TA Document 2001012

AV/C Digital Interface Command Set General Specification Version 4.1

December 11, 2001

Sponsored by:
1394 Trade Association

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

Abstract:

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

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

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

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

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

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

Editor, 1394 Trade Association
1111 South Main Street, Suite 122
Grapevine, TX, 76051, USA
Tel: (817) 410-5750

Table of contents

List of figures

List of tables

1. Overview

1.1 Purpose

This document specifies the AV/C protocol used for controlling consumer audio/video (AV/C) devices on not only a 1394 bus but also on other bus(es), describes the AV/C unit and subunit model, and defines common unit and subunit commands that are addressed using that model. This document also serves as a baseline for other AV/C subunit type and other unit specifications. A unit specification is one that defines functionality of a unit as a whole. For example, references [R12] and [R15] are unit specifications that define making connections among units and subunits.

This document builds upon an extensive body of standards work, underway and completed, as referenced in section 2. 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 part of the IEC 61883 standard. Similarly, the format of the isochronous data itself has been developed by the HD Digital VCR Conference.

1.2 Scope

This document replaces *AV/C Digital Interface Command Set General Specification 4.0* [R17].

Reference [R17]covers the AV/C general command and response model, unit/subunit model, and standard unit and subunit commands including connection commands. The predecessor documents, *AV/C Digital Interface Command Set General Specification Version 3.0* [R8], have been separated into two documents - the *AV/C Digital Interface Command Set General Specification Version 4.0* and the *AV/C Descriptor Mechanism Specification Version 1.0* documents [R10]. The document, - *Enhancement to the AV/C General Specification Version3.0, Version 1.1* [R9] has been contained in reference [R10] while *AV/C Info Block Types Specification Version 1.0* [R11] has been separated from reference [R9]. We encourage you strongly to read this document in conjunction with the references given below, as well as with any AV/C-related documents that may be created in the future.

2. References

The following standards contain provisions, which through reference in this document constitute provisions of this standard. All the standards listed are normative references.

2.1 Reference sources

All listed references are available at various web sites. Some web sites require membership to access the references, and other sites require payment for each reference. The following sites contain the references used in this document. The reader is encouraged to always consult these sites for information on the latest versions of specifications mentioned here, as well as specifications that may be developed in the future.

[R1] 1394TA web site, http://www.1394TA.org. This web site is kept up to date with the latest released and draft versions of AV/C specifications. You need to be a member to access draft specifications.

[R2] International Electro-technical Commission web site, http://www.iec.ch. This web site contains specifications that must be purchased.

[R3] IEEE standards web site. http://standards.ieee.org. This web site contains specifications that must be purchased.

2.2 Specifications

At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

[R4] IEEE Std 1394–1995, *Standard for a High Performance Serial Bus*, August 30 1996.

[R5] IEEE Std 1394a-2000, *Standard for High Performance Serial Bus-Amendment 1*.

[R6] ISO/IEC 13213:1994, *Control and Status Register (CSR) Architecture for Microcomputer Buses*, First Edition, October 5, 1994.

[R7] IEC 61883-1, *Consumer audio/video equipment – Digital Interface*, 1998-02.

[R8] TA Document 1998003, *AV/C Digital Interface Command Set General Specification, Version 3.0*, April 15, 1998.

[R9] TA Document 2000004, *Enhancement to the AV/C General Specification Version 3.0, Version 1.1*, October 24, 2000.

[R10] TA Document 1999025, *AV/C Descriptor Mechanism Specification Version 1.0* Draft

[R11] TA Document 1999045, *AV/C Information Block Types Specification Version 1.0* Draft

[R12] TA Document 1999032, *Connection and Compatibility Management, 1.0*, July 10, 2000.

[R13] TA Document 2000005, *AV/C Compatible Asynchronous Serial Bus Connections 2.0*, June 22, 2000.

[R14] TA Document 2000006, *AV/C Commands for Management of Asynchronous Serial Bus Connections 1.1*, June 22, 2000.

[R15] TA Document 1999037, *AV/C Command for Management of Enhanced Asynchronous Serial Bus Connections 1.0*, June 13, 2000.

[R16] TA Document 1999008, *AV/C Audio Subunit 1.0*, August 21, 2000.

[R17] TA Document 1999026, *AV/C Digital Interface Command Set General Specification 4.0*, July 23, 2001.

3. Changes from previous version

The following table shows the change history for this specification.

3.1 Version 1.0

Original Version

3.2 Version 2.0

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

Table 3.1 – Content change for version 2.0

Category	Description
Technical	The disc recorder/player type has been added, and the subunit type 05 has been changed to “Tuner” from “TV Tuner”.
Technical	A model for extended subunit addressing has been defined in section 7.3.3. As a result, item C3 in Annex C (extended subunit addressing) has been removed (what was item C4 - Notification Support - is now item C3).
Technical	A process for defining new device types and command sets has been defined in section 7.3.4.4.
Technical	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.
Editorial	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.

3.3 Version 2.0.1, January 5, 1998

Version 2.0.1 of this document differs from version 2.0 in the following ways:

Table 3.2 – Content change for version 2.0.1

Category	Description
Editorial	The AV/C Digital Interface Command Set 2.0 manual was separated into two books: General Specification and the VCR Subunit Specification, each assigned version 2.0.1

3.4 Version 3.0, April, 1998

Table 3.3 – Content change for version 3.0

Category	Description
Technical	The AV/C Descriptor Mechanism chapter was added.
Technical	The OPEN DESCRIPTOR, READ DESCRIPTOR, WRITE DESCRIPTOR, SEARCH DESCRIPTOR and OBJECT NUMBER SELECT commands were added.

3.5 Version 4.0, January, 2000

Table 3.4 – Content change for version 4.0

Category	Description
Editorial	Descriptor Mechanism was moved to a separate document.
Editorial	The new AVWG Template was applied to this document. The standard introductory sections were changed to conform to the new template.
Editorial	The References page was updated to include references on the Internet, and new relevant AV/C documents.
Editorial	The glossary was updated to include only those terms that are in the document.
Editorial/Technical	A data structure conventions chapter was added to explain the data structures presented in the document. The data structures include command frames with new *length* and *ck* columns, and command-response tables. Check *ck* fields to determine a NOT IMPLEMENTED response of a command.
Editorial	All the commands in the General 4.0 suite now include command-response tables which show in detail the field values in all response frames (ACCEPTED, REJECTED, INTERIM, etc.)
Editorial	The possible combinations of *ctypes* and *responses* were clarified using a table.
Technical	Clarified how to extend *subunit_type* and *subunit_ID* in the AV/C Command frame.
Technical	More subunit types were added to the subunit types table.
Editorial	Clarified the difference between unit commands, unit and subunit commands, and subunit commands.
Editorial/Technical	Added a section on target error checking of command frames. Based on the error checking levels, a clarification was made between REJECTED and NOT IMPLEMENTED responses.
Editorial	Discussed backward compatibility issues, and how future AV/C documents should and shouldn't generally change to ensure backward compatibility.
Editorial	Discussed legacy device behavior – what AV/C targets should do when new command frames are received at a legacy target.
Editorial	Clarified immediate and deferred transactions.

Category	Description
Technical	Clarified some of the AV/C Transaction rules for controllers and targets: 1. Recommended that subunit-type specifications define a time limit for responses after an INTERIM response. 2. Receipt acknowledgement is provided by a 1394 WRITE response or 1394 ack_complete. 3. If targets are preoccupied with another command, it should return with ack_busy or resp_conflict error. 4. After Notify interim response, the target receives a 1394 write response acknowledgement from the controller. 5. State change requests to a target already in that state shall return ACCEPTED. 6. Maximum packet size received by a target is specified in the target's max_rec field or 512 bytes, whichever is less.
Technical	Made it a recommendation that targets support a STATUS command after a NOTIFY INTERIM.
Editorial	Clarified the association between 1394 transactions and AV/C transactions.
Editorial	Developed figures describing an AV/C transaction using the 1394 split transaction and unified transaction models.
Technical	Single-tasking targets shall *not* ignore AV/C commands if preoccupied with another command, such as after returning an INTERM, but instead shall return a REJECTED response.
Editorial	For STATUS commands, clarified that a target has various states, and that each STATUS command has access to a subset of these states.
Technical	Added information about First Priority and Last Priority handling of NOTIFY command frames, and presented guidelines for NOTIFY handling.
Editorial	Added a section (10) on the AV/C Unit Model describing Serial Bus Isochronous, Serial Bus Asynchronous, and External plugs.
Editorial	Serial Bus Input and Output plugs are now termed Serial Bus Isochronous Input and Output plugs, to differentiate them from Serial Bus Asynchronous Input and Output plugs.
Editorial	General commands now include: POWER UNIT INFO SUBUNIT INFO RESERVE VERSION VENDOR-DEPENDENT
Technical	POWER command: Specified that the POWER command directed to a subunit may have an effect of turning on or off other subunits or the unit. Phy and link layers are not directly affected.
Technical	RESERVE command: Clarified the scope of the reservation that a target prevents controls by another controller.
Editorial	UNIT INFO command: Clarified that the unit_type field specifies a subunit_type that best describes the unit or is vendor unique.
Editorial	SUBUNIT INFO command: Clarified the use of page_data, and the use of extended subunit types and IDs.
Technical	The VERSION command was added.

Category	Description
Editorial	Connection commands now include: PLUG INFO CHANNEL USAGE CONNECT CONNECT AV CONNECTIONS DIGITAL INPUT DIGITAL OUTPUT DISCONNECT DISCONNECT AV INPUT PLUG SIGNAL FORMAT OUTPUT PLUG SIGNAL FORMAT
Technical	PLUG INFO Command: Included asynchronous input plugs for this command as given in TA Document 2000006, *AV/C Command for Management of Asynchronous Serial Bus Connections 1.1*.
Editorial	CONNECT Command: Clarified command and response field values.
Editorial	CONNECT AV Command: Clarified the difference between this command and CONNECT Command.
Editorial	CONNECTIONS Command: Added a ”connection information” frame to help organize layout of the command.
Editorial/Technical	INPUT and OUTPUT PLUG SIGNAL FORMAT command: Clarified **eoh**, **form**, **fmt** and **fdf** fields in these commands.
Editorial	Added information about target state change sources in Annex A.

3.6 Version 4.1, December, 2001

Table 3.5 – Content change for version 4.1

Category	Description
Technical	The AV/C General Bus Plug and related commands were added.

4. Definitions

4.1 Conformance levels

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

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

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

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

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

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

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

4.2 Glossary of terms

4.2.1 APR: Asynchronous Plug Control Register.

4.2.2 iAPR: Input plug control register for controlling asynchronous data streams.

4.2.3 oAPR: Output plug control register for controlling asynchronous data streams.

4.2.4 Asynchronous: asyn “any”– chronous – “time”. Asynchronous is an adjective used to describe data transfers are not sent at fixed time intervals. Asynchronous transfers are usually used for time in-sensitive data such as control commands.

4.2.5 AV/C unit: A consumer electronic device that throughputs Audio and/or Video data, *e.g.*, a camcorder or a VCR, attached as a Serial Bus node. This document describes a command set that can be built into AV/C units to control other AV/C units with the same architecture.

4.2.6 AV/C subunit: A part of an AV/C unit that is uniquely defined and offers a subset of functions that belong to the unit.

4.2.7 AV/C: Audio/video control. The AV/C Digital Interface Command Set of which a part is specified by this and other AV/C documents.

4.2.8 Byte: Eight bits of data.

4.2.9 Controller: A device at a serial bus node that sends AV/C commands to control a remote AV/C target device.

4.2.10 CSR: A Control and Status Register within a node, as defined by IEEE Std 1394–1995.

4.2.11 Data structure: A grouping of data fields in a particular and recognizable format.

4.2.12 FCP: Function Control Protocol, as defined by IEC 61883, Digital Interface for Consumer Electronic Audio/Video Equipment.

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

4.2.14 Isochronous: iso – "same" chronous – "time". Isochronous is an adjective used to describe data block transfers that occur at regular intervals. Isochronous transfers are used for time sensitive data such as audio and video.

4.2.15 Module: A hardware component that is designed to be removed and replaced easily.

4.2.16 Nibble: Four bits of data. A byte is composed of two nibbles.

4.2.17 Node: An addressable device attached to Serial Bus with at least the minimum set of control registers defined by IEEE Std 1394–1995.

4.2.18 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).

4.2.19 PCR: Plug Control Register, as defined by IEC 61883, Digital Interface for Consumer Electronic Audio/Video Equipment.

4.2.20 iPCR: Input plug control register for controlling isochronous data streams, as defined by IEC 61883.

4.2.21 oPCR: Output plug control register for controlling isochronous data streams, as defined by IEC 61883.

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

4.2.23 Port: A subcomponent of a plug that supports unidirectional data transfers.

4.2.24 Quadlet: Four bytes of data.

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

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

4.2.27 Serial Bus: The hardware interconnects and software protocols for the peer-to-peer transport of serialized data, as defined by IEEE Std 1394–1995.

4.2.28 Stream: A continuous flow of data originating from one source and terminating at zero or more destinations. A stream may be isochronous or asynchronous.

4.2.29 Target: A device at a serial bus node that receives and responds to AV/C commands from a remote controller device.

4.2.30 Unit architecture: Software-visible resources that have a form and function, and describe a class of units and their subunits. This document, in conjunction with the references above, defines the AV/C unit and subunit architecture

4.3 Acronyms and abbreviations

APR	Asynchronous Plug Register as defined by AV/C Compatible Asynchronous Serial Bus Connections
AV/C	Audio Video Control
CIP	Common Isochronous Packet
CSR	A Control and Status Register within a node, as defined by IEEE Std 1394–1995
PCR	Plug control register
FCP	Function Control Protocol, as defined by IEC 61883, Digital Interface for Consumer Electronic Audio/Video Equipment.
iAPR	Input asynchronous plug register
iPCR	Input plug control register
oAPR	Output asynchronous plug register
oPCR	Output plug control register
lsb	least significant bit
LSB	Least Significant Byte
msb	most significant bit
MSB	Most Significant Byte

5. Data structure conventions

The following information explains the conventions used in this specification for presenting information in tables and figures.

5.1 Endian-ness

Structures and command frames are always defined with the most significant byte (MSB) of multi-byte fields at the lowest address offset or operand (by number) in the structure or command frame. The most significant bit (msb) of a field is at the highest bit position. For example,

address offset	***msb***							***lsb***
00_{16} (***MSB***)								
01_{16}								
02_{16} (***LSB***)								

Figure 5.1 – MSB/LSB and msb/lsb positions

5.2 Command frame figures

Command frame figures in this specification contain a column for each of opcode/operands, field length, ck validation of fields, and the field name or value.

Field length may have a fixed and/or variable number of bytes. A column for field length denotes the length of the field in bytes.

The length of some fields may be determined by a preceding field or a formula. In the figure below, the length is transferred from the *field_C_length* field as *i* to the length column for *field_C*.

	length	ck	msb							lsb
operand[x]	2	√	field_A							
:										
:	1	√	field_B							
:	2	–	field_C_length= *i*							
:										
:	*i*	√	field_C							
:										

Figure 5.2 – Example of a variable length field

Variable length fields are denoted by a "see[x]" if a length field does not precede the field. The lengths of these fields are determined by some other means and are described in the footnote.

The following command frame example illustrates these concepts:

	length	ck	msb							lsb
opcode	1	√	COMMAND OPCODE (XX_{16})							
operand[0]	1	√	field A							
operand[1]	2	√	field B							
operand[2]										
operand[3]	see[1]	–	field C							
:										
:	1	–	field D							

[1] The length of this field is described here.

Figure 5.3 – Example command frame

NOTE — The opcode/operand column on the far left is used to map the fields to the opcode and operands of the command. When a field has a variable length, this mapping can no longer be determined, and colons ":" are used for all remaining operands.

Command frames shall specify the *ck* (check) column. A check "√" in this column indicates a field that should be validated to return a response of NOT IMPLEMENTED. A dash "–" in this column indicates a field that does not need validation. For more information, see 8.3 AV/C response rules.

5.3 Command-response tables

Command-response tables in this specification contain a column for fields of the command frame, a column for their values or description of their values, and a column for each response type except NOT IMPLEMENTED.

Table 5.1 – Generic command-response table example

Fields	Command	Response		
		ACCEPTED	REJECTED	INTERIM
field 1	FF_{16}	00_{16}	FF_{16}	FF_{16}
field 2	00_{16}	←	←	←
field 3	Explanation of the values for field 3	←	←	←

The arrow "←" for the fields in the command-response table indicate that the value is identical to that of the command frame.

5.4 Naming convention in specifications (informative)

To make the specification easier to read and understand, the following guidelines are suggested for naming fields in data structures:

1) Length fields, which precede fixed named fields are named "*xxxx_length*."
2) An AV/C command with *ctype*=XXXX is denoted by "a XXXX command". For example, "a CONTROL command".
3) AV/C Command names are placed in UPPERCASE. For example, "a UNIT INFO command" or "a UNIT INFO status command" when *ctype* is explicitly specified.
4) An AV/C response with *response*=YYYY is denoted by "a YYYY response". For example, "an ACCEPTED response".

6. Function control protocol (informative)

AV/C commands and responses are transported by the Function Control Protocol (FCP) defined by IEC 61883-1 [R7], Digital Interface for Consumer Electronic Audio/Video Equipment. FCP provides a simple structure to encapsulate device commands and responses within an IEEE Std 1394–1995 [R4] asynchronous Serial Bus block write packet.

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 below.

The format of an FCP frame, encapsulated within a Serial Bus block write packet, is illustrated below:

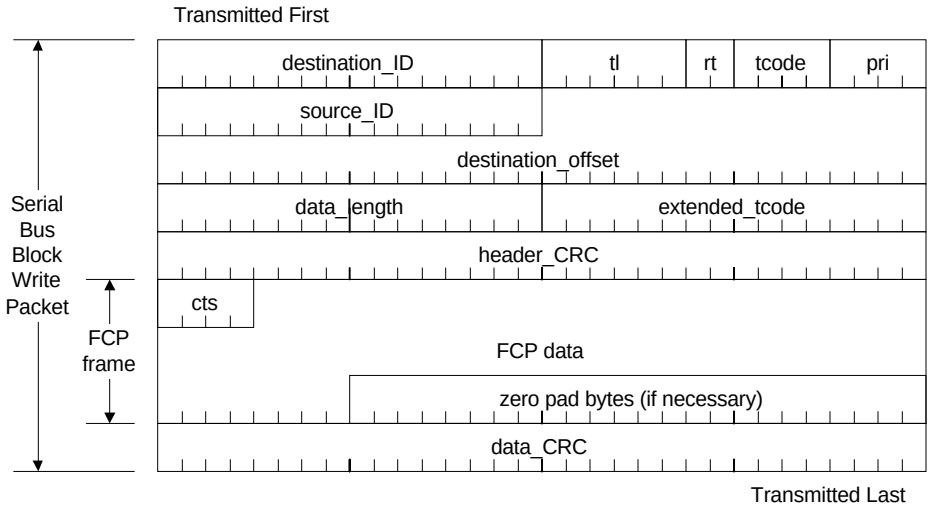

Figure 6.1 – FCP frame within a Serial Bus block write packet

6.1 Serial bus block write packet fields (informative)

The Serial bus block write packet fields are as formally defined by IEEE Std 1394–1995 [R4]. All are briefly described below:

1) **destination_ID:** The *destination_ID* field is used to specify the node ID of the recipient.
2) **tl:** The transaction label (*tl*) identifies a single transaction and is used to determine the corresponding response subaction.
3) **rt:** The retry code (*rt*) is used to specify a retried packet. It also specifies the retry mechanism and protocol that the destination device supports.
4) **tcode:** The transaction code (*tcode*) is used to specify the type of transaction, and thus its format. This field is set to 0001_2 for data block write requests, which are used for AV/C commands.
5) **pri:** The priority field (*pri*) is used to specify a priority for the transaction. When set to all zero, the fair arbitration method is used.
6) **source_ID:** The *source_ID* field is used to specify the node ID of the sender.
7) **destination_offset:** Commands originated by controller are addressed to the FCP_COMMAND register, *destination_offset* FFFF F000 $0B00_{16}$ at the target. The target returns its response(s) to the FCP_RESPONSE register, *destination_offset* FFFF F000 $0D00_{16}$, at the controller.

