of seven.

If the status command is accepted, the response frame returned has the structure illustrated by Figure 8-20 below.

	msb							lsb
opcode	SUBUNIT INFO (31 ₁₆)							
operand[0]	0	page			0	extension_code		
operand[1]	page_data							
...								
operand[n]								

Figure 8-20 — SUBUNIT INFO response format

The *page_data* returned is the four entries from the subunit table for the page requested. The subunit table is an array of byte entries; each entry has the format defined by Figure 8-21 below:

msb							lsb
subunit_type				max_subunit_ID			

Figure 8-21 — Subunit table entry

The *subunit_type* field of each entry is as defined in Table 5.3-1. The extension of *subunit_type* and *max_subunit_ID* follows the example explained in the CONNECT control command; so, a subunit table entry may be more than one byte in length.

The *max_subunit_ID* field is the count of subunits of *subunit_type* implemented by the AV unit, less one.

The subunit entries are not required to be in any particular order but are required to be uniquely identified by *subunit_type*. If fewer than 32 entries are present in the subunit table, they are terminated by a byte with the value FF₁₆. The value of entries past the terminating FF₁₆ is indeterminate and should be ignored by any controller that requests subunit information.

8.12 UNIT INFO command

The UNIT INFO status command is used to obtain information that pertains to the AV unit as a whole (distinct from subunit information, see 8.11). The format of the UNIT INFO status command is illustrated by Figure 8-22 below:

	msb						lsb
opcode	UNIT INFO (30 ₁₆)						
operand[0]	FF ₁₆						
...							
operand[4]							

Figure 8-22 — UNIT INFO status command format

If the status command is accepted by the target, a response frame with the format shown by Figure 8-23 below is returned.

	msb							lsb
opcode	UNIT INFO (30 ₁₆)							
operand[0]	07 ₁₆							
operand[1]	unit_type					unit		
operand[2]	(most significant byte)							
operand[3]	company_ID							
operand[4]	(least significant byte)							

Figure 8-23 — UNIT INFO response format

The *unit_type* field contains a value whose meaning is identical to those defined for subunit_type in Table 5.3-1. Value 1C₁₆ (vendor unique) should be returned in case none of the other values are considered to be appropriate. The *unit_type* field may take value 1E₁₆, which means that the field is extended to the following byte. In that case, an additional byte for *extended_unit_type* will be added immediately following operand[1], as shown in the example presented in the CONNECT control command. Further extension is possible when the value of *extended_unit_type* is FF₁₆, in which case another byte will be added.

The meaning of the *unit* field is not defined by this specification.

The *company_ID* field shall contain the 24-bit unique ID obtained from the IEEE Registration Authority Committee (RAC). It is expected that the value of company_ID returned by the UNIT INFO status command is the same as the vendor ID in the Node Unique ID leaf in the AV unit's configuration ROM. The most significant part of the company_ID is stored in *operand[2]* and the least significant part in *operand[4]*.

9. Common unit and subunit commands

This section defines commands that are applicable to an AV unit as well as a subunit independent of the functionality that these subunits represent indicated by their *subunit_type*. Table 8.12-1 below summarizes the common unit and subunit commands.

Table 8.12-1 — Common unit and subunit commands

Opcode	Value	Support level (by <i>ctype</i>)			Comments
		C	S	N	
POWER	B2 ₁₆	O	O	R	Control power state
RESERVE	01 ₁₆	O	O	R	Acquire or release exclusive control of a target
PLUG INFO	02 ₁₆	–	O	–	Information about Serial Bus and External plugs
VENDOR-DEPENDENT	00 ₁₆	V	V	V	Vendor-dependent commands

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 common subunit commands in the clauses that follow.

9.1 POWER command

The POWER control command is used to control the power status of the AV unit or one of its subunits determined by the AV/C address that is contained in the AV/C frame. The format of the POWER command is illustrated by Figure 9-1 below.

	msb							lsb
opcode	POWER (B2 ₁₆)							
operand[0]	power_state							

Figure 9-1 — POWER command format

When the POWER command is issued with a *ctype* value of CONTROL, the *power_state* field specifies the desired power state of the unit. Power on is encoded as 70₁₆ and power off as 60₁₆.

