TA Document 2001007
AV/C Music Subunit 1.0

April 8, 2002

Sponsored by:
1394 Trade Association

Accepted for Release by:
This document has not yet been accepted for release by the 1394 Trade Association Board of Directors.

Abstract:
This specification defines a model, data structure, command set for AV/C Music Subunit, operating over IEEE Std 1394-1995

Keywords:
MIDI, audio, SMPTE time code, Sample Count, audio SYNC, Music plug.

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

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

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

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

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

Editor, 1394 Trade Association
1111 South Main Street, Suite 100
Grapevine, TX, 76051, USA
Tel: (817) 410-5750

Table of contents

List of figures

List of tables

1. Overview

1.1 Purpose

The purpose of this document is to describe the information on the data structures and a command set of electric musical instruments and/or professional audio equipment that have MIDI functionality when they are connected to the IEEE1394 serial bus.

This document provides the method to establish the connection between devices that can use hundreds of MIDI, audio, SMPTE time code and Sample Count data signal flows on IEEE1394 physical layer. Especially the methods provided ensure low traffic volume and fast data processing.

Furthermore, in order to synchronize the audio and MIDI data between devices, this document provides the method to establish the audio reference synchronization as defined in “Audio and Music Data Transmission Protocol 2.0”[R12].

1.2 Scope

This specification defines the method of connecting AV/C devices, equipped with MIDI interfaces based on IEC 61883.

MIDI data is basically a command stream, such as music information and remote device control. Multiple MIDI command streams are defined in “Audio and Music Data Transmission Protocol 2.0”[R12]as Multiplexed MIDI. This specification specifies how to connect the multiplexed MIDI streams to a subunit which is defined as AV/C Music Subunit based on this document.

In many cases, MIDI devices also process audio, SMPTE time code and Sample Count data. This specification provides a method to compound and/or de-compound those data stream with multiplexed MIDI stream on AV/C Music Subunit.

This specification concerns itself with the syntax and semantics of commands and the structure of AV/C descriptors and information blocks.

2. References

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

[R1] IEEE Std 1394-1995, Standard for a High Performance Serial Bus.

[R2] IEC 61883-1, Consumer audio/video equipment – Digital interface – Part 1: General.

[R3] AV/C Digital Interface Command Set General Specification, Version 3.0. TA document number 1998003.

[R4] Enhancement to the AV/C General Specification 3.0. TA document number 1998010.

[R5] Enhancements to the AV/C General Specification 3.0 Version 1.1 TA document number 2000004

[R6] AV/C Digital Interface Command Set General Specification, Version 4.0. TA document number 1999026

[R7] AV/C Descriptor Mechanism Specification Version 1.0 TA document number 1999025.

[R8] AV/C Information Block Types Specification Version 1.0 TA document number 1999045.

[R9] AV/C Connection and Compatibility Management Specification 1.0. TA document number 1999031.

[R10] Audio and Music Data Transmission Protocol Version 1.0. TA document number 1997001.

[R11] Enhancement to Audio and Music Data Transmission Protocol 1.0. TA document number 1999014.

[R12] Audio and Music Data Transmission Protocol 2.0 TA document number 2001003

[R13] SMPTE Time Code and Sample Count Transmission Protocol Ver.1.0 TA document number 1999024.

[R14] RP-027, Specification for MIDI Media Adaptation Layer for IEEE1394

[R15] IEC 61883-6 PAS, Audio and music data transmission protocol

[R16] IEC 60958-1, Digital audio interface - Part 1: general

[R17] IEC 60958-3, Digital audio interface - Part 3: Consumer applications

[R18] IEC 60958-4, Digital audio interface - Part 4: Professional applications

[R19] IEC 61937, Digital audio - Interface for non-linear PCM encoded audio bitstreams applying IEC 60958

[R20] IEEE Std 754-1985, Binary Floating-Point Arithmetic

[R21] Complete MIDI 1.0 Detailed Specification

3. Definitions

3.1 Conformance levels

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

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

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

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

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

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

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

3.2 Glossary of terms

3.2.1 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.

3.2.4 AV/C descriptor: A data structure with defined fields used for sharing information with and/or modifying information by other AV/C devices. Descriptors have a set of commands for manipulation of their contents.

3.2.5 info block: An information block is a data structure embedded within a descriptor that describes acommon type of AV/C data.

3.2.6 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."