8) **data_length:** The *data_length* field is used to specify the length of data in the FCP frame. The FCP frame shall not exceed 512 bytes for AV/C commands.

9) **extended_tcode:** The extended transaction code (*extended_tcode*) field is only used for lock-request and lock-response transactions, and is set to zero for AV/C commands which use asynchronous split or unified (not recommended) write transactions.

10) **header_CRC and data_CRC:** These fields contain cyclic redundancy checking values generated using algorithms as stated in the 1394-1995 specification, and are used to validate the header and data block after transmission.

6.2 FCP frame fields

The FCP frame is composed of a cts field and FCP data.

1) **cts**: The command transaction format (cts) field defines the command transaction format used by the FCP frame. For the AV/C command set defined by this document, the *cts* field shall be zero (meaning AV/C).

2) **FCP data:** Data for the Function Control Protocol is defined. The data for the AV/C command set is defined in the subsequent sections in this document.

NOTE — If the data does not end at a quadlet boundary, remaining bytes are padded with zeros when the data is encapsulated in a Serial Bus Block Write Packet.

7. AV/C frames

AV/C command and response frames are encapsulated within FCP frames, as described in the previous section, and transmit AV/C commands and responses between the controller and target FCP_COMMAND and FCP_RESPONSE registers.

This document is primarily concerned with the data that appears only in the write request packets, and not in any other packets.

The format of both the AV/C command and the AV/C response frames are similar, as described in the clauses that follow.

7.1 AV/C command frame

An AV/C command frame contains up to 512 bytes of data and has the structure shown in Figure 7.1 below:

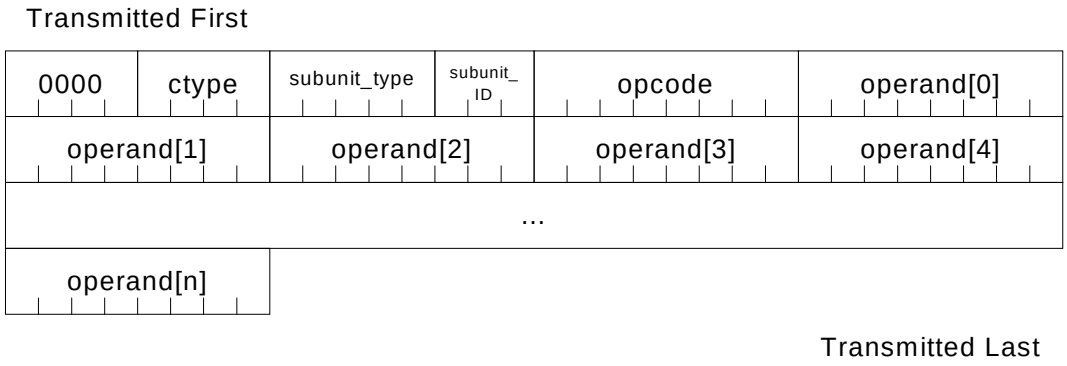

Figure 7.1 – AV/C command frame

All of the operands are optional and are defined based on the values of *ctype*, *subunit_type* and *opcode*.

NOTE — If the size of the AV/C command is greater than the value in the *max_rec* field of the target, then it may not handle the command.

7.2 AV/C response frame

An AV/C response frame has the structure shown in the figure below:

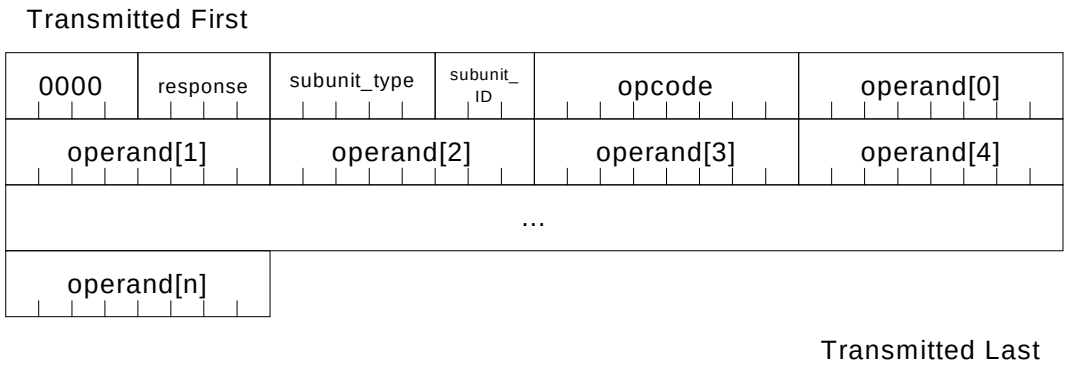

Figure 7.2 – AV/C response frame

All of the operands are optional and are defined based on the values of *response*, *subunit_type* and *opcode*.

7.3 AV/C frame fields

The fields and code values for AV/C command and response frames are defined below.

7.3.1 Command type codes (*ctype*)

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

Table 7.1 – ctype codes

Code	Command type	Description
0_{16}	CONTROL	Used to instruct a target to perform an operation. See section 9.1
1_{16}	STATUS	Used to check a device's current status. See section 9.2.
2_{16}	SPECIFIC INQUIRY	Used to check whether a target supports a particular CONTROL command. All operands are included. See section 9.3.
3_{16}	NOTIFY	Used for receiving notification of a change in a device's state. See section 9.4
4_{16}	GENERAL INQUIRY	Used to check whether a target supports a particular CONTROL command. Operands are not included. See section 9.5.
5_{16} - 7_{16}	Reserved for future specification	
8_{16} - F_{16}	Reserved for response codes	

The above command codes appear only in command frames.

7.3.2 Response codes (*response*)

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

Table 7.2 – Response codes

Value	Response	Description
0_{16} - 7_{16}	Reserved for command types	
8_{16}	NOT IMPLEMENTED	The target does not implement the command specified by the opcode and operand marked in the *ck* column, or doesn't implement the specified subunit. See section 8.3.
9_{16}	ACCEPTED	The target executed or is executing the command.
A_{16}	REJECTED	The target implements the command specified by the opcode and operands marked in the *ck* column, but cannot respond because the current state of the target doesn't allow it. Note that some commands may return a REJECTED response as a result of invalid operands not marked in the *ck* column. See section 8.3.
B_{16}	IN TRANSITION	The target implements the STATUS command, but it is in a state of transition. The STATUS command may be retried at a future time.
C_{16}	IMPLEMENTED / STABLE	For SPECIFIC INQUIRY or GENERAL INQUIRY commands, the target implements the command. For STATUS commands, the target returns STABLE and includes the status results.
D_{16}	CHANGED	The response frame contains a notification that the target device's state has changed.
E_{16}	Reserved for future specification	
F_{16}	INTERIM	For CONTROL commands, the target has accepted the request but cannot return information within 100 milliseconds. For NOTIFY commands, the target has accepted the command and will notify the controller of a change of target state at a future time.

7.3.3 Supported response codes

Each command in Table 7.1 can return a subset of the response codes given in Table 7.2. The following table shows the response codes that can be returned for each command type:

Table 7.3 – Supported response codes per command type

		Responses						
		NOT IMPLEMENTED	ACCEPTED	REJECTED	IN TRANSITION	IMPLEMENTED/ STABLE	CHANGED	INTERIM
Commands	CONTROL	√	√	√				√
	STATUS	√		√	√	√ (S[1])		
	SPECIFIC INQUIRY	√				√ (I[2])		
	NOTIFY	√		√			√	√
	GENERAL INQUIRY	√				√ (I[2])		

[1] S: STABLE.

[2] I: IMPLEMENTED.

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

Taken together, the *subunit_type* and *subunit_ID* fields define the command recipient's address within the target. These fields enable the target to determine whether the command is addressed to the target unit, or to a specific subunit within the target. The values in these fields remain unchanged in the response frame.

If either the *subunit_type or subunit*_ID values have been extended, then there will be one or more additional bytes used before the opcode byte.

7.3.4.1 Subunit_type and subunit_ID encoding

Version 1.0 of this specification limited subunit addressing to 32 subunit types and 5 subunits of a given type within a unit (refer to Table 5.2-1 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:

Table 7.4 – Subunit type encoding

Subunit Type	Subunit Name
0	Monitor
1	Audio
2	Printer
3	Disc
4	Tape recorder/player
5	Tuner
6	CA
7	Camera
8	Reserved for future specification
9	Panel
A	Bulletin Board
B	Camera Storage
0C – 1B	Reserved for future specification
1C[1]	Vendor unique
1D	Reserved for all subunit types
1E	subunit_type extended to next byte
1F	Unit

[1] The commands for a Vendor Unique subunit shall be defined by each vendor identified by the company_ID of the UNIT INFO status command.

Table 7.5 – Subunit type encoding for the first extension

Subunit Type	Meaning
01_{16} – FE_{16}	Reserved

Table 7.6 – Subunit ID encoding

Subunit ID	Meaning
0_{16} – 4_{16}	Subunit Instance number
5_{16}	subunit_ID extended to next byte
6_{16}	Reserved for all instances
7_{16}	Ignore (used when addressing units)

7.3.4.2 Addressing units and subunits

1) **Addressing a unit:** An AV/C address with *subunit_type* value $1F_{16}$ and *subunit_ID* value 7 addresses the complete AV/C unit instead of one of its subunits. The combinations of *subunit_type* value $1F_{16}$ and *subunit_ID* values 0 through 6 are reserved.

2) **Addressing a subunit:** If the *subunit_type* value is not equal to $1F_{16}$, the *subunit_ID* indicates the subunit indicated by *subunit_type* numbered sequentially. In this case, the *subunit_ID* starts at zero and is consecutively numbered up to the total number of instances minus one.

The *subunit_type* is extended to support new subunit types. The *subunit_ID* only needs to be extended if more than 5 subunits of a particular subunit type exist in a device, regardless of whether *extended subunit_types* are used or not.

When either *subunit_type* or *subunit_ID* is extended, an entire byte is used. This differs from the normal subunit address byte, in which both the type and ID are specified within a single byte. If *subunit_type* and *subunit_ID* are both extended, the *extended subunit_type* bytes always precede the *extended subunit_ID* bytes. Compare Figure 7.1 to the following diagram of an AV/C command frame with extended type and ID addresses:

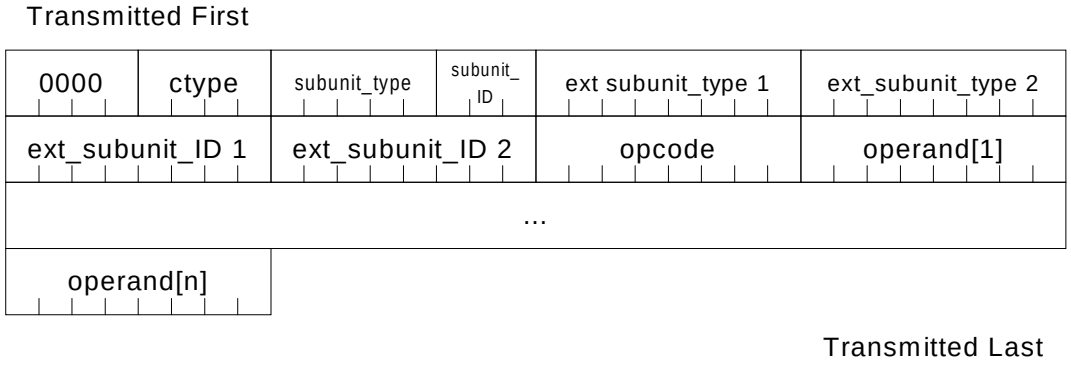

Figure 7.3 – AV/C command frame with two extended type and ID addresses

7.3.4.3 Extended subunit_type and extended subunit_ID encoding

The following tables define the values for *extended subunit_type* and *extended subunit_ID*:

Table 7.7 – Extended subunit_type values

Extended subunit type	Meaning
0	reserved for future specification
1- FE_{16}	extended subunit_type
FF_{16}	extended subunit_type extended to next byte

Table 7.8 – Extended subunit_ID values

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

7.3.4.4 Examples of subunit_type encoding

The following table shows an example of the encoding scheme for representing subunit IDs.

Table 7.9 – Subunit_type encoding examples

Example	Subunit Type Number	Subunit_type	Ext. Subunit_type 1	Ext. Subunit_type 2
1.	04_{16}	00100_2 (4_{16})	Not needed	Not needed
2.	31 (Unit)	11111_2 ($1F_{16}$)	Not needed	Not needed
3.	01_{16} (1^{st} extension)	11110_2 ($1E_{16}$) – extended	00000001_2 (01_{16})	Not needed
4.	01_{16} (2^{nd} extension)	11110_2 ($1E_{16}$) – extended	11111111_2 (FF_{16}) – extended	00000001_2 (01_{16})

In example 1, the *subunit_type* is represented fully by the original *subunit_type* field. The maximum number of *subunit_types* this field can hold is 29.

In example 2, the unit is represented fully by the original *subunit_type* field.

In example 3, specifying the 30^{th} *subunit_type* is represented by 01_{16} in the first *subunit_type* extension field. The maximum *subunit_types* extended fields can hold is 253.

In example 4, specifying the 283^{rd} subunit type (29 + 253 + 1) is represented by 01_{16} in the second *subunit_type* extension field.

In practice, the *subunit_type* and *subunit_ID* may never need to be extended. However, this capability provides for the *possibility* that this may occur:

7.3.4.5 Examples of subunit_ID encoding

The following table shows examples of the setting of *subunit_ID* and *extended subunit_ID* fields for addressing particular subunits.

Table 7.10 – Subunit_ID encoding examples

Example	Subunit_ID number	Subunit_ID	Extended_subunit_ID 1	Extended_subunit_ID 2
1.	4	011_2 (3_{16})	Not needed	Not needed
2.	7	101_2 (5_{16}) - extended	00000010_2 (02_{16})	Not needed
3.	260	101_2 (5_{16}) - extended	11111111_2 (FF_{16}) - extended	00000001_2 (01_{16})

In example 1, to address the fourth subunit, no *extended subunit_ID* fields need to be specified.

In example 2, since the *subunit_ID* field has the capacity to specify five subunits, only one *extended subunit_ID* field is needed to specify the seventh instance. The number to specify in the extended subunit_ID is 02_{16} (7 – 5).

In example 3, the 260^{th} instance requires two extended ID's. The number to specify in the second extended ID is 01_{16} (260 – 5 – 254).

7.3.5 Operation (*opcode*)

Within the five types of AV/C commands, CONTROL, STATUS, SPECIFIC INQUIRY, NOTIFY and GENERAL INQUIRY, the *opcode* field in Figure 7.3 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/C subunits, AV/C units or both, as follows.

Table 7.11 – Opcode values

Value	Addressing mode
00_{16} – $0F_{16}$	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
$E0_{16}$ – FF_{16}	Reserved for future specification

Unit commands are those commands that are addressed to a unit (*subunit_type* = 1F, *subunit_ID* = 7). All AV/C units shall recognize the opcode's value as the same command.

Unit and subunit commands are those commands that can be addressed to a unit or subunit. All AV/C units and subunits shall recognize the opcode's value as the same command.

Subunit commands are those commands that are addressed to a particular subunit_type and are defined by each subunit type specification. With these commands, it is possible for a controller to issue different commands to different subunits using the same opcode value. The controller can distinguish between the different commands since the combination of *subunit_type* and *opcode* indicates uniqueness.

7.3.6 Operands

The number and meaning of the *operand[n]* fields are determined by the *ctype*, *subunit_type* and *opcode* fields, as defined for each opcode.

7.3.7 New subunit_type classification process (informative)

The AV/C command set has been designed to accommodate the creation of new types of products that were not envisioned when the protocol was originally developed. When a manufacturer is designing a new piece of equipment, the following guidelines should be used to determine if the device falls into an existing category (as defined in Table 7.5), or if a new subunit 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 subunit type classification is a two step process:

1) Examine the MAIN functionality of the subunit in terms of the following:
   a) Transport mechanism - does it have one, or does it contain a distinctive or unique mechanism?
   b) 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)?
   c) Signal output - is this subunit a signal source?
   d) Signal processing - accepts input, performs some sort of processing, and then outputs modified data.
   e) 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 split the subunit type into separate audio and video categories.

While many subunits may have 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.

7.3.8 Selecting new subunit_type values (informative)

When selecting a new *subunit_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 7.5).
2) Select only the specified new *subunit_type* from the table; other values 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.

7.3.9 Reserved fields and values

Reserved fields and values within an AV/C frame are reserved for future specification. Reserved fields are bits in a data structure that do not contain any defined or as yet meaningful information. Reserved values are values in defined fields that are not used by this specification.

Except as otherwise indicated (see note below), all devices implementing any version of this specification shall follow the rules regarding reserved fields and reserved values as given below:

1) Controllers shall not set fields to reserved values, and shall set reserved fields to zero before sending an AV/C command frame.
2) Targets shall not set fields to reserved values, and only when a response frame includes reserved fields, it shall set the reserved fields to zero before returning AV/C response frames.

NOTE — In some instances, a specification may define reserved command operands or data structure fields as non-zero values. In these instances, targets shall not set the reserved fields to zero, but shall retain the values that defined in the specification.

7.3.10 Backward compatibility issues

As this specification develops, and as new devices implement new fields and values that were once reserved in this specification, the ability of older devices to operate with newer devices in a backward-compatible manner becomes a necessary design consideration. In order to support backward compatibly, it is important to review how this and other subunit type or unit specifications can and cannot change.

AV/C specifications can change in the following ways:

— Commands and data structures may be added.

— Fields may be appended to the existing data structures in descriptor structures See the *AV/C Descriptor Mechanism Specification Version 1.0* document [R10].

— Reserved fields may be defined.

— Reserved values may be defined.

AV/C specifications cannot change in the following ways:

— Fields should not be appended to existing commands.

— Commands should not be removed (but they can be deprecated.)

— Data structures should not be removed (but they can be deprecated.)

— Existing fields in commands and data structures should not be removed.

— The purpose of existing fields in commands and data structures should not be changed.

If commands are deprecated, they should be indicated as such in the specification.

8. AV/C Operations

8.1 AV/C transaction

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 and zero or more AV/C response frames addressed to the controller. Unless stated otherwise within individual command descriptions, it is assumed that at least one response will be returned.