Setting the power status of the AV unit to on or off shall cause the power of all of its subunits to be set in the same way. Setting the power status of a subunit shall not affect the power status of the AV unit or any of the other subunits.

The POWER command with a *ctype* value of STATUS may be used to determine the current power state of the AV unit or one of its subunits. In this case, *operand[0]* is set to 7F₁₆ when the command is issued and is updated to the current power state when the STABLE response is returned.

The POWER command may also be used as a 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 power state of the addressed unit or subunit changes. The notify response has the same format as the response frame.

9.2 RESERVE command

The RESERVE control command permits a controller to acquire or release exclusive control of the AV unit or one of its subunits determined by the AV/C address that is contained in the AV/C frame. The format of the command is illustrated by Figure 9-2 below.

	msb							lsb
opcode	RESERVE (01 ₁₆)							
operand[0]	priority							
operand[1]	text							
...								
operand[12]								

Figure 9-2 — RESERVE control command format

The *priority* field shall specify the relative priority associated with the command. Zero has special meaning and indicates that no controller has reserved the AV (sub)unit. The other values, between one and 0F₁₆, indicate that the target holds a reservation for a controller. A *priority* value of four is, by convention, the standard priority that controllers are expected to use in the absence of other reasons for choosing a higher or lower priority.

The *text* field provides for up to 12 bytes of ASCII characters. If no *text* string is present, the bytes are expected to have a value of FF₁₆.

An AV (sub)unit accepts RESERVE control commands according to the following rules:

after a power-on reset or a command reset, the AV (sub)unit is in a free state and reports a *priority* value of zero in response to any RESERVE inquiries (see the discussion of RESERVE when *c_{type}* has a value of STATUS, below).

- a) an AV (sub)unit that is in the free state may be reserved by any controller that issues a RESERVE control command. The target shall internally record the *priority* at which the reservation is made, the *text* string that accompanies the reservation and the 16-bit node ID of the controller. An ACCEPTED response guarantees to the controller that the reservation has succeeded.

NOTE: When a priority value is accepted by an AV (sub)unit and a reservation is established, the stored value is transformed according to the following table.

Command priority	Stored priority
0 — 1	priority
02 ₁₆ — 0E ₁₆	priority & 0E ₁₆
0F ₁₆	priority

This has the effect of rounding most odd priorities down to a smaller even value.

- b) while an AV (sub)unit holds a reservation for a controller, it shall reject any control commands other than RESERVE with a *c_{type}* of CONTROL that are issued by any other controller. The 16-bit node ID stored by the AV (sub)unit upon receipt of the RESERVE control command is the basis for accepting or rejecting control commands for controllers.
- c) if a RESERVE control command is received from the same controller that holds the reservation, it shall be accepted. This permits the original controller to raise or lower the *priority* associated with the reservation.

- d) if a RESERVE control command is received from a different controller than that which made the reservation, the AV (sub)unit shall reject the command unless the *priority* is greater than the current reservation priority. In the case where the new priority is greater than the current priority, the existing reservation is preempted and a reservation is established for the new controller according to the procedures already described in b).
- e) If a RESERVE control command is addressed to the AV unit but that AV unit contains a subunit that already holds a reservation with an equal or higher priority, the RESERVE control command shall return a REJECTED response.
- f) If a RESERVE control command is addressed to the AV unit and that AV unit contains no subunits that are already reserved with an equal or higher priority, then each existing reservation of a subunit shall be preempted and a reservation of the AV unit is established for the new controller according to the procedures already described in b).
- g) Any control command that is addressed to a subunit within an AV unit that is reserved by a different controller than the one which issued the control command, shall be rejected.

When an AV (sub)unit detects a Serial Bus reset, it shall reset its reservation priority to zero (free) and set both the reservation node ID and the reservation text to values of all ones. Then, until the reservation has been reestablished, or until a period of ten seconds has elapsed, it shall reject all commands with a *ctype* of CONTROL except for RESERVE commands. This procedure permits the original holder of the reservation to reestablish the reservation with its reassigned node ID after the bus reset.

