



AV/C commands for management of Asynchronous Serial Bus Connections

Version 1.0
January 26, 1999

Sponsored by:

Audio/Video Working Group of the 1394 Trade Association

Approved for Release by:

1394 Trade Association Board of Directors

Abstract:

This document describes AV/C commands for management of asynchronous serial bus connections. An asynchronous data-transfer service efficiently transfer data between simple Serial Bus compliant deices. Management commands include establishing, breaking, and monitoring the connection. And the extensions of existing AV/C commands are also described.

Keywords: , management

1394 Trade Association
Regency Plaza Suite 350, 2350 Mission College Blvd., Santa Clara, CA 95054, USA
<http://www.1394TA.org>

Copyright © 1998-1999 by the 1394 Trade Association. Permission is granted to members of the 1394 Trade Association to reproduce this document for their own use or the use of other 1394 Trade Association members only, provided this notice is included. All other rights reserved. Duplication for sale, or for commercial or for-profit use is strictly prohibited without the prior written consent of the 1394 Trade Association.

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

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

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

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

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

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

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

Table of Contents

- 1. Overview 1
 - 1.1 Scope 1
 - 1.2 Summary..... 1
- 2. References 2
- 3. Definitions and abbreviations 3
 - 3.1 Conformance glossary 3
 - 3.2 Technical glossary 3
 - 3.3 Size notation 5
 - 3.4 Numerical notation..... 5
 - 3.5 State machine notation 5
 - 3.6 C++ code notation 7
 - 3.7 Unimplemented locations 7
 - 3.7.1 Reserved locations 7
 - 3.7.2 Ignored locations 8
 - 3.7.3 Unused locations 8
- 4. Extension for existing AV/C commands 9
 - 4.1 Plug numbers for asynchronous plug definition..... 9
 - 4.2 Extended command for asynchronous plug support..... 10
- 5. AV/C connection sequences 11
 - 5.1 Establishing an asynchronous connection..... 11
 - 5.2 Breaking an asynchronous connection 11
 - 5.3 Failure of establishing an asynchronous connection 12
 - 5.4 Overlaying an asynchronous connection 13
 - 5.5 Breaking an overlaid connection..... 14
 - 5.6 Establish multicast connections 14
 - 5.7 Breaking multicast connections 15
 - 5.8 Reconnect procedures after bus reset..... 16
 - 5.8.1 AV/C reconnect timing 16
 - 5.8.2 AV/C reconnect commands..... 16
 - 5.9 Automatic disconnection 16
- 6. Asynchronous Connection management command 20
 - 6.1 Overview 20
 - 6.2 Common frame format 21
 - 6.3 CONTROL commands 28
 - 6.3.1 ALLOCATE subfunction..... 28
 - 6.3.2 ALLOCATE_ATTACH subfunction..... 29
 - 6.3.3 ALLOCATE_ATTACH_FRAME subfunction..... 31
 - 6.3.4 ATTACH subfunction 33
 - 6.3.5 ATTACH_FRAME subfunction..... 35
 - 6.3.6 ADD_OVERLAY subfunction..... 36
 - 6.3.7 DETACH subfunction 38
 - 6.3.8 DETACH_RELEASE subfunction 40
 - 6.3.9 RELEASE subfunction 41
 - 6.3.10 RESTORE_PORT subfunction..... 43
 - 6.3.11 RESTORE_PORT_FRAME subfunction..... 45
 - 6.3.12 SUSPEND_PORT subfunction..... 50



6.3.13	RESUME_PORT subfunction.....	52
6.4	STATUS command.....	54
7.	Asynchronous Connection port states.....	55
7.1	Code definitions.....	56
7.2	Consumer port states.....	57
7.2.1	Consumer port state machine.....	57
7.2.2	Consumer port state machine notes.....	59
7.3	Producer port states.....	63
7.3.1	Producer port state machines.....	63
7.3.2	Producer port state machine notes.....	65

List of Figures

Figure 3.5.1-	State machine notation.....	6
Figure 3.5.2 -	Equivalent state machine.....	6
Figure 4.2.1 -	Extended PLUG INFO status command generic format.....	10
Figure 4.2.2 -	Extended PLUG INFO status command format.....	10
Figure 4.2.3 -	Extended PLUG INFO status response format from an AV unit.....	10
Figure 5.1.1 -	Controller managed connection.....	11
Figure 5.2.1 -	Connection break sequence.....	12
Figure 5.3.1 -	Producer-plug connection failure.....	13
Figure 5.4.1 -	Connecting an overlaid plug.....	13
Figure 5.5.1 -	Disconnecting an overlaid plug.....	14
Figure 5.6.1 -	Establishing multicast connections.....	14
Figure 5.7.1 -	Breaking multicast connections.....	15
Figure 5.8.1 -	Reconnecting asynchronous plugs.....	16
Figure 5.9.1 -	Producer initiated disconnection.....	17
Figure 5.9.2 -	Consumer initiated disconnection.....	18
Figure 6.2.1 -	ASYNCHRONOUS CONNECTION common frame format.....	21
Figure 7.2.1 -	Consumer port state machine.....	58
Figure 7.2.2 -	Consumer port state machine (cont'd).....	59
Figure 7.3.1 -	Producer port state machine.....	64

List of Tables

Table 3.3-1 -	Size notation examples.....	5
Table 3.6-1 -	Specific expression summary.....	7
Table 3.6-2 -	Serial Bus data types.....	7
Table 4.1-1 -	Serial Bus and external plug numbers.....	9
Table 4.1-2 -	Existing AV/C commands which handles asynchronous plug.....	9
Table 6.1-1 -	Support level of asynchronous connection management command.....	20
Table 6.2-1 -	<i>subfunction</i> field definitions.....	22
Table 6.2-2 -	Support level of subfunctions.....	23
Table 6.2-3 -	<i>status</i> field values for CONTROL command response frame.....	23
Table 6.2-4 -	<i>status</i> field values for STATUS command response frame.....	24
Table 6.2-5 -	<i>plug ID</i> supported values.....	24
Table 6.2-6 -	<i>port ID</i> definitions.....	25
Table 6.2-7 -	<i>port bits</i> field definitions.....	25
Table 6.3-1 -	Command and ACCEPTED response frame of ALLOCATE.....	28
Table 6.3-2 -	Command and REJECTED response frame of ALLOCATE.....	28

Table 6.3-3 - Command and ACCEPTED response frame of ALLOCATE_ATTACH..... 29

Table 6.3-4 - Command and REJECTED response for ALLOCATE_ATTACH 31

Table 6.3-5 - Command and INTERIM response frame of ALLOCATE_ATTACH_FRAME..... 32

Table 6.3-6 - Command and REJECTED response for ALLOCATE_ATTACH_FRAME..... 33

Table 6.3-7 - Command and ACCEPTED response frame of ATTACH..... 34

Table 6.3-8 - Command and REJECTED response for ATTACH 34

Table 6.3-9 - Command and INTERIM response frame of ATTACH_FRAME..... 35

Table 6.3-10 - Command and REJECTED response for ATTACH_FRAME..... 36

Table 6.3-11 - Command and ACCEPTED response frame of ADD_OVERLAY 36

Table 6.3-12 - Command and REJECTED response frame of ADD_OVERLAY 38

Table 6.3-13 - Command and ACCEPTED response frame of DETACH..... 39

Table 6.3-14 - Command and REJECTED response for DETACH 40

Table 6.3-15 - Command and ACCEPTED response frame of DETACH_RELEASE 40

Table 6.3-16 - Command and REJECTED response frame of DETACH_RELEASE 41

Table 6.3-17 - Command and ACCEPTED response frame of RELEASE..... 42

Table 6.3-18 - Command and REJECTED response frame of RELEASE..... 42

Table 6.3-19 - Command and ACCEPTED response frame of RESTORE_PORT for the producer port ... 44

Table 6.3-20 - Command and REJECTED response frame of RESTORE_PORT for the producer port 44

Table 6.3-21 - Command and ACCEPTED response frame of RESTORE_PORT for the consumer port .. 44

Table 6.3-22 - Command and REJECTED response frame of RESTORE_PORT for the consumer port ... 45

Table 6.3-23 - Command and response frame of RESTORE_PORT_FRAME for the producer port 47

Table 6.3-24 - Command and REJECTED response for RESTORE_PORT_FRAME for the producer port 48

Table 6.3-25 - Command and response frame of RESTORE_PORT_FRAME for the consumer port..... 49

Table 6.3-26 - Command and REJECTED response for RESTORE_PORT_FRAME for the consumer port 50

Table 6.3-27 - Command and ACCEPTED response frame of SUSPEND_PORT..... 51

Table 6.3-28 - Command and REJECTED response frame of SUSPEND_PORT 51

Table 6.3-29 - Command and ACCEPTED response frame of RESUME_PORT 52

Table 6.3-30 - Command and REJECTED response frame of RESUME_PORT 53

Table 6.4-1 - STATUS Command and STABLE response frame 54

Table 6.4-2 - Possible STATUS response from the producer port 55

Table 6.4-3 - Possible STATUS response from the consumer port 55

Table 7.1-1 - State machine code definitions 56

Table 7.1-2 - State machine code definitions (cont'd) 57



1. Overview

1.1 Scope

This document describes AV/C commands for management of asynchronous serial bus connections. An asynchronous data-transfer service efficiently transfers data between simple Serial Bus compliant devices. The scope and purpose of this activity are summarized below:

Scope. The definition of AV/C commands for management of asynchronous connections, where management includes establishing, breaking, and monitoring the connection, and the definition of existing AV/C commands extension for the support of asynchronous connections.

The AV/C management commands assume that the controller, producer, and consumer are attached to the same bus.

Purpose. To provide mechanisms for existing AV/C devices and controllers to easily manage asynchronous connection services.

1.2 Summary

Current AV equipment which supports IEC 61883 use Isochronous Channel for real-time data transmission. This document describes how "AV/C Compatible Asynchronous Serial Bus Connections" may be established using the AV/C command set for the non-real-time data transmission between the AV units. Although asynchronous connections are expected to be established using AV/C commands, other commands can also be used. The connection commands are generated by the controller and are sent to producer and consumer nodes. The controller may be an independent node, or may be collocated on the producer-plug or consumer-plug nodes.

An asynchronous connection is designed to survive bus resets. However, asynchronous communications may be temporarily disrupted until the controller has the opportunity to restore the connection. Restoration of the connection involves the distribution of new node Id values, since the node Id address of the connected port is sometimes affected by the bus reset.

2. References

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

- IEEE Std 1394-1995, Standard for a High Performance Serial Bus.¹
- IEC-61883, Consumer audio/video equipment - Digital interface - Part 1: General.²
- ISO/IEC 13213: 1994 [ANSI/IEEE Std 1212, 1994 Edition], Information Technology—Microprocessor systems—Control and Status Register (CSR) Architecture for Microcomputer Buses.³
- AV/C Digital Interface Command Set General Specification, Version 3.0, April 15 1998⁴
- 1394 Open Host Controller Interface Specification (release 1.00), October 20, 1997⁵
- AV/C Compatible Asynchronous Serial Bus Connections, Version 1.0FC2, January 11, 1999⁶

-
- 1) IEEE publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.
 - 2) Document 100C/182/FDIS from: U.S. National Committee of The IEC ANSI, 11, West 42nd Street, 13th Floor, New York, NY 10036. Phone: (212)642-4900(Questions) (212)642-4980 (Sales). FAX: (212)398-0023
 - 3) ISO/IEC [ANSI/IEEE] publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA or from International Electrotechnical Commission, 3 rue de arembé, Case Postale 131, CH 1211, Genève 20, Switzerland/Suisse.
 - 4) Approved 1394TA documents may be ordered through: 1394 Trade Association, Regency Plaza Suite 350, 2350 Mission College Blvd., Santa Clara, CA 95054, USA; by contacting taadmin@1394TA.org, or by fetching a pdf copy from the 1394TA web site: <http://www.1394TA.org/abouttech/specifications/techspec.html>
 - 5) A "pdf" copy of this specification may be fetched from: <ftp://ftp.austin.ibm.com/pub/chrptech/1394ohci/>
 - 6) Approved 1394TA documents may be ordered through: 1394 Trade Association, Regency Plaza Suite 350, 2350 Mission College Blvd., Santa Clara, CA 95054, USA; by contacting taadmin@1394TA.org, or by fetching a pdf copy from the 1394TA web site: <http://www.1394TA.org/abouttech/specifications/techspec.html>

3. Definitions and abbreviations

3.1 Conformance glossary

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

expected:	A keyword used to describe the behavior of the hardware or software in the design models assumed by this specification. Other hardware and software design models may also be implemented.
may:	A keyword that indicates flexibility of choice with no implied preference.
shall:	A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products conforming to this specification.
should:	A keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase "is recommended."

3.2 Technical glossary

asynchronous connection:

A logical point-to-point communication path established between producer and consumer ports, that supports robust high-bandwidth flow-controlled transfers of one or more data frames.

asynchronous-connection consumer (abbreviated as **consumer**):

The node that consumes data frames provided by the asynchronous-connection producer.

asynchronous-connection producer (abbreviated as **producer**):

The node that produces data frames for consumption by the asynchronous-connection consumer.

asynchronous plug:

A collection of externally visible components (called ports) that can be connected to a subunit for the purposes of sending sequences of variable-length frames.

APR:

Asynchronous connection Port Register. One asynchronous plug may have one *iAPR* or more than one up-to 14 *oAPR*. An asynchronous connection is established between *iAPR* on the consumer and *oAPR* on the producer.

input Asynchronous Port Register (abbreviated as ***iAPR***):

A consumer-resident register affiliated with a consumer port, that is updated by the producer to indicate how much of data has been written to the segment buffer. This register also has other bits for data flow control purpose.

output Asynchronous Port Register (abbreviated as ***oAPR***):

A producer-resident register affiliated with a producer port on a plug, that is updated by the consumer to indicate how much data can be safely written by the producer. This register also has other bits for data flow control purpose.

AV unit:

The physical instantiation of a consumer electronic device, *e.g.*, a camcorder or a VCR, within a Serial Bus node.

AV subunit:

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

AV/C:

Audio/video control, as in the AV/C Digital Interface Command Set

byte:

Eight bits of data, used as a synonym for octet.

consumer:

(see asynchronous connection consumer).

CompareSwap4:

A bus transaction that conditionally stores a *next* value to a specified address and returns the previous data value from that address. The store occurs when the addressed

- memory value and a second *test* value are equal. In the CSR Architecture, this is called a 4-byte compare_swap transaction.
- consumer port:** A port that is the sink of data frames and is flow controlled by updates of its externally visible *iAPR*.
- CSR Architecture:** A abbreviation of the following reference: ISO/IEC 13213 : 1994 [ANSI/IEEE Std 1212, 1994 Edition], Information Technology—Microprocessor systems—Control and Status Register (CSR) Architecture for Microcomputer Buses.
- data frame** (abbreviated as **frame**):
A contiguous group of data bytes sent between producer and consumer.
- data segment** (abbreviated as **segment**):
A largest portion of a data frame that can be written into the segment buffer before updating the consumer's *iAPR* register.
- EUI-64:** Extended Unique Identifier, 64-bits, as defined by the IEEE. The EUI-64 is a concatenation of the 24-bit company_ID obtained from the IEEE Registration Authority Committee (RAC) and a 40-bit number (typically a silicon serial number) that the vendor identified by company_ID guarantees to be unique for all of its products. The EUI-64 is also known as the node unique ID and is redundantly present in a node's configuration ROM in both the Bus_Info_Block and the Node_Unique_Id leaf.
- FCP:** Function Control Protocol, as defined by IEC 61883.
- frame:** (see data frame).
- IEEE:** The Institute of Electrical and Electronics Engineers, Inc.
- isochronous:** A term that indicates the essential characteristic of a time-scale or signal, such that the time intervals between consecutive instances either have the same duration or durations that are integral multiples of the shortest duration. In the context of Serial Bus, "isochronous" is taken to mean a bounded worst-case latency for the transmission of data; physical and logical constraints that introduce jitter preclude the exact definition of "isochronous."
- module:** The smallest component of physical management, *i.e.*, a replaceable device.
- node:** An addressable device attached to Serial Bus with at least the minimum set of control registers defined by IEEE Std 1394-1995.
- node ID:** A 16-bit number, unique within the context of an interconnected group of Serial Buses. The node ID is used to identify both the source and destination of Serial Bus asynchronous data packets. It can identify one single device within the addressable group of Serial Buses (unicast), or it can identify all devices (broadcast).
- plug:** A physical or virtual end-point of connection implemented by an AV unit or subunit that may receive or transmit isochronous or other data. Plugs may be Serial Bus isochronous plugs, accessible through the PCR's; they may be asynchronous plugs; they may be external, physical plugs on the AV unit; or they may be internal virtual plugs implemented by the AV subunits.
- PCR:** Plug Control Register, as defined by IEC 61883, Digital Interface for Consumer Electronic Audio/Video Equipment.
- iPCR:** Input PCR, as defined by IEC 61883.
- oPCR:** Output PCR, as defined by IEC 61883.
- port:** A subcomponent of a plug that supports unidirectional data transfers.
- producer:** (see asynchronous connection producer).
- producer port:** A port that is the source of data frames and is flow controlled by updates of its externally visible *oAPR*.
- quadlet:** Four bytes of data.
- segment:** (see data segment).
- segment buffer:** An externally visible address space on a consumer into which data is written by the connected producer.
- Serial Bus:** The physical interconnects and higher level protocols for the peer-to-peer transport of serial data, as defined by IEEE Std 1394-1995.
- quadlet:** Four bytes of data.
- unit architecture:** The formal specification of the format and function of the software-visible resources

and behaviors of a class of units. This document, in conjunction with the references above, defines a unit architecture for the class of AV devices.

3.3 Size notation

The Serial Bus description avoids the confusing terms half-word, word, and double-word, which have widely different definitions depending on the processor’s word size. In their place the Serial Bus description uses terms established in previous IEEE bus standards, which are independent of the processor. These terms are illustrated in Table 3.3.1.