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

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

3.2.9 controller: A device that issues AV/C commands to targets.

3.2.10 target: A device that receives AV/C commands from controllers.

3.2.11 source: A subunit source plug, unit output plug, or unit input plug that serves as the originating source of data streams. There are some restrictions on which type of plug can be a source, depending on what type of connection is referred.

3.2.12 destination: A subunit destination plug, unit input plug, or unit output plug that receives data streams transmitted by a source. There are some restrictions on which type of plug can be a destination, depending on what type of connection is referred.

3.2.13 AM824: A 32-bit data field that has an 8-bit label and 24-bit data field defined in Audio and Music Data Transmission Protocol Ver. 1.0.

3.2.14 MPX MIDI Data: Channel:A discontiguous sequence of AM824 elements used to carry a particular MIDI data stream. A number of MPX-MIDI Data Channels are multiplexed together within a single MIDI Conformant Data Channel. MPX-MIDI Data Channels are used only inside of the IEEE1394 media adaptation layer.

3.3 Acronyms and abbreviations

AV/C	Audio Video Control
IEEE	The Institute of Electrical and Electronics Engineers, Inc.
MIDI	Musical Instrument Digital Interface
SMPTE	Society of Motion Picture & Television Engineers
SYNC	Synchronization
FS	Frequency Sampling
lsb	least significant bit
LSB	Least Significant Byte
msb	most significant bit
MSB	Most Significant Byte

4. Music Subunit Logical Model

4.1 The AV/C Logical Model

The following diagram illustrates the logical model of a Music subunit.

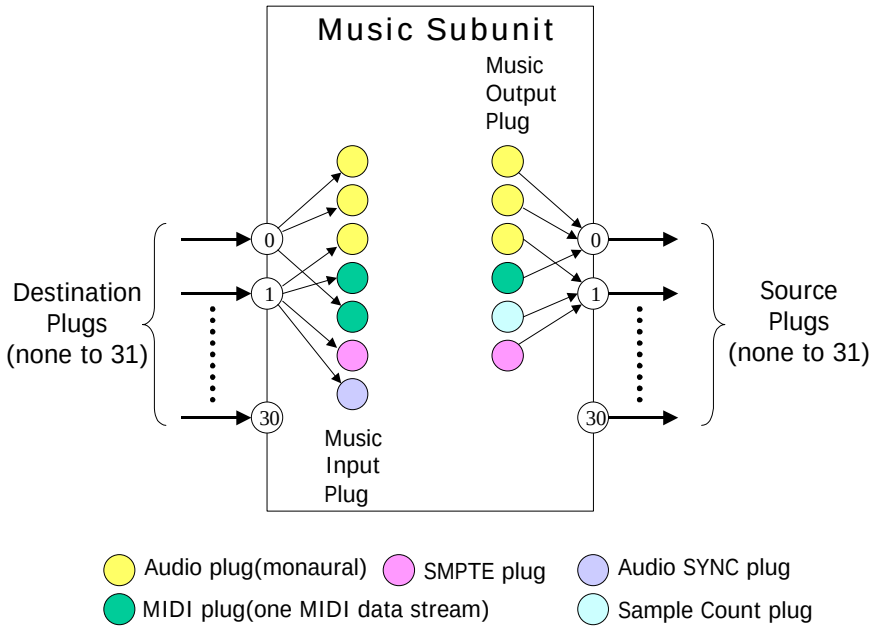


Figure 4.1 – Logical Model of a Music Subunit

Data streams such as MIDI command/control, audio, SMPTE and Sample Count are carried by isochoronous channels. Incoming data streams are connected to subunit destination plugs on the Music subunit. Within the component, each subunit destination plug may be connected to one or more Music input plugs, so that the component can process or use the stream.

Data streams generated by the Music subunit are sourced at one or more Music output plugs. A Music output plug may be connected to a subunit source plug in order to generate a data stream on an outgoing isochronous channel.

One subunit destination plug may be connected to one or more Music input plugs. One or more Music output plugs may be connected to a single subunit source plug.

destination plug[0] – [30] –> Music subunit –> source plug[0] – [30]

Within a Music subunit, some of the attributions for the end points of

Music input plug[0] – [65534]

Music output plug[0] – [65534]

are described in the info block and the remaining part of the attributions are able to get by the command directly.

