TA Document 1999010

Editorial Corrections to the Asynchronous Serial Bus Connections, Version 1.0

January 3, 1999

Sponsored by:
1394 Trade Association

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

Abstract:
This document describes the editorial corrections to the documents “AV/C Compatible Asynchronous Serial Bus Connections, Version 1.0” and “AV/C commands for management of Asynchronous Serial Bus Connections, Version 1.0”.

Keywords:
Audio, Video, 1394, Digital, Interface, Asynchronous, Connection, Management.

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

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

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

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

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

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

Contact Information

Much of the information in this document is preliminary and subject to change. Members of the AVWG are encouraged to review and provide inputs for this proposal. For document status updates, please contact:

Takahiko Nankou, Workitem Project Leader
Matsushita Electric Industrial Co., Ltd.
NBD Group, 4th NBD Project
AVC Products Development Laboratory, AVC Company
2-15, Matsuba-cho, Kadoma,
Osaka, 571-8503 Japan
E-Mail: nankou@avrl.mei.co.jp
Phone: +81-6-6901-1161
Fax: +81-6-6906-6549

You can also submit comments using the 1394 TA reflector. Its web site address is:

www.1394ta.org

Go to the members-only section and select the AVWG reflector. You will need to have signed up for the reflector.

NOTE —The information on this page should be removed when this document is accepted.

Table of Contents

List of Figures

List of Tables

1. Overview

1.1 Purpose of this document

This document describes the editorial corrections to the following two documents:

TA Document 1998016 : "AV/C Compatible Asynchronous Serial Bus Connections, Version 1.0, January 26,1999"

TA Document 1998011 : "AV/C commands for management of Asynchronous Serial Bus Connections, Version 1.0, January 26,1999"

The corrections described in this document will be included in the future version of the above two documents.

2. References

This standard shall be used in conjunction with the following publications.

[R1] TA Document 1998016, AV/C Compatible Asynchronous Serial Bus Connections, Version 1.0, January 26, 1999

[R2] TA Document 1998011, AV/C management commands for management of Asynchronous Serial Bus Connections, Version 1.0, January 26, 1999

3. Definitions

3.1 Conformance Levels

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

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

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

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

3.1.5 reserved codes: A set of codes for a reserved field that are defined in this specification, but not otherwise used. Future specifications may implement the use of these codes. A product implementing this specification shall not generate these codes.

3.1.6 reserved fields: A set of bits for a reserved field that are defined in this specification, but are not otherwise used. Products that implement this specification shall zero these fields. Products that implement future revisions of this specification may set these codes as defined by the specification.

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

3.2 Glossary of Terms

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

3.2.2 CSR Architecture: A convenient abbreviation of the following reference (see clause 2): ISO/IEC 13213 : 1994 [ANSI/IEEE Std 1212, 1994 Edition], Information Technology—Microprocessor systems—Control and Status Register (CSR) Architecture for Microcomputer Buses.

3.2.3 quadlet: Four bytes of data.

3.3 Acronyms and Abbreviations

Many bus and interconnect-related technical terms are used in this document. These terms are described below:

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

3.3.2 asynchronous-connection consumer (abbreviated as **consumer**): The component of a node that consumes data frames provided by the asynchronous connection producer.

3.3.3 asynchronous-connection producer (abbreviated as **producer**): The component of a node that produces data frames for consumption by the asynchronous connection consumer.

3.3.4 connection: The attachment of a producer plug to a consumer plug for the purpose of sending an asynchronous stream of data frames.

3.3.5 consumer: (see asynchronous connection consumer).

3.3.6 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.

3.3.7 compound plug: A collection of plugs that can be simultaneously connected to matching set of plugs using one sequence of connection-establishment commands.

3.3.8 consumer port: A port that is the sink of data frames and is flow controlled by updates of its externally visible *iAPR* control register.

3.3.9 data frame (abbreviated as **frame**): A contiguous group of data bytes sent between producer and consumer.

3.3.10 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*..

3.3.11 file-type transfers. An asynchronous connection data transfer that has no real-time delivery constraints that could force discarding of selected frames (as distinguished from stream-type transfers).

3.3.12 frame: (see data frame).

3.3.13 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 that are used for demarcation of variable-length frames, and to support the connection disconnect sequence.

3.3.14 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 that are used for demarcation of variable-length frames, and to support the connection disconnect sequence.

3.3.15 payload: The portion of a request or response packet that contains data defined by an application layer.

3.3.16 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.

3.3.17 port: A subcomponent of a plug that supports unidirectional data transfers.

3.3.18 producer: (see asynchronous connection producer).

3.3.19 producer port: A port that is the source of data frames and is flow controlled by updates of its externally visible *oAPR* control register.