Table 3.3-1 - Size notation examples

Size in bits	16-bit word notation	32-bit word notation	IEEE Std. notation (used in this standard)
8	Byte	Byte	Byte
16	Word	Half-word	Doublet
32	Long-word	Word	Quadlet
64	Quad-word	Double	Octlet

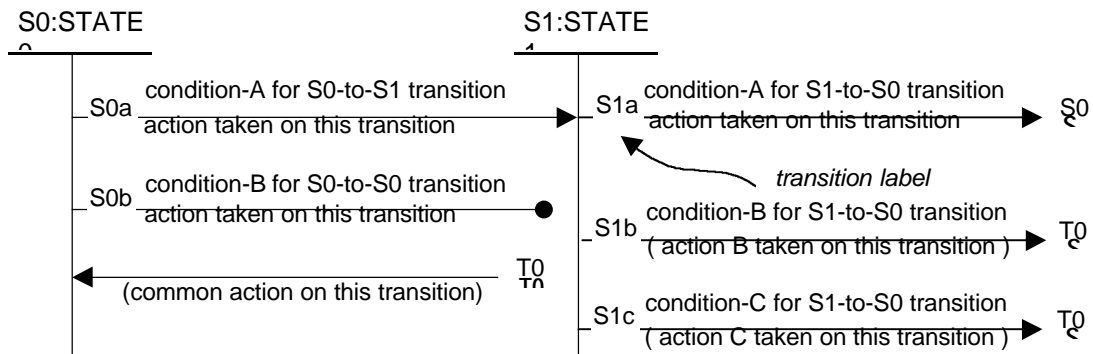
3.4 Numerical notation

Decimal, hexadecimal, and binary numbers are used within this document. For clarity, decimal numbers are generally used to represent counts, hexadecimal numbers are used to represent addresses, and binary numbers are used to describe bit patterns within binary fields.

Decimal numbers are represented in their usual 0, 1, 2, ... format. Hexadecimal numbers are represented by a string of one or more hexadecimal (0-9,A-F) digits followed by the subscript 16, except in C++ code contexts, where they are written as 0x123EF2 etc. Binary numbers are represented by a string of one or more binary (0,1) digits, followed by the subscript 2. Thus the decimal number “26” may also be represented as “1A₁₆” or “11010₂”.

3.5 State machine notation

All state machines in this standard use the style shown in Figure 3.5.1. This is similar to the notation used in the Serial Bus standard, with modifications to more compactly and consistently illustrates state transition actions. To illustrative the functionality of transition-destination labels, the equivalent state machines without transition-destination labels is also illustrated in Figure 3.5.2.



Notes:
When appropriate, this note may specify that remaining state machine states are located and specified in other figures

Figure 3.5.1- State machine notation

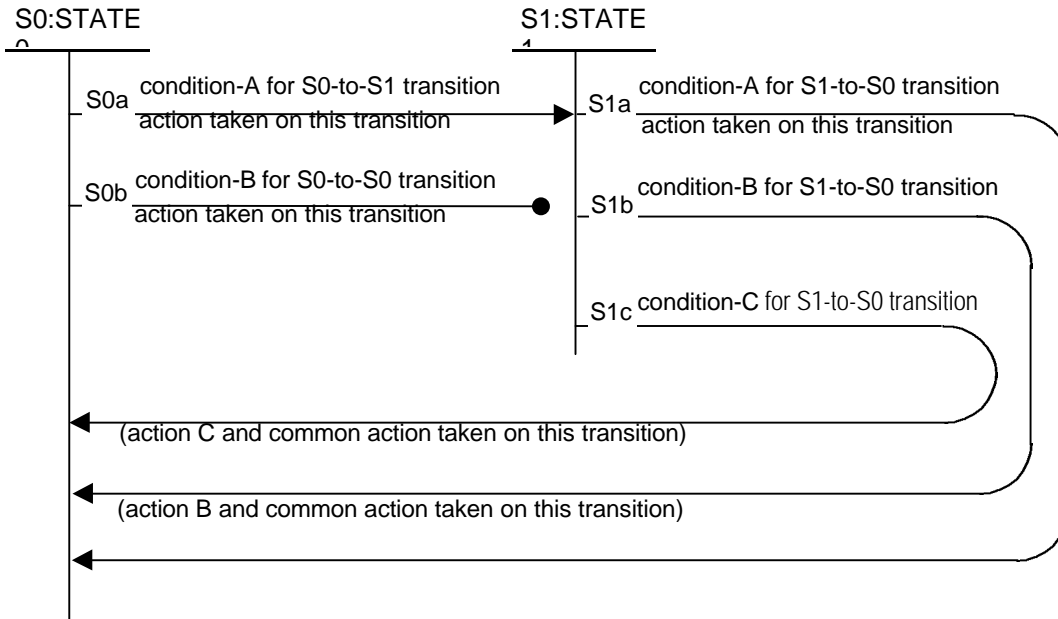


Figure 3.5.2 - Equivalent state machine

3.6 C++ code notation

The conditions and actions of the state machines are formally defined by C++ code. Since many C++ code operators are non-obvious to the casual reader, their meanings are summarized in table 3.6-1.

Table 3.6-1 - Specific expression summary

Expression	Description
$\sim I$	Bitwise complement of integer I
$++I$	Pre-increment of integer I (I is incremented, then used in the expression)
$--I$	Pre-decrement of integer I (I is decremented, then used in the expression)
$I \wedge J$	Bitwise XOR of integers I and J
$I \& J$	Bitwise AND of integer values I and J
$I J$	Bitwise OR of integer values I and J
$I \ll J$	Value of I, shifted left by J bits, zero fill
$I \gg J$	Value of I, shifted right by J bits, zero fill if I is an unsigned number, sign extension if I is signed
$I == J$	Equality test, true if I is equal to J
$I != J$	Inequality test, true if I is not equal to J
$!B$	Logical negation of boolean variable B
$A \&\& B$	Logical AND of boolean values A and B
$A B$	Logical OR of boolean values A and B

In addition, the following nonstandard data types (actually, object classes) are supported:

Table 3.6-2 - Serial Bus data types

Data type	Description
timer	real value (units of seconds) that autonomously increments at a defined rate
boolean	One bit value where 1 is true and 0 is false

Other, more specific data types are defined in the clauses where they are relevant.

All C++ code is executed as if it takes zero time. Time only elapses when the following function is called:

```
void wait (real time); // wait for "time" to elapse
```

3.7 Unimplemented locations

The capabilities of reserved, ignored, and unused values are exactly defined, to minimize conflicts between current implementations and future definitions. These notations shall apply to all fields unless otherwise noted.

3.7.1 Reserved locations

Some locations or portions of locations are not implemented and are defined to be *reserved* (abbreviated as *res* or *r*). When a reserved value is written, a zero values shall be provided; when read, the returned value shall be ignored.

3.7.2 Ignored locations

Selected locations or portions of locations are partially implemented and are defined to be *ignored* (abbreviated as *ign* or *i*). An ignored values has affiliated storage elements, but the values in these storage elements has no side effect. When an *ignored* value is written, a zero value shall be provided; when read, the returned value shall be ignored.

3.7.3 Unused locations

Selected locations or portions of locations may be not implemented or partially implemented and are defined to be either *unused* (abbreviated as *un* or *u*). For *unused* locations, the selection between *reserved* and *ignored* behaviors is implementation dependent.

4. Extension for existing AV/C commands

This chapter describes how the existing AV/C commands are extended to support asynchronous connections. The asynchronous connections and asynchronous plugs are handled according to existing AV/C unit architecture, like isochronous connections and isochronous plugs.

4.1 Plug numbers for asynchronous plug definition

An AV Unit, which supports asynchronous connections, shall have more than one asynchronous plug. To support asynchronous plug, existing plug definitions are extended. Table 9.2-1 of [4] is changed, as described by Table 4.1-1 below.

Table 4.1-1 - Serial Bus and external plug numbers

value	source plug	destination plug
0 – 1E ₁₆	Serial Bus iPCR[0] – iPCR[30]	Serial Bus oPCR[0] – oPCR[30]
1F ₁₆ – 7E ₁₆	Reserved for future specification	Reserved for future specification
7F ₁₆	Any available Serial Bus plug iPCR[x]	Any available Serial Bus plug oPCR[x]
80 ₁₆ – 9E ₁₆	External input plug 0 – 30	External output plug 0 – 30
9F ₁₆	Reserved for future specification	Reserved for future specification
A0₁₆ – BE₁₆	Serial Bus Asynchronous input plug [0] – [30]	Serial Bus Asynchronous output plug [0] – [30]
BF₁₆	Any available Serial Bus Asynchronous input plug [x]	Any available Serial Bus Asynchronous output plug [x]
C0 ₁₆ – FC ₁₆	Reserved for future specification	Reserved for future specification
FD ₁₆	Reserved for future specification	Multiple plugs
FE ₁₆	Invalid	Invalid
FF ₁₆	Any available external input plug	Any available external output plug

Table 4.1-2 below shows the list of AV/C commands, in which these new defined values may be used.

Table 4.1-2 - Existing AV/C commands which handles asynchronous plug

opcode	value
CONNECT	24 ₁₆
CONNECTIONS	22 ₁₆
DISCONNECT	25 ₁₆

4.2 Extended command for asynchronous plug support

PLUG INFO status command defined in 10.8 of [4], is extended to support asynchronous plugs. The extended format of PLUG INFO status command is illustrated by Figure 4.2.1 below.

	msb						lsb
opcode	PLUG INFO (02 ₁₆)						
operand[0]	subfunction						
operand[1]	subfunction dependent fields						
operand[2]							
operand[3]							
operand[4]							

Figure 4.2.1 - Extended PLUG INFO status command generic format

When subfunction field is set to 00₁₆, PLUG INFO status command is all the same as defined in [4], which have operand[0] ~ operand[4]. When subfunction field is set to 01₁₆, PLUG INFO status command format is extended as illustrated by Figure 4.2.2 below.

	msb						lsb
opcode	PLUG INFO (02 ₁₆)						
operand[0]	01 ₁₆						
operand[1]	FF ₁₆						
operand[2]							
operand[3]							
operand[4]							

Figure 4.2.2 - Extended PLUG INFO status command format

If the PLUG INFO status command with subfunction set to 01₁₆ was addressed to an AV unit, the response frame returned is illustrated by Figure 4.2.4 below.

	msb						lsb
opcode	PLUG INFO (02 ₁₆)						
operand[0]	01 ₁₆						
operand[1]	Serial_Bus_asynchronous_input_plugs						
operand[2]	Serial_Bus_asynchronous_output_plugs						
operand[3]	FF ₁₆						
operand[4]							

Figure 4.2.3 - Extended PLUG INFO status response format from an AV unit

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

5. AV/C connection sequences

This chapter describes the overview of asynchronous connections management. Since AV/C command that manages the asynchronous connections has only one common frame format, the subfunction field value specifies the action to request the target. In this chapter, these subfunctions are described just as “command”. The detail of AV/C management command (ASYNCHRONOUS CONNECTION) is described in Chapter 6.

5.1 Establishing an asynchronous connection

An asynchronous connection is established by a controller, which sends an ALLOCATE command to the consumer node, as illustrated by Figure 5.1.1. The consumer port is effectively locked from other changes, including connection overlays, between the processing of the initial ALLOCATE and final ATTATCH command.

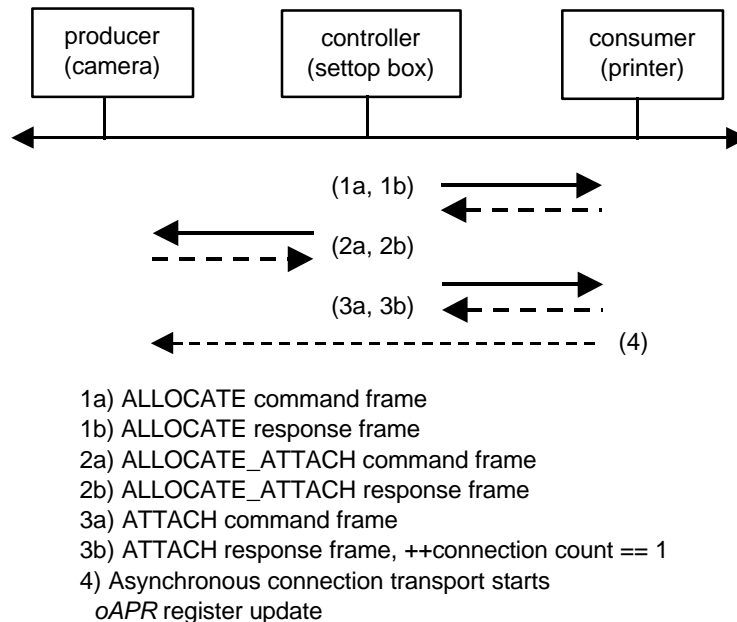


Figure 5.1.1 - Controller managed connection

The ALLOCATE command (1a,1b) allocates the port resources and returns the address of the consumer port to the controller. The following ALLOCATE_ATTACH (2a,2b) command is responsible for allocating and connecting the producer port resources; the final ATTACH (3a,3b) command is responsible for connecting the consumer port, resulting its connection count value incremented to be one. The producer port remains inactive until the consumer updates the producer-resident oAPR register (4), thereby activating the producer port.

NOTE: There is timeout constraints between (1b)~(3a) and (2b)~(4) to avoid unnecessary idle state. If the next request did not observed within 5 seconds, the port detects timeout. See Chapter 7 for detail.

5.2 Breaking an asynchronous connection

The disconnection of asynchronous ports is initiated by a controller, which sends a DETATCH command

to the consumer node, as illustrated by Figure 5.2.1.

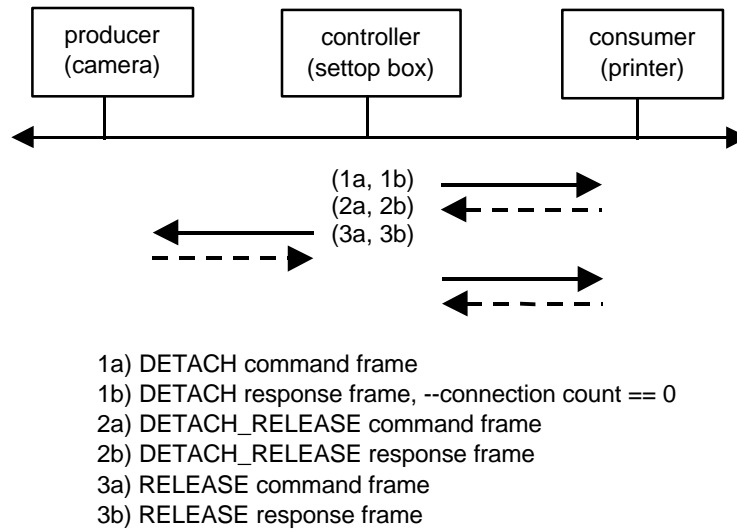


Figure 5.2.1 - Connection break sequence

The DETACH command (1a,1b) decrements its connection count value. If the connection count value which is returned by the DETACH response (1b) is zero, it indicates that the consumer entered the successful disconnection sequence. If the returned connection count value is not zero, it indicates that the consumer port still have more than one connections. In such a case, the controller should not issue the DETACH_RELEASE command (2a,2b) to the producer, which would break the connection. For detailed description of this case, please refer to Section 5.5.

If the connection count value is zero, the consumer-port resource is left in a inactive state. While inactive, the consumer-plug resource accepts register updates and segment-buffer writes from the producer, while inhibiting the generation of producer-port updates and segment-buffer writes.

The following DETACH_RELEASE command (2a,2b) disconnects the producer-port. If the producer holds multicast connections, and if the disconnected producer-port is also the last-connected producer-plug port, the producer-plug resources is also released. The final RELEASE command (3a,3b) disconnects the consumer-port and releases its resources.

NOTE: There is timeout constraints between (1b)~(3a) to avoid unnecessary idle state. If the next request did not observed within 5 seconds, the port detects timeout. See Chapter 7 for detail.

5.3 Failure of establishing an asynchronous connection

The controller’s connection sequence may fail if the affiliated producer-port connection attempt is rejected. In this case, the controller is responsible for releasing the prepared consumer-port resource, as illustrated in Figure 5.3.1.

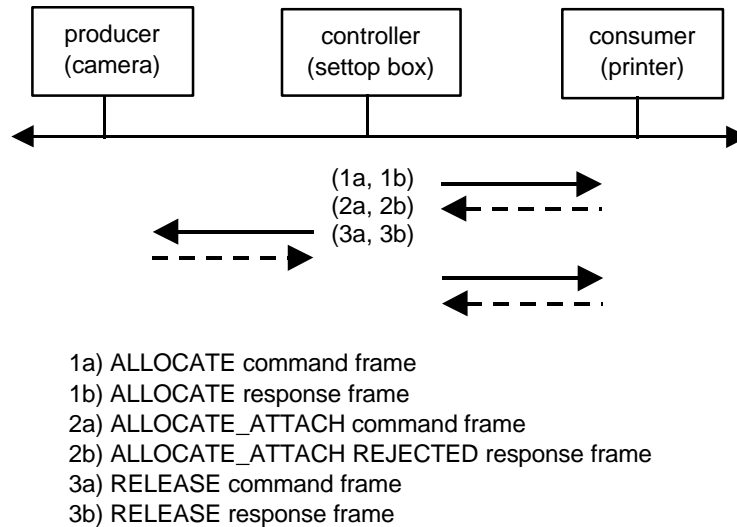


Figure 5.3.1 - Producer-plug connection failure

NOTE: There is timeout constraints between (1b)~(3a) to avoid unnecessary idle state. If the next request did not observed within 5 seconds, the port detects timeout. See Chapter 7 for detail.

5.4 Overlaying an asynchronous connection

An asynchronous connection supports overlay of existing connection as optional, because existing isochronous connections defined in [2] support overlay, and an asynchronous connection may be used with isochronous together. The overlaid connection can not be broken while the connection count value is greater than one because there would be other controllers who are working with this connection. After one connection has been established, the existing connection can be overlaid by the same controller or other controllers, as illustrated by Figure 5.4.1.