An AV/C command may be addressed to a specific AV/C unit or it identifies all AV/C units on the bus (broadcast). Unless stated otherwise within individual command descriptions, it is assumed that a single AV/C unit is addressed by the command.

There are two types of AV/C transactions based on how quickly a device can respond to commands – *immediate* transactions and *deferred* transactions.

8.1.1 Immediate transactions

An immediate AV/C transaction is one where the target is able to execute the entire transaction within 100 milliseconds. An example of a simple immediate AV/C transaction is shown below:

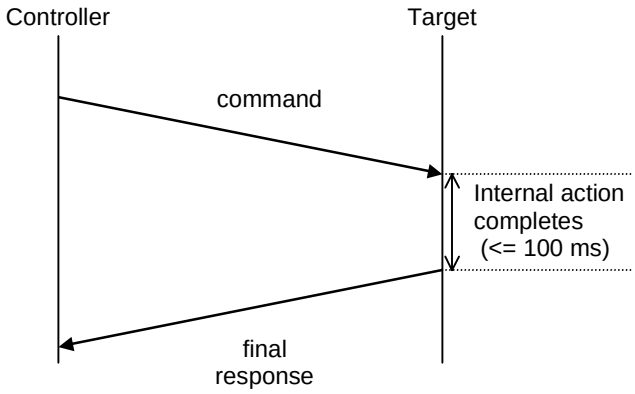

Figure 8.1 – AV/C immediate transaction

In an immediate transaction any response except CHANGED and INTERIM may be returned. This is described in more detail later. The transaction is complete when the target writes the AV/C response frame to the controller.

8.1.2 Deferred transactions

For some CONTROL command 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 INTERIM response with the expectation that a final response (ACCEPTED or REJECTED) will follow later.

When a NOTIFY command transaction is sent, an INTERIM response shall always return before a final CHANGED response.

The figure below illustrates an AV/C deferred transaction.

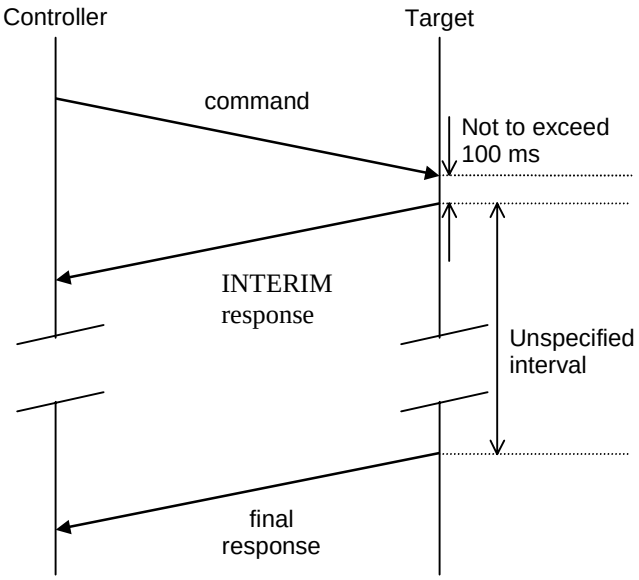

Figure 8.2 – AV/C deferred transaction

The following table shows the combination of the AV/C transaction types and the final responses based on the command type:

Table 8.1 – AV/C transaction types and final responses

ctype	transaction type	final response
CONTROL	immediate	NOT IMPLEMENTED ACCEPTED REJECTED
	deferred	ACCEPTED REJECTED
STATUS	immediate	NOT IMPLEMENTED REJECTED IN TRANSITION STABLE
SPECIFIC INQUIRY	immediate	NOT IMPLEMENTED IMPLEMENTED
NOTIFY	immediate	NOT IMPLEMENTED REJECTED
	deferred	REJECTED CHANGED
GENERAL INQUIRY	immediate	NOT IMPLEMENTED IMPLEMENTED

Note that the STATUS, SPECIFIC INQUIRY and GENERAL INQUIRY commands do not contain deferred transactions. The response code of NOT IMPLEMENTED shall be returned in immediate transactions.

8.2 AV/C transaction rules

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

1) 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. The receipt of the AV/C command is acknowledged by the target via a 1394 WRITE response with resp_complete or a 1394 ack_complete for split or unified transaction respectively.

   NOTE — The controller should assume that it could take longer than 100 milliseconds to receive an AV/C response frame after it confirms the command acknowledge, considering various bus/repeater delays and possible 1394 busy-retry. If the controller does not receive the AV/C response frame within the above period, the controller may retry by resending the same AV/C command frame.

2) For immediate transactions, the target shall return a response code other than INTERIM within 100 milliseconds as a final response. Some combinations of *ctype* and response code require immediate transactions. See Table 8.1 – AV/C transaction types and final responses. The return of any response code other than INTERIM marks the transaction completed and the target is normally ready to accept other AV/C transactions.

3) For deferred transactions, the target shall promptly return an intermediate response code of INTERIM within 100 milliseconds. Subsequent to a first 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, but subunit specifications are recommended to define a time limit for a CONTROL command on command-by-command basis. The target shall ultimately send a final response when the command completes.

4) Before sending a first response to a command, the number of AV/C commands that a target can process is implementation dependent. If the target cannot process an additional command, it should return a 1394 ack_busy, ack_conflict_error, or resp_conflict_error to the additional command.

   NOTE — Some targets that are compliant to the previous version of this specification may ignore the additional command in the above case. The controller may send the additional command after receiving the first response for the previous command. The controller that is compliant to IEEE 1394-1995 handles a 1394 ack_conflict_error as ack missing.

5) In case of deferred transactions, the target should be able to process a STATUS command between an INTERIM response and a final response. If the target cannot process an additional CONTROL or NOTIFY commands during the period, it should first check if the additional command is implemented. If not implemented, the target shall return a NOT IMPLEMENTED response. Otherwise, the target should either return a REJECTED response to the previous command and process the additional command, or return a REJECTED response to the additional commands and continue processing the previous command

6) If the size of an AV/C command frame sent to a target exceeds the maximum size the target can receive, the target should return a 1394 transaction error.

   NOTE — Though the above is recommended, note that some targets may ignore the command, and others may return a NOT IMPLEMENTED response.

7) If the target detects a Serial Bus reset, it shall reset its state to be able to accept AV/C command frames. In this case, any in progress AV/C transactions shall be discarded without the return of a response frame.

8) When the target receives a NOTIFY command, the target should check if 1394 write transaction for an INTERIM response is completed successfully. If it fails, the target may resend another INTERIM response within 100 milliseconds of the receipt of the AV/C command frame, or it may cancel the execution of the NOTIFY command.

NOTE — Without the receipt of an INTERIM response to a NOTIFY command, there is no means for the controller to check if the NOTIFY command has been successfully accepted by the target. On the other hand, the controller may check the execution status of a CONTROL command by the corresponding STATUS command when it does not receive an INTERIM response for the CONTROL command.

8.3 AV/C response rules

A target shall follow the rules to respond to an AV/C command frame.

NOTE — If the cts field in the FCP frame is not zero (not AV/C command frame), the target shall behave according to the command transaction specified by the cts value. If the target does not support the cts value, the target shall ignore the FCP frame and shall not generate any FCP frame.

1) 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.
2) If the target receives an AV/C command frame whose *subunit_type* and *subunit_ID* fields address the command to a nonexistent subunit, the target shall return a NOT IMPLEMENTED response.
3) If the target receives a command frame that includes unsupported combination of *ctype* and *opcode* for the unit or subunit, the target shall return a NOT IMPLEMENTED response.
4) If the size of the AV/C command frame is not compatible with ctype, opcode and operands, the target shall return a NOT IMPLEMENTED response.
5) If the fields marked in the ck column of the command frame include the unsupported value, the target shall return a NOT IMPLEMENTED response. See section 5.2.
6) If the target can not execute the CONTROL, STATUS, or NOTIFY command, and a NOT IMPLEMENTED response would not be required for the command, the target shall return a REJECTED response.
7) If a target receives an AV/C command frame using the broadcasting *node_ID*, and a NOT IMPLEMENTED response would be required for the command, the target shall not generate a response frame.
8) If a CONTROL command requests a target to change to a particular state, and if the target is already in that state, the target shall return an ACCEPTED response.

   NOTE — Note that some unit or subunit specifications may place the exception of this rule.

A response of NOT IMPLEMENTED requires an immediate transaction, and the target shall return it within 100 milliseconds. To determine whether the target shall return a NOT IMPLEMENTED response, the target validates the command frame according to the above rules. Checking whether the operands are supported values is performed only on the fields marked in the *ck* column. See section 5.2 for more information about the *ck* column. It is strongly recommended that all unit and subunit specifications include the *ck* column for each field of their commands.

Note that it is possible to specify fields that within their individual context are valid, but within the context of the command are invalid. When this occurs, a REJECTED or NOT IMPLEMENTED response shall be returned. It is also recommended that the unit or subunit specifications define the rules for the validation within a context.

For example, a multiple date field includes a month and a day. If "2" is specified for the month, and "31" is specified for the day, the combination of values is invalid. Taken within their individual field contexts, however, they are valid. It is up to the unit or subunit type specification to determine how to check for context sensitive information or other error cases that do not appear in the above categories.

8.3.1 Error checking procedure(informative)

There can be a variety of reasons, which prevent an AV/C command from being executed by a target. The following table illustrates the procedure of error checking in AV/C command frames:

Table 8.2 – Error levels

Error Level	Response Frame
1. Check for reserved values of ctype	No response
2. Check for supported combination of subunit_type and subunit_ID	NOT IMPLEMENTED
3. Check for supported combinations of opcode and ctype	NOT IMPLEMENTED
4. Check for compatible frame size	NOT IMPLEMENTED
5. Check fields marked in the *ck* column of the command frame for supported value.	NOT IMPLEMENTED
6. Check all fields of the command frame for execution	REJECTED

8.3.2 Legacy device behavior

It is the responsibility of controllers to issue understandable commands to all compatible targets. A controller, however, could specify commands that are incompatible to legacy devices for various reasons.

The following types of controller errors could occur when specifying *opcode* and *operand[n]* values:

1) A controller could send a command to a legacy target containing a previously reserved field.
2) A controller could send a command to a legacy target containing a previously reserved value.
3) A controller could send a command to a legacy target containing an appended field in a descriptor

The rules below were created to define the behavior of legacy devices when they receive commands using a future version of this document as a baseline. The Table 8.3 below shows rules for receiving fields that were previously reserved.

Table 8.3 – Rules for reserved fields, reserved values and appended fields

Situation	Rule
A new controller sends an AV/C command frame to a legacy target specifying a field that was previously reserved.	The legacy target shall return an AV/C response frame of NOT IMPLEMENTED.
A new controller sends an AV/C command frame to a legacy target specifying a field value that was previously reserved.	The legacy target shall return an AV/C response frame of NOT IMPLEMENTED.
A new controller sends a command with an appended field(s) in a descriptor to a legacy target.	The target shall return an AV/C response frame of REJECTED.

8.4 Matching response frames with command frames

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, depending upon the command type, *subunit_type*, *opcode and operand[n]*.

NOTE — In the case of the TRANSPORT STATE command and the SEARCH MODE command in the Tape Recorder/Player subunit, the *opcode* field is altered in the response frame. In all other commands, the *opcode* field is not altered.

8.5 AV/C with split and unified 1394 transactions (informative)

The AV/C command and response frame are delivered by 1394 asynchronous write transactions. A 1394 write transaction is a split transaction or an unified transaction as illustrated by Figure 8.3 and Figure 8.4, respectively.

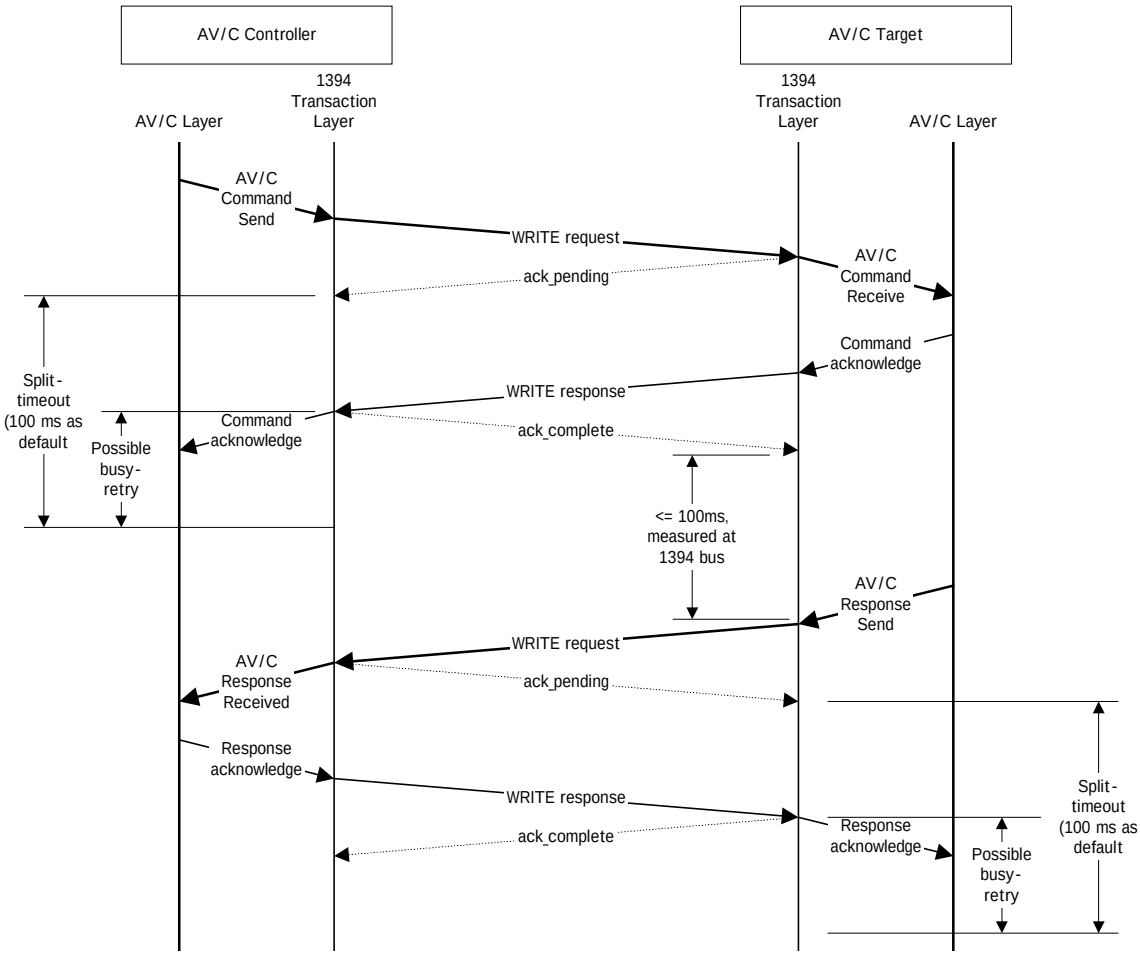

Figure 8.3 – Anatomy of an AV/C transaction with 1394 split transactions

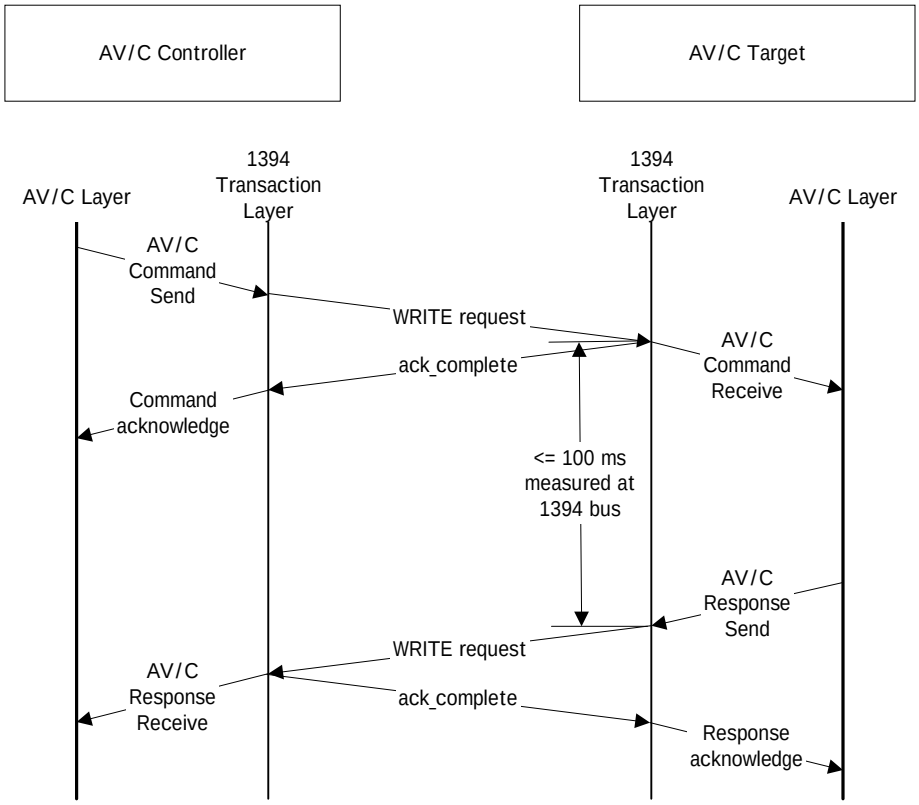

Figure 8.4 – Anatomy of an AV/C transaction with 1394 unified transactions

9. AV/C command types

AV/C commands are variable-length strings of bytes that are embedded within a command frame and addressed to a particular AV/C unit or subunit. A command consists of a command type (*ctype*), a unit or subunit to which the command is addressed (*subunit_type* and *subunit_ID*), an operation code (*opcode*) and zero 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.

9.1 CONTROL commands

A CONTROL command is sent by a controller to another device, the target, to instruct the target to perform an operation. Either the AV/C unit or subunit in the target 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. The following figure shows the command and response transactions for CONROL commands.

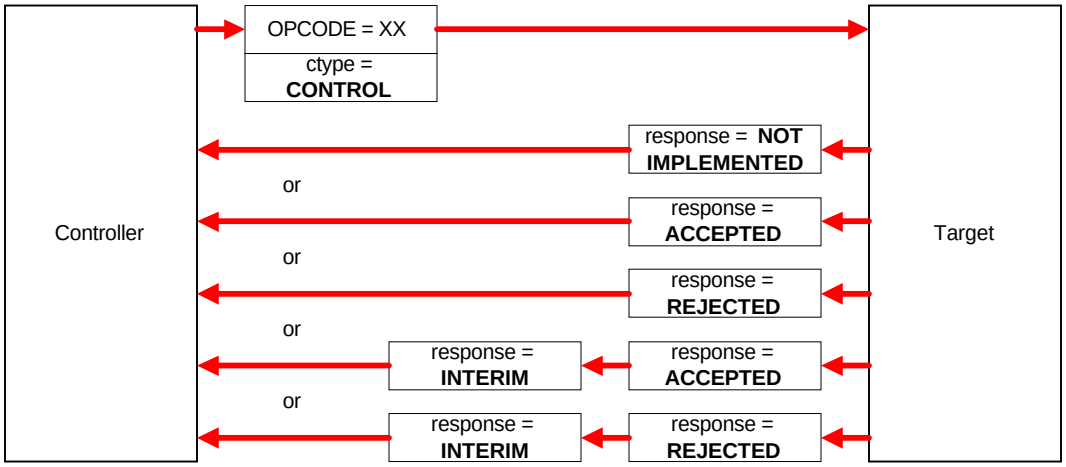

Figure 9.1 – Command and responses for CONTROL commands

Subject to the procedures described in chapter 8 “AV/C Operations”, 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 implement the CONTROL command specified by *opcode* and *operand[n]* marked in the *ck* column or the command is addressed to a subunit not implemented by the target. The target’s state is not modified. See section 8.3 for more information about the conditions under which NOT IMPLEMENTED is returned.