Music plug types are;

Audio (monaural)

MIDI (for one MIDI data stream)

SMPTE time code

Sample Count

Audio SYNC

5. Music Subunit Identifier Descriptor

The general subunit identifier structure is described in reference [R3]. The Music subunit identifier descriptor contains the following information:

Address	Length, Byte	External Read/Writ	Contents
00 00$_{16}$	2	R	descriptor_length
00 01$_{16}$			
00 02$_{16}$	1	R	generation_ID
00 03$_{16}$	1	R	size_of_list_ID = i
00 04$_{16}$	1	R	size_of_object_ID
00 05$_{16}$	1	R	size_of_object_position
00 06$_{16}$	2	R	number_of_root_object_lists = n
00 07$_{16}$			
00 08$_{16}$	i	R	root_object_list_id[0]
:			
:			:
:	i	R	root_object_list_id[n-1]
:			
:	2	R	music_subunit_type_dependent_information_length = j
:			
:	j	R/W	music_subunit_type_dependent_information
:			
:			
:	2	R	manufacture_dependent_information_length = k
:			
:	k	R/W	manufacture_dependent_information
:			
:			

Figure 5.1 – Music Subunit Identifier Descriptor

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

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

Table 5.1– Generation ID Values

generation_ID	Meaning
00_{16}	Data structure and command sets as specified in the AV/C General Specification version 3.0.
01_{16}	Data structure and command sets as specified in the AV/C General Specification version 3.0 and the Enhancement to the AV/C General Specification 3.0, version 1.0 and 1.1.
02_{16}	Data structure and command sets as specified in the AV/C General Specification version 4.0
all others	reserved for future specification

5.1 Music Subunit Dependent Information

The *music_subunit_dependent_information* field indicates the fixed information which depend on the Music subunit of a subunit and the format of it is as follows:

Address	Length, Byte	External Read/Writ	Contents
00 00_{16}	2	R	music_subunit_dependent_info_fields_length
00 01_{16}			
00 02_{16}	1	R	attributes
00 03_{16}	1	R	music_subunit_version
00 04_{16}	2	R	music_subunit_specific_information_length = i
00 05_{16}			
00 06_{16}	i	R	music_subunit_specific_information
:			
:	k	R	optional info blocks for future expansion
:			

Figure 5.2 – Music Subunit Dependent Information

attributes: The *attributes* field is defined as follows:

Table 5.2 – Attributes Field

Attribute Value	Meaning
1xxx xxxx	**Has more attributes:** If this bit is set to 1, then the next byte is also an attributes byte. If this bit is 0, then the next byte is as defined for this structure.
all others	reserved for future specification

music_subunit_version: The *music_subunit_version* field indicates the version number of Music subunit. The upper 4bits shows major version number, and lower 4 bits shows minor version number.

Table 5.3 – Music Subunit Version

music_subunit_version	Meaning
10_{16}	Version 1.0 of the AV/C Music Subunit specification
all others	reserved for future specification

5.2 Music Subunit Specific Information

The format of music_subunit_specific_information is as follows:

Address	Length, Byte	External Read/Writ	Contents
00 00_{16}	1	R	capability_attributes
00 01_{16}	i	R	general_capability
:			
:	j	R	audio_capability
:			
:	k	R	MIDI_capability
:			
:	l	R	SMPTE_time_code_capability
:			
:	m	R	Sample_count_capability
:			
:	n	R	audio_SYNC_capability
:			

Figure 5.3 – Music Subunit Specific Information

capability_attribute: The *capability_attribute* field indicates the existence of a capability field that is specific to each type of plug. It is defined as follows:

Table 5.4 – Capability Attribute

Attribute Value	Meaning
1xxx xxxx	**Has more attributes:** If this bit is set to 1, then the next byte is also an attributes byte. If this bit is 0, then the next byte is as defined for this structure.
xxxx xxx1	**General capability:** If this bit is set to 1, then the *music_subunit_specific_information* field contains *general_capability* field.
xxxx xx1x	**Audio capability:** If this bit is set to 1, then the *music_subunit_specific_information* field contains *audio_capability* field.
xxxx x1xx	**MIDI capability:** If this bit is set to 1, then the *music_subunit_specific_information* field contains *MIDI_capability* field.
xxxx 1xxx	**SMPTE_time_code capability:** If this bit is set to 1, then the *music_subunit_specific_information* field contains *SMPTE_time_code_capability* field.
xxx1 xxxx	**Sample Count capability:** If this bit is set to 1, then the *music_subunit_specific_information* field contains *Sampel_count_capability* field.
xx1x xxxx	**Audio SYNC capability:** If this bit is set to 1, then the *music_subunit_specific_information* field contains *audio_SYNC_capability* field.
all others	reserved for future specification