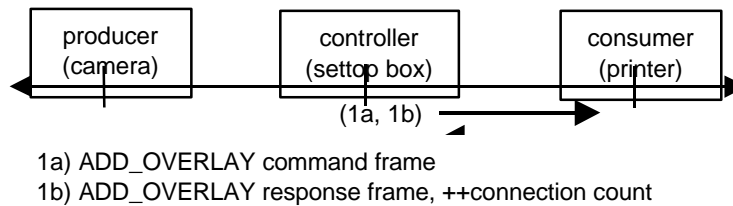


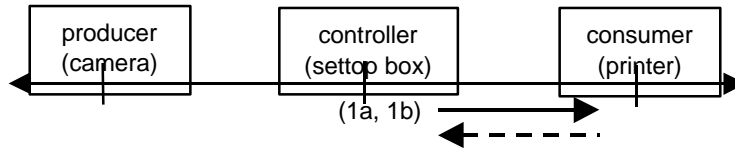
Figure 5.4.1 - Connecting an overlaid plug

Note that connection count information is the property of the consumer port, the overlay command shall be issued to the consumer only. When overlaying the existing connection, the following conditions shall be met at the consumer port:

- 1) Same producer. The node ID of the to-be-connected producer is the same as the node ID of the currently connected producer.
- 2) Same plug and port. The plug ID and port ID on the to-be-connected producer matches the plug ID and port ID of the currently connected producer.

5.5 Breaking an overlaid connection

When more than one overlaid connection has been established, the disconnection of overlaid port using the same command as normal disconnect operation is more efficient, as illustrated by Figure 5.5.2.



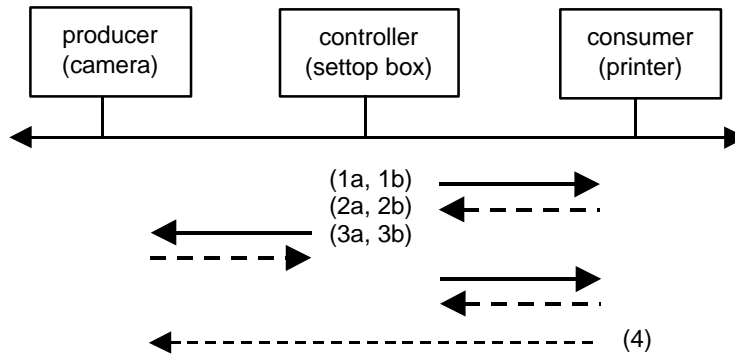
- 1a) DETACH command frame
- 1b) DETACH response frame, --connection count != 0

Figure 5.5.1 - Disconnecting an overlaid plug

If the DETACH response frame (1b) includes the connection count field set to zero, this indicates that the consumer has no connection. In such a case, the controller is responsible to break the connection by issuing the DETATCH_RELEASE command and RELEASE command, as described “5.2 Breaking an asynchronous connection”.

5.6 Establish multicast connections

As asynchronous connection supports optional multicast connections capability, one producer plug can be connected up-to 14 consumer nodes using 14 ports inside. After one connection has been established, the controller can establish the connection between the current producer plug (with different port) and the other consumer port (if the producer plug supports the multicast connections), as illustrated by Figure 5.6.1.



- 1a) ALLOCATE command frame
- 1b) ALLOCATE response frame
- 2a) ALLOCATE_ATTACH command frame
- 2b) ALLOCATE_ATTACH response frame
- 3a) ATTACH command frame
- 3b) ATTACH response frame
- 4) Asynchronous connection transport starts
oAPR register update

Figure 5.6.1 - Establishing multicast connections

Multicast connection can be established if the following conditions are met when the

ALLOCATE_ATTACH control command frame (2a) is processed at the producer port:

- 1) Same producer. The to-be-connected producer is the same as currently connected producer.
- 2) Same plug. The plug ID on the to-be-connected producer matches the plugId of the currently connected producer.
- 3) Different port. The port ID on the to-be-connected producer shall be different from the port ID of currently connected producer. The controller may specify the port by using “any available port” value in the ALLOCATE_ATTACH (2a) command frame to get the available port ID value, which may be assigned by the producer, returned by the response frame.
- 4) Different consumer plug. The plug ID of the to-be-connected consumer is different from any of the already-connected consumers.

NOTE: There is timeout constraints between (1b)~(3a) and (2b)~(4) to avoid unnecessary idle state. If the next request did not observed within 5 seconds, the port detects timeout. See Chapter 7 for detail.

5.7 Breaking multicast connections

When the multicast connections had been established, one connection of the multicast connections can be broken as illustrated by Figure 5.7.1.

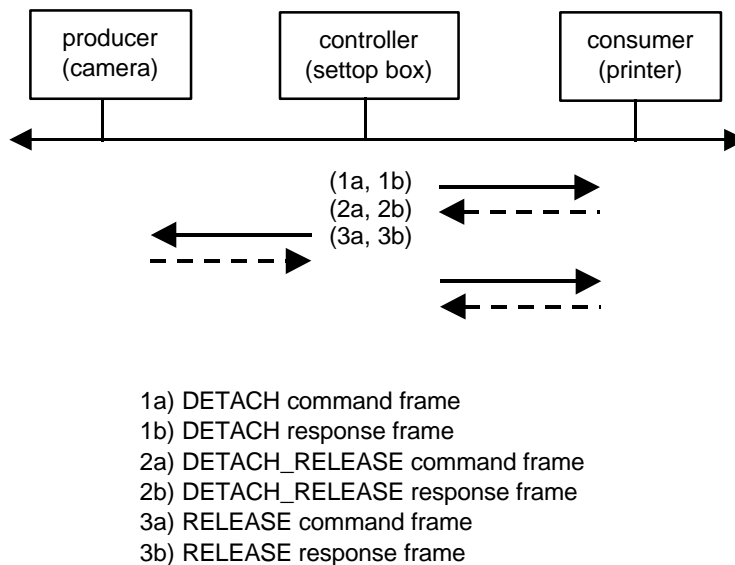


Figure 5.7.1 - Breaking multicast connections

Note that DETACH_RELEASE (2a,2b) command frame should specify the producer port ID to which the consumer port is connected. If the more than one connection had been established on the plug, the DETACH_RELEASE (2a,2b) command only breaks the specified connection. On the other hand, the other connections, which the producer plug holds, would be still remaining.

NOTE: There is timeout constraints between (1b)~(3a) to avoid unnecessary idle state. If the next request did not observed within 5 seconds, the port detects timeout. See Chapter 7 for detail.

5.8 Reconnect procedures after bus reset

5.8.1 AV/C reconnect timing

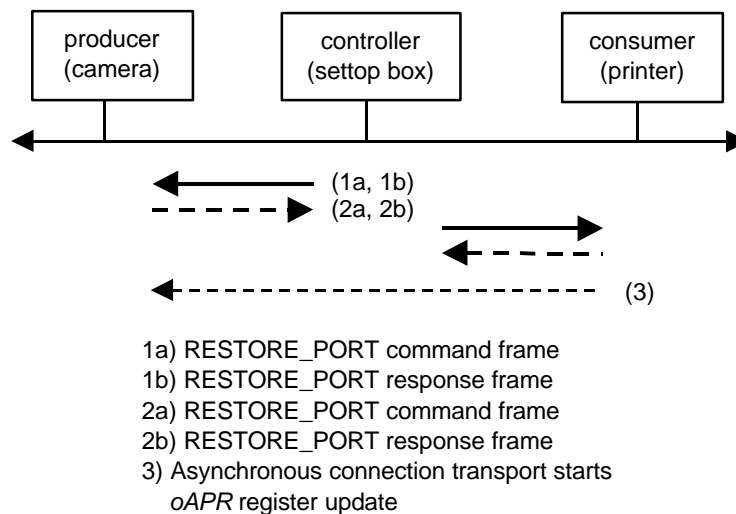
Asynchronous connections are affected by bus resets, since the nodeID portions of their connected ports may have changed. The controller is responsible for resuming the connections (providing each port with a revised nodeID of the other) after the bus reset.

To keep the consistency with RESERVE control command, the resumption of asynchronous connections shall complete within 10 seconds by the controller that established the connection. A previously active producer or consumer rejects all connection and disconnection commands during these first 10 seconds, accepting only its expected resumption commands. After the 10 seconds delay, resumption commands are rejected, because the targets had been turned into their initial states; the connection and disconnection commands are accepted in the normal fashion. A previously inactive producer or consumer accepts connection and disconnection commands, with no distinction between the commands that are received in the first 10 seconds and those commands that follow.

5.8.2 AV/C reconnect commands

The RESTORE_PORT command is used to re-establish the connection. The use of distinct reconnection commands allows reconnections and new connections to proceed concurrently, while providing the consumer port with sufficient information to distinguish between the two operations, as illustrated by Figure 5.8.1. A new asynchronous connection in the first 10 seconds is also allowed, as described in section 5.1.

Figure 5.8.1 - Reconnecting asynchronous plugs



NOTE: There is timeout constraints between (1b)~(3) to avoid unnecessary idle state. If the next request did not observed within 5 seconds, the port detects timeout. See Chapter 7 for detail.

5.9 Automatic disconnection

The controller can direct the producer node to break the connection automatically after a single frame had been transmitted, using the ALLOCATE_ATTACH_FRAME subfunction. If this subfunction is used in the command frame, the target shall return the INTERIM

response, followed by the ACCEPTED or REJECTED response after a single frame transmission.

A controller can request the automatic disconnection to the producer node, as illustrated by Figure 5.9.1 below.

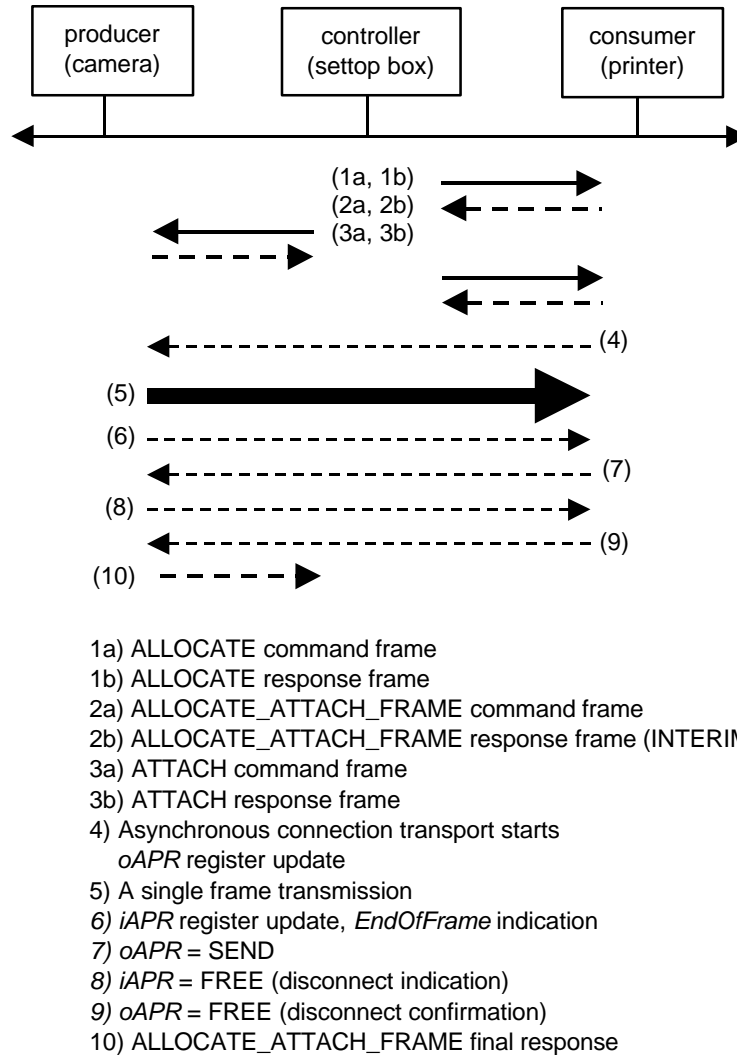


Figure 5.9.1 - Producer initiated disconnection

The ALLOCATE command (1a, 1b) allocates the port resources and returns the address of the consumer port to the controller. The following ALLOCATE_ATTACH_FRAME (2a,2b) command is responsible for allocating and connecting the producer port resources; the final ATTACH (3a,3b) command is responsible for connecting the consumer port. The port remains inactive until the consumer updates the producer-resident *oAPR* register (4), thereby activating the producer port.

When the subfunction of ALLOCATE_ATTACH_FRAME (2a), the target (producer) shall return the INTERIM response frame (2b), which shall include the port information (ex. address of the producer port, port bits, and so on). The controller shall use these information when the following ATTACH command (3a) is issued. The ATTACH command (3a,3b) connects the consumer port to the producer port, resulting the start of asynchronous connection transport by updating the *oAPR* register (4).

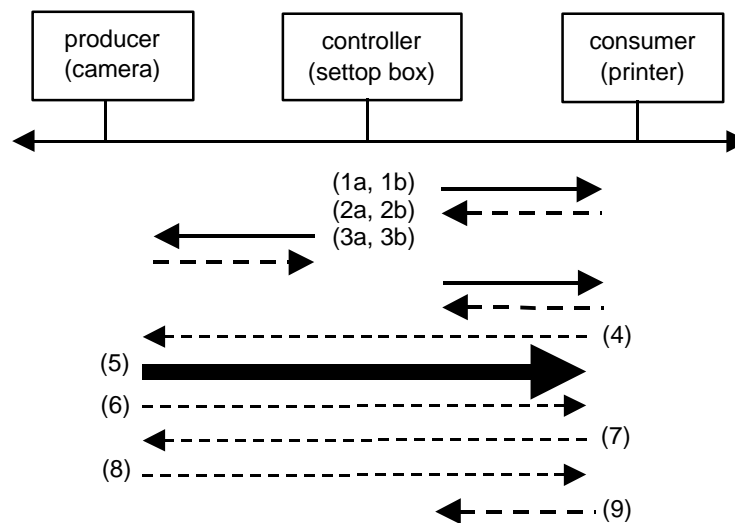
After a single frame data had been transmitted (5), the producer indicates the *EndOfFrame* by updating the consumer-resident *iAPR* register's **mode** value to LAST, LESS, JUNK, TOSS or LOST (6). Then the consumer requests the producer to send next frame by updating the producer-resident *oAPR* register with *mode* value set to SEND (7).

As the producer had been requested to disconnect automatically, the producer requests disconnection by updating the *iAPR.mode* with FREE value (8), then the consumer confirms disconnection by updating the *oAPR.mode* with FREE value (9).

After successful disconnection (*iAPR* register's **mode** value was LAST), the producer returns ACCEPTED response for ALLOCATE_ATTACH_FRAME command (10). If error condition occurred during a single frame transmission (*iAPR* register's **mode** value was LESS, JUNK, TOSS or LOST), and the connection might have been broken), the producer returns REJECTED response for ALLOCATE_ATTACH_FRAME command (10) with error code value stored in *status* field.

NOTE: There is timeout constraints between (1b)~(3a) and (2b)~(4) to avoid unnecessary idle state. If the next request did not observed within 5 seconds, the port detects timeout. See Chapter 7 for detail.

And a controller can request the automatic disconnection to the consumer node, as illustrated by Figure 5.9. 2 below.



- 1a) ALLOCATE command frame
- 1b) ALLOCATE response frame
- 2a) ALLOCATE_ATTACH command frame
- 2b) ALLOCATE_ATTACH response frame
- 3a) ATTACH_FRAME command frame
- 3b) ATTACH_FRAME response frame (INTERIM)
- 4) Asynchronous connection transport starts
oAPR register update
- 5) A single frame transmission
- 6) *iAPR* register update, *EndOfFrame* indication
- 7) *oAPR* = FREE (disconnect indication)
- 8) *iAPR* = FREE (disconnect confirmation)
- 9) ATTACH_FRAME final response

Figure 5.9.2 - Consumer initiated disconnection

The ALLOCATE command (1a, 1b) allocates the port resources and returns the address of the consumer port to the controller. The following ALLOCATE_ATTACH (2a,2b) command is responsible for allocating and connecting the producer port resources; the final ATTACH_FRAME (3a,3b) command is responsible for connecting the consumer port. The consumer returns the INTERIM response frame (3b) to the controller, then start an asynchronous connection transport by updating the *oAPR* register (4). The producer port remains inactive until the consumer updates the producer-resident *oAPR* register, thereby activating the producer port.

After a single frame data had been transmitted (5), the producer indicates the *EndOfFrame* by updating the consumer-resident *iAPR* register's **mode** value to LAST, LESS, JUNK, TOSS or LOST (6).

As the consumer had been requested to disconnect automatically, the consumer requests disconnection by updating the *oAPR.mode* with FREE value (7), then the producer confirms disconnection by updating the *iAPR.mode* with FREE value (8).

After successful disconnection(*iAPR* register's **mode** value was LAST), the consumer returns the ACCEPTED response for ATTACH_FRAME command (9). If error condition occurred during a single frame transmission (*iAPR* register's **mode** value was LESS, JUNK, TOSS or LOST), and the connection might have been broken), the consumer returns the REJECTED response for ATTACH_FRAME command with error code value stored in *status* field (9).

NOTE: There is timeout constraints between (1b)~(3a) and (2b)~(4) to avoid unnecessary idle state. If the next request did not observed within 5 seconds, the port detects timeout. See Chapter 7 for detail.

6. Asynchronous Connection management command

6.1 Overview

A controller establishes an Asynchronous Connection between the Producer Node and the Consumer Node by using the AV/C commands. After the connection had been established, asynchronous data frames are going to be transmitted between the nodes.

Although the AV/C command which is addressed to the AV subunit might be the trigger to start the transmission, this kind of definition is subunit design dependent issue and is out of scope of this document. The AV/C command, which is used for management of asynchronous connections, has only one opcode and common frame format. Support level of asynchronous connection command is defined in Table 6.1-1 below.

Note that an AV unit can have asynchronous plugs, this management command is a unit command. A target node which has Asynchronous Plug shall support ASYNCHRONOUS CONNECTION command according to the following table.