ACCEPTED: The target implements the CONTROL command specified by *opcode* and *operand[n]* marked in the *ck* column 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.

NOTE — If a command requires that the target returns ACCEPTED when the execution completes, a unit or subunit specification should define it.

REJECTED: The target implements the CONTROL command specified by *opcode* and *operands[n]* marked in the *ck* column but the target's present 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's state may be modified as a result of the CONTROL command. Note that some commands may return a REJECTED response as a result of invalid operands.

INTERIM: If the CONTROL command specified by *opcode* and *operand[n]* marked in the *ck* column 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. After the first response of INTERIM, the target shall not send any additional INTERIM responses for this command. 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.

To avoid waiting a for a final response indefinitely, each specification is recommended to define a time limit, a corresponding STATUS command, or another command that cancels the CONTROL command.

9.2 STATUS commands

A STATUS command is sent by a controller to a device to request the device's current status that is within the context of the command. STATUS commands may be sent to either AV/C units or subunits. STATUS commands shall not alter a target's state.

The definition of "target state" varies according to the context of each command. For example, a change in state in the context of a descriptor command could indicate that the descriptor is in an open state or a read-only state. It may not have any affect on other states that may be defined within the unit or subunit.

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

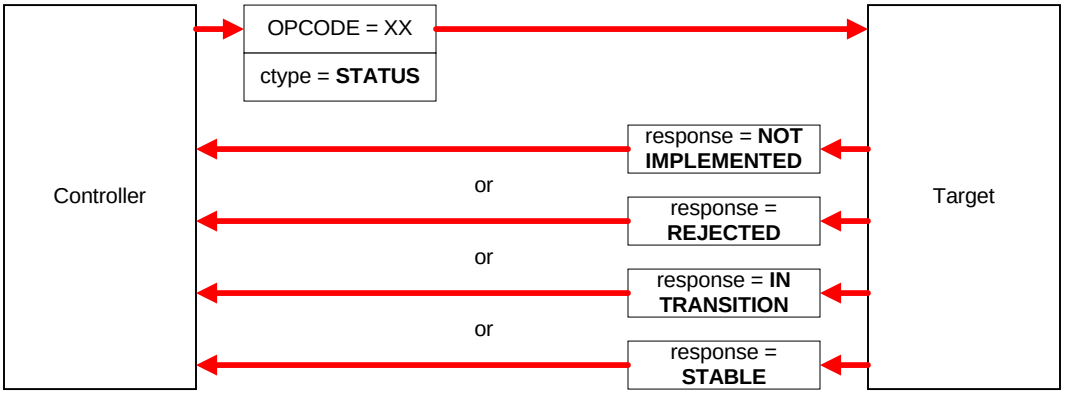

Figure 9.2 – Command and responses for STATUS commands

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 implement the STATUS command specified by *opcode* and *operand[n]* marked in the *ck* column or the command is addressed to a subunit not implemented by the target. See section 8.3 for more information about the conditions under which NOT IMPLEMENTED is returned.

REJECTED: The target implements the STATUS command specified by *opcode* but the target state does not presently permit the return of status for the command. Note that some commands may return a REJECTED response as a result of invalid operands.

IN TRANSITION: The target implements the STATUS commands specified by *opcode* and *operand[n]* marked in the *ck* column 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 a STABLE status.

NOTE — The IN TRANSITION response frame shall include the expected state that the target is transiting into.

STABLE: The target implements the STATUS command specified by *opcode* and *operand[n]* marked in the *ck* column and the information requested is reported in the *opcode* and *operand[n]* 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.

In the NOT IMPLEMENTED and REJECTED 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.

9.3 SPECIFIC INQUIRY commands

A SPECIFIC INQUIRY command may be used by a controller to determine whether or not a target supports the particular CONTROL command. Except for the *ctype* field, the AV/C command frame for a SPECIFIC INQUIRY command is identical to the corresponding control command.

A controller may reliably use SPECIFIC 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 a SPECIFIC INQUIRY command. The following figure shows the command and response mechanism for SPECIFIC INQUIRY commands.

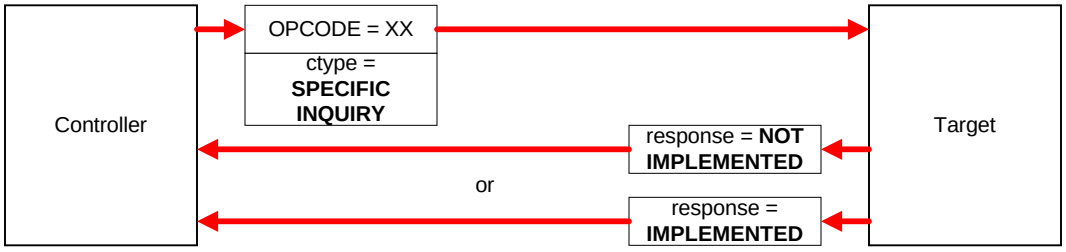

Figure 9.3 – Command and responses for SPECIFIC INQUIRY commands

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]* marked in the *ck* column is implemented by the target AV/C device.

NOTE — If an IMPLEMENTED response returns from a SPECIFIC INQUIRY command, its corresponding CONTROL command can return ACCEPTED or REJECTED responses. If a NOT IMPLEMENTED response returns from a SPECIFIC INQUIRY command, its corresponding CONTROL command shall return a NOT IMPLEMENTED response. See section 8.3 for more information about the conditions under which NOT IMPLEMENTED is returned.

An AV/C 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 STATUS 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, the SPECIFIC INQUIRY command type does not have a support level since they return information about the support level of the corresponding CONTROL command. However, the ability of an AV/C device to provide a response to a SPECIFIC INQUIRY command for any *opcode* is mandatory. This insures that a controller shall always receive a response to a support level SPECIFIC INQUIRY command.

The broadcasting *node_ID* shall not be used for SPECIFIC INQUIRY commands.

9.4 NOTIFY commands

A controller that desires to receive notification of future changes in a device's state may use a 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. The following figure shows the command and response mechanism for NOTIFY commands:

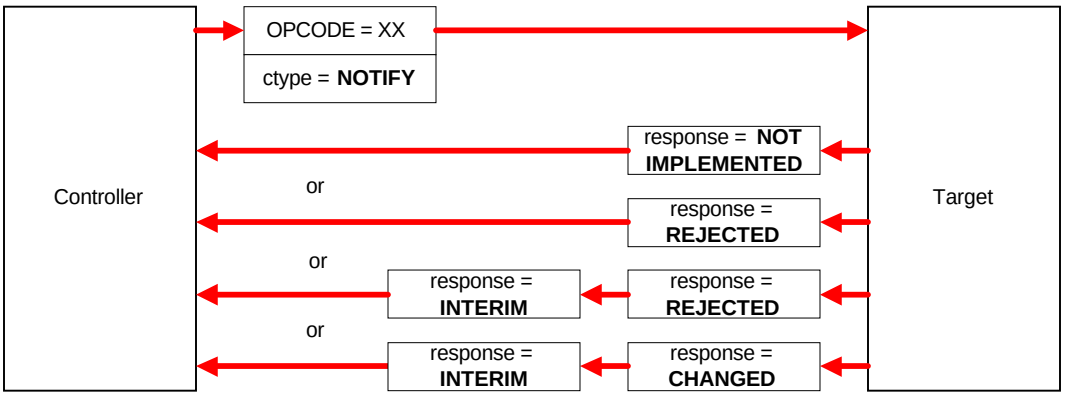

Figure 9.4 – Command and responses for NOTIFY commands

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

NOT IMPLEMENTED: The target does not implement the NOTIFY command specified by *opcode* and *operand[n]* marked in the *ck* column or the command is addressed to a subunit not implemented by the target. See section 8.3 for more information about the conditions under which NOT IMPLEMENTED is returned.

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 *opcode* and *operand[n]* data returned in the response frame. At 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 the target state that is within the context of the command opcode. The future change of the 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.

REJECTED: The target implements event notification for the condition specified by *opcode* and *operands[n]* marked in the *ck* column but is not able to presently supply the requested information. Note that some commands may return a REJECTED response as a result of invalid operands.

A target might be capable of processing only one NOTIFY command at a time. To handle this situation, a first priority and last priority handling is defined:

First Priority: In this case, if one controller has requested a NOTIFY to a target, and a second controller requests the same NOTIFY to the target, the second controller will receive a REJECTED response.

Last Priority: In this case, if another controller issues the same NOTIFY command before the CHANGED response is returned for the first controller, the first controller's command will be superseded, and a REJECTED response will be returned to the first controller.

When a controller receives a REJECTED response after an INTERIM response to a NOTIFY command, the controller should revert to status polling to determine any status changes on the target. This prevents the possible race condition when two controllers are asking for the same status information. Controllers shall carefully choose polling intervals such that extraneous bandwidth is not used on the bus, yet status updates are received in a timely manner.

CHANGED: The target supports the event notification specified by *opcode* and *operand[n]* marked in the *ck* column 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 first response to a TRANSPORT STATE notify command is an INTERIM response 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.

If the status is changed from STABLE to IN TRANSITION, a CHANGED response of a NOTIFY command shall be returned as illustrated by Figure 9.5 below.

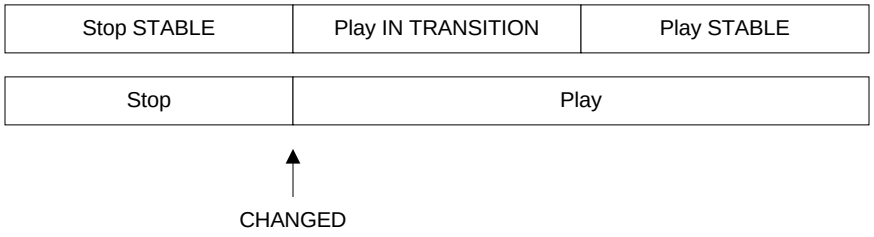

Figure 9.5 – Timing of the CHANGED response

9.4.1 Guidelines for NOTIFY handling

To support *first priority* and *last priority* situations, the target must store the controller's node_ID in a NOTIFY buffer. This specification allows both implementations. Other unit or subunit specifications or guidelines may select one.

1) Guideline for a controller

— When a controller receives a REJECTED response to a NOTIFY command, the controller is strongly recommended to start polling using STATUS commands until the state changes, or until the controller decides that it is no longer interested in that information. A controller may send a NOTIFY command again after a REJECTED response to the previous NOTIFY command after an appropriate time interval (e.g. 5 sec) to avoid a race condition when more than one controller is asking for the same information. In this case, a first priority target will reject the new NOTIFY command again. On the other hand, a last priority target will accept the new NOTIFY command while rejecting the previous NOTIFY command.

— If a controller has set a NOTIFY command and not received any response within 100 milliseconds, it may ignore any further INTERIM or CHANGED responses for that NOTIFY command.

2) Guideline for a target

— If a controller sends multiple NOTIFY commands asking for the same information to the same target, then the target need only store one instance of the NOTIFY command. This guideline prevents all of the plural NOTIFY queues being occupied with the same command from the same controller. The target need only store one instance of that NOTIFY command by the rule above, especially in the case of a first priority target.

9.5 GENERAL INQUIRY commands

A GENERAL INQUIRY command may be used by a controller to determine whether or not a target supports the particular CONTROL command *without* being required to specify a particular set of parameters for that command. The format of the GENERAL INQUIRY command frame shall consist of only the *opcode* of the command that 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. The following figure shows the command and response mechanism for GENERAL INQUIRY commands:

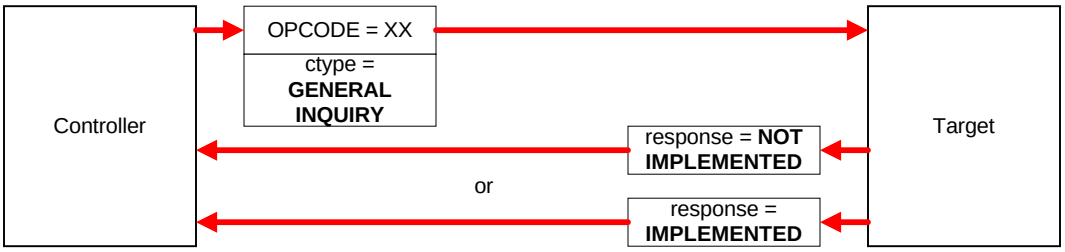

Figure 9.6 – Command and responses for GENERAL INQUIRY commands

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 on. A response of IMPLEMENTED specifies that at least one of the corresponding CONTROL command variations specified by *opcode* is implemented by the target AV/C 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 an IMPLEMENTED response for the BACKWARD general inquiry command.

Unlike the other command types, the GENERAL INQUIRY command type does not have a support level since they return information about the support level of the corresponding CONTROL command. However, the ability of an AV/C device to provide a response to a GENERAL INQUIRY command for any

opcode is mandatory. This insures that a controller shall always know the support level of the CONTROL 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 that do not respond to this command type. A controller that does not receive a response may try the SPECIFIC INQUIRY command as a fallback measure.

The broadcasting *node_ID* shall not be used for GENERAL INQUIRY commands.

9.6 Support levels

9.6.1 Command support levels

Each AV/C unit or subunit may implement a subset of the AV/C command set. An unsupported command shall be returned 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 all audio/video devices that claim compliance with the governing specification and that implement the AV/C unit or subunit type(s) for which the command is defined. AV/C compliant devices are identified by their configuration ROM entries.

Recommended: For an AV/C compliant device, the command is optional but it represents a basic functionality, *e.g.*, video and audio insert modes for a VCR subunit's RECORD command. If the device has unit or subunit type(s) with functionality that corresponds to the command, it is recommended that the command be implemented.

Optional: The command is optional for an AV/C compliant device.

Vendor-dependent: The device vendor defines support for and interpretation of the command.

10. AV/C model

10.1 AV/C unit model

An AV/C unit is an instantiation of a logical entity that represents an electronic device on a 1394 node. An AV/C unit has a set of coherent functions that are common to other AV/C units.

The AV/C unit model consists of AV/C subunits, input plugs, and output plugs, as illustrated in the following figure:

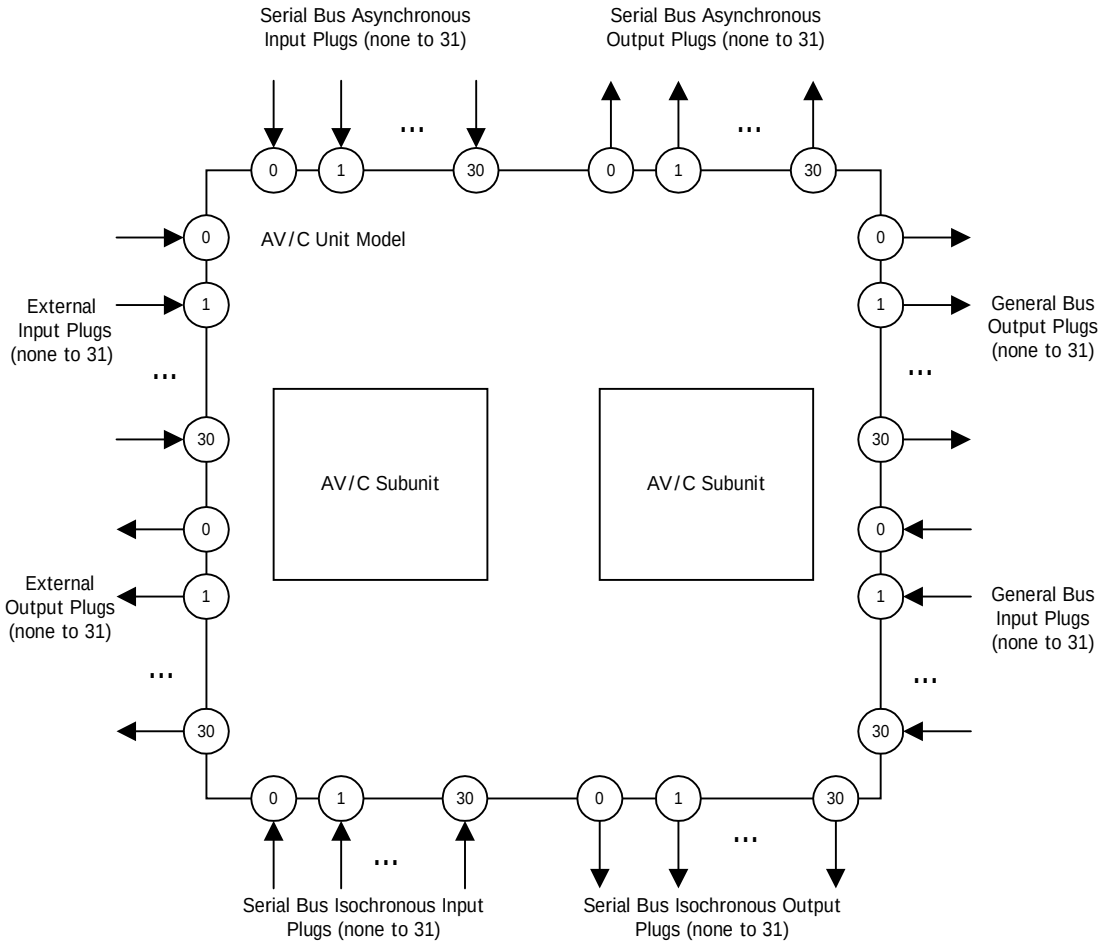

Figure 10.1 – AV/C unit model

An AV/C unit does not contain another AV/C unit.

10.1.1 Unit plugs

A unit plug is a virtual entity which represents the functions to input or output data. Unit plugs are divided into two kinds of plugs, one for input and one for output. Unit plugs consist of “Serial Bus Isochronous Plugs”, “Serial Bus Asynchronous Plugs”, “General Bus plugs”,

and “External Plugs”.

10.1.1.1 Serial bus isochronous plugs

Serial Bus Isochronous Plugs are virtual connection points on the AV/C unit that, together with the Plug Control Registers (iPCRs and oPCRs), are used to transmit 1394 isochronous data flows through the unit and to control their attributes, as described in reference [R7] and the related specifications. There is a one-to-one relationship between the Serial Bus Isochronous Plugs and the Plug Control Registers (iPCRs and oPCRs). The iMPRs and oMPRs are master registers that pertain to all input and all output plugs.

A Serial Bus Isochronous Input Plug inputs one isochronous stream from the serial bus interface into the AV/C unit. A Serial Bus Isochronous Output Plug outputs one isochronous stream from the AV/C unit to the serial bus interface. An AV/C unit can have from 0 to 31 Serial Bus Isochronous Input Plugs. An AV/C unit can have from 0 to 31 Serial Bus Isochronous Output Plugs. An AV/C unit does not have to implement any Serial Bus Isochronous Plug if it does not send or receive any isochronous stream.