5.2.1 General Capability

The *general_capability* field indicates the information about general capability of the subunit and it contains the following information:

Address Offset	Length, Byte	External Read/Write	Contents
00 00$_{16}$	1	R	general_capability_field_length = i - 1
00 01$_{16}$	i -1	R	general_capability_information
:			

Figure 5.4 – General Capability Field

general_capability_field_length: The *general_capability_field_length* field specifies the number of bytes for the remainder of this field.

general_capability_information: The *general_capability_information* field contains the following information.

Address	Length, bytes	External Read/Write	Contents
00 00_{16}	1	R	transmit_capability
00 01_{16}	1	R	receive_capability
00 02_{16}	4	R	latency_capability = FFFF $FFFF_{16}$
:			
:			

Figure 5.5 – General Capability Information

transmit_capability: The *transmit_capability* field specifies the capability of transmission. The following table illustrates the *transmit_capability*.

Table 5.5 – Transmit Capability

Transmit Capability	Meaning
xxxx xxx1	**Non-Blocking:** If this bit is set to 1, then this subunit can transmit the data with Non-Blocking transmission method.
xxxx xx1x	**Blocking:** If this bit is set to 1, then this subunit can transmit the data with Blocking transmission method.
all others	reserved for future specification

receive_capability: The *receive_capability* field specifies the capability of transmission. The format of *receive_capabilit* is as follows.

Table 5.6 – Receive Capability

Receive Capability	Meaning
xxxx xxx1	**Non-Blocking:** If this bit is set to 1, then this subunit can receive the data with Non-Blocking transmission method.
xxxx xx1x	**Blocking:** If this bit is set to 1, then this subunit can receive the data with Blocking transmission method.
all others	reserved for future specification

latency_capability: The *latency_capability* field specifies the latency of transmission. This field is reserved for future specification. The size of this field is 4 Bytes and should be set to FFFF $FFFF_{16}$ at this document.

5.2.2 Audio Capability

The *audio_capability* field of *music_subunit_specific_information* indicates the information about audio capability of the subunit and it contains the following information:

Address Offset	Length, Byte	External Read/Write	Contents
00 00$_{16}$	1	R	audio_capability_field_length = j - 1
00 01$_{16}$	j - 1	R	audio_capability_information
:			

Figure 5.6 – Audio Capability Field

audio_capability_field_length: The *audio_capability_field_length* field specifies the number of bytes for the remainder of this field.

The *audio_capability_information* field contains the following information.

Address Offset	Length, Byte	External Read/Write	Contents
00 00$_{16}$	1	R	available_audio_format_info_number = p
00 01$_{16}$	6	R	available_audio_format_info[0]
00 02$_{16}$			
00 03$_{16}$			
00 04$_{16}$			
00 05$_{16}$			
00 06$_{16}$			
:			:
:	6	R	available_audio_format_info[p-1]
:			
:			
:			
:			
:			

Figure 5.7 – Audio Capability Information

available_audio_format_info_number: The *available_audio_format_info_number* field specifies the number of formats that are available in the subunit.

The *available_audio_format_info[]* field contains the following information.

Address Offset	Length, Byte	External Read/Write	Contents
00 00$_{16}$	2	R	number_of_maximum_audio_input_channels
00 01$_{16}$			
00 02$_{16}$	2	R	number_of_maximum_audio_output_channels
00 03$_{16}$			
00 04$_{16}$	2	R	available_audio_format
00 05$_{16}$			

Figure 5.8 – Available Audio Format Info

number_of_maximum_audio_input_channels: The *number_of_maximum_audio_input_channels* field specifies the maximum number of audio input channels that can be connected to the subunit.

number_of_maximum_audio_output_channels: The *number_of_maximum_audio_output_channels* field specifies the maximum number of audio output channels that can be connected to the subunit.

available_audio_format: The format *available_audio_format* field contains following information.

Address	Length, bytes	External Read/Write	Contents
00 00_{16}	1	R	FDF
00 01_{16}	1	R	AM824 LABEL

Figure 5.9 – Available Audio Format

FDF: The *FDF* field is defined in "Audio and Music Data Transmission Protocol 2.0"[R12].