Table 6.1-1 - Support level of asynchronous connection management command

Opcode	Value	Support level (by ctype)			Comments
		C	S	N	
ASYNCHRONOUS CONNECTION	26 ₁₆	*	M	-	Manage asynchronous connections

NOTE: * The support level of ASYNCHRONOUS CONNECTION control command is depned on its subfunctions and described in Section 6.2.

6.2 Common frame format

The common frame format for ASYNCHRONOUS CONNECTION command (for *ctype* of both CONTROL and STATUS) and response, which provides the Asynchronous Connection and Asynchronous Plug management functionality, is illustrated by the Figure 6.2.1 below.

	msb					lsb	
opcode	ASYNCHRONOUS CONNECTION (26_{16})						
operand[0]	subfunction						
operand[1]	status						
operand[2]	plug id						
operand[3]	(msb)	plug offset					
operand[4]							
operand[5]							
operand[6]							
operand[7]							
operand[8]	(lsb)	port id			port bits		
operand[9]	connected node ID						
operand[10]	connected node ID						
operand[11]	(msb)	connected plug offset					
operand[12]							
operand[13]							
operand[14]							
operand[15]							
operand[16]	(lsb)	connected port ID			connected port bits		
operand[17]	connected plug ID						
operand[18]	ex	res	connection count				
operand[19]	write interval			retry count			
operand[20]	reserved						

Figure 6.2.1 - ASYNCHRONOUS CONNECTION common frame format

subfunction

subfunction field indicates the action to be taken by the target in the command frame, whose *ctype* field is CONTROL, as follows.

Table 6.2-1 - subfunction field definitions

Symbol	value	meaning
ALLOCATE	01 ₁₆	Allocate the consumer port resource
ATTACH	02 ₁₆	Connect the consumer port to the producer port
ALLOCATE_ATTACH	03 ₁₆	Allocate the producer port resource and connect it to the consumer port
RELEASE	05 ₁₆	Release the port resource
DETACH	06 ₁₆	Disconnect the consumer port
DETACH_RELEASE	07 ₁₆	Disconnect and release the producer port resource
ADD_OVERLAY	0A ₁₆	Add the overlaid connection to the consumer port
SUSPEND_PORT	10 ₁₆	Suspend the consumer port
RESUME_PORT	20 ₁₆	Resume the consumer port
RESTORE_PORT	40 ₁₆	Restore the connection after a bus reset
ATTACH_FRAME	82 ₁₆	Connect the consumer port to the producer port and disconnect it after the transmission of a frame
ALLOCATE_ATTACH_FRAME	83 ₁₆	Allocate the producer port resource and connect it to the consumer port and disconnect and release it after the transmission of a frame
RESTORE_PORT_FRAME	C0 ₁₆	Restore the port resource that is allocated by the ALLOCATE_ATTACH_FRAME or ATTACH_FRAME subfunction command after a bus reset
-	other values	reserved

If the *ctype* field of the command frame is STATUS, this field shall be set to FF₁₆.

When value of subfunction is ATTACH_FRAME, ALLOCATE_ATTACH_FRAME, or RESTORE_PORT_FRAME, INTERIM response can be used as a first response, and for other subfunctions, INTERIM response should not be used.

As the MSB of *subfunction* field indicates the “disconnect request”, the subfunction can be described as follows:

```
#define dr_flag 0x80 // disconnect request bit.
// if set to one, disconnect after a
// single frame transmission.

#define ATTACH_FRAME dr_flag|ATTACH
#define ALLOCATE_ATTACH_FRAME dr_flag|ALLOCATE_ATTACH
#define RESTORE_PORT_FRAME dr_flag|RESTORE_PORT
```

The detailed descriptions of these subfunctions are described in the following sections.

The following Table 6.2-2 shows the support level of these subfunctions. In Table 6.2-2, “M” means “Mandatory”, “O” means “Optional”, and “-” means “Not used”.

Table 6.2-2 - Support level of subfunctions

symbol	producer	consumer
ALLOCATE	-	M
ATTACH	-	M
ALLOCATE_ATTACH	M	-
RELEASE	-	M
DETACH	-	M
DETACH_RELEASE	M	-
ADD_OVERLAY	-	O
SUSPEND_PORT	-	O
RESUME_PORT	-	O
RESTORE_PORT	M	M
ATTACH_FRAME	-	O
ALLOCATE_ATTACH_FRAME	O	-
RESTORE_PORT_FRAME	O	O

status

The **status** field indicates the states of the asynchronous port or the result of command execution. The states of the asynchronous port are defined in Chapter 7.

In case of the CONTROL command, this field indicates the result of command execution or the status of asynchronous port. Table 6.2-3 below shows the status field values of the response frame in case of CONTROL command.

Table 6.2-3 - status field values for CONTROL command response frame

value	symbol	response code of AV/C frame	meaning
01 ₁₆	FREE	ACCEPTED	the specified port is in a FREE state
02 ₁₆	FIXED	ACCEPTED	the specified port is in a FIXED state
03 ₁₆	ACTIVE	ACCEPTED	the specified port is in a ACTIVE state
04 ₁₆	INACTIVE	ACCEPTED	the specified port is in a INACTIVE state
05 ₁₆	WAIT	ACCEPTED	the specified port is in a WAIT state
06 ₁₆	SUSPENDED	ACCEPTED	the specified port is in a SUSPENDED state
80 ₁₆	NO_PLUG	REJECTED	no plug is available now
81 ₁₆	NO_PORT	REJECTED	no port of the specified plug is available now
82 ₁₆	PLUG_BUSY	REJECTED	the specified plug is not available now
83 ₁₆	PORT_BUSY	REJECTED	the specified port is not available now
84 ₁₆	INVALID_OFFSET	REJECTED	the invalid offset address value passed
85 ₁₆	NO_CONNECTION	REJECTED	no connection to break no connection to add overlay
86 ₁₆	CONNECTED_NODE_ERROR	REJECTED	connected port not responding
87 ₁₆	BROKEN	REJECTED	disconnected by the connected port
88 ₁₆	MAX_OVERLAY	REJECTED	connection count is already maximum value
89 ₁₆	FRAME_ERROR	REJECTED	the frame transmission failure. frame is not ended with <i>iAPR.mode</i> = LAST
FE ₁₆	ANY_OTHER_ERR	REJECTED	other internal errors
other values	-	-	reserved

NOTE: If the msb of returned status value was set to one, this field indicates the error code. Note that in case of command frame (both CONTROL and STATUS), this field shall be set to FF₁₆, and status value is returned in a response frame from the target.

Table 6.2-4 below shows the status field values of the response frame in case of STATUS command.

Table 6.2-4 - status field values for STATUS command response frame

value	symbol	response code of AV/C frame	meaning
01 ₁₆	FREE	STABLE	the specified port is in a FREE state
02 ₁₆	FIXED	STABLE	the specified port is in a FIXED state
03 ₁₆	ACTIVE	STABLE	the specified port is in a ACTIVE state
04 ₁₆	INACTIVE	STABLE	the specified port is in a INACTIVE state
05 ₁₆	WAIT	STABLE	the specified port is in a WAIT state
06 ₁₆	SUSPENDED	STABLE	the specified port is in a SUSPENDED state
other values	-	-	reserved

plug ID

The **plug ID** field indicates the plug identifier of the target, and this field should include the values defined in Table 6.2-5.

Table 6.2-5 - plug ID supported values

value	meaning
A0 ₁₆	asynchronous plug [0]
:	:
BE ₁₆	asynchronous plug [30]
BF ₁₆	any available asynchronous Plug

The value of BF₁₆ (any available asynchronous plug) may be used to indicate the controller has no preference and the plug number may be assigned by the target.

plug offset

The **plug offset** field has 42bits length and this field indicates the base offset address of the plug of the target.

port ID

The **port ID** field specifies which port of the plug is selected, as specified below:

Table 6.2-6 - port ID definitions

value	Meaning
0 ₁₆	consumer port
1 ₁₆	producer port [1]
:	:
E ₁₆	producer port[14]
F ₁₆	any available producer port

When setting up the connection with the producer plug, the controller should use F₁₆ as **port ID**. After bus reset, the controller should use the allocated **port ID** value to restore the port.

The **plug offset** and **port ID** indicates the 48 bits offset address of the port of the target node, as calculated below:

$$(\text{the offset address of the port}) = (\text{plug offset}) \ll 6 \mid (\text{port ID}) \ll 2;$$

port bits

The **port bits** field indicates the constraints and capabilities of specified port. The meaning of this field depends on the type of the port. If the port is the consumer port, this field indicates constraints it has. If the port is the producer port, this field indicates the capability of it. See [1], "4.3.4 Consumer port address" and "4.3.5 Producer port address" for detail. The Table 6.2-7 below shows the meaning of this field.

Table 6.2-7 - port bits field definitions

port ID Value	port bits value	symbol	meaning
0 ₁₆	X1 ₂	ct == 1	The consumer port supports the concurrent writes
	X0 ₂	ct == 0	The consumer port supports only the sequential writes
	1X ₂	m == 1	The consumer port supports multicast connections
	0X ₂	m == 0	The consumer port does not support multicast connections
1 ₁₆ ~E ₁₆	00 ₂	FIXED	Segment buffer size shall never change for this port
	01 ₂	FRAME	Segment buffer size may change at the next frame boundary for this port
	10 ₂	SEGMENT	Segment buffer size may change at the next segment boundary for this port
	11 ₂	-	Reserved

connected node ID

The **connected node ID** field indicates the node identifier which the target is (going to be) connected with.

connected plug offset

The **connected plug offset** field has 42bits length and this field indicates the base offset address of the plug to which the target plug is (going to be) connected with.

connected port ID

The **connected port ID** field indicates the port identifier or the port to which the target plug is (going to be) connected with, and this field has values defined as Table 6.2-6.

The **connected node ID**, **plug offset** and **port ID** indicates the 64 bits Serial Bus Address of the connected port, as calculated below:

$$(\text{address of the port}) = (\text{connected node ID}) \ll 48 \mid (\text{plug offset}) \ll 6 \mid (\text{port ID}) \ll 2;$$

connected port bits

The **connected port bits** field indicates constraints and capabilities of specified port. The meaning of this field depends on the type of the port. If the port is the consumer port, this field indicates constrains it has. If the port is the producer port, this field indicates the capability of it. The Table 6.2-7 shows the meaning of this field.

connected plug ID

The **connected plug ID** field indicates the plug identifier of the target, and has the values defined by Table 6.2-5.

connection count

The **connection count** field indicates the number of overlaid connections the consumer node holds. This means how many controllers concern for this connection. If the consumer port is not connected to any other producer port, this field is zero. After the consumer port is connected to the producer port, this field would be incremented, up to 63 (3F₁₆). Note that the value of 3F₁₆ shall be used for the command frame and the response frame may include the current value. As the producer port does not hold this information, this field shall be always 3F₁₆ for the producer port.

ex (exclusive)

The **ex** bit specifies whether the other controllers are expected to be excluded from this plug or not. If the target accepted the control command whose **ex** bit is one, the target shall check the following control command addressed to the plug, whether the node ID of the controller is same as the previous one. If the node ID of the controller is different from the previous one, the target shall reject the requested control commands. If this bit is set to zero, the target may accept the control commands addressed to the specified plug from other controllers.

Note that **ex** = 1 may be accepted if one of the following conditions had been met:

- ✓ The specified plug is the consumer plug.
- ✓ The specified producer plug is not used.
- ✓ The controller, which already holds the exclusive access to the producer plug, is going to add a new (muticast) connection on it.

If a port is used using **ex**=1, the following ASYNCHRONOUS CONNECTION control command with **ex**=1 should be used by the owner. When **subfunction** field specifies ATTACH_FRAME, ALLOCATE_ATTACH_FRAME, or RESTORE_PORT_FRAME, the **ex** bit shall be set to one.

res

res field is reserved for the futuer extension and shall be zero.

write interval

write interval indicates the required interval of Serial Bus Write Transactions that would be issued to the consumer port, to avoid a number of "ack_busy" returning. Interval time of write transactions can be calculated as follows:

$$(\text{Interval Time}) = (\text{NOMINAL_CYCLE_TIME}) \times 2^{(\text{write interval})}$$

If the *ct* bit of the consumer port is zero, this field has no meaning and its value is F_{16} .

retry count

retry count indicates the required retry count for Serial Bus Transactions failure issued to the consumer port.

The values of write interval and retry count would be provided by the consumer node by using the RESPONSE frame. The controller shall pass these values to the producer node. The producer node is recommended to pace the generation of Serial Bus Transactions according to provided values. Otherwise, if the producer node ignored these values and generated Serial Bus Transactions at its own pace, these transactions may fail and the producer node may receive ack_busy.

6.3 CONTROL commands

This section describes the format of the ASYNCHRONOUS CONNECTION CONTROL command.

For the CONTROL command, the target returns NOT IMPLEMENTED response when any fields, except for *plug offset*, *connected node ID* and *connected plug offset* field, includes the unsupported values.

Note that all fields shall be the same as the command frame in the NOT IMPLEMENTED response frame.

6.3.1 ALLOCATE subfunction

ALLOCATE subfunction is used to retrieve the offset address of the specified or available consumer port from the target, and temporary reserve the port from other changes before ATTACH subfunction. The controller can specify the plug ID of the target, or allow the consumer to assign the available plug with using “any available plug” value.

In the case that this subfunction is used with “any available plug” value for the *plug ID* field and the consumer has an available plug, its plug ID and its offset address are returned in the ACCEPTED response frame.

Table 6.3-1 illustrates the value of each field in the CONTROL command frame and ACCEPTED response frame of ALLOCATE subfunction.

Table 6.3-1 - Command and ACCEPTED response frame of ALLOCATE

field	CONTROL command frame	ACCEPTED response frame
subfunction	ALLOCATE (01 ₁₆)	
status	not used (FF ₁₆)	FIXED (02 ₁₆)
plug ID	specified plug ID or any available plug (BF ₁₆)	specified plug ID or allocated plug ID
plug Offset	not used (3 FF FF FF FF FF ₁₆)	Offset address of the plug
port ID	consumer port (0 ₁₆)	←
port bits	not used (11 ₂)	supported value
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (11 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	00 ₁₆ (current value)
write interval	not used (F ₁₆)	required write interval value
retry count	not used (F ₁₆)	required retry count value

NOTE: “←” means “same as the command frame”

Note that ALLOCATE subfunction can be used for only the consumer, so the *port ID* field shall be always 0₁₆.

If the target cannot allocate the requested plug resource, or any error condition had been occurred, the target returns the REJECTED response.

Table 6.3-2 illustrates the value of each field in the CONTROL command frame and REJECTED response frame of ALLOCATE subfunction.

Table 6.3-2 - Command and REJECTED response frame of ALLOCATE

field	CONTROL command frame	REJECTED response frame
subfunction	ALLOCATE (01 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID or any available plug (BF ₁₆)	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (0 ₁₆)	←
port bits	not used (1 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (1 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the command frame”

In the REJECTED response frame, the *status* field includes the error code as defined in Table 6.2-3.

6.3.2 ALLOCATE_ATTACH subfunction

ALLOCATE_ATTACH subfunction is used for the producer to retrieve the offset address of the specified or available producer plug and/or port from the target, and to inform the producer node about the consumer port information for establishing a connection.

Furthermore, ALLOCATE_ATTACH subfunction is used for the producer to establish the multicast connection. When adding a new connection to the already-used plug (multicast), the plug ID value may be retrieved by using the STATUS command before issuing this subfunction. And the controller may use “any available port” value for the port ID field of this command, to allow the producer plug to allocate a new port.

A controller may specify the plug ID, or it may allow the target to allocate the available plug when the controller has no preference and the plug number may be assigned by the producer. And the controller shall specify the to-be-connected port that had been retrieved from the consumer node by the ALLOCATE response frame.

In the case that this subfunction is used with “any available plug” value for the *plug ID* field and the producer has an available plug, its plug ID and its offset address would be returned in the ACCEPTED response frame.

Table 6.3-3 illustrates the value of each field in the CONTROL command frame and ACCEPTED response frame for the ALLOCATE_ATTACH subfunction.

Table 6.3-1 - Command and ACCEPTED response frame of ALLOCATE_ATTACH

field	CONTROL command frame	ACCEPTED response frame
subfunction	ALLOCATE_ATTACH (03 ₁₆)	
status	not used (FF ₁₆)	ACTIVE (03 ₁₆)
plug ID	specified plug ID or any available plug (BF ₁₆)	specified plug ID or allocated plug ID
plug Offset	not used (3 FF FF FF FF FF ₁₆)	Offset address of the plug
port ID	specified port ID or any available port (F ₁₆)	specified port ID or allocated port ID
port bits	not used (1 ₂)	supported value
connected node ID	specified node ID to be connected (consumer)	←
connected plug offset	offset address of the specified plug to be connected (consumer)	←
connected port ID	consumer port (0 ₁₆)	←
connected port bits	supported value from consumer plug	←
connected plug ID	plug ID of the consumer	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	retrieved value from the consumer	←
retry count	retrieved value from the consumer	←

NOTE: “←” means “same as the command frame”

After the target returned the ACCEPTED response to the controller, the target waits *oAPR* register update from the consumer.

Note that ALLOCATE_ATTACH can be used for only the producer, so the *port ID* field shall not be 0₁₆.

If the fields which are used to specify the parameters (other fields which are not described “not used”), include invalid value, or the target cannot allocate the requested port resource, or any error condition had been occurred, the target returns the REJECTED response.

Table 6.3-4 illustrates the value of each field in the CONTROL command and the REJECTED response frame for ALLOCATE_ATTACH subfunction.

Table 6.3-2 - Command and REJECTED response for ALLOCATE_ATTACH

field	CONTROL command frame	REJECTED response frame
subfunction	ALLOCATE_ATTACH (03 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID or any available plug (BF ₁₆)	←
plug Offset	not used (3 FF FF FF FF ₁₆)	←
port ID	specified port ID or any available port (F ₁₆)	←
port bits	not used (11 ₂)	←
connected node ID	specified node ID to be connected (consumer)	←
connected plug offset	offset address of the specified plug to be connected (consumer)	←
connected port ID	consumer port (0 ₁₆)	←
connected port bits	supported value from consumer plug	←
connected plug ID	plug ID of the consumer	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	retrieved value from the consumer	←
retry count	retrieved value from the consumer	←

NOTE: "←" means "same as the command frame"

In the REJECTED response frame, the *status* field includes the error status value as defined in Table 6.2-3.