10.1.1.2 Serial bus asynchronous plugs

Serial Bus Asynchronous Plugs are virtual connection points on the AV/C unit that are used to transmit and receive large 1394 asynchronous data flows that do not require real-time delivery guarantees and are described in reference [R13].

A Serial Bus Asynchronous Input Plug inputs one asynchronous data flow from the serial bus interface into the AV/C unit. A Serial Bus Asynchronous Output Plug outputs one asynchronous data flow from AV/C unit to the serial bus interface. An AV/C unit can have from 0 to 31 Serial Bus Asynchronous Input Plugs. An AV/C unit can have from 0 to 31 Serial Bus Asynchronous Output Plugs. An AV/C unit does not have to implement any Serial Bus Asynchronous Plug if it does not support Asynchronous Connections.

10.1.1.3 General bus plugs

General Bus Plugs are virtual connection points on the AV/C unit that are used to transmit and receive data flows through various types of bus other than 1394 Serial Bus. Actual connection for the data flow between devices is made by a certain method defined by the bus specification. The GENERAL BUS SETUP command defined in this specification is used to setup or associate General Bus Plugs with the connections. How this command is used and the format of the bus dependent field are defined by each bus specification.

A General Bus Input Plug inputs one stream from a type of bus listed in this specification into the AV/C unit. A General Bus Output Plug outputs one stream from AV/C unit to a type of bus listed in this specification. On the General Bus Plugs, the streams conform to a signal format that is defined by each bus specification. An AV/C unit can have from 0 to 31 General Bus Input Plugs. An AV/C unit can have from 0 to 31 General Bus Output Plugs. An AV/C unit does not have to implement any General Bus Plug if it does not send or receive any stream through General Bus Plug.

10.1.1.4 External plugs

An External Plug inputs or outputs one stream from an external interface into the AV/C unit. External Plugs are plugs that transmit on media other than 1394 and other types of bus. On External Plugs, the streams may be either digital or analog. An AV/C unit can have from 0 to 31 External Input Plugs. An AV/C unit can have from 0 to 31 External Output Plugs. External Plugs do not contain associated plug control registers. An AV/C unit does not have to implement any External Plug if it does not send or receive any external signals. The details of this type of plug are not defined in this specification.

10.1.2 AV/C unit plug addresses

The AV/C unit plug addresses are defined in the table below:

Table 10.1 – AV/C unit plug addresses

Value	Unit Input Plug	Unit Output Plug
00_{16} – $1E_{16}$	Serial Bus Isochronous Input Plug 0 – 30 (iPCR 0 – 30)	Serial Bus Isochronous Output Plug 0 – 30 (oPCR 0 – 30)
$1F_{16}$ – $3F_{16}$	Reserved	Reserved
40_{16} – $5E_{16}$	General Bus Input Plug 0 – 30	General Bus Output Plug 0 – 30
$5F_{16}$ – $7E_{16}$	Reserved	Reserved
$7F_{16}$	Any available Serial Bus Isochronous Input Plug	Any available Serial Bus Isochronous Output Plug
80_{16} – $9E_{16}$	External Input Plug 0 – 30	External Output Plug 0 – 30
$9F_{16}$	Reserved	Reserved
$A0_{16}$ – BE_{16}	Serial Bus Asynchronous Input Plug 0 – 30	Serial Bus Asynchronous Output Plug 0 – 30
BF_{16}	Any available Serial Bus Asynchronous Input Plug	Any available Serial Bus Asynchronous Output Plug
$C0_{16}$ – FC_{16}	Reserved	Reserved
FD_{16}	Reserved	Multiple Plugs
FE_{16}	Invalid	Invalid
FF_{16}	Any available External Input Plug	Any available External Output Plug

When the “any available” plug is used in a command, the controller expects that the target will specify a plug. The "invalid" value is used in STATUS and NOTIFY commands, and in their response frame when no specific plug is returned. The "multiple plugs" value is returned in the response frame of STATUS and NOTIFY commands if one plug is not uniquely specified. For example, a unit input plug or subunit source plug is connected to multiple plugs, either unit output plugs, subunit destination plugs, or both. When the "multiple plugs" value is returned, the destination_subunit_type and destination_subunit_ID fields of the response frame have no meaning.

10.2 AV/C subunit model

An AV/C subunit is an instantiation of a logical entity that can be identified within an AV/C unit. An AV/C subunit has a set of coherent functions that the electronic device provides. Functions are defined for each category of devices in its subunit specification.

An AV/C subunit model consists of source plugs, and destination plugs, and sometimes function blocks as illustrated in the figure below:

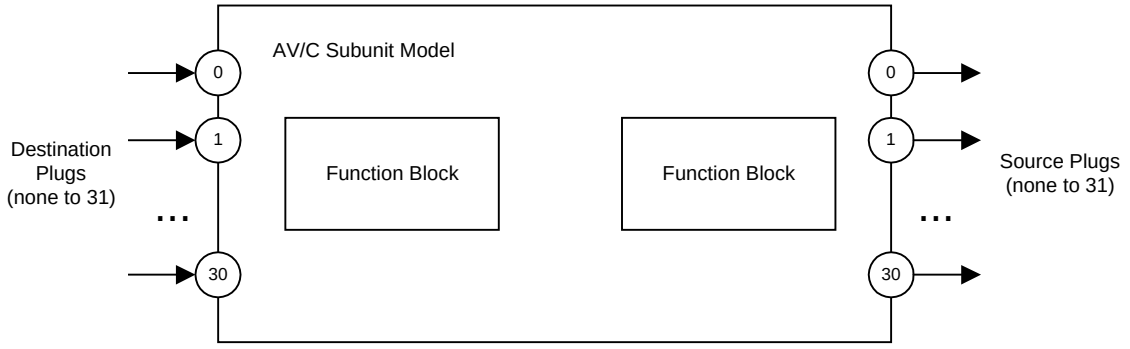

Figure 10.2 – AV/C subunit model

An AV/C subunit does not contain another AV/C subunit.

10.2.1 Function blocks

Function Blocks within an AV/C subunit process the input streams and may send the processed streams to the source plugs. The streams may also be stored internally within the subunit. The processes or functions performed by these Function Blocks are subunit implementation dependent and are specified in reference [R16].

10.2.2 Subunit plugs

A subunit plug is a virtual entity, which represents the functions for inputting or outputting one stream to a unit plug or another subunit plug. Subunit plugs are divided into two types, one is the destination plug, which is used for inputting the data to a subunit, and the other is the source plug, which is used for outputting the data from a subunit.

An AV/C subunit can have from 0 to 31 source plugs. An AV/C subunit can have from 0 to 31 destination plugs. An AV/C subunit does not have to implement any subunit plug if it does not send or receive any signals.

10.2.3 AV/C subunit plug address

The AV/C subunit plug address is defined in the table below:

Table 10.2 – AV/C subunit plug address

Value	Source Plug	Destination Plug
00_{16} – $1E_{16}$	Source Plug 0 – 30	Destination Plug 0 - 30
$1F_{16}$ – FC_{16}	Reserved	Reserved
FD_{16}	Reserved	Multiple Plugs
FE_{16}	Invalid	Invalid
FF_{16}	Any available source plug	Any available destination plug

When the “any available” plug is used in a command, the controller expects that the target will specify a plug. The "invalid" value is used in STATUS and NOTIFY commands, and in their response frame when no specific plug is returned. The "multiple plugs" value is returned in the response frame of STATUS and NOTIFY commands if one plug is not uniquely specified. For example, a unit input plug or subunit source

plug is connected to multiple plugs, either unit output plugs, subunit destination plugs, or both. When the "multiple plugs" value is returned, the destination_subunit_type and destination_subunit_ID fields of the response frame have no meaning.

10.3 Internal connections

This section describes the connectivity of AV/C unit plugs and AV/C subunit plugs. A connection may be permanent, meaning it is hardwired and cannot be changed, or locked, meaning the flow of the data stream cannot be disrupted. The AV/C unit is solely responsible to make, change, or break connections.

10.3.1 Rules for connecting internal plugs

It is possible to connect the following plug pairs:

1) Unit input plug –> Unit output plug
2) Unit input plug –> Subunit destination plug
3) Subunit source plug –> Unit output plug
4) Subunit source plug –> Subunit destination plug

Other connection rules:

5) A unit input plug or subunit source plug can connect to multiple plugs simultaneously.
6) A unit output plug and a subunit destination plug can have only one connection.

All other types of connections are invalid.

NOTE — The CONNECT AV is an exception to rule 6 above, and can allow multiple connections to a subunit destination plug or a unit output plug.

10.3.2 Connection commands

Commands for establishing and managing connections between unit, subunit, and function block plugs can be found in the following section and in references [R12] [R15] and [R16].

11. General commands

General commands are used for controlling or querying a unit and/or subunit, and are expected to be used broadly on all types of AV/C devices. Table 11.1 below summarizes the AV/C General commands.

Table 11.1 – General commands

Opcode	Value	Support level[1] (by *ctype*)			Target	Comments
		C	S	N		
POWER	$B2_{16}$	O	O	R	Unit or subunit	Control power state
UNIT INFO	30_{16}	–	M	–	Unit only	Report unit information
SUBUNIT INFO	31_{16}	–	M	–	Unit only	Report subunit information
RESERVE	01_{16}	O	O	R	Unit or subunit	Acquire or release exclusive control of a target
VERSION	$B0_{16}$	–	O	–	Unit or subunit	Get information about the version of an AV/C unit or subunit
VENDOR-DEPENDENT	00_{16}	V	V	V	Unit or subunit	Vendor-dependent commands

[1] C: CONTROL, S: STATUS, N: NOTIFY, M: Mandatory, R: Recommended, O: Optional, V: Vendor dependent, –: Not Defined

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.

11.1 POWER command

11.1.1 POWER control command

The POWER command is used to control or determine the power status of an AV/C unit or one of its subunits specified by the AV/C address that is contained in the AV/C frame.

NOTE – The states of the PHY and LINK layers are not affected by this control command. However, the POWER control command to power off may affect the states of the PHY and LINK layers for energy conservation.

Setting the power status of the AV/C 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 to on or off may, according to product design, cause the power of the AV/C unit and/or other subunit(s) in the AV/C unit to be set in the same way.

NOTE – Some subunits, such as Bulletin board subunit and Panel subunit, may not have the power state and work regardless of the power state of the unit or other subunits.

The format of the POWER control command frame is shown by Figure 11.1 below:

	length	ck	msb							lsb
opcode	1	√	POWER ($B2_{16}$)							
operand[0]	1	√	power_state							

Figure 11.1 – POWER control command frame

11.1.1.1 Field definitions

power_state: the *power_state* field specifies the desired power state of the unit. Power on is encoded as 70_{16} and power off as 60_{16}.

11.1.1.2 POWER control command responses

All response frames of POWER control command shall have the same format as the command frame.

11.1.1.3 POWER control and response field values

The following table shows the field values in the POWER control command and response frames:

Table 11.2 – Field values in the POWER control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
power_state	60_{16} or 70_{16}	←	←	←

← means “same as the command frame”.

11.1.2 POWER status command

The POWER status command may be used to determine the current power state of the AV/C unit or one of its subunits. The POWER status command shall have the same format as the corresponding control command. In this case, *operand[0]* is set to $7F_{16}$ when the command is issued and is updated to the current power state when the STABLE response is returned.

The format of POWER status command frame is shown by Figure 11.2 below:

	length	ck	msb							lsb
opcode	1	√	POWER ($B2_{16}$)							
operand[0]	1	√	power_state = $7F_{16}$							

Figure 11.2 – POWERS status command frame

11.1.2.1 POWER status responses

All response frames of POWER status command shall have the same format as the command frame.

11.1.2.2 POWER status command and response field values

The following table shows the field values in the POWER status command and response frames:

Table 11.3 – Field values in the POWER status command: REJECTED, IN TRANSITION and STABLE response frames

Fields	Command	Response		
		REJECTED	IN TRANSITION	STABLE
power_state	$7F_{16}$	←	60_{16} or 70_{16}	60_{16} or 70_{16}

← means “same as the command frame”.

11.1.3 POWER notify command

The NOTIFY command type can also be used with the POWER command. The POWER notify command has the same format as the POWER 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 format of the POWER notify command frame is shown by Figure 11.3 below.

	length	ck	msb							lsb
opcode	1	√	POWER ($B2_{16}$)							
operand[0]	1	√	power_state = $7F_{16}$							

Figure 11.3 – POWER notify command frame

11.1.3.1 POWER notify command responses

All responses of POWER notify command frames have the same format as the command frame.

11.1.3.2 POWER notify command and response field values

The following table shows the field values in the POWER notify command and response frames:

Table 11.4 – Field values in the POWER notify command: REJECTED, INTERIM and CHANGED response frames

Fields	Command	Response		
		REJECTED	INTERIM	CHANGED
power_state	$7F_{16}$	←	60_{16} or 70_{16}	60_{16} or 70_{16}

11.2 UNIT INFO command

The UNIT INFO command is used to obtain information that pertains to the AV/C unit as a whole (distinct from subunit information, see section 11.3).

11.2.1 UNIT INFO status command

For the UNIT INFO command, only the STATUS command type is supported. The format of the UNIT INFO status command frame is shown by Figure 11.4 below:

	length	ck	msb							lsb
opcode	1	√	UNIT INFO (30_{16})							
operand[0]	5	√	all FF_{16}							
…										
operand[4]										

Figure 11.4 – UNIT INFO status command frame

11.2.1.1 UNIT INFO status command responses

The format of response is shown by Figure 11.5 below:

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

Figure 11.5 – UNIT INFO status command response format

11.2.1.1.1 Field definitions

unit_type: The *unit_type* field contains a value whose meaning is identical to those defined for *subunit_type*. The *unit_type* shall be the subunit type that best describes the unit. For example, the VCR device may return the tape recorder/player *unit_type*, even though the VCR has a tuner. Value $1C_{16}$ (vendor unique) should be returned in case none of the other values are considered to be appropriate. The *unit_type* field may take value $1E_{16}$, 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].

Further extension is possible when the value of *extended_unit_type* is FF_{16}, in which case another byte will be added.

unit: The definition of the unit field is vendor dependent.

company_ID: 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 Root Directory in the AV/C 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].

11.2.1.2 UNIT INFO status command and response field values

The following table shows the field values in the UNIT INFO status command and response frames:

Table 11.5 – Field values in the UNIT INFO status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
unit_type	$1F_{16}$	←	$1C_{16}$, or most appropriate subunit_type See Table 7.4.
unit	7_{16}	←	vendor-dependent
extended_unit_type	Not supplied in command frame	←	extended_subunit_ type as necessary. See Table 7.4.
company_ID	all FF_{16}	←	company_ID

← means “same as the command frame”.

NOTE — The IN TRANSITION response frame does not apply to the UNIT INFO status command.

11.3 SUBUNIT INFO command

The SUBUNIT INFO command is used to obtain information about the subunit(s) of an AV/C unit.

11.3.1 SUBUNIT INFO status command

The format of the SUBUNIT INFO status command frame is shown by Figure 11.6 below:

	length	ck	msb							lsb
opcode	1	√	SUBUNIT INFO (31_{16})							
operand[0]	1	√	0	page			0	extenstion_code		
operand[1]	4	√	page_data = all FF_{16}							
:										
operand[4]										

Figure 11.6 – SUBUNIT INFO status command frame

11.3.1.1 Field definitions

page: The *page* field value specifies which part of the subunit table is to be returned. Each page consists of at most four subunits, and each AV/C unit contains up to 32 AV/C subunits.

If the specified page contains no data, then the target should return a NOT IMPLEMENTED response.

extension_code: The *extension_code* field may be used in a future revision of this specification. It shall presently have a value of 7.

page_data: The *page_data* field shall be all FF_{16} in the command frame.

11.3.1.2 SUBUNIT INFO status command responses

The format of the response frame is shown by Figure 11.7 below:

	length	msb							lsb
opcode	1	SUBUNIT INFO (31_{16})							
operand[0]	1	0	page			0	extension_code		
operand[1]	n	page_data							
:									
operand[n]									

Figure 11.7 – SUBUNIT INFO response format

page_data: The *page_data* returned is the four entries from the subunit table for the page requested. The subunit table is an array of entries; each entry has the format defined by Figure 11.8 below:

offset	length	msb							lsb
00_{16}	1	subunit_type					max_subunit_ID		
:	1	extended_subunit_type 1							
:		…							
:	1	extended_subunit_type n							
:	1	extended_max_subunit_ID 1							
:		…							
:	1	extended_max_subunit_ID n							

Figure 11.8 – Subunit page table entry

In the above table entry, the *extended_subunit_type* and *extended_max_subunit_ID* fields may not exist based on the subunit type specified and its maximum number.

subunit_type: The *subunit_type* field of each entry is as defined in the subunit_type table.

max_subunit_ID: The *max_subunit_ID* field is the count of subunits of *subunit_type* implemented by the AV/C unit, less one. For example, if there are 5 subunits of the *subunit_type*, this field has the value 4.

extended_subunit_type 1 – n: The extended_subunit_type 1 – n shall be specified if the *subunit_type* is an extended value. In case the *subunit_type* is extended more than once, they shall be placed together in the response frame.

extended_subunit_ID 1 – n: The *extended_subunit_ID 1 – n* shall be specified if the *subunit_ID* is extended. In case the *subunit_ID* is extended more than once, they shall be placed together in the response frame.

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 shall be appended, one after the other, from the first *page_data*. Then, if the last page contains less than four entries, the field(s) shall be set to FF_{16}(s).

11.3.1.3 SUBUNIT INFO status command and response field values

The following table shows the field values in the SUBUNIT INFO status command and response frames:

Table 11.6 – Field values in the SUBUNIT INFO status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
page	0 - 7	←	←
page_data	all FF_{16}	←	<= four page table entries. See Figure 11.8.

← means “same as the command frame”.

NOTE — The IN TRANSITION response frame does not apply to the SUBUNIT INFO status command.

11.4 RESERVE command

11.4.1 RESERVE control command

The RESERVE control command permits a controller to acquire or release exclusive control of the AV/C unit or one of its subunits determined by the AV/C address that is contained in the AV/C frame. The format of RESERVE control command frame is shown by Figure 11.9 below:

	length	ck	msb							lsb
opcode	1	√	RESERVE (01_{16})							
operand[0]	1	√	priority							
operand[1]	12	–	Text							
:										
operand[12]										

Figure 11.9 – RESERVE control command frame

11.4.1.1 Field definitions

priority: 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/C (sub)unit. The other values, between one and $0F_{16}$, 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.

text: 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 FF_{16}.

11.4.1.2 Rules for using the RESERVE control command

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

1) After a power-on reset, the AV/C (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 status commands, below).

2) An AV/C (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/C (sub)unit and a reservation is established, the stored value is transformed according to the following table:

Table 11.7 – Priority codes

Command priority	Stored priority
00_{16} — 01_{16}	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.

3) While a controller holds the reservation of an AV/C (sub)unit, the target shall reject any CONTROL commands issued by any other controller that interfere with the control by the holder of the reservation other than a RESERVE control command . The (sub)unit specifications may define special handling rules of a CONTROL command appropriate for the application of the command. The 16-bit node ID stored by the AV/C (sub)unit upon receipt of the RESERVE control command is the basis for accepting or rejecting CONTROL commands for controllers.