AM824 LABEL: The *AM824 LABEL* field is also defined in "Audio and Music Data Transmission Protocol 2.0"[R12]. The following table illustrates the relation between *FDF* field and *AM824_LABEL* field in *available_audio_format* field. For more detail information, "Audio and Music Data Transmission Protocol 2.0"[R12] shall be referred.

Table 5.7 – Relation of FDF and AM824 LABEL

FDF	AM824 LABEL
00_{16}	00_{16} to FF_{16}
all others	Always set to 00_{16}

The detailed encoding of *AM824 LABEL* is as follows:

Table 5.8 – AM824 LABEL Value

Value	Meaning
00_{16} – $3F_{16}$	IEC 60958 conformant
40_{16} – $4F_{16}$	Multi-bit Linear Audio
50_{16} – 57_{16}	One Bit Audio(Plain)
58_{16} – $5F_{16}$	One Bit Audio(Encoded)
60_{16} – 67_{16}	High Precision Multi-bit Linear Audio
68_{16} – $7F_{16}$	reserved
80_{16} – 83_{16}	MIDI Conf.
84_{16} – 87_{16}	reserved
88_{16} – $8B_{16}$	SMPTE time code conformant
$8C_{16}$ – $8F_{16}$	Sample Count
90_{16} – BF_{16}	reserved
$C0_{16}$ – EF_{16}	Ancillary Data
others	reserved

5.2.3 MIDI Capability

The MIDI capability field shows the MIDI version number, revision number and MIDI adaptation layer version that is specified in “RP-027, Specification for MIDI Media Adaptation Layer for IEEE1394”[R14]. For more detail information, “RP-027, Specification for MIDI Media Adaptation Layer for IEEE1394” shall be referred.

The *MIDI_capability* field of *music_subunit_specific_information* contains the following information:

Address Offset	Length, Byte	External Read/Write	Contents
00 00_{16}	1	R	MIDI_capability_field_length = k - 1
00 01_{16}	k - 1	R	MIDI_capability_information
:			

Figure 5.10 – MIDI Capability Field

MIDI_capability_field_length: *The MIDI_capability_field_length* field specifies the number of bytes for the remainder of this field.

The *MIDI_capability_information* field contains the following information.

Address Offset	msb							lsb
00 00_{16}	MIDI_version				MIDI_revision			
00 01_{16}	MIDI_adaptation_layer_version							
00 02_{16}	number_of_maximum_MIDI_input_ports							
00 03_{16}								
00 04_{16}	number_of_maximum_MIDI_output_ports							
00 05_{16}								

Figure 5.11 – MIDI Capability Information

MIDI_version, MIDI_revision: The *MIDI_version* and *MIDI_revision* field specifies the version of MIDI that is described in "Complete MIDI 1.0 Detailed Specification"[R21].

MIDI_adaptation_layer_version: The *MIDI_adaptation_layer_version* field specifies the version of MIDI adaptation layer that is specified in "RP-027, Specification for MIDI Media Adaptation Layer for IEEE1394"[R14].

Table 5.9 – MIDI Adaptation Layer Version

MIDI Adaptation Layer Version	Meaning
00_{16}	**RP-027:** MIDI adaptation layer as specified in the RP-027, Specification for MIDI Media Adaptation Layer for IEEE1394[R14]
all others	reserved for future specification

number_of_maximum_MIDI_input_ports: The *number_of_maximum_MIDI_input_ports* field specifies the maximum number of MIDI input ports that can be received in the subunit.

number_of_maximum_MIDI_output_ports: The *number_of_maximum_MIDI_output_ports* field specifies the maximum number of MIDI output ports that can be transferred from the subunit.

5.2.4 SMPTE Time Code Capability

The *SMPTE_time_code_capability* field of *music_subunit_specific_information* indicates the information about SMPTE time code conformant capability of the subunit and it contains the following information:

Address Offset	Length, Byte	External Read/Write	Contents
00 00_{16}	1	R	SMPTE_time_code_capability_field_length = l - 1
00 01_{16}	l - 1	R	SMPTE_time_code_capability_information
:			

Figure 5.12 – SMPTE Time Code Capability Field

SMPTE_time_code_capability_field_length: The *SMPTE_time_code_capability_field_length* field specifies the number of bytes for the remainder of this field.