6.3.3 ALLOCATE_ATTACH_FRAME subfunction

ALLOCATE_ATTACH_FRAME subfunction is used for a producer to transfer a single frame. After a frame had been transmitted, the connection will be broken automatically by the producer.

The producer returns the INTERIM response on successful port allocation. After the INTERIM response, ACCEPTED or REJECTED response will be returned to the controller as a final response and the port will become FREE state.

After the connection had been broken, both of the producer port and the consumer port states changed to FREE state, so there is no need to issue the DETACH_RELEASE subfunction to the producer port, and no need to issue the DETACH and RELEASE subfunction to the consumer port.

Table 6.3-5 illustrates the value of each field in the CONTROL command frame and the successful INTERIM response, and following ACCEPTED response frame or REJECTED response frame, for the ALLOCATE_ATTACH_FRAME subfunction .

Table 6.3-1 - Command and INTERIM response frame of ALLOCATE_ATTACH_FRAME

field	CONTROL command frame	INTERIM response frame	ACCEPTED response frame	REJECTED response frame
subfunction	ALLOCATE_ATTACH_FRAME (83 ₁₆)			
status	not used (FF ₁₆)	ACTIVE (03 ₁₆)	FREE (01 ₁₆)	error code
plug ID	specified plug ID or any available plug (BF ₁₆)	specified plug ID or allocated plug ID	←	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	Offset address of the plug	←	←
port ID	producer port[1] (1 ₁₆)	←	←	←
port bits	not used (11 ₂)	supported value	←	←
connected node ID	specified node ID to be connected (consumer)	←	←	←
connected plug offset	offset address of the specified plug to be connected (consumer)	←	←	←
connected port ID	consumer port (0 ₁₆)	←	←	←
connected port bits	supported value from consumer plug	←	←	←
connected plug ID	plug ID of the consumer	←	←	←
ex	exclusive (1 ₂)	←	←	←
connection count	not used (3F ₁₆)	←	←	←
write interval	retrieved value from the consumer	←	←	←
retry count	retrieved value from the consumer	←	←	←

NOTE: "←" means "same as the previous frame"

After the target returned the INTERIM response to the controller, the target waits *oAPR* register update from the consumer.

Note that ALLOCATE_ATTACH_FRAME can be used for only the producer, so the *port ID* field shall not be 0₁₆.

After a frame had been transmitted to the consumer, the target returns the final response according the result. If a frame had been transmitted successfully, the target returns the ACCEPTED response.

If a frame had not been transmitted with error conditions, the target returns the REJECTED response.

If REJECTED response returned, the status field includes the error code as defined Table 6.2-3.

The ALLOCATE_ATTACH_FRAME may be rejected without returning INTERIM response, if the fields which are used to specify the parameters (other fields which are not described "not used") include invalid value, or the target cannot allocate the requested port resource, or any error condition had been occurred.

Table 6.3-6 illustrates the CONTROL command frame and REJECTED response frame for ALLOCATE_ATTACH_FRAME subfunction.

Table 6.3-2 - Command and REJECTED response for ALLOCATE_ATTACH_FRAME

field	CONTROL command frame	REJECTED response frame
subfunction	ALLOCATE_ATTACH (FRAME) (83 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID or any available plug (BF ₁₆)	←
plug Offset	not used (3 FF FF FF FF ₁₆)	←
port ID	producer port[1] (1 ₁₆)	←
port bits	not used (11 ₂)	←
connected node ID	specified node ID to be connected (consumer)	←
connected plug offset	offset address of the specified plug to be connected (consumer)	←
connected port ID	consumer port (0 ₁₆)	←
connected port bits	supported value from consumer plug	←
connected plug ID	plug ID of the consumer	←
ex	exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	retrieved value from the consumer	←
retry count	retrieved value from the consumer	←

NOTE: "←" means "same as the command frame"

In the REJECTED response frame, the *status* field includes the error code as defined at Table 6.2-3.

6.3.4 ATTACH subfunction

ATTACH subfunction is used for the consumer to inform the offset address of the producer plug and port for establishing a connection.

Table 6.3-7 illustrates the value of each field in the CONTROL command frame and ACCEPTED response frame for the ATTACH subfunction.

Table 6.3-1 - Command and ACCEPTED response frame of ATTACH

field	CONTROL command frame	ACCEPTED response frame
subfunction	ATTACH (02 ₁₆)	
status	not used (FF ₁₆)	ACTIVE (03 ₁₆)
plug ID	allocated plug ID	←
plug Offset	Offset address of the plug	←
port ID	consumer port (0 ₁₆)	←
port bits	supported value	←
connected node ID	specified node ID to be connected (producer)	←
connected plug offset	offset address of the specified plug to be connected (producer)	←
connected port ID	port ID of the producer port	←
connected port bits	supported value from producer plug	←
connected plug ID	plug ID of the producer	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	01 ₁₆
write interval	not used (F ₁₆)	required write interval value
retry count	not used (F ₁₆)	required retry count value

NOTE: “←” means “same as the command frame”

After the target returned the ACCEPTED response to the controller, the target updates *oAPR* register in the producer.

Note that ATTACH can be used for only the consumer, so the *port ID* field shall be 0₁₆. If the fields which are used to specify the parameters (other fields which are not described “not used”), include invalid value, or any error condition had been occurred, the target returns the REJECTED response.

Table 6.3-8 illustrates the CONTROL command frame and REJECTED response frame for ATTACH subfunction.

Table 6.3-2 - Command and REJECTED response for ATTACH

field	CONTROL command frame	REJECTED response frame
subfunction	ATTACH (02 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	allocated plug ID	←
plug Offset	Offset address of the plug	←
port ID	consumer port (0 ₁₆)	←
port bits	supported value	←
connected node ID	specified node ID to be connected (producer)	←
connected plug offset	offset address of the specified plug to be connected (producer)	←
connected port ID	port ID of the producer port	←
connected port bits	supported value from producer plug	←
connected plug ID	plug ID of the producer	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

In the REJECTED response frame, the *status* field includes the error code as defined at Table 6.2-3.

6.3.5 ATTACH_FRAME subfunction

ATTACH_FRAME subfunction is used for consumer to break the connection automatically after a frame has been transmitted, in addition to “ATTACH” subfunction. The consumer returns the INTERIM response on success, followed by the ACCEPTED or REJECTED response after a frame transmission.

Table 6.3-9 illustrates the value of each field in the CONTROL command frame and successful INTERIM response, and following ACCEPTED response and REJECTED response frame for the ATTACH_FRAME subfunction.

Table 6.3-1 - Command and INTERIM response frame of ATTACH_FRAME

field	CONTROL command frame	INTERIM response frame	ACCEPTED response frame	REJECTED response frame
subfunction	ATTACH_FRAME (82 ₁₆)			
status	not used (FF ₁₆)	ACTIVE (03 ₁₆)	FREE (01 ₁₆)	error code
plug ID	allocated plug ID	←	←	←
plug Offset	Offset address of the plug	←	←	←
port ID	consumer port (0 ₁₆)	←	←	←
port bits	supported value	←	←	←
connected node ID	specified node ID to be connected (producer)	←	←	←
connected plug offset	offset address of the specified plug to be connected (producer)	←	←	←
connected port ID	port ID of the producer port	←	←	←
connected port bits	supported value from producer plug	←	←	←
connected plug ID	plug ID of the producer	←	←	←
ex	exclusive (1 ₂)	←	←	←
connection count	not used (3F ₁₆)	1 ₁₆	0 ₁₆	0 ₁₆
write interval	not used (F ₁₆)	required write interval value	←	←
retry count	not used (F ₁₆)	required retry count value	←	←

NOTE: “←” means “same as the previous frame”

After the target returned the INTERIM response to the controller, the target updates *oAPR* register in the producer.

Note that ATTACH_FRAME can be used for only the consumer, so the *port ID* field shall be 0₁₆.

After a frame had been received from the producer, the target returns the final response according the result. If a frame had been received successfully, the target returns ACCEPTED response. If a frame had not been received with error conditions, the target returns REJECTED response.

For the REJECTED response frame, the status field indicates the error code defined as Table 6.2-3.

The ATTACH_FRAME may be rejected without returning INTERIM response, if the fields which are used to specify the parameters (other fields which are not described “not used”) include invalid value, or any error condition had been occurred.

Table 6.3-10 illustrates the value of each field in the CONTROL command frame and REJECTED response frame for ATTACH_FRAME subfunction.

Table 6.3-2 - Command and REJECTED response for ATTACH_FRAME

field	CONTROL command frame	REJECTED response frame
subfunction	ATTACH_FRAME (82 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	allocated plug ID	←
plug Offset	Offset address of the plug	←
port ID	consumer port (0 ₁₆)	←
port bits	supported value	←
connected node ID	specified node ID to be connected (producer)	←
connected plug offset	offset address of the specified plug to be connected (producer)	←
connected port ID	port ID of the producer port	←
connected port bits	supported value from producer plug	←
connected plug ID	plug ID of the producer	←
ex	exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

In the REJECTED response frame, the *status* field includes the error code as defined at Table 6.2-3.

6.3.6 ADD_OVERLAY subfunction

ADD_OVERLAY subfunction is used to overlay the existing connection by the same controller that established the connection, or other controllers.

The successful ADD_OVERLAY results the connection count value to be incremented. When the *connection count* is 63, which means there are maximum value of overlay for the connection, ADD_OVERLAY command shall be rejected by returning REJECTED response.

The controller shall specify the target port information (plug ID, port ID, offset address, and so on) and the connected port information, which may be previously retrieved from the target by using STATUS command.

Table 6.3-11 illustrates the value of each field in the CONTROL command frame and ACCEPTED response frame for the ADD_OVERLAY subfunction.

Table 6.3-1 - Command and ACCEPTED response frame of ADD_OVERLAY

field	CONTROL command frame	ACCEPTED response frame
subfunction	ADD_OVERLAY (0A ₁₆)	
status	not used (FF ₁₆)	ACTIVE (03 ₁₆) or SUSPENDED (06 ₁₆)
plug ID	allocated plug ID	←
plug Offset	Offset address of the plug	←
port ID	consumer port (0 ₁₆)	←
port bits	supported value	←
connected node ID	specified node ID to be connected (producer)	←
connected plug offset	offset address of the specified plug to be connected (producer)	←
connected port ID	port ID of the producer port	←
connected port bits	supported value from producer plug	←
connected plug ID	plug ID of the producer	←
ex	not exclusive (0 ₂)	←
connection count	not used (3F ₁₆)	incremented connection count value
write interval	not used (F ₁₆)	required write interval value
retry count	not used (F ₁₆)	required retry count value

NOTE: "←" means "same as the previous frame"

When the target returns ACCEPTED response, the *connection count* field of the response frame includes the updated *connection count* value, which had been incremented

Note that ADD_OVERLAY can be used for only the consumer, so the *port ID* field shall be 0₁₆.

If the fields which are used to specify the parameters (other fields which are not described "not used"), include invalid value, or any error condition had been occurred, the target returns the REJECTED response.

Table 6.3-12 indicates the command frame and the REJECTED response frame for the ADD_OVERLAY subfunction.

Table 6.3-2 - Command and REJECTED response frame of ADD_OVERLAY

field	CONTROL command frame	REJECTED response frame
subfunction	ADD_OVERLAY (0A ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	allocated plug ID	←
plug Offset	Offset address of the plug	←
port ID	consumer port (0 ₁₆)	←
port bits	supported value	←
connected node ID	specified node ID to be connected (producer)	←
connected plug offset	offset address of the specified plug (producer)	←
connected port ID	port ID of the producer port	←
connected port bits	supported value from producer plug	←
connected plug ID	plug ID of the producer	←
ex	not exclusive (0 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: "←" means "same as the previous frame"

When REJECTED response returned, the *status* field contains the error code as defined at Table 6.2-3.

6.3.7 DETACH subfunction

DETACH subfunction is used for the consumer, to remove an overlaid connection if the connection count is greater than one, or to break the connection if the connection count is 1. The controller shall specify the plug ID and port ID values of the target.

The successful DETACH addressed to the overlaid port results the connection count value to be decremented. When the *connection count* is one, which means there is a single connection, decremented connection count results the connection to be broken (detached).

If the controller detects the connection count is set to zero in the ACCEPTED response frame for DETACH subfunction, the controller is responsible to break this connection, by issuing the DETACH_RELEASE and RELEASE subfunctions.

Table 6.3-13 illustrates the value of each field in the CONTROL command frame and ACCEPTED response frame for the DETACH subfunction.

Table 6.3-1 - Command and ACCEPTED response frame of DETACH

field	CONTROL command frame	ACCEPTED response frame
subfunction	DETACH (06 ₁₆)	
status	not used (FF ₁₆)	ACTIVE (03 ₁₆) or SUSPENDED (06 ₁₆) or INACTIVE (04 ₁₆)
Plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	Offset address of the plug
port ID	consumer port (0 ₁₆)	←
port bits	not used (1 ₂)	supported value
connected node ID	not used (FF FF ₁₆)	connected node ID
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	offset address of the connected plug
connected port ID	not used (F ₁₆)	port ID of the producer port
connected port bits	not used (1 ₂)	supported value from producer plug
connected plug ID	not used (FF ₁₆)	plug ID of the producer
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	decremented count
write interval	not used (F ₁₆)	required write interval value
retry count	not used (F ₁₆)	required retry count value

NOTE: "←" means "same as the previous frame"

If the returned value of *connection count* field is zero, the returned value of status is INACTIVE. If the returned value of connection count field is non-zero, it indicates that the connection still exists, and returned value of status is ACTIVE or SUSPENDED.

After the target returned the ACCEPTED response with *connection count* = 0 to the controller, the target becomes INACTIVE state and stop updating *oAPR* register in the producer.

Note that DETACH can be used for the consumer only, so the *port ID* field value shall be always 0₁₆.

If the fields which are used to specify the parameters (other fields which are not described "not used"), include invalid value, or any error condition had been occurred, the target returns the REJECTED response.

Table 6.3-14 illustrates the value of each field in the CONTROL command frame and REJECTED response frame.

Table 6.3-2 - Command and REJECTED response for DETACH

field	CONTROL command frame	REJECTED response frame
subfunction	DETACH (06 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (0 ₁₆)	←
port bits	not used (1 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (1 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

In the REJECTED response frame, the *status* field includes the error code as defined in Table 6.2-3.

6.3.8 DETACH_RELEASE subfunction

DETACH_RELEASE subfunction is used for the producer, to break the connection and to release the port resource. The controller shall specify the plug ID and port ID values of the target.

Table 6.3-15 illustrates the value of each field in the CONTROL command frame and ACCEPTED response frame for the DETACH_RELEASE subfunction.

Table 6.3-1 - Command and ACCEPTED response frame of DETACH_RELEASE

field	CONTROL command frame	ACCEPTED response frame
subfunction	DETACH_RELEASE (07 ₁₆)	
status	not used (FF ₁₆)	FREE (01 ₁₆)
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	specified port ID	←
port bits	not used (1 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (1 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the command frame”

After the target returned the ACCPTED response to the controller, the target stops issuing Serial Bus transactions addressed to the consumer.

Note that DETACH_RELEASE can be used for the producer only, so the *port ID* field value shall be specified and shall not be 0₁₆.

If the fields which are used to specify the parameters (other fields which are not described “not used”), include invalid value, or any error condition had been occurred, the target returns the REJECTED response.

Table 6.3-16 illustrates the value of each field in the CONTROL command frame and REJECTED response frame for DETACH_RELEASE subfunction.

Table 6.3-2 - Command and REJECTED response frame of DETACH_RELEASE

field	CONTROL command frame	REJECTED response frame
subfunction	DETACH_RELEASE(07 ₁₆)	
status	not used (FF ₁₆)	status value
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	specified port ID	←
port bits	not used (11 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (11 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the command frame”

In the REJECTED response frame, the *status* field includes the error code as defined in Table 6.2-3.

6.3.9 RELEASE subfunction

RELEASE subfunction is used to release the consumer port. The controller shall specify the plug ID and port ID values of the target.

Table 6.3-17 illustrates the value of each field in the CONTROL command frame and the ACCEPTED response frame of RELEASE subfunction.

Table 6.3-1 - Command and ACCEPTED response frame of RELEASE

field	CONTROL command frame	ACCEPTED response frame
subfunction	RELEASE (05 ₁₆)	
status	not used (FF ₁₆)	FREE (01 ₁₆)
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (0 ₁₆)	←
port bits	not used (1 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (1 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

Note that RELEASE can be used for the consumer only, so the *port ID* field value shall be always 0₁₆.

If the fields which are used to specify the parameters (other fields which are not described “not used”), include invalid value, or any error condition had been occurred, the target returns the REJECTED response.

Table 6.3-18 illustrates the value of each field in the CONTROL command frame and REJECTED response frame for RELEASE subfunction.

Table 6.3-2 - Command and REJECTED response frame of RELEASE

field	CONTROL command frame	REJECTED response frame
subfunction	RELEASE (05 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (0 ₁₆)	←
port bits	not used (1 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (1 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

In the REJECTED response frame, the *status* field includes the error code as defined in Table 6.2-3.

6.3.10 RESTORE_PORT subfunction

RESTORE_PORT subfunction is used for both the producer and the consumer, to re-establish the connection after bus reset. And RESTORE_PORT subfunction is used to re-establish the multicast connections with specifying the port ID value that had been connected.

Note that when a bus reset occurred, the node (producer or consumer) shall keep its connection-related information for 10 seconds. If no RESTORE_PORT had been issued within 10 seconds, the node shall release the connection-related information and turn to its initial FREE state. After receiving RESTORE_PORT, the producer shall wait the update of the *oAPR* register by the consumer, then start the segment transmission. According to this re-establishment procedure, the controller shall issue RESTORE_PORT to the producer at first and issue RESTORE_PORT to the consumer later. Note that during the data might be lost at the producer during these procedures if the producer had been sending timing-dependent data frames.