4) 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. A zero value of the priority can be sent to release the reservation. It should be the responsibility of the controller to always release a target when it is finished with its reservation.

5) If a RESERVE control command is received from a different controller other than that which made the reservation, the AV/C (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).

6) If a RESERVE control command is addressed to the AV/C unit but that AV/C unit contains a subunit that already holds a reservation with an equal or higher priority, the RESERVE control command shall return a REJECTED response.

7) If a RESERVE control command is addressed to the AV/C unit and that AV/C 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/C unit is established for the new controller according to the procedures already described in b).

8) Any CONTROL command that is addressed to a subunit within an AV/C unit that is reserved by a different controller other than the one that issued the CONTROL command shall be rejected.

When an AV/C (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 CONTROL commands 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 have established a reservation with the target AV/C (sub)unit prior to the bus reset. Because the node ID of the AV/C 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/C (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.

11.4.1.3 RESERVE control command responses

All response frames of RESERVE control command shall have the same format as the command frame.

11.4.1.4 RESERVE control and response field values

The following table shows the field values in the RESERVE control command and response frames:

Table 11.8 – Field values in the RESERVE control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
priority	00_{16} – $0F_{16}$	←	←	←
text	variable	←	←	←

← means "same as the command frame".

11.4.2 RESERVE status command

Any controller may request the current reservation status of an AV/C (sub)unit by issuing a RESERVE status command. The format of RESERVE status command frame is shown in Figure 11.10 below.

	length	ck	msb							lsb
opcode	1	√	RESERVE (01_{16})							
operand[0]	13	√	all FF_{16}							
:										
operand[12]										

Figure 11.10 – RESERVE status command frame

11.4.2.1 RESERVE status command responses

All response frames of RESERVE status command shall have the same format as the command frame.

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.

11.4.2.2 RESERVE status command response field values

The following table shows the field values in the RESERVE control command and response frames:

Table 11.9 – Field values in the RESERVE status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
priority	FF_{16}	←	0 to $0F_{16}$
text	all FF_{16}	←	text

← means "same as the command frame".

NOTE — The IN TRANSITION response frame does not apply to the RESERVE status command.

11.4.3 RESERVE notify command

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 11.10 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.

11.4.3.1 RESERVE notify command responses

All response frames of RESERVE notify command shall have the same format as the command frame.

11.4.3.2 RESERVE notify command and response field values

The following table shows the field values in the RESERVE notify command and response frames:

Table 11.10 – Field values in the RESERVE notify command: REJECTED, INTERIM and CHANGED response frames

Fields	Command	Response		
		REJECTED	INTERIM	CHANGED
priority	FF_{16}	←	0 to $0F_{16}$	0 to $0F_{16}$
text	all FF_{16}	←	text	text

← means “same as the command frame”.

11.5 VERSION command

11.5.1 VERSION status command

The VERSION status command is used to get the version of the specification to which an AV/C unit or subunit complies. The VERSION status command is also used to get the capabilities and other related information of AV/C unit or subunit. The format of VERSION status command frame is shown by Figure 11.11 below.

	lengt	ck	msb							lsb
opcode	1	√	VERSION ($B0_{16}$)							
operand[0]	1	√	subfunction							
operand[1]	32	– [1]	subfunction_dependent_field							
:										
operand[32]										

[1] subfunction dependent.

Figure 11.11 – VERSION status command frame

11.5.1.1 Field definitions

subfunction: The subfunction operand specifies the way to inquire about the version information.

Table 11.11 – subfunction operand meaning

subufunction	Action
00_{16} – FE_{16}	reserved for future definition
FF_{16}	Get the latest version or the support level of the specified version

The format of VERSION STATUS command frame when subfunction FF_{16} is shown in Figure 11.12 – VERSION status command frame when subfunction = FF_{16}

subfunction_dependent_field: The definition of this field depends on each subfunction.

11.5.1.2 VERSION status command response

All response frames of VERSION status command have the same format as the command frame.

11.5.1.3 VERSION status command when subfunction = FF_{16}

This subfunction can be used to get the latest specification version information to which the unit or subunit complies, or to inquire whether the specific version number is supported or not. The format of the VERSION status command when subfunction = FF_{16} is shown by Figure 11.12 – VERSION status command frame when subfunction = FF_{16} below.

	length	ck	msb							lsb
opcode	1	√	VERSION ($B0_{16}$)							
operand[0]	1	√	subfunction =FF_{16}							
operand[1]	1	√	version_information							
operand[2]	31	√	unit and subunit_dependent_field							
:										
operand[32]										

Figure 11.12 – VERSION status command frame when subfunction = FF_{16}

11.5.1.3.1 Field definitions

version_information: The *version_information* field is used to specify whether to get the support level of the specified version or to get the latest version information. The values are described in the table below.

Table 11.12 – version_information field

version_information field value	Action
00_{16} – FE_{16}	Get the support level of the specified version
FF_{16}	Get the latest version information to which the unit or subunit complies

unit and subunit_dependent_field: The format and contents of *unit and subunit_dependent_field* depends on the (sub)unit specification.

11.5.1.3.2 VERSION status command response field values to get the latest version information

The VERSION STATUS command can be used to get the latest version of an AV/C unit's or subunit's specification. All (sub)units that implement VERSION status commands shall support the subfunction to get the latest version information (when *version_information* = FF_{16}). The following table shows the field values in the VERSION status command and response frames:

Table 11.13 – Field values in the VERSION status command: REJECTED and STABLE response frames to get the latest version information

Fields	Command	Response	
		REJECTED	STABLE
version_information	FF_{16}	←	latest_version_information
unit and subunit_dependent_field	all FF_{16}	←	unit and subunit_dependent_field

← means "same as the command frame".

The AV/C unit or subunit returns a STABLE response with the *latest_version_information* set to indicate the latest version of the specification to which a unit or subunit complies. The format and contents of the *latest_version_information* depends on the unit or subunit specification.

11.5.1.3.3 VERSION status command response field values to get the support level of the specified version

The VERSION status command can be used to get the support level of the specified version of an AV/C unit's or subunit's specification. The following table shows the fields values of VERSION status command and response frames to get the support level of the specified version:

Table 11.14 – Field values in the VERSION status command: REJECTED and STABLE response frames to get the support level of the specified version

Fields	Command	Response	
		REJECTED	STABLE
version_information	specified version information (00_{16}- FE_{16})	←	←
unit and subunit_dependent_field	all FF_{16}	←	unit and subunit_dependent_field

← means "same as the command frame".

If an AV/C unit or subunit is designed to comply with the specification specified by the *version_information* field value, it returns a STABLE response with the *unit_and_subunit_dependent_field* set to indicate its support information. The format and contents of the *version_information* field depends on the unit or subunit specification.

If the unit or subunit does not support the subfunction to get the support level of the specified version or does not comply with the specification specified by the *version_information* field value, it returns a NOT IMPLEMENTED response.

11.6 VENDOR-DEPENDENT commands

11.6.1 VENDOR-DEPENDENT commands

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

	length	ck	msb							lsb
opcode	1	√	VENDOR-DEPENDENT (00_{16})							
operand[0]	3	√	(most significant byte)							
operand[1]			company_ID							
operand[2]										(least significant byte)
operand[3]	See[1]	See[1]	vendor_dependent_data							
:										
operand[n]										

[1] Vendor dependent.

Figure 11.13 – VENDOR-DEPENDENT command frame

11.6.1.1 Field definitions

company_ID: The *company_ID* field shall contain the 24-bit unique ID obtained from the IEEE Registration Authority Committee (RAC). The value of *company_ID* provided in the operands of VENDOR-DEPENDENT commands indicates the company or the organization that defines the VENDOR-DEPENDENT command. The most significant part of the *company_ID* is stored in *operand[0]* and the least significant part in *operand[2]*.

vendor_dependent_data: 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 commands are defined in the same five command types, CONTROL, SPECIFIC INQUIRY, STATUS, NOTIFY and GENERAL INQUIRY, specified by the *ctype* field.

12. Connection commands

Connection commands are those commands that are used to establish, maintain, and break connections between unit and subunit plugs. Table 12.1 below summarizes the AV/C connection commands.

Table 12.1 – Connection Commands

Opcode	Value	Support level (by *ctype*)			Target	Comments
		C	S	N		
PLUG INFO	02_{16}	–	O	–	Unit or subunit	Information about unit and subunit plugs
CHANNEL USAGE	12_{16}	–	R	R	Unit only	Report information on IEEE 1394 isochronous channel usage
CONNECT	24_{16}	O	O	R	Unit only	Establish connections for unspecified streams between plugs and subunits
CONNECT AV	20_{16}	O	O	O	Unit only	Establish AV connections between plugs and subunits
CONNECTIONS	22_{16}	–	O	–	Unit only	Report connection status
DIGITAL INPUT	11_{16}	O	O	–	Unit only	Make or break broadcast Serial
DIGITAL OUTPUT	10_{16}	O	O	–	Unit only	Bus connections
DISCONNECT	25_{16}	O	–	–	Unit only	Break unspecified stream connections between plugs and subunits
DISCONNECT AV	21_{16}	O	–	–	Unit only	Break AV connections between plugs and subunits
INPUT PLUG SIGNAL FORMAT	19_{16}	O	R	O	Unit only	Set or report signal formats for
OUTPUT PLUG SIGNAL FORMAT	18_{16}	O	R	O	Unit only	Serial Bus plugs
GENERAL BUS SETUP	$1F_{16}$	O	O	O	Unit only	Setup General Bus

[1] C: CONTROL, S: STATUS, N: NOTIFY, M: Mandatory, R: Recommended, O: Optional, –: Not Defined

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

12.1 PLUG INFO command

12.1.1 PLUG INFO status command

The PLUG INFO status command is used to inquire about the information of plugs on the AV/C unit or one of its subunits determined by the AV/C address contained in the AV/C frame. The format of PLUG INFO status command frame is shown by Figure 12.1 below.

	length	ck	msb							lsb
opcode	1	√	PLUG INFO (02_{16})							
operand[0]	1	√	subfunction							
operand[1]	4	√	all FF_{16}							
:										
operand[4]										

Figure 12.1 – PLUG INFO status command frame

NOTE — This command has been extended to support asynchronous plugs in reference [R14] and [R15].

12.1.1.1 Field definitions

subfunction: The *subfunction* field specifies type of plug. When this command is addressed to a subunit, the *subfunction* field has the value 00_{16} and the number of subunit plugs are referred. When this command is addressed to an AV/C unit, the *subfunction* field has one of the values described in the Table 12.2 below.

Table 12.2 – Field values for subfunction (unit plugs)

Value	Subfunction
00_{16}	Serial Bus Isochronous and External Plug
01_{16}	Serial Bus Asynchronous Plug
02_{16} –$3F_{16}$	Reserved
40_{16} –$7F_{16}$	General Bus Plug (See Table 12.6)
80_{16} –FF_{16}	Reserved

12.1.1.2 PLUG INFO status command response from a subunit

If the PLUG INFO status command is addressed to an AV/C subunit, the format of the response frame is shown below.

	length	msb							lsb
opcode	1	PLUG INFO (02_{16})							
operand[0]	1	00_{16}							
operand[1]	1	destination_plugs							
operand[2]	1	source_plugs							
operand[3]	1	FF_{16}							
operand[4]	1	FF_{16}							

Figure 12.2 – PLUG INFO status command response format from an AV/C subunit

12.1.1.2.1 Field definitions

destination_plugs and source_plugs: For the AV/C subunit response frame, operand[1] and operand[2] shall indicate the number of destination and source plugs of that AV/C subunit, and operand[3] and operand[4] shall have the value FF_{16}.

12.1.1.2.2 PLUG INFO status command response field values from a subunit

Table 12.3 – Field values in the PLUG INFO status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
destination_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)
source_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)

← means "same as the command frame".

NOTE — The IN TRANSITION response is not valid for the PLUG INFO command.

12.1.1.3 PLUG INFO status command response from a unit when subfunction = 00_{16}

If the PLUG INFO status command is addressed to an AV/C unit, and the subfunction value is 00_{16}, then the format of the response is shown below.

	length	msb							lsb
opcode	1	PLUG INFO (02_{16})							
operand[0]	1	00_{16}							
operand[1]	1	serial_bus_isochronous_input_plugs							
operand[2]	1	serial_bus_isochronous_output_plugs							
operand[3]	1	external_input_plugs							
operand[4]	1	external_output_plugs							

Figure 12.3 – PLUG INFO status command response format from an AV/C unit when subfunction = 00_{16}

12.1.1.3.1 Field definitions

serial_bus_isochronous input plugs, serial_bus_isochronous_output_plugs, external_input_plugs and external_output_plugs: If the PLUG INFO status command is addressed to the AV/C unit, and the subfunction value is 00_{16}, operand[1] and operand[2] shall indicate the number of Serial Bus Isochronous Input and Output Plugs, respectively, while operand[3] and operand[4] shall indicate the number of External Input and Output Plugs, respectively.

12.1.1.3.2 PLUG INFO status command response field values from a unit when subfunction = 00_{16}

Table 12.4 – Field values in the PLUG INFO status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
serial_bus_isochronous_input_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)
serial_bus_isochronous_output_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)
external_input_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)
external_output_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)

← means “same as the command frame”.

NOTE — The IN TRANSITION response frame is not valid for the PLUG INFO command.

12.1.1.4 PLUG INFO status command response from unit when subfunction = 01_{16}

If the PLUG INFO status command is addressed to an AV/C unit, and the subfunction value is 01_{16}, then the format of the response frame is shown below.

	length	msb							lsb
opcode	1	PLUG INFO (02_{16})							
operand[0]	1	01_{16}							
operand[1]	1	serial_bus_asynchronous_input_plugs							
operand[2]	1	serial_bus_asynchronous_output_plugs							
operand[3]	1	FF_{16}							
operand[4]	1	FF_{16}							

Figure 12.4 – PLUG INFO status command response format from an AV/C unit when subfunction = 01_{16}

12.1.1.4.1 Field definitions

serial_bus_asynchronous_input_plugs and serial_bus_asynchronous_output_plugs: If the PLUG INFO status command is addressed to the AV/C unit, and the subfunction is 01_{16}, operand[1] and operand[2] shall indicate the number of Serial Bus Asynchronous Input and Output Plugs, respectively.

12.1.1.4.2 PLUG INFO status command response field values from a unit when subfunction = 01_{16}

Table 12.5 – Field values in the PLUG INFO status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
serial_bus_asynchronous_input_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)
serial_bus_asynchronous_output_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)

← means "same as the command frame".

NOTE − The IN TRANSITION response frame is not valid for the PLUG INFO command.

12.1.1.5 PLUG INFO status command response from unit when subfunction = 40_{16}– $7F_{16}$

If the PLUG INFO status command is addressed to an AV/C unit, and the subfunction value is in the range of 40_{16} – $7F_{16}$, then the format of the response frame is shown below.

	length	msb							lsb
opcode	1	PLUG INFO (02_{16})							
operand[0]	1	subfunction (40_{16}-$7F_{16}$)							
operand[1]	1	first_input_plug							
operand[2]	1	number_of_input_plugs							
operand[3]	1	first_output_plug							
operand[4]	1	number_of_output_plugs							

Figure 12.5 – PLUG INFO status command response format from an AV/C unit when subfunction = 40_{16} - $7F_{16}$

12.1.1.5.1 Field definitions

subfunction: When the PLUG INFO status command is used to inquire information about a general bus plug, the *subfunction* field has the value in the range of 40_{16}-$7F_{16}$ and specifies a bus type defined in the Table 12.6 below.

Table 12.6 – Bus type indicated by subfunction value

Value	Bus type
40_{16}	Bluetooth
41_{16} – $7F_{16}$	Reserved

first_input_plug: The *first_input_plug* field has the value of the smallest input plug number for the bus type specified by the *subfunction* field. The *first_input_plug* field value shall be in the range of 40_{16}-$5E_{16}$ (see Table 10.1).

number_of_input_plugs: This field has the number of input plugs for the bus type specified by the *subfunction* field. Those plugs shall have the consecutive number start from *first_input_plug*.

first_output_plug: The *first output_plug*_field has the value of the smallest output plug number for the bus type specified by the *subfunction* field. The *first_output_plug* field value shall be in the range of 40_{16}-$5E_{16}$ (see Table 10.1).

number_of_output_plugs: This field has the number of output plugs the bus type specified by the *subfunction* field. Those plugs shall have the consecutive value start from *first_output_plug*.

12.1.1.5.2 PLUG INFO status command response field values from a unit when subfunction = 40_{16}– $7F_{16}$

Table 12.7 – Field values in the PLUG INFO status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
first_input_plug	FF_{16}	←	40_{16} to $5E_{16}$
number_of_input_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)
first_output_plug	FF_{16}	←	40_{16} to $5E_{16}$
number_of_output_plugs	FF_{16}	←	0 to $1F_{16}$ (0 to 31)

← means “same as status command”.

NOTE − The IN TRANSITION response frame is not valid for the PLUG INFO command.

12.2 CHANNEL USAGE command

12.2.1 CHANNEL USAGE status command

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

Using a channel means that one of the AV/C unit's oPCRs indicates that there exists a connection that 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 format of CHANNEL USAGE status command frame is shown by Figure 12.6 below.

	length	ck	msb							lsb
opcode	1	√	CHANNEL USAGE (12_{16})							
operand[0]	1	√	IEEE 1394 isochronous channel							
operand[1]	2	√	node_ID = $FFFF_{16}$							
operand[2]										
operand[3]	1	√	oPCR number = FF_{16}							

Figure 12.6 – CHANNEL USAGE status command frame

12.2.1.1 Field definitions

IEEE 1394 isochronous channel: The target checks to see if it is using the channel denoted by the *IEEE 1394 isochronous channel* field.

node_ID: This field shall be set to $FFFF_{16}$ in the command frame.

oPCR number: This field shall be set to FF_{16} in the command frame.

12.2.1.2 CHANNEL USAGE status command responses

The CHANNEL USAGE status response has the format as shown by Figure 12.7 below.

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

Figure 12.7 – CHANNEL USAGE status command response format

12.2.1.2.1 Field definitions

IEEE 1394 isochronous channel: This value is the same as that specified in the STATUS command.

node_ID: If this value returns $FFFF_{16}$, it indicates that the target is not using the above isochronous channel for output. If a *node_id* is returned, the target is using the above channel.

oPCR number: If the *node_ID* above is $FFFF_{16}$ this value is also FF_{16}. If a *node_ID* is returned, this value is the output plug control register that is used to transmit on the isochronous channel.

In case the CHANNEL USAGE status command is broadcast (as opposed to unicast), the response obligation on this command exists only for the target that outputs the channel. Because at most one target can meet this condition, at most one response frame will be returned and that response shall have a valid *node_ID* and *oPCR number*.

12.2.1.3 CHANNEL USAGE status command and response field values

The following table shows the field values in the CHANNEL USAGE status command and response frames:

Table 12.8 – Field values in the CHANNEL USAGE status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
IEEE 1394 Isochronous Channel	$0 – 3F_{16}$	←	←
node_ID	$FFFF_{16}$	←	node_ID or $FFFF_{16}$
oPCR Number	FF_{16}	←	$0 – 1F_{16}$ or FF_{16}

← means “same as the command frame”.

NOTE — The IN TRANSITION response frame does not apply to the CHANNEL USAGE status command.