NOTE: Controllers shall not issue RESERVE control commands within ten seconds of a bus reset unless they had established a reservation with the target AV (sub)unit prior to the bus reset. Because the node ID of the AV unit may have changed after the bus reset, a controller that wishes to reestablish (sub)unit reservations is expected to examine the unique identifier, EUI-64, in configuration ROM to locate the AV (sub)unit previously reserved.

Because of this restriction, the target can assume that a RESERVE command received within 10 seconds of a bus reset is legitimate, and shall therefore accept the reservation. Any controller may request the current reservation status of an AV (sub)unit by issuing a RESERVE command with a *ctype* field of STATUS, in the format shown in Figure 9-3.

	msb							lsb
opcode	RESERVE (01 ₁₆)							
operand[0]	FF ₁₆							
...								
operand[12]								

Figure 9-3 — RESERVE status command format

If a response frame is returned that indicates STABLE, *operand[0]* holds the current reservation priority and *operand[1]* through *operand[12]* hold the text string stored at the time the reservation was established. There is no way to determine the identity of the controller that holds the reservation.

Controllers that wish to be advised of a possible change of status of their own reservations, for example preemption by another controller by means of a higher priority reservation, should issue a RESERVE command in the format shown in Figure 9-3 but with a *ctype* value of NOTIFY. If a new reservation is established, the original reservation holder is notified by an AV/C response frame with CHANGED status and operand values that reflect the new reservation.

NOTE: Any new reservation results in CHANGED status, even a reservation made by the same controller that already holds a reservation. A response frame is returned to any outstanding notify command in all of these cases.

9.3 PLUG INFO command

The PLUG INFO status command is used to inquire about the number of plugs on the AV unit or one of its subunits determined by the AV/C address contained in the AV/C frame. The format of the PLUG INFO status command is illustrated by Figure 9-4 below.

	msb							lsb
opcode	PLUG INFO (02 ₁₆)							
operand[0]	00 ₁₆							
operand[1]	FF ₁₆							
operand[3]								
operand[4]								

Figure 9-4 — PLUG INFO status command format

If the PLUG INFO status command was addressed to an AV subunit, the format of the response frame is shown in Figure 9-5 below.

	msb							lsb
opcode	PLUG INFO (02 ₁₆)							
operand[0]	00 ₁₆							
operand[1]	destination_plugs							
operand[2]	source_plugs							
operand[3]	FF ₁₆							
operand[4]	FF ₁₆							

Figure 9-5 — PLUG INFO status response format from an AV subunit

For the AV subunit response frame, operand[1] and operand[2] shall indicate the number of destination and source plugs of that AV subunit, and operand[3] and operand[4] shall have the value FF₁₆.

If the PLUG INFO status command was addressed to an AV unit, the response frame returned is illustrated by Figure 9-6 below.

	msb							lsb
opcode	PLUG INFO (02 ₁₆)							
operand[0]	00 ₁₆							
operand[1]	Serial_Bus_input_plugs							
operand[2]	Serial_Bus_output_plugs							
operand[3]	External_input_plugs							
operand[4]	External_output_plugs							

Figure 9-6 — PLUG INFO response format from an AV unit

If the PLUG INFO status command is addressed to the AV unit, operand[1] and operand[2] shall indicate the number of Serial Bus input and output plugs, respectively, while operand[3] and operand[4] shall indicate the number of external input and output plugs, respectively.

9.4 VENDOR-DEPENDENT commands

The VENDOR-DEPENDENT command permits module vendors to specify their own set of commands and responses for AV units or subunits determined by the AV/C address that is contained in the AV/C frame. The structure of the command is illustrated by Figure 9-7 below.

	msb						lsb
opcode	VENDOR-DEPENDENT (00 ₁₆)						
operand[0]	(most significant byte)						
operand[1]	company_ID						
operand[2]	(least significant byte)						
operand[3]	vendor_dependent_data						
...							
operand[n]							

Figure 9-7 — VENDOR-DEPENDENT command format

The *company_ID* field shall contain the 24-bit unique ID obtained from the IEEE Registration Authority Committee (RAC). It is expected that the value of *company_ID* provided in the operands of vendor-dependent commands be the same as the vendor ID in the Node Unique ID leaf in configuration ROM of the AV unit to which the command is addressed. The most significant part of the *company_ID* is stored in *operand[0]* and the least significant part in *operand[2]*.