The *SMPTE_time_code_capability_information* field contains the following information.

Address	msb							lsb
00 00_{16}	reserved						Tx	Rx

Figure 5.13 – SMPTE Time Code Capability Information

The following table illustrates the *SMPTE_time_code_capability_information.*

Table 5.10 – SMPTE Time Code Capability Information

SMPTE Time Code Capability Information	Meaning
xxxx xxx1	**Rx:** If this bit is set to 1, then this subunit can receive the SMPTE time code data.
xxxx xx1x	**Tx:** If this bit is set to 1, then this subunit can transmit the SMPTE time code data.
all others	reserved for future specification

5.2.5 Sample Count Capability

The *sample_count_capability* field of *music_subunit_specific_information* indicates the information about Sample Count capability of the subunit and it contains the following information:

Address Offset	Length, Byte	External Read/Write	Contents
00 00$_{16}$	1	R	sample_count_capability_field_length = m - 1
00 01$_{16}$	m - 1	R	sample_count_capability_information
:			

Figure 5.14 – Sample Count Capability Field

Sample_count_capability_field_length: The *sample_count_capability_field_length* field specifies the number of bytes for the remainder of this field.

The *sample_count_capability_information* field contains the following information.

Address	msb							lsb
00 00$_{16}$	reserved						Tx	Rx

Figure 5.15 – Sample Count Capability Information

The following table illustrates the *sample_count_capability_information.*

Table 5.11 – Sample Count Capability Information

Sample Count Capability Information	Meaning
xxxx xxx1	**Rx:** If this bit is set to 1, then this subunit can receive the Sample Count data.
xxxx xx1x	**Tx:** If this bit is set to 1, then this subunit can transmit the Sample Count data.
all others	reserved for future specification

5.2.6 Audio SYNC Capability

The *audio_SYNC_capability* field of *music_subunit_specific_information* indicates the information about audio SYNC capability of the subunit and it contains the following information:

Address Offset	Length, Byte	External Read/Write	Contents
00 00$_{16}$	1	R	audio_SYNC_capability_field_length = n - 1
00 01$_{16}$	n - 1	R	audio_SYNC_capability_information
:			

Figure 5.16 – Audio SYNC Capability Field

audio_SYNC_capability_field_length: The *audio_SYNC_capability_field_length* field specifies the number of bytes for the remainder of this field.

The *audio_SYNC_capability_information* field contains the following information.

Address	msb							lsb
00 00$_{16}$	reserved						Ex	Bus

Figure 5.17 – Audio SYNC Capability Information

The following table illustrates the *audio_SYNC_capability_information.*

Table 5.12 – Audio SYNC Capability Information

Audio SYNC Capability Information	Meaning
xxxx xxx1	**Bus:** If this bit is set to 1, then this subunit can receive the audioSYNC from 1394 bus.
xxxx xx1x	**Ex:** If this bit is set to 1, then this subunit can receive the audio SYNC from sync. source.
all others	reserved for future specification

6. Music Subunit Status Descriptor

The Music subunit status descriptor shows internal status of the component and is Read Only information, which can be revised only by the subunit itself.

This descriptor shows dynamically changed information as compared with the Music subunit identifier descriptor.

6.1 Music Subunit-Specific Descriptor Identifiers

The Music subunit defines those types in addition to a general descriptor identifier type.

Table 6.1 – AV/C Music Subunit-Specific Identifier Types

AV/C Music Subunit-Specific Descriptor Identifier Types	
descriptor_type	**Meaning**
80_{16}	Music subunit Status Descriptor-defined in section 6.2
all others in the subunit-specific range (81_{16} – bf_{16})	reserved for future specification

6.2 Music Subunit Status Descriptor

The Music subunit status descriptor consists of multiple variations and multiple information blocks.

The format of Music subunit status descriptor is as follows:

Address	Length, Byte	External Read/Writ	Contents
00 00_{16}	2	R	descriptor_length
00 01_{16}			
00 02_{16}	i	R	general_music_subunit_status_area_info_block
00 03_{16}			
:			
:	j	R	music_output_plug_status_area_info_block
:			
:			
:	k	R	optional_info_block
:			
:			

Figure 6.1 – Music Subunit Status Descriptor

6.2.1 General Music Subunit Status Area Info Block (81 00_{16})

The *general_music_subunit_status_area_info_block* field shows the information that belongs to each subunit and the contents of that field will be changed by the configuration. For example, if the sampling

frequency for a device that belongs to the Music subunit changes from 48kHz to 96kHz, then the *general_music_subunit_status_area_info_block* field shall be modified by cutting the number of Music input plugs in half.