12.2.2 CHANNEL USAGE notify command

The CHANNEL USAGE command may also be used as a NOTIFY command. For the CHANNEL USAGE notify command, it is permissible to use the broadcasting *node_ID*. Note that at most one node can broadcast on an isochronous channel for output at one time.

12.2.2.1 CHANNEL USAGE notify command responses

All response frames of CHANNEL USAGE notify command shall have the same format as the command frame.

12.2.2.2 CHANNEL USAGE notify command and response field values

In case the CHANNEL USAGE command is 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 FF_{16}. If an interim response has been returned, a changed response shall be returned with operand[1] through operand[3] all equal to FF_{16} once the target stops using the specified channel.

In case the CHANNEL USAGE notify command is broadcast, the response obligation on this command exists only for the target that outputs 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 FF_{16}. If an interim response has been returned, a changed response shall be returned with operand[1] through operand[3] all equal to FF_{16} once the target stops using the specified channel.

The following table shows the field values in the CHANNEL USAGE notify command and response frames:

Table 12.9 – Field values in the CHANNEL USAGE notify command: REJECTED, INTERIM and CHANGED response frames

Fields	Command	Response		
		REJECTED	INTERIM	CHANGED
IEEE 1394 Isochronous Channel	$0 – 3F_{16}$	←	←	←
node_ID	$FFFF_{16}$	←	node_ID	$FFFF_{16}$
oPCR Number	FF_{16}	←	$0 – 1F_{16}$	FF_{16}

← means “same as the command frame”.

12.3 CONNECT command

12.3.1 CONNECT control command

The CONNECT control command establishes an internal connection as given in section 10.3.

NOTE — In reference [R12], the SIGNAL SOURCE command can be used in some cases as a preferable alternative to the CONNECT command.

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

	length	ck	msb							lsb
opcode	1	√	CONNECT (24_{16})							
operand[0]	1	√	$3F_{16}$						lock	perm
operand[1]	1	√[1]	source_subunit_type					source_subunit_ID		
:	1	√[1]	source_plug							
:	1	√[1]	destination_subunit_type					destination_subunit_ID		
operand[n]	1	√[1]	destination_plug							

1 If both the source plug and the destination plug are implemented but the connection between the plugs is not implemented, a NOT IMPLEMENTED response shall be returned.

Figure 12.8 – CONNECT control command frame

12.3.1.1 Field definitions

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

perm: The definition of the perm field is described in section 12.3.2.3. The target ignores this field in a CONNECT control command and returns the same value in the response.

source_subunit_type and ID, subunit_destination_type and ID: 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.

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:

	length	ck	msb							lsb
opcode	1	√	CONNECT (24_{16})							
operand[0]	1	√	$3F_{16}$						lock	perm
operand[1]	1	√[1]	source_subunit_type					source_subunit_ID		
:	1	√[1]	extended_source_subunit_type							
:	1	√[1]	extended_source_subunit_ID							
:	1	√[1]	source_plug							
:	1	√[1]	destination_subunit_type					destination_subunit_ID		
:	1	√[1]	extended_destination_subunit_type							
:	1	√[1]	extended_destination_subunit_ID							
operand[n]	1	√[1]	destination_plug							

1 If both the source plug and the destination plug are implemented but the connection between the plugs is not implemented, a NOT IMPLEMENTED response shall be returned.

Figure 12.9 – CONNECT control command frame 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.

source_plug and destination_plug: The *source_plug* and *destination_plug* fields are defined by Table 10.2.

When the stream flows from or to one of the AV/C unit's Isochronous Serial Bus, Asynchronous Serial Bus or External plugs, the *subunit_type* field shall have a value of $1F_{16}$ (AV/C 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 10.1.

If "any available plug" is specified in the *source_plug* or *destination_plug* fields, then the target shall assign the plug number, and place it in the response frame.

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

12.3.1.2 CONNECT control command responses

All response frames of CONNECT control command shall have the same format as the command frame.

12.3.1.3 CONNECT control and response field values

The following table shows the field values in the CONNECT control command and response frames:

Table 12.10 – Field values in the CONNECT control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
lock	0 or 1	←	←	←
perm	0 or 1	←	←	←
source_subunit_type	subunit_type[1]	←	←	←
source_subunit_ID	subunit_ID[2]	←	←	←
source_plug	source_plug[3]	←	←	updated source_plug
destination_subunit_type	subunit_type[1]	←	←	←
destination_subunit_ID	subunit_ID[2]	←	←	←
destination_plug	destination_plug[3]	←	←	updated destination_plug

[1] See subunit type table.
[2] See subunit ID table.
[3] See plug tables.
← means "same as the command frame".

12.3.2 CONNECT status command

The CONNECT command may also be used as a STATUS command to determine the current state of the connections within an AV/C 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 12.10 and Figure 12.11 below, and have the same meaning as the corresponding fields of the CONNECT control command.

	length	Ck	msb							lsb
opcode	1	√	CONNECT (24_{16})							
operand[0]	1	√	$3F_{16}$						lock = 1	prm = 1
operand[1]	1	√	source_subunit_type					source_subunit_ID		
:	1	√	source_plug							
:	1	√	FF_{16}							
operand[n]	1	√	FE_{16}							

Figure 12.10 – CONNECT status command frame for a source plug

	length	Ck	msb							lsb
opcode	1	√	CONNECT (24_{16})							
operand[0]	1	√	$3F_{16}$						lock = 1	prm = 1
operand[1]	1	√	FF_{16}							
:	1	√	FE_{16}							
:	1	√	destination_subunit_type					destination_subunit_ID		
operand[n]	1	√	destination_plug							

Figure 12.11 – CONNECT status command frame for a destination plug

12.3.2.1 Field definitions

The field definitions for STATUS commands are the same as CONTROL commands.

12.3.2.2 CONNECT status command responses

The CONNECT status response frame has the same format for all fields as the CONNECT control command.

12.3.2.3 CONNECT status command and response field values

Except for the *perm* bit, the CONNECT status response frame contains exact copies of the CONNECT response frame that is 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/C unit are connections that cannot be altered by the CONNECT control command or deleted by the DISCONNECT command, in which case a NOT IMPLEMENTED 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 FE_{16} (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 FE_{16} (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 FD_{16} (multiple plugs). The *perm* bit and *lock* bit in the response frame are the same as those in the command frame.

The following table shows the field values in the CONNECT status command and response frames:

Table 12.11 – Field values in the CONNECT status command: REJECTED and STABLE response frames when the source is specified

Fields	Command	Response	
		REJECTED	STABLE
lock	1	←	0, 1or ←
perm	1	←	0, 1or ←
source_subunit_type	subunit_type	←	←
source_subunit_ID	subunit_ID	←	←
source_plug	source_plug	←	←
destination_subunit_type	$1F_{16}$	←	subunit_type or ←
destination_subunit_ID	7_{16}	←	subunit_ID or ←
destination_plug	FE_{16}	←	destination_plug, FD_{16}, or FE_{16}[1]

[1] If the *destination_plug* returns FD_{16} or FE_{16}, the *lock*, the *perm*, the *destination_subunit_type* and *destination_subunit_ID* in the response frame are the same as those in the command frame.

← means "same as the command frame".

NOTE — The IN TRANSITION response frame does not apply to the CONNECT status command.

Table 12.12 – Field values in the CONNECT status command: REJECTED and STABLE response frames when the destination is specified

Fields	Command	Response	
		REJECTED	STABLE
lock	1	←	0, 1 or ←
perm	1	←	0, 1 or ←
source_subunit_type	$1F_{16}$	←	subunit_type or ←
source_subunit_ID	7_{16}	←	subunit_ID or ←
source_plug	FE_{16}	←	source_plug, or FE_{16}[1]
destination_subunit_type	subunit_type	←	←
destination_subunit_ID	subunit_ID	←	←
destination_plug	destination_plug	←	←

[1] If the *source_plug* returns FE_{16}, the *lock*, the *perm*, *source_subunit_type* and *source_subunit_ID* in the response frame are the same as those in the command frame.

← means "same as the command frame".

NOTE — The IN TRANSITION response frame does not apply to the CONNECT status command.

12.3.3 CONNECT notify command

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.

12.3.3.1 Field definitions

The field definitions for STATUS commands are the same as CONTROL commands.

12.3.3.2 CONNECT notify command responses

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.

12.3.3.3 CONNECT notify command and response field values

In the interim response frame, if the source or destination plug is connected, the plug is indicated. If it is not connected, the plug shall be indicated as invalid (plug field value FE_{16}). In the changed response frame, if the source or destination plug becomes unconnected, the plugs shall be indicated as invalid (plug field value FE_{16}), otherwise if connected, the plug is indicated. If the specified source plug or the specified destination plug can have the permanent connections only, the CONNECT notify command shall return a NOT IMPLEMENTED response.

The following table shows the field values in the CONNECT notify command and response frames:

Table 12.13 – Field values in the CONNECT notify command: REJECTED, INTERIM and CHANGED response frames when source is specified

Fields	Command	Response		
		REJECTED	INTERIM	CHANGED
lock	1	←	0, 1 or ←	0, 1 or ←
perm	1	←	0, 1 or ←	0, 1 or ←
source_subunit_type	subunit_type	←	←	←
source_subunit_ID	subunit_ID	←	←	←
source_plug	source_plug	←	←	←
destination_subunit_type	$1F_{16}$	←	subunit_type or ←	subunit_type or ←
destination_subunit_ID	7_{16}	←	subunit_ID, or ←	subunit_ID, or ←
destination_plug	FE_{16}	←	destination_plug, FD_{16} or FE_{16}[1]	destination_plug, FD_{16} or FE_{16}[1]

[1] If the *destination_plug* returns FD_{16} or FE_{16}, the *lock*, the *perm*, *destination_subunit_type* and *destination_subunit_ID* in the response frame are the same as those in the command frame.

← means "same as the command frame".

Table 12.14 – Field values in the CONNECT notify command: REJECTED, INTERIM and CHANGED response frames when destination is specified

Fields	Command	Response		
		REJECTED	INTERIM	CHANGED
lock	1	←	0, 1 or ←	0, 1 or ←
perm	1	←	0	0
source_subunit_type	$1F_{16}$	←	subunit_type or $1F_{16}$	subunit_type or $1F_{16}$
source_subunit_ID	7_{16}	←	subunit_ID or 7_{16}	subunit_ID or 7_{16}
source_plug	FE_{16}	←	source_plug or FE_{16}[1]	source_plug or FE_{16}[1]
destination_subunit_type	subunit_type	←	←	←
destination_subunit_ID	subunit_ID	←	←	←
destination_plug	destination_plug	←	←	←

[1] If the *source_plug* returns FE_{16}, the *lock*, the *perm*, *source_subunit_type* and *source_subunit_ID* in the response frame are the same as those in the command frame.

← means "same as the command frame".

12.4 CONNECT AV command

The CONNECT AV control command establishes audio/video connection as given in clause10.3.1. The CONNECT AV and DISCONNECT AV (in a following section) are used for DV systems, however, these commands cannot support multiple plugs nor asynchronous connections, and both audio and video plugs instead of specific subunit plugs must be indicated when using these commands. Therefore, it is not recommended to apply these commands to other systems, but to use CONNECT or SIGNAL SOURCE in reference [R12] instead.

12.4.1 CONNECT AV control command

The format of CONNECT AV control command frame is shown below.

	length	Ck	msb							lsb
opcode	1	√	CONNECT AV (20_{16})							
operand[0]	1	√	video_src_type		audio_src_type		video_dest_type		audio_dest_type	
operand[1]	1	√[1]	video_source							
:	1	√[1]	audio_source							
:	1	√[1]	video_destination							
operand[n]	1	√[1]	audio_destination							

[1] If both the source plug and the destination plug are implemented but the connection between the plugs is not implemented, a NOT IMPLEMENTED response shall be returned.

Figure 12.12 – CONNECT AV control command frame for audio/video stream

12.4.1.1 Field definitions

audio/video_src/dest_type: The fields *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:

Table 12.15 – Source and destination identifying fields

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

audio/video_source/destination: If the source or destination type is zero, the corresponding source or destination operand is a subunit address encoded as described in the subunit_type table. 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 FF_{16} is a special case and indicates that the AV/C 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/C unit does not implement the connection type. For example, in a CONNECT AV control command sent to an AV/C unit

that has 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:

Table 12.16 – Serial bus or external plug values

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/C device to select, at its option, an available subunit, Serial Bus Isochronous 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 Isochronous input plug is active and configured for HD signals, a CONNECT AV control command for an audio/video stream that specifies “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 Isochronous 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.

12.4.1.2 CONNECT AV control command responses

All response frames of CONNECT AV control command shall have the same format as the command frame.

12.4.1.3 CONNECT AV control command and response field values

The following table shows the field values in the CONNECT AV control command and response frames:

Table 12.17 – Field values in the DISCONNECT control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
video_src_type	0, 1, or 2	←	←	←
audio_src_type	0, 1, or 2	←	←	←
video_dest_type	0, 1, or 2	←	←	←
video_src_type	0, 1, or 2	←	←	←
video_source	subunit_type & ID, or plug	←	←	←, or plug[1]
audio_source	subunit_type & ID, or plug	←	←	←, or plug[1]
video_destination	subunit_type & ID, or plug	←	←	←, or plug[1]
audio_destination	subunit_type & ID, or plug	←	←	←, or plug[1]

[1]Plug number is returned if “any plug” ($7F_{16}$ or FF_{16}) is specified in the command frame.

← means “same as the command frame”.

12.4.2 CONNECT AV status command

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 an internal A/V connection for a unit or subunit. The response shows the connected source plugs or (sub)unit(s). The format of the CONNECT AV status command frame is shown in Figure 12.13 below.

	length	ck	msb							lsb
opcode	1	√	CONNECT AV (20_{16})							
operand[0]	1	√	vid_src_type=3_{16}		aud_src_type=3_{16}		video_dest_type		audio_dest_type	
operand[1]	1	√	video_source = FF_{16}							
:	1	√	audio_source = FF_{16}							
:	1	√	video_destination							
operand[n]	1	√	audio_destination							

Figure 12.13 – CONNECT AV status command frame for audio/video stream

12.4.2.1 Field definitions

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

12.4.2.2 CONNECT AV status command responses

The response frame returned for the CONNECT AV status command shows the source of the connection(s) and has the same format as described in Figure 12.12.

12.4.2.3 CONNECT AV status command and response field values

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 FF_{16} (invalid), and the *video_source_type* and *audio_source_type* fields shall both have the value 1 (ignore).

The following table shows the field values in the CONNECT AV status command and response frames:

Table 12.18 – Field values in the CONNECT AV status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
video_src_type	3	←	0, 1, or 2
audio_src_type	3	←	0, 1, or 2
video_dest_type	0, 1, or 2	←	←
video_src_type	0, 1, or 2	←	←
video_source	FF_{16}	←	subunit_type & ID, plug, or FF_{16}
audio_source	FF_{16}	←	subunit_type & ID, plug, or FF_{16}
video_destination	subunit_type & ID, or plug	←	←
audio_destination	subunit_type & ID, or plug	←	←

← means "same as the command frame".

NOTE — The IN TRANSITION response frame does not apply to the CONNECT AV status command.

12.4.3 CONNECT AV notify command

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.

12.4.3.1 Field definitions

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

12.4.3.2 CONNECT AV notify command responses

All response frames of CONNECT AV notify command shall have the same format as the command frame.

12.4.3.3 CONNECT AV notify command and response field values

The following table shows the field values in the CONNECT AV notify command and response frames:

Table 12.19 – Field values in the CONNECT AV notify command: REJECTED, INTERIM and STABLE response frames

Fields	Command	Response		
		REJECTED	INTERIM	CHANGED
video_src_type	3	←	0, 1, or 2	0, 1, or 2
audio_src_type	3	←	0, 1, or 2	0, 1, or 2
video_dest_type	0, 1, or 2	←	←	←
video_src_type	0, 1, or 2	←	←	←
video_source	FF_{16}	←	subunit_type & ID, plug, or FF_{16}	subunit_type & ID, plug, or FF_{16}
audio_source	FF_{16}	←	subunit_type & ID, plug, or FF_{16}	subunit_type & ID, plug, or FF_{16}
video_destination	subunit_type & ID, or plug	←	←	←
audio_destination	subunit_type & ID, or plug	←	←	←

← means “same as the command frame”.

12.5 CONNECTIONS command

12.5.1 CONNECTIONS status command

The CONNECTIONS status command is used to inquire the state of all connections for unspecified streams. The format of the CONNECTIONS status command frame is shown by Figure 12.14 below.

	length	ck	msb							lsb
opcode	1	√	CONNECTIONS (22_{16})							
operand[0]	1	√	FF_{16}							

Figure 12.14 – CONNECTIONS status command frame

12.5.1.1 CONNECTIONS status command responses

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 the figure below.

	length	msb							lsb
opcode	1	CONNECTIONS (22_{16})							
operand[0]	1	total_connections							
:	see[1]	connection_information[0]							
:	:	:							
operand[n]	see[1]	connection_information[total_connections – 1]							

[1] The length may vary depending on the total connections and whether the subunit_type extension is used or not.

Figure 12.15 – CONNECTIONS status command response format

12.5.1.1.1 Field definitions

total_connections: The *total_connections* field specifies the number of connection information fields returned in the operands that follow.

connection_information: The connection information is shown below:

	length	msb							lsb
operand[x]	1	$3F_{16}$						lock	perm
:	see[1]	connection[x].source							
:									
:	see[1]	connection[x].destination							
:									

[1] The length may vary depending on whether the subunit_type extension is used or not.

Figure 12.16 – Connection information

The format of each connection information is identical to operand[1] through operand[4] of the CONNECT control command on page 78. For a connection that includes an extended *subunit_type* or *subunit_ID*, these addresses may change depending on the number of extended fields.

12.5.1.2 CONNECTIONS status command and response field values

The following table shows the field values in the CONNECTIONS status command and response frames:

Table 12.20 – Field values in the CONNECTIONS status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
total_connections	FF_{16}	←	0 – 101
lock[x]	not supplied	←	0 or 1
perm[x]	not supplied	←	0 or 1
source_subunit_type[x]	not supplied	←	subunit_type[1]
source_subunit_ID[x]	not supplied	←	subunit_ID[2]
source_plug[x]	not supplied	←	source_plug[3]
destination_subunit_type[x]	not supplied	←	subunit_type[1]
destination_subunit_ID[x]	not supplied	←	subunit_ID[2]
destination_plug[x]	not supplied	←	source_plug[3]

[1] See subunit type table.
[2] See subunit ID table.
[3] See plug tables.
← means “same as the command frame”.

NOTE — The IN TRANSITION response frame does not apply to the CONNECTIONS status command.

12.6 DIGITAL INPUT command

12.6.1 DIGITAL INPUT control command

The DIGITAL INPUT control command permits an AV/C unit to establish a broadcast-in connection according to its own preferences. The format of the DIGITAL INPUT control command frame is shown below.

	length	ck	msb							lsb
opcode	1	√	DIGITAL INPUT (11_{16})							
operand[0]	1	√	connection_state							

Figure 12.17 – DIGITAL INPUT command frame

12.6.1.1 Field definitions

connection_state: The *connection_state* field specifies whether the AV/C unit is expected to establish (70_{16}) or break (60_{16}) a broadcast input connection.