3.3.20 segment: (see data segment).

3.3.21 segment buffer: An externally visible address space on a consumer into which data is written by the connected producer.

3.3.22 stream-type transfers. An asynchronous connection data transfer that has real-time delivery

constraints, where these delivery constraints can force discarding of selected frames (as distinguished from file-type transfers).

4. Editorial corrections

4.1 Corrections to "AV/C Compatible Asynchronous Serial Bus Connections, Version 1.0"

4.1.1 The definitions of "LESS" and "LOST"

In the page 31 of the document [R1], "LESS" and "LOST" values are defined as follows.

Table 4.1 – iAPR.mode values

mode	Name	Description
0	FREE	Producer initiated disconnection sequence (Initial state)
1	MORE	The frame has not yet ended
2	SUSPENDED	Suspend confirmation; suspended frame transfers
3	-	reserved
4	LAST	A successful frame transfer
5	LESS	A truncated-length frame was transferred
6	JUNK	A corrupted frame was transferred
7	LOST	A zero-length pseudo frame, making a discarded frame location

The following is the detailed description of LESS value:

> "The **LESS** indication labels the **last segment in abnormal frame**, when the abnormal frame contains valid data but the frame was truncated early."

The following is the detailed description of "LOST" value:

> "The **LOST** indication labels a zero length **frame**, marking where one or more **frames** have been lost."

On the other hand, State Machine definitions and other parts of the document describe the "LOST" indication as the confirmation of "TOSS" indication.

Furthermore, in case that frame transmission had been aborted by "TOSS" indication, if any segment data had been transferred, the confirmation should be "LESS" but there are some descriptions that "LOST" is used. These are considered as editorial errors.

4.1.2 List of the figures to be corrected

The following is Figure 27 from page 37 of the document [R1], where the "LOST" indication is incorrect, shall be "LESS", because the 32KB of the frame A had been transmitted:

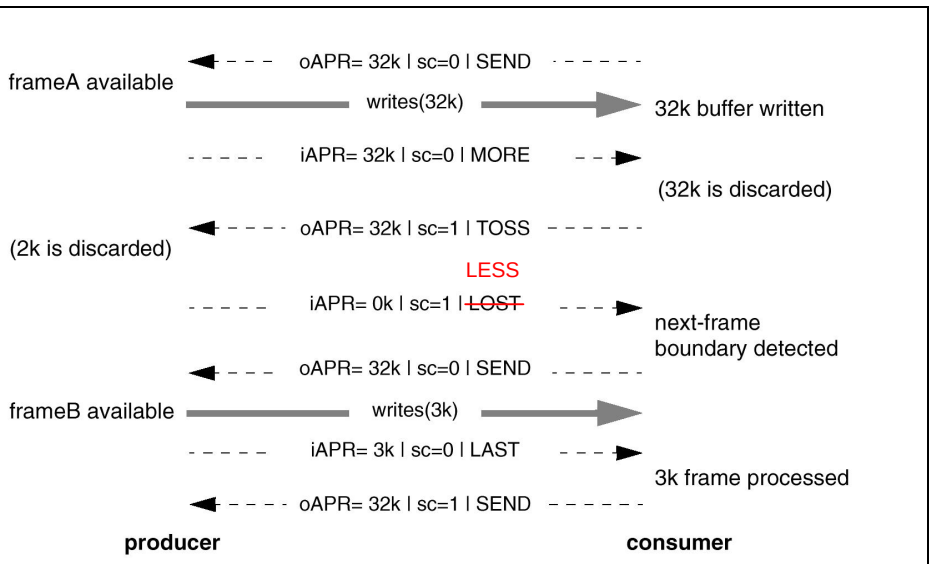

Figure 27—34k/3k data frame illustration, first frame discarded

Figure 4.1 – Correction of "Figure 27" of the document [R1]

The following is Figure 28 from page 38 of the document [R1], where the "LOST" indication is incorrect, shall be "LESS", because the 32KB of the frame A had been transmitted:

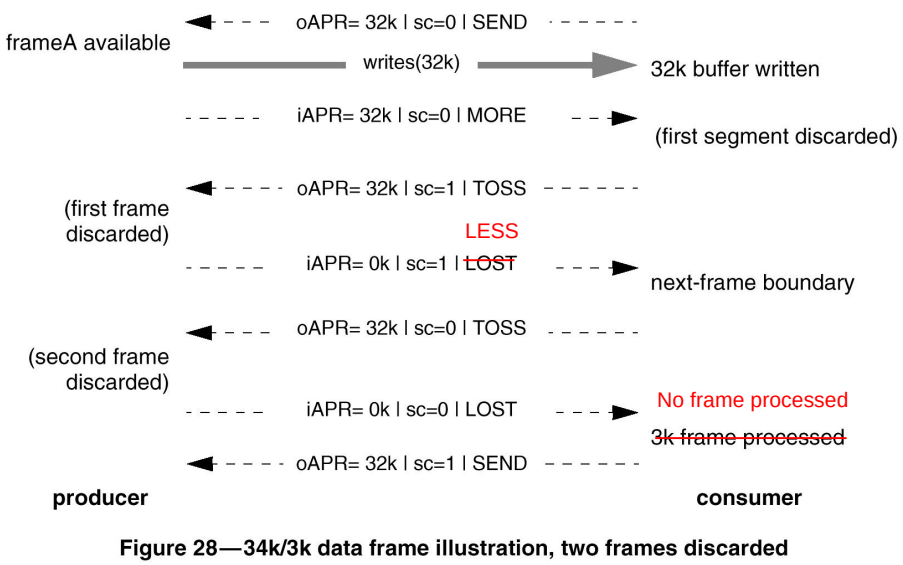

Figure 28—34k/3k data frame illustration, two frames discarded

Figure 4.2 – Correction of "Figure 28" of the document [R1]

4.1.3 List of editorial corrections

The following table is the list of corrections of the document [R1].

Table 4.2 – List of corrections of the document [R1]

Where	Incorrect description	Correct description
Page 37, Figure 27	iAPR= 0k | sc=1 | LOST	iAPR= 0k | sc=1 | LESS
Page 37, last paragraph	The consumer returns a TOSS indication, acknowledging the *iAPR* update but indicating the segment buffer was discarded due to a congested consumer condition. In this example, the producer commits to discarding this frame and provides the consumer with a LOST indication.	The consumer returns a TOSS indication, acknowledging the *iAPR* update but indicating the segment buffer was discarded due to a congested consumer condition. In this example, the producer commits to discarding this frame and provides the consumer with a LESS indication.
Page 38, Figure 28	iAPR= 0k | sc=1 | LOST	iAPR= 0k | sc=1 | LESS
	3k frame processed	No frame processed
Page 38, last paragraph	The consumer returns a TOSS indication, acknowledging the *iAPR* update but indicating the segment buffer was discarded due to a congested consumer condition. After the producer has provided a confirming LOST indication, the consumer has the option of discarding the second segment by providing a TOSS (rather than SEND) indication.	The consumer returns a TOSS indication, acknowledging the *iAPR* update but indicating the segment buffer was discarded due to a congested consumer condition. After the producer has provided a confirming LESS indication, the consumer may discard the second frame by providing a TOSS indication.
Page 50, Figure 40: TX1b actions	iAPRCopy.mode = LOST;	iAPRCopy.mode = EndOfFrame(iAPRCopy.mode) ? LOST : LESS ;
Page 53, TX1b description	TX1b. A changed oAPR with a TOSS indication causes the remainder of the frame to be discarded and a LOST confirmation is provided.	TX1b. When iAPRCopy.mode == MORE, in this case, an updated oAPR as a TOSS indication causes the remainder of the frame to be discarded and a LESS indication is provided. In the others (such as LAST, LESS, JUNK, LOST), an updated oAPR as a TOSS indication causes one or more Frame to be discarded and a LOST indication is provided.
Page 58, Figure 43, RX3c condition	BusRestEvent() == TRUE	BusResetEvent() == TRUE
Page 31	NOTE—The LOST indication is intended to be used by stream-type subunits, and is not expected to be used by file-type subunit.	(remove)

Where	Incorrect description	Correct description
Page 60, explanation of "Flush()"	It is application dependent that the consumer transfers or discards. When buffer space for the next frame is available, this function returns **SEND** and otherwise returns **WAIT**.	It is application dependent that the consumer transfers or discards. When buffer space for the next frame is available, this function returns **SEND** and otherwise returns **WAIT**. If iAPR.mode is LAST or JUNK or LESS or LOST, this function issues the EndOfFrameInd(iAPR.mode) to the application layer.
Page 50~51, Page 57~59 transitions TX1h, TX3e, TX4F, TX5c, TX6c, RX1f, RX4e, RX5c, RX6c, RX8b, RX9b	TimeoutEvent()	TimeoutInd()
Page 52, 60	The TimeoutEvent() returns TRUE to the application layer when a timeout occurs.	The TimeoutInd() indicates the application layer that a timeout occurs.
Page 52, 60	(no description)	The *break_ind()* indicates the application layer that the connection had been broken.

4.2 Corrections to "AV/C commands for management of Asynchronous Serial Bus Connections, Version 1.0"

4.2.1 Description about "Reserved locations"

The section 3.7.1 "Reserved locations", page 7 shall be replaced with the following sentence.