The controller shall specify the *plug ID*, *port ID*, *connected node ID* and *ex* fields. The *connected node ID* field shall be updated with the new node ID if it changes after a bus reset. If these values are not consistent with the previous values before a bus reset except for the *connected node ID* field, the target returns the REJECTED response.

Table 6.3-19 illustrates the value of each field in the CONTROL command frame and the ACCEPTED response frame of RESTORE_PORT for the producer port.

Table 6.3-1 - Command and ACCEPTED response frame of RESTORE_PORT for the producer port

field	CONTROL command frame	ACCEPTED response frame
subfunction	RESTORE_PORT (40 ₁₆)	
status	not used (FF ₁₆)	ACTIVE (03 ₁₆)
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	offset address of the plug
port ID	port ID of the producer port	←
port bits	not used (11 ₂)	supported value
connected node ID	node ID of consumer	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	offset address of connected plug
connected port ID	consumer port (00 ₁₆)	←
connected port bits	not used (11 ₂)	supported value from the consumer port
connected plug ID	not used (FF ₁₆)	connected plug ID of the consumer
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	required value from the consumer
retry count	not used (F ₁₆)	required value from the consumer

NOTE: "←" means "same as the command frame"

Table 6.3-20 illustrates the value of each field in the CONTROL command frame and the REJECTED response frame of RESTORE_PORT for the producer port.

Table 6.3-2 - Command and REJECTED response frame of RESTORE_PORT for the producer port

field	CONTROL command frame	REJECTED response frame
subfunction	RESTORE_PORT (40 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	port ID of the producer port	←
port bits	not used (11 ₀₂)	←
connected node ID	node ID of consumer	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	consumer port (00 ₁₆)	←
connected port bits	not used (11 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: "←" means "same as the command frame"

In the REJECTED response frame, the *status* field includes the error code as defined in Table 6.2-3.

Table 6.3-21 illustrates the value of each field in the CONTROL command frame and the ACCEPTED response frame of RESTORE_PORT for the consumer port.

Table 6.3-3 - Command and ACCEPTED response frame of RESTORE_PORT for the consumer port

field	CONTROL command frame	ACCEPTED response frame
subfunction	RESTORE_PORT (40 ₁₆)	
status	not used (FF ₁₆)	ACTIVE (03 ₁₆) or SUSPENDED (06 ₁₆)
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	offset address of the plug
port ID	consumer port (00 ₁₆)	←
port bits	not used (1 ₂)	supported value
connected node ID	node ID of producer	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	offset address of connected plug
connected port ID	port ID of the producer port	←
connected port bits	not used (1 ₂)	supported value from the producer port
connected plug ID	not used (FF ₁₆)	connected plug ID of the producer
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	01 ₁₆
write interval	not used (F ₁₆)	required write interval value
retry count	not used (F ₁₆)	required retry count value

NOTE: "←" means "same as the command frame"

Table 6.3-22 illustrates the value of each field in the CONTROL command frame and the REJECTED response frame of RESTORE_PORT for the consumer port.

Table 6.3-4 - Command and REJECTED response frame of RESTORE_PORT for the consumer port

field	CONTROL command frame	REJECTED response frame
subfunction	RESTORE_PORT (40 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (00 ₁₆)	←
port bits	not used (1 ₂)	←
connected node ID	node ID of producer	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	port ID of the producer port	←
connected port bits	not used (1 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: "←" means "same as the command frame"

In the REJECTED response frame, the *status* field includes the error code as defined in Table 6.2-3.

6.3.11 RESTORE_PORT_FRAME subfunction

RESTORE_PORT_FRAME subfunction is used for the producer and consumer to re-establish the connection, which had been established by the ALLOCATE_ATTACH_FRAME or ATTACH_FRAME subfunction command.

NOTE: Note that when a bus reset occurred, the node (producer or consumer) shall keep its connection-related information for 10 seconds. If no RESTORE_PORT_FRAME had been issued within 10 seconds, the node shall release the connection-related information and turn to its initial FREE state. After receiving RESTORE_PORT_FRAME, the producer shall wait the update of the *oAPR* register by the consumer, then start the segment transmission. According to this re-establishment procedure, the controller shall issue RESTORE_PORT_FRAME or RESTORE_PORT to the producer at first and issue RESTORE_PORT_FRAME or RESTORE_PORT to the consumer later.

If RESTORE_PORT_FRAME is issued to the producer, it returns the INTERIM response on success, followed by the ACCEPTED/REJECTED response after a frame transmission.

Table 6.3-23 shows the command frame CONTROL command frame and successful INTERIM response, and following ACCEPTED response REJECTED response frame for the RESTORE_PORT_FRAME subfunction.

Table 6.3-1 - Command and response frame of RESTORE_PORT_FRAME for the producer port

field	CONTROL command frame	INTERIM response frame	ACCEPTED response frame	REJECTED response frame
subfunction	RESTORE_PORT_FRAME (C0 ₁₆)			
status	not used (FF ₁₆)	ACTIVE (03 ₁₆)	FREE (01 ₁₆)	error code
plug ID	specified plug ID	←	←	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	offset address of the plug	←	←
port ID	port ID of the producer port	←	←	←
port bits	not used (11 ₂)	supported value	←	←
connected node ID	node ID of consumer	←	←	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	offset address of connected plug	←	←
connected port ID	consumer port (00 ₁₆)	←	←	←
connected port bits	not used (11 ₂)	supported value from the consumer port	←	←
connected plug ID	not used (FF ₁₆)	connected plug ID of the consumer	←	←
ex	exclusive (1 ₂)	←	←	←
connection count	not used (3F ₁₆)	←	←	←
write interval	not used (F ₁₆)	required value from the consumer	←	←
retry count	not used (F ₁₆)	required value from the consumer	←	←

NOTE: “←” means “same as the previous frame”

After a frame had been transmitted to the consumer, the producer returns the final response according the result. If a frame had been transmitted successfully, the producer returns ACCEPTED response. If a frame had not been transmitted with error conditions, the producer returns REJECTED response. For the REJECTED response frame, the status field indicates the error code defined as Table 6.2-3.

The RESTORE_PORT_FRAME may be rejected without returning INTERIM response, if the command included the invalid parameters, or any error conditions had been occurred.

Table 6.3-24 illustrates the value of each field in the CONTROL command frame and REJECTED response frame for ATTACH_FRAME subfunction for the producer port.

Table 6.3-2 - Command and REJECTED response for RESTORE_PORT_FRAME for the producer port

field	CONTROL command frame	REJECTED response frame
subfunction	RESTORE_PORT_FRAME (C0 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	port ID of the producer port	←
port bits	not used (11 ₂)	←
connected node ID	node ID of consumer	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	consumer port (00 ₁₆)	←
connected port bits	not used (11 ₂)	←
connected plug ID	not used (FF ₁₆)	←
connection count	exclusive (1 ₂)	←
ex	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

If RESTORE_PORT_FRAME is issued to the consumer, it returns the INTERIM response on success, followed by the ACCEPTED/REJECTED response after a frame transmission.

Table 6.3-25 shows the command frame CONTROL command frame and successful INTERIM response, and following ACCEPTED response REJECTED response frame for the RESTORE_PORT_FRAME subfunction for the consumer port.

Table 6.3-3 - Command and response frame of RESTORE_PORT_FRAME for the consumer port

field	CONTROL command frame	INTERIM response frame	ACCEPTED response frame	REJECTED response frame
subfunction	RESTORE_PORT_FRAME (C0 ₁₆)			
status	not used (FF ₁₆)	ACTIVE (03 ₁₆)	FREE (01 ₁₆)	error code
plug ID	specified plug ID	←	←	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	offset address of the plug	←	←
port ID	consumer port (00 ₁₆)	←	←	←
port bits	not used (11 ₂)	supported value	←	←
connected node ID	node ID of producer	←	←	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	offset address of the plug to be connected	←	←
connected port ID	port ID of the producer port	←	←	←
connected port bits	not used (11 ₂)	supported value from producer plug	←	←
connected plug ID	not used (FF ₁₆)	plug ID of the producer	←	←
ex	exclusive (1 ₂)	←	←	←
connection count	not used (3F ₁₆)	01 ₁₆	00 ₁₆	00 ₁₆
write interval	not used (F ₁₆)	required write interval value	←	←
retry count	not used (F ₁₆)	required retry count value	←	←

NOTE: “←” means “same as the previous frame”

After a frame had been transmitted to the consumer, the consumer returns the final response according the result. If a frame had been transmitted successfully, the producer returns ACCEPTED response. If a frame had not been transmitted with error conditions, the producer returns REJECTED response.

For the REJECTED response frame, the status field indicates the error code defined as Table 6.2-3.

Table 6.3-26 illustrates the value of each field in the CONTROL command frame and REJECTED response frame for ATTACH_FRAME subfunction for the consumer port.

Table 6.3-4 - Command and REJECTED response for RESTORE_PORT_FRAME for the consumer port

field	CONTROL command frame	REJECTED response frame
subfunction	RESTORE_PORT_FRAME (C0 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (00 ₁₆)	←
port bits	not used (11 ₂)	←
connected node ID	node ID of producer	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	port ID of the producer port	←
connected port bits	not used (11 ₂)	←
connected plug ID	not used (FF ₁₆)	←
connection count	exclusive (1 ₂)	←
ex	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

In the REJECTED response frame, the status field includes the error code defined at Table 6.2-3.

6.3.12 SUSPEND_PORT subfunction

SUSPEND_PORT subfunction is used for the consumer port to suspend the connection. When a connection is suspended, the remaining of the frame data is discarded by the producer, and no data transmission occurred until the RESUME_PORT subfunction is issued.

Table 6.3-27 illustrates the value of each field in the CONTROL command frame and the ACCEPTED response frame of SUSPEND_PORT subfunction.

Table 6.3-1 - Command and ACCEPTED response frame of SUSPEND_PORT

field	CONTROL command frame	ACCEPTED response frame
subfunction	SUSPEND_PORT (10 ₁₆)	
status	not used (FF ₁₆)	SUSPENDED (06 ₁₆)
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (0 ₁₆)	←
port bits	not used (11 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (11 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

Note that SUSPEND_PORT can be used for the consumer only, so the *port ID* field value shall be 0₁₆.

If the passed parameters are invalid, or the consumer port is already in a SUSPENDED state, or any error condition had been occurred, the target returns the REJECTED response.

Table 6.3-28 illustrates the value of each field in the CONTROL command frame and REJECTED response frame for SUSPEND_PORT subfunction.

Table 6.3-2 - Command and REJECTED response frame of SUSPEND_PORT

field	CONTROL command frame	REJECTED response frame
subfunction	SUSPEND_PORT (10 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (0 ₁₆)	←
port bits	not used (11 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (11 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

In the REJECTED response frame, the *status* field includes the error code as defined in Table 6.2-3.

6.3.13 RESUME_PORT subfunction

RESUME_PORT subfunction is used for the suspended-consumer port to resume the frame transmission.

The resumed asynchronous connection layer restarts the frame transmission from the beginning of the next outgoing frame, and during the SUSPENDED state, frames may be discarded by the producer.

Table 6.3-29 illustrates the value of each field in the CONTROL command frame and the ACCEPTED response frame of RESUME_PORT subfunction.

Table 6.3-1 - Command and ACCEPTED response frame of RESUME_PORT

field	CONTROL command frame	ACCEPTED response frame
subfunction	RESUME_PORT (20 ₁₆)	
status	not used (FF ₁₆)	ACTIVE (03 ₁₆)
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (0 ₁₆)	←
port bits	not used (11 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (11 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: “←” means “same as the previous frame”

Note that RESUME_PORT can be used for the consumer only, so the *port ID* field value shall be always 0₁₆.

If the passed parameters are invalid, or the consumer port is not in a SUSPENDED state, or any error condition had been occurred, the target returns the REJECTED response.

Table 6.3-30 illustrates the value of each field in the CONTROL command frame and REJECTED response frame for RELEASE subfunction.

Table 6.3-2 - Command and REJECTED response frame of RESUME_PORT

field	CONTROL command frame	REJECTED response frame
subfunction	RESUME_PORT (20 ₁₆)	
status	not used (FF ₁₆)	error code
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	←
port ID	consumer port (0 ₁₆)	←
port bits	not used (1 ₂)	←
connected node ID	not used (FF FF ₁₆)	←
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	←
connected port ID	not used (F ₁₆)	←
connected port bits	not used (1 ₂)	←
connected plug ID	not used (FF ₁₆)	←
ex	not exclusive (0 ₂) or exclusive (1 ₂)	←
connection count	not used (3F ₁₆)	←
write interval	not used (F ₁₆)	←
retry count	not used (F ₁₆)	←

NOTE: "←" means "same as the previous frame"

In the REJECTED response frame, the *status* field includes the error code as defined in Table 6.2-3.

6.4 STATUS command

ASYNCHRONOUS CONNECTION command supports *ctype* of STATUS. The STATUS command is used for retrieving the information about the port and its connection state, so the controller shall specify the plug ID and port ID values.

If the target supports the specified asynchronous plug and/or port, the target returns the STABLE response frame, which includes the port information it has. Otherwise, if the target does not have the specified plug and/or port, the target returns the NOT IMPLEMENTED response with the response frame which has all the same field values as the command frame. Furthermore, if the “not used” field included unspecified value, the target returns NOT IMPLEMENTED response, respectively. The following Table 6.4.1 illustrates the STATUS command frame values.

Table 6.4-1 - STATUS Command and STABLE response frame

field	STATUS command frame	STABLE response frame
subfunction		FF ₁₆
status	not used (FF ₁₆)	current value
plug ID	specified plug ID	←
plug Offset	not used (3 FF FF FF FF FF ₁₆)	current value
port ID	specified port ID	←
port bits	not used (11 ₂)	current value
connected node ID	not used (FF FF ₁₆)	current value
connected plug offset	not used (3 FF FF FF FF FF ₁₆)	current value
connected port ID	not used (F ₁₆)	current value
connected port bits	not used (11 ₂)	current value
connected plug ID	not used (FF ₁₆)	current value
ex	not used (1 ₂)	current value
connection count	not used (3F ₁₆)	current value
write interval	not used (F ₁₆)	current value
retry count	not used (F ₁₆)	current value

NOTE: “←” means “same as the command frame”

When a STATUS command is issued to the consumer port, *port ID* value shall be set to 0₁₆. The response frame may include the current value of each fields according to the current state of the port, as defined in Table 6.4-2 below:

Table 6.4-2 - Possible STATUS response from the producer port

fields	port state	FREE	ACTIVE	WAIT
status		FREE (01 ₁₆)	ACTIVE (03 ₁₆)	WAIT (05 ₁₆)
plug ID		←	←	←
plug Offset		O	O	O
port ID		←	←	←
port bits		O	O	O
connected node ID		X	O	X
connected plug offset		X	O	O
connected port ID		X	O	O
connected port bits		X	O	O
connected plug ID		X	O	O
ex		X	O	O
connection count		X	X	X
write interval		X	O	O
retry count		X	O	O

NOTE: “←” means “same as the specified value in the command frame”

“O” means “returning the current value”

“X” means “no information”, returns the same value as the command frame

NOTE: The states of the asynchronous port are defined in Chapter 7.

When a STATUS command is issued to the consumer port, *port ID* value shall be set to specified port ID value (1₁₆ ~E₁₆), which identifies the producer port. The response frame may include the current value of each fields according to the current state of the port, as defined in Table 6.4-3 below:

Table 6.4-3 - Possible STATUS response from the consumer port

fields	port state	FREE / FIXED	ACTIVE / INACTIVE / SUSPENDED	WAIT
status		FREE (01 ₁₆) / FIXED(02 ₁₆)	ACTIVE(03 ₁₆) / INACTIVE(04 ₁₆) / SUSPENDED(06 ₁₆)	WAIT (05 ₁₆)
plug ID		←	←	←
plug Offset		O	O	O
port ID		←	←	←
port bits		O	O	O
connected node ID		X	O	X
connected plug offset		X	O	O
connected port ID		X	O	O
connected port bits		X	O	O
connected plug ID		X	O	O
ex		X	O	O
connection count		X	O	O
write interval		O	O	O
retry count		O	O	O

NOTE: “←” means “same as the command frame”

“O” means “returning the current value”

“X” means “no information”, returns the same value as the command frame

NOTE: The states of the asynchronous port are defined in Chapter 7.

7. Asynchronous Connection port states

7.1 Code definitions

All of the procedures and definitions in this chapter use the syntax specified in the Table 7.1-1 below.