12.6.1.2 DIGITAL INPUT control command responses

All response frames of DIGITAL INPUT control command shall have the same format as the command frame.

12.6.1.3 DIGITAL INPUT control command and response field values

The following table shows the field values in the DIGITAL INPUT control command and response frames:

Table 12.21 – Field values in the DIGITAL INPUT control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
connection_state	70_{16} or 60_{16}	←	←	←

← means "same as the command frame".

12.6.2 DIGITAL INPUT status command

The DIGITAL INPUT status command may also be used to determine the current input broadcast connection state of the unit. In this case, *operand[0]* is set to FF_{16} when the STATUS command is issued and is updated to the current *connection_state* when the STABLE response frame is returned.

12.6.2.1 DIGITAL INPUT status command responses

All response frames of DIGITAL INPUT status command shall have the same format as the command frame.

12.6.2.2 DIGITAL INPUT status command and response field values

The following table shows the field values in the DIGITAL INPUT status command and response frames.

Table 12.22 – Field values in the DIGITAL INPUT status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
connection_state	FF_{16}	←	60_{16} or 70_{16}

← means “same as the command frame”.

NOTE — The IN TRANSITION response frame does not apply to the DIGITAL INPUT status command.

12.7 DIGITAL OUTPUT command

12.7.1 DIGITAL OUTPUT control command

The DIGITAL OUTPUT control command permits an AV/C unit to establish a broadcast-out connection according to its own preferences. The format of the DIGITAL OUTPUT control frame is shown below.

	length	ck	msb							lsb
opcode	1	√	DIGITAL OUTPUT (10_{16})							
operand[0]	1	√	connection_state							

Figure 12.18 – DIGITAL OUTPUT command frame

12.7.1.1 Field definitions

connection_state: The *connection_state* field specifies whether the AV/C unit is expected to establish (70_{16}) or break (60_{16}) a broadcast output connection. The AV/C 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.

12.7.1.2 DIGITAL OUTPUT control command responses

All response frames of DIGITAL OUTPUT control command shall have the same format as the command frame.

12.7.1.3 DIGITAL OUPUT control command and response field values

The following table shows the field values in the DIGITAL OUTPUT control command and response frames:

Table 12.23 – Field values in the DIGITAL OUTPUT control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
connection_state	70_{16} or 60_{16}	←	←	←

← means “same as the command frame”.

12.7.2 DIGITAL OUTPUT status command

The DIGITAL OUTPUT status command may also be used to determine the current output broadcast connection state of the unit. In this case, *operand[0]* is set to FF_{16} when the STATUS command is issued and is updated to the current *connection_state* when the STABLE response frame is returned.

12.7.2.1 Field definitions

The field definitions in the STATUS command are the same as in the CONTROL command.

12.7.2.2 DIGITAL OUTPUT status command responses

All response frames of DIGITAL OUTPUT status command shall have the same format as the command frame.

12.7.2.3 DIGITAL OUTPUT status command and response field values

The following table shows the field values in the DIGITAL OUTPUT status command and response frames:

Table 12.24 – Field values in the DIGITAL OUTPUT status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
connection_state	FF_{16}	←	60_{16} or 70_{16}

← means “same as the command frame”.

NOTE — The IN TRANSITION response frame does not apply to the DIGITAL OUTPUT status command.

12.8 DISCONNECT command

The DISCONNECT command is used to disconnect the connection which has been made by the CONNECT command.

12.8.1 DISCONNECT control 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 frame is shown below.

	length	ck	msb							lsb
opcode	1	√	DISCONNECT (25_{16})							
operand[0]	1	√	FF_{16}							
operand[1]	1	√	source_subunit_type					source_subunit_ID		
operand[2]	1	√	source_plug							
operand[3]	1	√	destination_subunit_type					destination_subunit_ID		
operand[4]	1	√	destination_plug							

Figure 12.19 – DISCONNECT command frame

12.8.1.1 Field definitions

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.

12.8.1.2 DISCONNECT control command responses

All response frames of the DISCONNECT control command shall have the same format as the command frame.

12.8.1.3 DISCONNECT control command and response field values

The following table shows the field values in the DISCONNECT control command and response frames:

Table 12.25 – Field values in the DISCONNECT control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
source_subunit_type	subunit_type[1]	←	←	←
source_subunit_ID	subunit_ID[2]	←	←	←
source_plug	source_plug[3]	←	←	←
destination_subunit_type	subunit_type[1]	←	←	←
destination_subunit_ID	subunit_ID[2]	←	←	←
destination_plug	destination_plug[3]	←	←	←

[1] See subunit type table.
[2] See subunit ID table.
[3] See plug tables.
← means “same as the command frame”.

12.9 DISCONNECT AV command

The DISCONNECT AV command is used to disconnect the connection which has been made by the CONNECT AV command.

12.9.1 DISCONNECT AV control command

The DISCONNECT AV control command is used to remove audio/video connections between subunits and plugs as described in the CONNECT AV command. The value of *operand[0]* is other than FF_{16} and the syntax is shown in the figure below.

	length	ck	msb							lsb
opcode	1	√	DISCONNECT AV (21_{16})							
operand[0]	1	√	video_source_typ		audio_source_typ		video_dest_type		audio_dest_type	
operand[1]	1	√	video_source							
operand[2]	1	√	audio_source							
operand[3]	1	√	video_destination							
operand[4]	1	√	audio_destination							

Figure 12.20 – DISCONNECT AV command frame

12.9.1.1 Field definitions

The field definitions and their uses for DISCONNECT AV are identical to the field definitions given in Figure 12.12 for the CONNECT AV command. This includes the extended source and destination *subunit_type* and *subunit_ID* if they are used.

12.9.1.2 DISCONNECT AV control command responses

All response frames of DISCONNECT AV control command shall have the same format as the command frame.

12.9.1.3 DISCONNECT AV control command and response field values

The following table shows the field values in the DISCONNECT AV control command and response frames:

Table 12.26 – Field values in the DISCONNECT AV control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
video_src_type	0, 1, or 2	←	←	←
audio_src_type	0, 1, or 2	←	←	←
video_dest_type	0, 1, or 2	←	←	←
video_src_type	0, 1, or 2	←	←	←
video_source	subunit_type & ID, or plug	←	←	←
audio_source	subunit_type & ID, or plug	←	←	←
video_destination	subunit_type & ID, or plug	←	←	←
audio_destination	subunit_type & ID, or plug	←	←	←

← means “same as the command frame”.

12.10 INPUT PLUG SIGNAL FORMAT command

12.10.1 INPUT PLUG SIGNAL FORMAT control command

The INPUT PLUG SIGNAL FORMAT control command is used to configure a specified unit Serial Bus isochronous input plug to receive data in the designated signal format. The format of the INPUT PLUG SIGNAL FORMAT control command frame is shown in the figure below.

	length	ck	msb							lsb
opcode	1	√	INPUT PLUG SIGNAL FORMAT (19_{16})							
operand[0]	1	√	plug							
operand[1]	1	√	eoh=1	form=0	fmt					
operand[2]	3	√	(most significant byte)							
operand[3]			fdf							
operand[4]										(least significant byte)

Figure 12.21 – INPUT PLUG SIGNAL FORMAT control command frame

NOTE — The INPUT PLUG SIGNAL FORMAT command does not apply to asynchronous or external connections.

12.10.1.1 Field definitions

plug: The *plug* field specifies which of the 31 Serial Bus isochronous input plugs are referenced. For more information on specifying this field, see Table 10.1.

eoh: Indicates the end of the CIP header of an isochronous packet. This value shall always be 1 in this command. See reference [R7]and the related specifications for more information.

form: In combination with *eoh*, indicates the additional structure of the CIP header field. This value shall always be 0 in this command. See reference [R7] and the related specifications for more information.

fmt and fdf: The fields *fmt* and *fdf* are as defined in IEC 61883, Digital Interface for Consumer Electronic Audio/Video Equipment [R7] and the related specifications. Together they specify the desired signal format for the Serial Bus Isochronous input plug identified by *plug*.

If the first bit of the fmt field is 0, then the fdf field shall contain $FFFF_{16}$ in its last two bytes (the SYT field) meaning “no information”.

If the *fmt* field’s type is 100000_2 (MPEG2-TS), then the Time Shift Flag in the *fdf* field has no meaning for this command. The Time Shift Flag may contain any values and the target ignores it.

12.10.1.2 INPUT PLUG SIGNAL FORMAT control command responses

All response frames of INPUT PLUG SIGNAL FORMAT control command shall have the same format as the command frame.

12.10.1.3 INPUT PLUG SIGNAL FORMAT control command and response field values

The following table shows the field values in the INPUT PLUG SIGNAL FORMAT control command and response frames:

Table 12.27 – Field values in the INPUT PLUG SIGNAL FORMAT control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
plug	0 – 30	←	←	←
eoh	1	←	←	←
form	0	←	←	←
fmt	see[1]	←	←	←
fdf	see[1]	←	←	←

[1] This value is defined in reference [R7]and the related specifications.

← means "same as the command frame".

12.10.2 INPUT PLUG SIGNAL FORMAT status command

The INPUT PLUG SIGNAL FORMAT status command is used to inquire which signal format a specified Serial Bus Isochronous input plug is configured to receive. The format of the INPUT PLUG SIGNAL FORMAT status command frame is shown in the figure below.

	length	ck	msb							lsb
opcode	1	√	INPUT PLUG SIGNAL FORMAT (19_{16})							
operand[0]	1	√	plug							
operand[1]	4	√	all FF_{16}							
:										
operand[4]										

Figure 12.22 – INPUT PLUG SIGNAL FORMAT status command frame

12.10.2.1 Field definitions

The field definitions for STATUS commands are the same as CONTROL commands. The *fmt* and *fdf* fields specify the signal format that the Serial Bus input plug identified by *plug* is configured to receive.

The Time Shift Flag (For MPEG2-TS) in the *fdf* field may be used for the STATUS command as an exception. Some target implementations return the Time Shift Flag of the receiving signal.

12.10.2.2 INPUT PLUG SIGNAL FORMAT status command responses

All response frames of INPUT PLUG SIGNAL FORMAT status command shall have the same format as the command frame.

12.10.2.3 INPUT PLUG SIGNAL FORMAT status command and response field values

The following table shows the field values in the INPUT PLUG SIGNAL FORMAT status command and response frames:

Table 12.28 – Field values in the INPUT PLUG SIGNAL FORMAT status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
plug	0 – 30	←	←
eoh	1	←	←
form	1	←	0
fmt	$3F_{16}$	←	see[1]
fdf	all FF_{16}	←	see[1]

[1] This field value is defined in reference [R7] and the related specifications.

← means "same as the command frame".

NOTE — The IN TRANSITION response frame does not apply to the INPUT PLUG SIGNAL FORMAT status command.

12.10.3 INPUT PLUG SIGNAL FORMAT notify command

The INPUT PLUG SIGNAL FORMAT command may also be used as a NOTIFY command. The NOTIFY command frame has the same format as the STATUS command frame. 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 Isochronous input plug is receiving changes.

12.10.3.1 Field definitions

The field definitions for NOTFY commands are the same as STATUS commands.

12.10.3.2 INPUT PLUG SIGNAL FORMAT notify command responses

All response frames of INPUT PLUG SIGNAL FORMAT notify command shall have the same format as the command frame.

12.10.3.3 INPUT PLUG SIGNAL FORMAT notify command and response field values

The following table shows the field values in the INPUT PLUG SIGNAL FORMAT notify command and response frames.

Table 12.29 – Field values in the INPUT PLUG SIGNAL FORMAT notify command: REJECTED, INTERIM and CHANGED response frames

Fields	Command	Response		
		REJECTED	INTERIM	CHANGED
plug	0 – 30	←	←	←
eoh	1	←	←	←
form	1	←	0	0
fmt	$3F_{16}$	←	see[1]	see[1]
fdf	all FF_{16}	←	see[1]	see[1]

[1] This field value is defined in reference [R7] and the related specifications.

← means “same as the command frame”.

12.11 OUTPUT PLUG SIGNAL FORMAT command

The OUTPUT PLUG SIGNAL FORMAT control command is used to configure a specified Serial Bus isochronous output plug to transmit data in the designated signal format. The general format of the OUTPUT PLUG SIGNAL FORMAT control command frame is shown in the figure below.

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

Figure 12.23 – OUTPUT PLUG SIGNAL FORMAT command frame

12.11.1 OUTPUT PLUG SIGNAL FORMAT control command

The format of the OUTPUT PLUG SIGNAL FORMAT control command frame is shown in the figure below.

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

Figure 12.24 – OUTPUT PLUG SIGNAL FORMAT control command frame

12.11.1.1 Field definitions

plug: The *plug* field specifies which of the 31 Serial Bus isochronous output plugs is referenced. For more information on specifying this field, see Table 10.1.

eoh: Indicates the end of the CIP header of an isochronous packet. This value shall always be 1 in this command. See reference [R7] and the related specifications for more information.

form: In combination with *eoh*, indicates the additional structure of the CIP header field. This value shall always be 0 in this command. See reference [R7] and the related specifications for more information.

fmt and fdf: The fields *fmt* and *fdf* are as defined in IEC 61883, Digital Interface for Consumer Electronic Audio/Video Equipment [R7] and the related specifications. Together they specify the desired signal format for the Serial Bus Isochronous output plug identified by *plug*.

If the first bit of the *fmt* field is 0, then the *fdf* field shall contain $FFFF_{16}$ in its last two bytes (the SYT field) meaning “no information”.

If the *fmt* field’s type is 100000_2 (MPEG2-TS), then the Time Shift Flag in the *fdf* field has no meaning for this command. The Time Shift Flag may contain any values and the target ignores it.

12.11.1.2 OUTPUT PLUG SIGNAL FORMAT control command responses

All response frames of OUTPUT PLUG SIGNAL FORMAT control command shall have the same format as the command frame.

12.11.1.3 OUTPUT PLUG SIGNAL FORMAT control command and response field values

The following table shows the field values in the OUTPUT PLUG SIGNAL FORMAT control command and response frames:

Table 12.30 – Field values in the OUTPUT PLUG SIGNAL FORMAT control command: REJECTED, INTERIM and ACCEPTED response frames

Fields	Command	Response		
		REJECTED	INTERIM	ACCEPTED
plug	0 – 30	←	←	←
eoh	1	←	←	←
form	0	←	←	←
fmt	see[1]	←	←	←
fdf	see[1]	←	←	←

[1] This value is defined in reference [R7] and the related specifications.

← means “same as the command frame”.

12.11.2 OUTPUT PLUG SIGNAL FORMAT status command

The OUTPUT PLUG SIGNAL FORMAT status command is used to inquire which signal format a specified Serial Bus Isochronous output plug is configured to transmit. The format of the OUTPUT PLUG SIGNAL FORMAT command frame is shown by Figure 12.25 below.

	length	ck	msb							lsb
opcode	1	√	OUTPUT PLUG SIGNAL FORMAT (18_{16})							
operand[0]	1	√	plug							
operand[1]	4	√	all FF_{16}							
...										
operand[4]										

Figure 12.25 – OUTPUT PLUG SIGNAL FORMAT status command frame

12.11.2.1 Field definitions

The field definitions for STATUS commands are the same as CONTROL commands. The *fmt* and *fdf* fields specify the signal format that the Serial Bus output plug identified by *plug* is configured to send.

The Time Shift Flag (For MPEG2-TS) in the *fdf* field may be used for the STATUS command. Some target implementations return the Time Shift Flag of the sending signal.

12.11.2.2 OUTPUT PLUG SIGNAL FORMAT status command responses

All response frames of OUTPUT PLUG SIGNAL FORMAT status command shall have the same format as the command frame.

12.11.2.3 OUTPUT PLUG SIGNAL FORMAT status command and response field values

The following table shows the field values in the OUTPUT PLUG SIGNAL FORMAT status command and response frames:

Table 12.31 – Field values in the OUTPUT PLUG SIGNAL FORMAT status command: REJECTED and STABLE response frames

Fields	Command	Response	
		REJECTED	STABLE
plug	0 – 30	←	←
eoh	1	←	←
form	1	←	0
fmt	$3F_{16}$	←	see[1]
fdf	all FF_{16}	←	see[1]

[1] This field value is defined in reference [R7] and the related specifications.

← means “same as the command frame”.

NOTE — The IN TRANSITION response frame does not apply to the OUTPUT PLUG SIGNAL FORMAT status command.

12.11.3 OUTPUT PLUG SIGNAL FORMAT notify command

The OUTPUT PLUG SIGNAL FORMAT command may also be used as a NOTIFY command. The NOTIFY command frame has the same format as the STATUS command frame. 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 Isochronous output plug is transmitting changes.

12.11.3.1 Field definitions

The field definitions for NOTIFY commands are the same as STATUS commands.

12.11.3.2 OUTPUT PLUG SIGNAL FORMAT notify command responses

All response frames of OUTPUT PLUG SIGNAL FORMAT notify command shall have the same format as the command frame.

12.11.3.3 OUTPUT PLUG SIGNAL FORMAT notify command and response field values

The following table shows the field values in the OUTPUT PLUG SIGNAL FORMAT notify command and response frames:

Table 12.32 – Field values in the OUTPUT PLUG SIGNAL FORMAT notify command: REJECTED, INTERIM and CHANGED response frames

Fields	Command	Response		
		REJECTED	INTERIM	CHANGED
plug	0 – 30	←	←	←
eoh	1	←	←	←
form	1	←	0	0
fmt	$3F_{16}$	←	see[1]	see[1]
fdf	all $FF1_6$	←	see[1]	see[1]

[1] This field value is defined in reference [R7] and the related specifications.

← means “same as the command frame”.

12.12 GENERAL BUS SETUP commands

12.12.1 GENERAL BUS SETUP, ctype = all

GENERAL BUS SETUP command is used to set up General Bus Plug. The structure of the command is shown by Figure 12.26 below.

	length	ck	msb							lsb
opcode	1	√	GENERAL BUS SETUP ($1F_{16}$)							
operand[0]	1	√	bus_type							
operand[1]	See[1]	See[1]	bus_type dependent_data							
:										
operand[n]										

[1] Bus dependent

Figure 12.26 – GENERAL BUS SETUP command frame

12.12.1.1 Field definitions

bus_type: The field specifies a type of general bus plug. The value available is the same as in Table 12.6.

bus_type_dependent_data: The format and meaning of the *bus_type_dependent_data* field are specified by the type of bus identified by *bus_type*.

Annexes

Annex A: Target State Change Sources (informative)

AV/C target devices can have multiple states that are dependent or independent from each other. For example, a Disc subunit with a descriptor mechanism (used for storing various types of descriptive data) can be in a play state, and have a descriptor in an open state at the same time. Connections also have states, which exist independently of other unit or subunit states.

Dependent and independent target states can be changed in the following ways:

1) Via AV/C CONTROL commands.
2) Via a front panel controller or a non-AV/C remote control device.
3) Via an internal event.

Since a target can have various sources of user input, each input may actually contain a subset of the controllable features of the target. The manufacturer of the target device should determine the source of usable features.

In some cases, there may be a conflict when two different state change sources request the target to enter states that can only exist when the other doesn't. For example, a descriptor may be open for write by a particular AV/C controller, and a front panel command such as RECORD requires immediate closure of the descriptor. It is an implementation's responsibility to assign priority to the conflicting commands, and to have one command yield to the other.