The format and meaning of the *vendor_dependent_data* field are specified by the vendor identified by *company_ID*.

Although the behavior of vendor-dependent commands is beyond the scope of this specification, it is recommended that vendor-dependent be defined in the same five command types, CONTROL, SPECIFIC INQUIRY, STATUS, NOTIFY and GENERAL INQUIRY, specified by the *ctype* field described in 5.3.1.

A. AV/C commands in numerical order (normative)

The table below lists all the AV/C commands, in numerical order by *opcode*. Commands that pertain to subunits in addition to units are indicated by an X in the Subunit commands column. The legend for the subunit types follows the table.

Value	Opcode	Unit command	Subunit commands	Support level (by ctype)		
				C	S	N
00 ₁₆	VENDOR-DEPENDENT	X	X	V	V	V
01 ₁₆	RESERVE	X	X	O	O	R
02 ₁₆	PLUG INFO	X	X	–	O	–
10 ₁₆	DIGITAL OUTPUT	X		O	O	–
11 ₁₆	DIGITAL INPUT	X		O	O	–
12 ₁₆	CHANNEL USAGE	X		–	R	R
18 ₁₆	OUTPUT PLUG SIGNAL FORMAT	X		O	R	O
19 ₁₆	INPUT PLUG SIGNAL FORMAT	X		O	R	O
20 ₁₆	CONNECT AV	X		O	O	O
21 ₁₆	DISCONNECT AV	X		O	–	–
22 ₁₆	CONNECTIONS	X		–	O	–
24 ₁₆	CONNECT	X		O	O	R
25 ₁₆	DISCONNECT	X		O	–	–
30 ₁₆	UNIT INFO	X		–	M	–
31 ₁₆	SUBUNIT INFO	X		–	M	–
B2 ₁₆	POWER	X	X	O	O	R

In the preceding table, an asterisk in the support level column indicates that the command operands or the type of subunit determine whether the command is mandatory (M), recommended (R), optional (O), or vendor-dependent (V).

B. Unresolved issues (informative)

This annex describes areas of the AV/C Digital Interface Command Set that are not yet fully resolved or subject to ambiguous interpretations by implementors.

It is recommended that this informative annex remain a part of the specification until the 1394 Trade Association is able to resolve the ambiguities.

B.1 Command execution model

There is no well articulated model for how commands are to be executed, to what degree command queuing is possible, whether or not response frames are required to be returned in the same order as the corresponding commands were initially issued, how (precisely) response frames are to be correlated with their command frames and so on.

The document editors have been unable to describe consistent behavior for AV devices in this area. The document editors are concerned that these ambiguities will permit varying implementations of both peripherals and host software such that interoperability is compromised.

The matter of response frame identification is troublesome because there is no way to uniquely guarantee that a particular response frame can be matched with its command frame.

B.2 Remote bus resets

In section 6, the behavior of an AV device when it detects a Serial Bus reset on the local bus is described. The AV device shall discard any in progress transactions without the return of a response frame. This is intended to prevent the return of a response frame to the incorrect controller, since the 6-bit physical ID of the controller may have changed as the result of a bus reset.

The same considerations apply if the controller is located on a remote bus that experiences a bus reset: the physical ID's on that bus may change. This in turn implies that an AV device, if it is to be able to be controlled by a remote controller must have some way to detect the occurrence of a bus reset on the remote bus, in order to be able to discard an in progress transactions.

B.3 Notification support

It is desirable for a controller to have the opportunity to receive notification for any change of state in a target. This implies that all commands which cause state changes need to have notification specifications. There are some commands currently defined without notification support as an option, such as DIGITAL INPUT, DIGITAL OUTPUT, and a few others. Future work should focus on defining them.

B.4 Identifying the Specific Type of Subunit

In the situation where given subunit type has variations, it would be useful for a controller to be able to distinguish what variant it is talking to. For example, the VCR subunit type now has DVCR and D-VHS variants (or subtypes?). We need a method of classifying this new relationship model, and for providing this information to controllers.