3.7.1 Reserved fields

The reserved field shall be set as defined in the section 3.1, "Rules for the reserved fields", page 5 of "Enhancements to the AV/C General Specification 3.0, Version 1.0, January 26,1999".

4.2.2 Definition of "write interval " field

In the page 27 of the document [R2], says that "write interval" field has no meaning when ct=0.

The background of this definition had been that "if *ct* = 0, producer node shall not issue the next write request, producer would not receive ack_busy".

But, even if the *ct* bit is zero, the consumer port might return "ack_busy".

- Considerable situation:

- ✓ There might be another producer node writing into the another segment buffer on the consumer node.
- ✓ The controller may issue (a lot of) AV/C commands to the consumer during the segment writes.

In such situations above, the consumer may return "ack_busy" to the producer, because it might be busy to process transactions requested from other node than the connected producer, although the previous write transaction had been completed.

Therefore, it should be defined that "write interval field is valid regardless of *ct* value".
The definition of "write interval" field should be changed as follows:

4.2.3 List of editorial corrections

The following table is the list of corrections of the document [R2].

Table 4.3 – List of corrections of the document [R2]

Where	Incorrect description	Correct description
Page 26, "write interval" description	***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: (**Interval Time**) = (NOMINAL_CYCLE_TIME) x 2 $^{(write\ interval)}$ If the *ct* bit of the consumer port is zero, this field has no meaning and its value is F_{16}.	If the *ct* bit of the consumer port is one, ***write interval*** indicates the required interval of Serial Bus Write request (TR_DATA.req) that would be issued to the consumer port, to avoid a number of "ack_busy" returning. If the *ct* bit of the consumer port is zero, this field indicates the required interval from failed transaction (TR_DATA.conf with "Request status = RETRY LIMIT") to the next write request retry (TR_DATA.req). Interval time of write transactions request can be calculated as follows: (**Interval Time**) = (NOMINAL_CYCLE_TIME) x 2 $^{(write\ interval)}$ The detailed definitions of "TR_DATA.req" and "TR_DATA.conf" are described in the reference [1] (Section 7.1.2).
Page 64, FIgure 7.3.1 P2e conditions	avc_cmd(&ctrl,req) != NULL && (***conditions P2a, P2b and P2c don't apply***)	avc_cmd(&ctrl,req) != NULL && (***conditions P2a, P2b and P2d don't apply***)
Page 65, Transition P1d.	While the *dr* isone, …	While the *dr* is one, …
Page 18	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).	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 or LOST (6).

Where	Incorrect description	Correct description
Page 18	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), …	If error condition occurred during a single frame transmission (*iAPR* register's ***mode*** value was LESS, JUNK or LOST), and the connection might have been broken), …
Page 19	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).	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 or LOST (6).
Page 19	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), …	If error condition occurred during a single frame transmission (*iAPR* register's ***mode*** value was LESS, JUNK or LOST), and the connection might have been broken), …
Page 54, Last sentence	When a STATUS command is issued to the consumer port, *port ID* value shall be set to 0_{16}.	When a STATUS command is issued to the producer port, *port ID* value shall be set to specified port ID value (1_{16} ~E_{16}), which identifies the producer port.
Page 55, above Table 6-4.3	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.	When a STATUS command is issued to the consumer port, *port ID* value shall be set to 0_{16}.

Page 58,
C1a action
	avc_rsp(ctrl,ACCEPTED,req);
	AttachEvent_req(cinfo);
		avc_rsp(ctrl,ACCEPTED,req);
		if(cint->ex) ctrl_bak=ctrl;
		AttachEvent_req(cinfo);

Page 58, C1b condition	avc_cmd(&ctrl,req)==ATTACH_FRAME && ex && legal_access	avc_cmd(&ctrl,req)==ATTACH_FRAME && req->ex && legal_access

Page 58,
C1b action
	memcpy(hold,req);
	avc_rsp(ctrl,INTERIM,req);
		memcpy(hold,req);
		ctrl_bak=ctrl;
		avc_rsp(ctrl,INTERIM,req);

Page64,
P2a action
	setupInfo(pinfo,req);
	avc_rsp(ctrl,ACCEPTED,req);
		setupInfo(pinfo,req);
		if(pinfo->ex) ctrl_bak=ctrl;
		avc_rsp(ctrl,ACCEPTED,req);

Page 64,
P2b action
	setupInfo(pinfo,req);
	avc_rsp(ctrl,INTERIM,req);
		setupInfo(pinfo,req);
		memcpy(hold,req);
		ctrl_bak=ctrl;
		avc_rsp(ctrl,INTERIM,req);