Table 7.1-1 - State machine code definitions

```
typedef struct {
    Byte subfnc;           // subfunction
    Byte plugID;          // target plug identifier
    Byte status;          // plug status or result of command execution
    Octlet plugAddr;      // offset address of plug, including portID and portBits
    Doublet connectNodeID; // node identifier of (to-be) connected node
    Octlet connectPlugOffset; // offset address of (to-be) connected plug,
                                // including connectPortID and connectPortBits
    Byte connectPlugID;   // plug identifier of (to-be)connected port
    boolean ex;           // exclusive request bit
    Byte connectionCount; // connection count which the (consumer) port holds
    Byte writeInterval;   // write interval value requested from the consumer
    Byte retryCount;      // retry count value requested from the consumer
} portInfo;

portInfo *cInfo;        // portInfo stored in the consumer port of the target
portInfo *pInfo;        // portInfo stored in the producer port of the target
portInfo *req;          // portInfo stored in the command frame which is requested by the controller
portInfo *hold;         // portInfo stored for deferred response
Doublet ctrl;           // controller nodeID, retrieved from AV/C command frame.
Doublet ctrl_bak;       // backuped controller nodeID stored in the target "PLUG" to check the exclusive
                        // access. Its initial value is 0xFFFF.
boolean dr;             // flag for disconnect request stored in the target port, set to 1 if the MSB
                        // of subfunction field had been set to 1.
boolean suspendflg;    // flag for suspended state before bus reset

#define legal_access    ( !cInfo->ex || ( ctrl_bak == ctrl ) )
                        // true if the requested command is from the controller who claimed to access
                        // the plug by issuing command with ex=1, or the plug is NOT allocated
                        // by a controller by issuing command with ex=0.

void initialize( portInfo *xInfo) // do the power-reset initialization, as follows:
{
    invalidate( xInfo );
    xInfo->plugID = PLUG_ID;        // Defined Plug ID value assigned for this port
    xInfo->plugAddr = PORT_ADDRESS; // Defined Plug Address value assigned for this port
    if ( isConsumer( xInfo ) ){
        xInfo->writeInterval = INITIAL_INT_VALUE; // INITIAL_INT_VALUE is the device specific value
        xInfo->retryCount = INITIAL_RETRY_VALUE; // INITIAL_RETRY_VALUE is the device specific value
    }
    dr = suspendflg = FALSE;
}

Byte avc_cmd( Doublet *ctrlID, portInfo *req );
// Valid "ASYNCHRONOUS CONNECTION" AV/C command frame reception event,
// returns subfunction, or NULL when no event arrived.
// As req->plugID field value should be specified, the command itself is passed
// to the proper plug management components. If the initial command includes "any
// available port" for producer, the node searches the available port to start
// its state machine.
// In the state machine diagrams, other AV/C commands than ASYNCHRONOUS CONNECTION
// are not described and make no state transitions, but they shall be handled as
// defined in [3] or other specifications.
// And also "ASYNCHRONOUS CONNECTION" commands with invalid field values are not
// fully described and make no state transitions, but shall be rejected with REJECTED
// response or NOT IMPELEMENTED response, as specified in Chapter 6.

Byte avc_rsp( Doublet ctrlID, Byte rsp, portInfo * );
// generate AV/C response frame to the controller specified as ctrlID.
// rsp specifies the response code of AV/C response frame, and other fields of the
// response frame are set by using the portInfo structure.

void invalidate( portInfo * ); // invalidate portInfo, setting all the bit to 1 except for PlugID and
                               // PlugAddr, then set portInfo->status to FREE value.

bool EndOfFrame_ind( BYTE *mode ); // event indication from Asynchronous Connections layer,
// which indicates the connection layer detects "LAST" or "TOSS" or "LESS" or "JUNK".
// "mode" returns register mode value from AsyncConnections layer.
```

Table 7.1-2 - State machine code definitions (cont'd)

```

bool EndOfFrame_ind( BYTE *mode );           // event indication from Asynchronous Connections layer,
                                             // which indicates the connection layer detects "LAST", "TOSS",
                                             // "LESS"
                                             // or "JUNK". "mode" returns register mode value from
                                             // AsyncConnections
                                             // layer.

void memcpy( portInfo *dst, portInfo *src); // copy the src structure to the dst

void setupInfo( portInfo *stat, portInfo *req ) // exchange the valid field value between stat and req
{
    stat->subfunc = req->subfunc;           // as follows:
    stat->status = req->status;             // copy the subfunction value
    setupInfo()                            // store status value, req->status value should be set before
    switch( req->subfunc ){
        case ALLOCATE:
            stat->ex = req->ex;              // set the ex bit
            req->plugAdr = stat->plugAdr;    // return plug address
            req->connectionCount = stat->connectionCount = 0; // set connectionCount to 0
            req->writeInterval = stat->writeInterval; // return (initial) writeInterval
            req->retryCount = stat->retryCount; // return (initial) retryCount
            break;
        case ALLOCATE_ATTACH:
        case ALLOCATE_ATTACH_FRAME:
            stat->ex = req->ex;              // set the ex bit
            req->plugAdr=stat->plugAdr;      // return the plug address
            memcpy(stat,req);              // set other fields as requested
            break;
        case ATTACH:
        case ATTACH_FRAME:
            req->connectionCount = 1;        // set connectionCount to 1
            memcpy(stat,req);              // set other fields as requested
            break;
        case ADD_OVERLAY:
            stat->connectionCount++;        // increment a connectionCount value
            memcpy(req,stat);              // return the current state values
            break;
        case DETACH:
            stat->connectionCount--;        // decrement a connectionCount value
            memcpy(req,stat);              // return the current state values
            break;
        case RESTORE_PORT:
        case RESTORE_PORT_FRAME:
            stat->connectNodeID = req->connectNodeID; // set connectNodeID
            if ( isConsumer( stat ) )
                stat->connectionCount = 1; // set connectionCount to 1
            memcpy(req,stat);              // return the current state values
            break;
        case SUSPEND_PORT:
        case RESUME_PORT:
        case DETACH_RELEASE:
        case RELEASE:
            break;                          // no fields to be exchanged
    }
}

bool Break_ind(); // returns TRUE when an indication from Asynchronous Connections layer occurred,
// which indicates the connection is broken because of the following conditions
// occurred:
// (a) connected port indicated "FREE", so the port confirmed "FREE".
// (b) other error conditions occurred.

bool Timeout_ind(); // returns TRUE when an indication from Asynchronous Connections layer, which
// indicates the connection timeout observed at the transport layer.

Byte SetErrorCode(); // returns the error code from the execution results from AsyncConnections layer,
// or from other internal conditions.

void AttachEvent_req( portInfo * ); // ATTACH event request to AsyncConnections layer
void DetachEvent_req( portInfo * ); // DETACH event request to AsyncConnections layer
void RestoreEvent_req( portInfo * ); // RESTORE event request to AsyncConnections layer
void ReleaseEvent_req( portInfo * ); // RELEASE event request to AsyncConnections layer
void BreakEvent_req( portInfo * ); // BREAK event request to AsyncConnections layer
void SuspendEvent_req( portInfo * ); // SUSPEND event request to AsyncConnections layer
void ResumeEvent_req( portInfo * ); // RESUME event request to AsyncConnections layer

```

7.2 Consumer port states

7.2.1 Consumer port state machine

The behavior of a consumer plug is specified by its state machine definition, as illustrated by Figure 7.2.1.

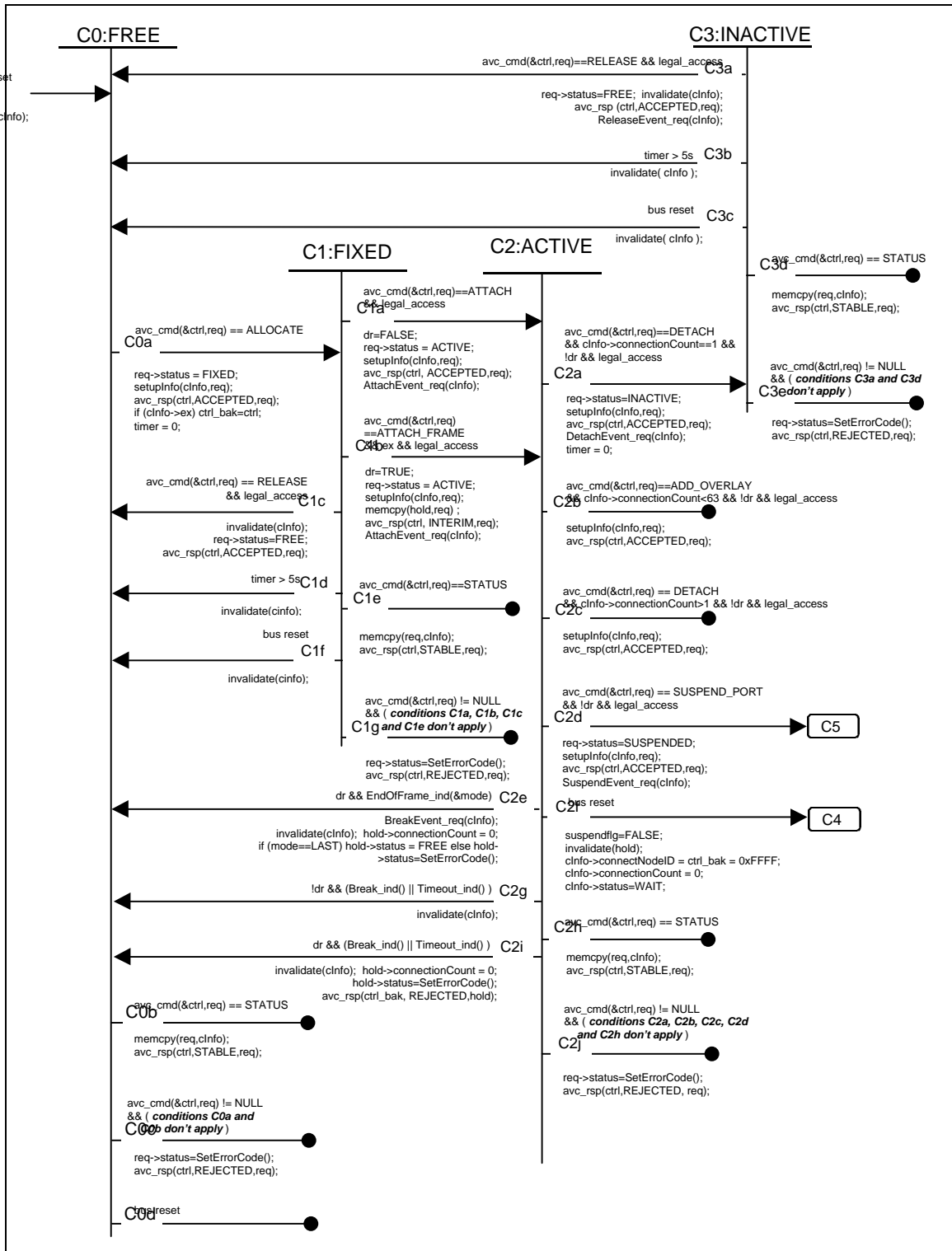


Figure 7.2.1 - Consumer port state machine

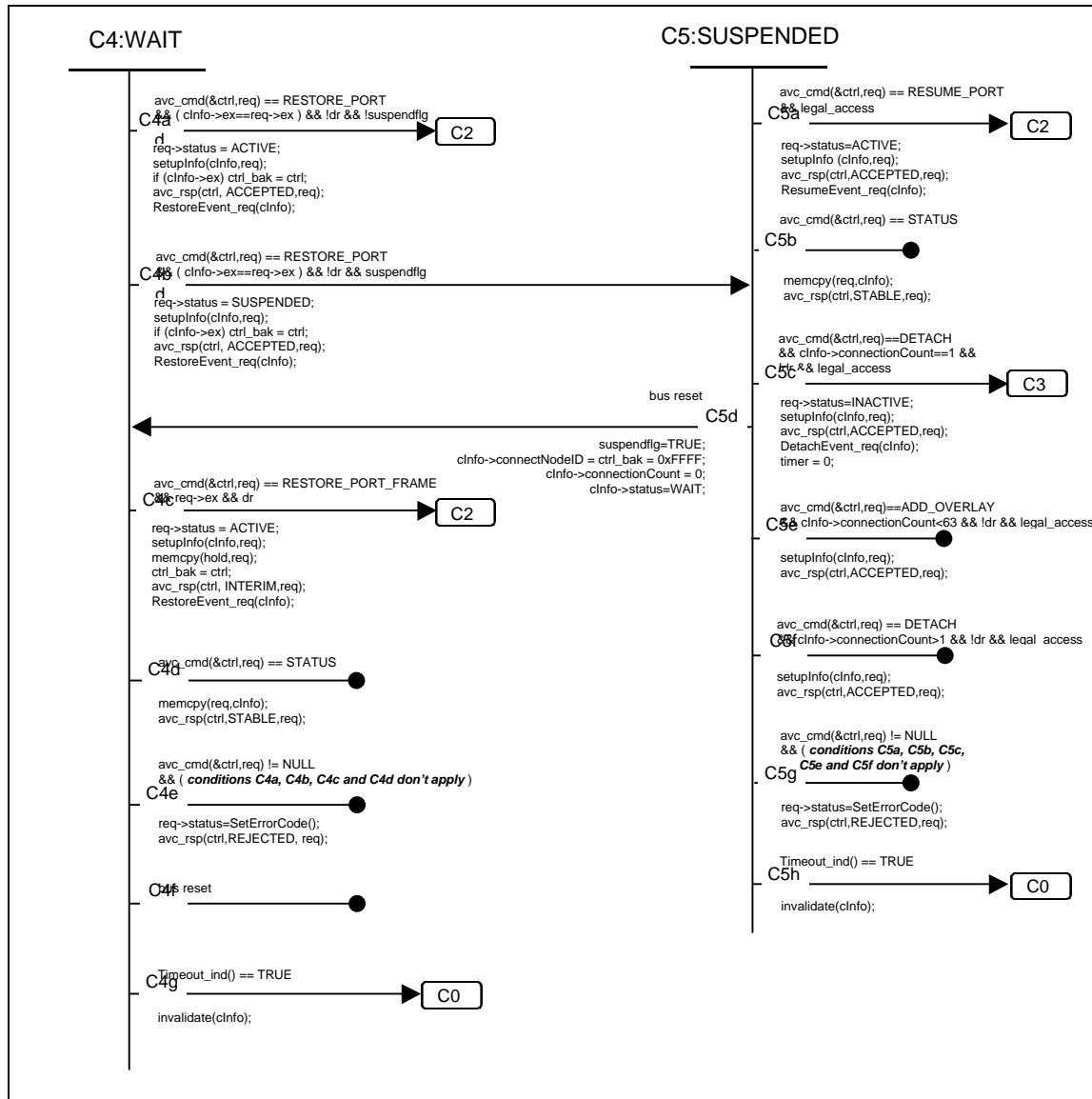


Figure 7.2.2 - Consumer port state machine (cont'd)

7.2.2 Consumer port state machine notes

State C0:FREE.

The *FREE* state is the initial state of the port, with no committed resources. While in the *C0:FREE* state, the purpose of the *ALLOCATE* subfunction is to transition the port to the *C2:ACTIVE* (connected) state, by passing through the intermediate *C1:FIXED* state. These state transitions are listed below.

Transition C0a.

When valid *ALLOCATE* subfunction is received, the node copies a port information to return response frame to the controller, then transitions to the *C1:FIXED* state. If the *ex* bit in the command frame had been set to one, the target stores the node ID of the controller to reject the following control commands from other controllers. And the port resets the timer to detect 5 seconds timeout for transitions from *C1:FIXED* to *C2:ACTIVE*.

- Transition C0b.** When STATUS command is requested, the port reports the current port information by returning STABLE response frame.
- Transition C0c.** All other connection management subfunctions are rejected, returning REJECTED response frame with the error code in the *status* field.
- Transition C0d.** Bus resets while in the *C0:FREE* state leaves the port in the same state.

State C1:FIXED.

In the *FIXED* state the ports resources have been allocated, but the port is not yet prepared to accept write or lock transactions and cannot send lock transactions to the producer port (since this port has not yet been identified). While in the *C1:FIXED* state, the purpose of the ATTACH or ATTACH_FRAME subfunction is to transition the port to the *C2:ACTIVE* (connected) state. If there had not been valid transition from *C1:FIXED* to *C2:ACTIVE* within 5 seconds, the port transitions to *C0:FREE* state. These state transitions are listed below.

- Transition C1a.** The valid ATTACH subfunction transitions the port to the fully-connected *C2:ACTIVE* state. ATTACH subfunction command frame includes the connection information such as producer port address. After returning ACCEPTED response frame to the controller, the port requests to an asynchronous connections layer to start by *AttachEvent_req()*. Although not illustrated, the *oAPR* register on the connected producer port is also updated to activate asynchronous connection communications.
- Transition C1b.** The valid ATTACH_FRAME subfunction transitions the port to the fully-connected *C2:ACTIVE* state. ATTACH_FRAME subfunction command frame includes the connection information such as producer port address. The port stores the requested connection information (for the final response), then returns INTERIM response frame to the controller. Then the port requests to an asynchronous connections layer to start by *AttachEvent_req()*. Although not illustrated, the *oAPR* register on the connected producer port is also updated to activate asynchronous connection communications.
- Transition C1c.** The valid RELEASE subfunction returns the port to the initial *C0:FREE* state. (This is normally the final step in a connection-abort sequence, which releases the consumer after a producer-port connection failure is discovered). The port invalidates the port information and returns ACCEPTED response frame to the controller.
- Transition C1d.** After 5 seconds connection timeout, the consumer port transitions to the *C0:FREE* state. The port invalidates the port information.
- Transition C1e.** When STATUS command with subfunction set to STATUS is requested, the port reports the current port information by generating STABLE response frame.
- Transition C1f.** A bus reset returns the port to its initial *C0:FREE* state. The port invalidates the port information.
- Transition C1g.** Other subfunctions are rejected, because they have no meaning while in this transient state or the following stable state. The node returns REJECTED response to the controller with the error code in the *status* field.

State C2:ACTIVE.

In the *ACTIVE* state the port is operational. While in the *C2:ACTIVE* state, the purpose of the subfunctions are to allow overlays (which leave the port in the *C2:ACTIVE* state, but increment the connection count) and to allow disconnection by transitioning the port to the *C3:INACTIVE* state. These state transitions are listed below.

- Transition C2a.** The valid DETACH subfunction addressed to the port holding no overlays (*connection count == 1*) state transitions the port to a half-disconnected *C3:INACTIVE* state. After returning ACCEPTED response to the controller, the port requests to an asynchronous connections layer to stop requesting the

- data by *DetachEvent_req()*. And the port resets the timer to detect 5 seconds timeout for transitions from *C3:INACTIVE* to *C0:FREE*.
- Transition C2b.** The valid *ADD_OVERLAY* subfunction increments the connection count if the *connection count* was less than 63, leaving the port connected in its *C2:ACTIVE* state. The node returns *ACCEPTED* response frame on success.
- Transition C2c.** The valid *DETACH* subfunction decrements the connection count if the *connection count* value was bigger than 1, leaving the port connected in its *C2:ACTIVE* state. The node returns *ACCEPTED* response frame on success.
- Transition C2d.** The valid *SUSPEND_PORT* subfunction transitions to *C5:SUSPENDED* state. After returning *ACCEPTED* response to the controller, the port requests to an asynchronous connections layer to suspend by *SuspendEvent_req()*.
- Transition C2e.** When the previous issued attachment subfunction had been *ATTACH_FRAME* and *dr* had been set to one and *EndOfFrame* was indicated from an asynchronous connections layer, the port requests the asynchronous connections layer to break the connection by issuing *BreakEvent_req()* to start the disconnection sequence described in Figure 5.9.2. Then invalidates the port information, and returns final response frame to the controller which indicates the previous issued *ATTACH_FRAME* execution had been finished. If the mode value retrieved from the asynchronous connections layer had been *LAST*, the *ACCEPTED* response will be used, otherwise *REJECTED* response with the error code in the *status* field will be used.
- Transition C2f.** A bus reset transitions the port to the *C4:WAIT* state, in preparation of accepting the expected *RESTORE_PORT* or *RESTORE_PORT_FRAME* subfunction (these command restores the node ID of its connected plug). The port sets its *connect node ID* value to 0xFFFF (invalid node ID), and *connection count* to zero.
- Transition C2g.** When the previous issued attachment subfunction had been *ATTACH* (*dr* had been set to zero) and there had been an indication from an asynchronous connections layer that shows the connection is broken by connected node, timeout or other reasons, the port invalidates the port information, then transitions to *C0:FREE* state.
- Transition C2h.** When *STATUS* command is requested, the port reports the current port information by generating *STABLE* response frame.
- Transition C2i.** When the previous issued attachment subfunction had been *ATTACH_FRAME* and *dr* had been set to one, and there had been an indication from an asynchronous connections layer that shows the connection is broken by connected node, timeout or other reasons, the port invalidates the port information, then returns *REJECTED* response frame to the controller which indicates the previously issued *ATTACH_FRAME* execution had been failed. In the *REJECTED* response frame, the error code is stored in the *status* field
- Transition C2j.** Other subfunctions are rejected. The node returns *REJECTED* response to the controller with the error code in the *status* field.

State C3:INACTIVE.

In the *INACTIVE* state the port is half disabled; write or lock transactions are accepted (because the connected producer port may still be active) but lock requests are not generated by this port. While in the *C3:INACTIVE* state, the purpose of the subfunctions is to allow disconnection by transitioning the port to the *C0:FREE* state. If there had not been valid transition from *C3:INACTIVE* to *C0:FREE* within 5 seconds, the port transitions to *C0:FREE* state. These state transitions are listed below.

- Transition C3a.** The valid *RELEASE* subfunction releases the port's allocated resources and returns the port to the *C0:FREE* state. The port invalidates the port information, and returns *ACCEPTED* response to the controller, then requests to an asynchronous connections layer to release the port by *DisconnectEvent_req()*.
- Transition C3b.** After 5 seconds release timeout, the consumer port transitions to the *C0:FREE* state. The port invalidates the port information.
- Transition C3c.** A bus reset returns the port to its initial *C0:FREE* state. The port invalidates the port

- information.
- Transition C3d.** When STATUS command with subfunction set to STATUS is requested, the port reports the current port information by generating STABLE response frame.
- Transition C3e.** Other subfunctions are rejected. The port returns REJECTED response to the controller with the error code in the *status* field.
- State C4:WAIT.**
In the WAIT (post reset) state the port is disabled (asynchronous connection requests are not accepted or generated). Only RESTORE_PORT or RESTORE_PORT_FRAME subfunction, which restores the node ID value, is accepted. The intent is to temporarily maintain knowledge of pre-existing connections, so that the re-establishment of the connection can be given precedence over new connection command sequences.
- Transition C4a.** When the *suspendflg* set to FALSE (this means the port had been C2:ACTIVE state before bus reset), the valid RESTORE_PORT subfunction transitions the port to the fully connected C2:ACTIVE state, setting the connection count to one when this is done. The port stores the connected node ID values to the port information, then sets its connection count value to one (executed in *setupInfo()*). Then the port returns ACCEPTED response frame to the controller, then requests an asynchronous connections layer to restart by *RestoreEvent_req()*. Although not illustrated, the *oAPR* register on the connected producer port is also updated to re-activate asynchronous connection communications. Note that the *ex* flag shall be the same value as previously requested before the bus reset, and *dr* shall be zero.
- Transition C4b.** When the *suspendflg* set to TRUE (this means the port had been C5:SUSPENDED state before bus reset), the valid RESTORE_PORT subfunction transitions the port to the fully connected (but suspended) C5:SUSPENDED state, setting the connection count to one when this is done. The port stores the connected node ID values to the port information, then sets its connection count value to one. The port returns ACCEPTED response frame to the controller, and requests an asynchronous connections layer to restart by *RestoreEvent_req()*. Although not illustrated, the *oAPR* register on the connected producer port is also updated, to re-activate asynchronous connection communications. Note that the *ex* flag shall be the same value as previously requested before the bus reset, and the *dr* bit shall be zero.
- Transition C4c.** The valid RESTORE_PORT_FRAME subfunction transitions the port to the fully-connected C2:ACTIVE state, setting the connection count to one when this is done. The port stores the connected node ID values to the port information. The port returns an INTERIM response frame to the controller, and requests an asynchronous connections layer to restart by *RestoreEvent_req()*. Although not illustrated, the *oAPR* register on the connected producer port is also updated, to re-activate asynchronous connection communications. Note that the *ex* and *dr* values shall be the same as previously requested before the bus reset, and the *dr* bit shall be one.
- Transition C4d.** When STATUS command with subfunction set to STATUS is requested, the node reports the current port information by generating STABLE response frame.
- Transition C4e.** Other subfunctions are rejected. The node returns REJECTED response to the controller with the error code in the *status* field.
- Transition C4f.** A bus reset leaves the port in an unchanged state.
- Transition C4g.** When an asynchronous connections layer indicates *Timeout_ind()* after 10 seconds reconnection timeout, the consumer port transitions to the C0:FREE state. The port invalidates the port information.

State C5:SUSPENDED.

In the SUSPENDED state the port is suspended (asynchronous connection is

suspended to skip the frame transmissions). Only the RESUME_PORT subfunction, which re-enables the frame transmission, is accepted.

- Transition C5a.** The valid RESUME_PORT subfunction sets the asynchronous connections layer to resume from the *C5:SUSPENDED* state to *C2:ACTIVE* state. The port returns ACCEPTED response frame to the controller, then requests to an asynchronous connections layer to resume by *ResumeEvent_req()*.
- Transition C5b.** When STATUS command is requested, the node reports the current port information by generating STABLE response frame.
- Transition C5c.** The valid DETACH subfunction addressed to the port holding no overlays (*connection count == 1*) state transitions the port to a half-disconnected *C3:INACTIVE* state. After returning ACCEPTED response to the controller, the port requests to an asynchronous connections layer to stop requesting the data by *DetachEvent_req()*. And the port resets the timer to detect 5 seconds timeout for transitions from *C3:INACTIVE* to *C0:FREE*.
- Transition C5d.** A bus reset transitions the port to the *C4:WAIT* state, in preparation of accepting the expected RESTORE_PORT subfunction (this subfunction restores the node ID of its connected plug). The port sets the flag indicating that the state had been a *C5:SUSPENDED* state, then sets its *connect node ID* value to 0xFFFF (invalid node ID), and *connection count* to zero.
- Transition C5e.** The valid ADD_OVERLAY subfunction increments the connection count if the *connection count* was less than 63, leaving the port connected in its *C5:SUSPENDED* state. The node returns ACCEPTED response frame on success.
- Transition C5f.** The valid DETACH subfunction decrements the connection count if the *connection count* value was bigger than 1, leaving the port connected in its *C5:SUSPENDED* state. The node returns ACCEPTED response frame on success.
- Transition C5g.** Other subfunctions are rejected. The node returns REJECTED response to the controller with the error code in the *status* field.
- Transition C5h.** When the previous issued attachment subfunction had been ATTACH (*dr* had been set to zero) and there had been an indication from an asynchronous connections layer that shows the connection is broken by connected node, timeout or other reasons, the port invalidates the port information, then transitions to *C0:FREE* state.

7.3 Producer port states

7.3.1 Producer port state machines

As described in Section 5.6, the producer plug may contain more than one producer port to support the multicast connections.

As not the producer “plug” but the producer “port” is connected to the consumer, the producer port state transitions may occur independently from other ports within the same plug.

As the connection management commands should specify the *plugID* value, the received command is passed to its internal plug management components.

When the target received the ALLOCATE_ATTACH subfunction with “*any available port*”, the target searches the available port, then activates the port.

The behavior of a producer port is specified by its state machine definition, illustrated by the Figure 7.3.1.

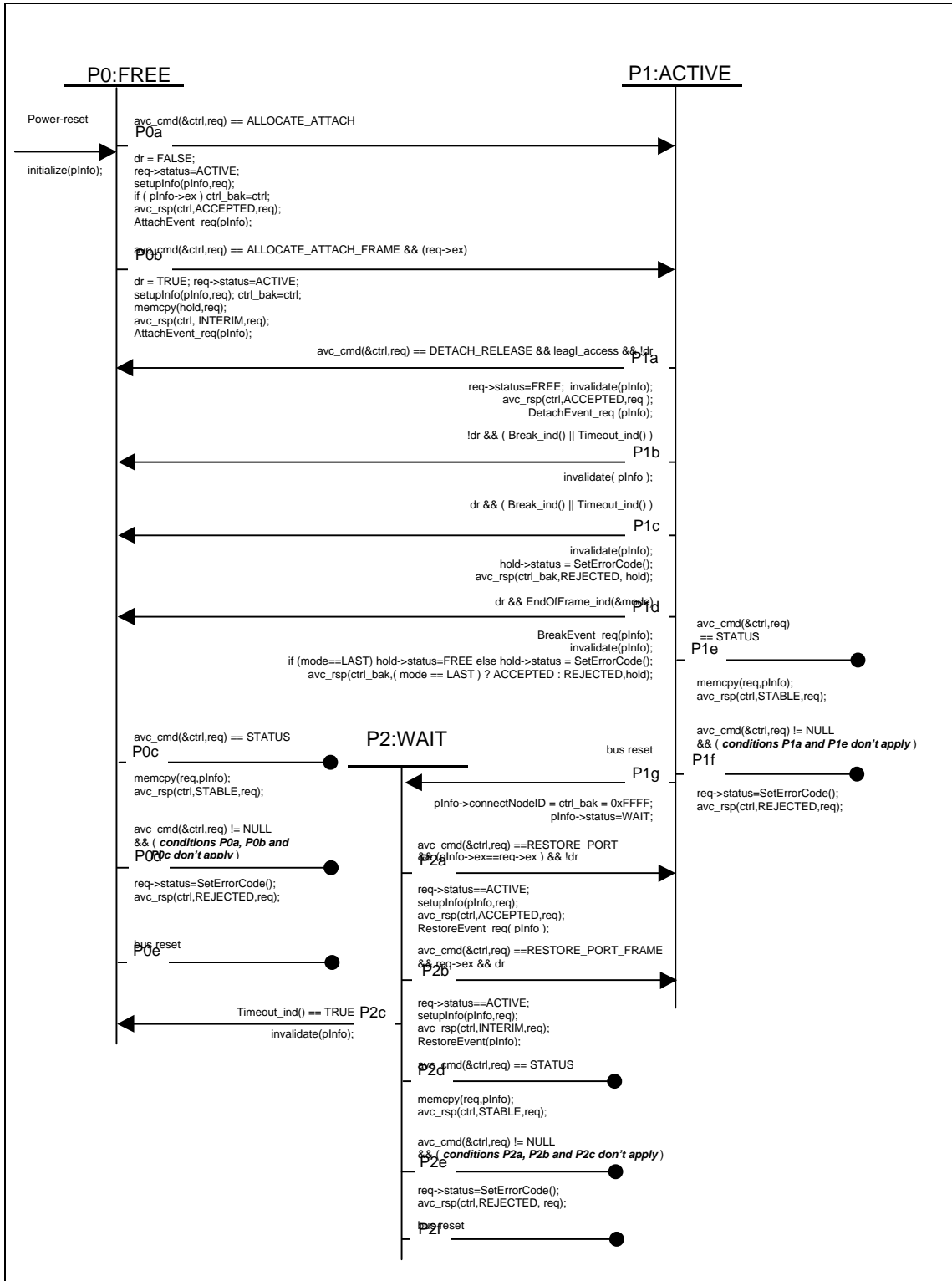


Figure 7.3.1 - Producer port state machine

7.3.2 Producer port state machine notes

State P0:FREE.

The *FREE* state is the initial state of the port, with no committed resources. While in the *P0:FREE* state, the purpose of the *ALLOCATE_ATTACH* or *ALLOCATE_ATTACH_FRAME* subfunction is to transition the port to the *PI:ACTIVE* (connected) state. These state transitions are listed below.

Transition P0a.

The valid *ALLOCATE_ATTACH* subfunction transitions the port to the connected to be *PI:ACTIVE* state. The *ALLOCATE_ATTACH* subfunction command frame includes the connection information such as consumer port address. If the *ex* bit in a command frame had been set to one, the target stores the node ID of the controller to reject the following control commands from other controllers.

The port returns *ACCEPTED* response frame to the controller, then requests to an asynchronous connections layer to start by *AttachEvent_req()*. Although not illustrated, the *oAPR* register on the connected producer port is also updated by the consumer to activate asynchronous connection communications.

Transition P0b.

The valid *ALLOCATE_ATTACH_FRAME* subfunction transitions the port to the connected to be *PI:ACTIVE* state. The *ALLOCATE_ATTACH_FRAME* subfunction command frame includes the connection information such as consumer port address. The command frame shall include *ex* bit set to one, and the port sets its internal *dr* value to one. The port stores the requested connection information (for the final response), then returns *INTERIM* response frame to the controller, then requests to an asynchronous connections layer to start by *AttachEvent_req()*. Although not illustrated, the *oAPR* register on the connected producer port is also updated by the consumer to activate asynchronous connection communications.

Transition P0c.

When *STATUS* command is requested, the node reports the current port information by generating *STABLE* response frame.

Transition P0d.

All other connection management subfunctions are rejected, generating *REJECTED* response frame with the error code in the *status* field.

Transition P0e.

Bus resets while in the *C0:FREE* state leave the port in the same state.

State P1:ACTIVE.

In the *ACTIVE* state the ports resources have been allocated and connected, and the port can accept lock transactions from its connected port. The first thing the connected consumer has to do is to set the *oAPR.run* bit to one. Only after that, the producer port can start sending data by using write transactions.

Transition P1a.

While the *dr* is zero, the valid *DETACH_RELEASE* subfunction transitions the port to an initial *P0:FREE* state. The port invalidates the port information and returns *ACCEPTED* response frame to the controller, then requests to an asynchronous connections layer to stop sending the data by *DetachEvent_req()*.

Transition P1b.

While the *dr* bit is zero, the *Break_ind()* indication (which indicates the unexpected disconnection) or the *Timeout_ind()* indication (which indicates the timeout detection) from the asynchronous connections layer, transitions the port to a initial *P0:FREE* state. The port invalidates the port information.

Transition P1c.

While the *dr* is one, the *Break_ind()* indication or the *Timeout_ind()* indication from the asynchronous connections layer transitions the port to an initial *P0:FREE* state. The port invalidates the port information and returns *REJECTED* response frame to the controller, with error code value stored in the *status* field.

Transition P1d.

While the *dr* is one, the *EndOfFrame_ind()* indication (which indicates the end of frame had been indicated by the producer port) from the asynchronous connections layer transitions the port to an initial *P0:FREE* state. The port requests to an asynchronous connections layer to break by *BreakEvent_req()* to start the

disconnection sequence described in Figure 5.9.1.

Then the port invalidates the port information. According to the mode value indicated from the asynchronous connections layer, the target returns response frame to the controller. If mode value had been LAST, which means the normal end of frame, the ACCEPTED response will be returned, otherwise REJECTED response will be returned.

Transition P1e. When STATUS command is requested, the port reports the current port information by generating STABLE response frame.

Transition P1f. All other connection management subfunctions are rejected, generating REJECTED response frame with the error code in the *status* field.

Transition P1g. A bus reset transitions the port to the P2:WAIT state, in preparation of accepting the expected RESTORE_PORT or RESTORE_PORT_FRAME subfunction (these command restores the nodeId address of its connected plug). The port sets its *connect node ID* value to 0xFFFF (invalid node ID).

State P2:WAIT.

In the WAIT (post reset) state the port is disabled (asynchronous connection requests are not accepted or generated). Only the connection management CONTROL command with *subfunction* set to RESTORE_PORT or RESTORE_PORT_FRAME, which recovers the nodeId value of the consumer node, is accepted. The intent is to temporarily maintain knowledge of pre-existing connections, so that the reestablishment of the connection can be given precedence over new connection command sequences.

Transition P2a. The valid RESTORE_PORT subfunction when *dr* bit set to zero, transitions the port to the connected P1:ACTIVE state. The port stores the connect node ID values to the port information. Then the port returns ACCEPTED response frame to the controller, then requests an asynchronous connections layer to restart by *RestoreEvent_req()*. Although not illustrated, the *oAPR* register on the connected producer port is also generated, to re-activate asynchronous connection communications. Note that the *ex* flag shall be the same value as previously requested before the bus reset, and the *dr* bit shall be zero.

Transition P2b. The valid RESTORE_PORT_FRAME subfunction when *dr* bit set to one, transitions the port to the connected P1:ACTIVE state. The port stores the connect node ID values to the port information. Then the port returns an AV/C response frame (INTERIM) to the controller, then requests an asynchronous connections layer to restart by *RestoreEvent_req()*. Although not illustrated, the *oAPR* register on the connected producer port is also updated, to re-activate asynchronous connection communications. Note that the *ex* and *dr* flags shall be the same value as previously requested before the bus reset, and the *dr* bit shall be to one.

Transition P2c. When an asynchronous connections layer indicates *Timeout_ind()* after 10 seconds reconnection timeout, the producer port transitions to the C0:FREE state. The port invalidates the port information.

Transition P2d. When STATUS command with subfunction set to STATUS is requested, the port reports the current port information by generating an AV/C STABLE response frame.

Transition P2e. Other subfunctions are rejected. The node returns an AV/C REJECTED response to the controller with the error code in the *status* field.

Transition P2f. A bus reset leaves the port in an unchanged state.