The format of *general_music_subunit_status_area_info_block* is as follows:

Address	Length, Byte	External Read/Writ	Contents
00 00$_{16}$	2	R	compound_length
00 01$_{16}$			
00 02$_{16}$	2	R	info_block_type = 81 00$_{16}$
00 03$_{16}$			
00 04$_{16}$	2	R	primary_fields_length
00 05$_{16}$			
00 06$_{16}$	i	R	current_general_capability_information
:			
:			

Figure 6.2 – General Music Subunit Status Area Info Block

current_general_capability_information: The *current_general_capability_information* field contains the following information.

Address	Length, bytes	External Read/Write	Contents
00 00$_{16}$	1	R	current_transmit_capability
00 01$_{16}$	1	R	current_receive_capability
00 02$_{16}$	4	R	current_latency_capability = FFFF FFFF$_{16}$
:			
:			

Figure 6.3 – Current General Capability Information

current_transmit_capability: The *current_transmit_capability* field specifies the current capability of transmission and the format is described in Table 5.5.

current_receive_capability: The *current_receive_capability* field specifies the current capability of transmission and the format is described in Table 5.6.

current_latency_capability: The *current_latency_capability* field specifies the current latency of transmission. This field is reserved for future specification. The size of this field is 4 Bytes and should be set to FFFF FFFF$_{16}$.

6.2.2 Music Output Plug Status Area Info Block (81 01$_{16}$)

The *music_output_plug_status_area_info_block* field is described below.

Address	Length, Byte	External Read/Writ	Contents
00 00$_{16}$	2	R	compound_length
00 01$_{16}$			
00 02$_{16}$	2	R	info_block_type = 81 01$_{16}$
00 03$_{16}$			
00 04$_{16}$	2	R	primary_fields_length
00 05$_{16}$			
00 06$_{16}$	1	R	number_of_source_plugs
00 07$_{16}$	i	R	nested plug_status_info_block structure
:			
:			

Figure 6.4 – Music Output Plug Status Area Info Block

number_of_source_plugs: *number_of_source_plugs* field indicates the number of source plugs, which is currently configured by subunit.

plug_status_info_block: The nested *plug_status_info_block* structure field can be described multiple *plug_status_info_block*.

6.2.3 Source Plug Status Info Block (81 02$_{16}$)

The *source_plug_status_info_block* format is as follows:

Address	Length, Byte	External Read/Writ	Contents
00 00$_{16}$	2	R	compound_length
00 01$_{16}$			
00 02$_{16}$	2	R	info_block_type = 81 02$_{16}$
00 03$_{16}$			
00 04$_{16}$	2	R	primary_fields_length
00 05$_{16}$			
00 06$_{16}$	1	R	source_plug_number
00 07$_{16}$	i	R	audio_info_block
:			
:	j	R	MIDI_info_block
:			
:	k	R	SMPTE_time_code_info_block
:			
:	m	R	sample_count_info_block
:			
:	n	R	audio_SYNC_info_block
:			

Figure 6.5 – Source Plug Status Info Block

source_plug_status_info_block: The *source_plug_status_info_block* field is described streams of audio, MIDI, SMPTE time code, Sample Count, and audio SYNC that connected each plug.

source_plug_number: The *source_plug_number* field indicates source plug number.

6.2.3.1 Audio Info Block (81 03_{16})

The *audio_info_block* indicates audio stream status connected to the source plug.

The format of *audio_info_block* is as follows:

Address	Length, Byte	External Read/Writ	Contents
00 00_{16}	2	R	compound_length
00 01_{16}			
00 02_{16}	2	R	info_block_type = 81 03_{16}
00 03_{16}			
00 04_{16}	2	R	primary_fields_length
00 05_{16}			
00 06_{16}	1	R	number_of_audio_streams
00 07_{16}	p	R	name_info_block
:			
:	q	R	optional info_block
:			

Figure 6.6 – Audio Info Block

number_of_audio_streams: The *number_of_audio_streams* field indicates the number of audio streams included in the source plug.

name_info_block: The *name_info_block* is one of general AV/C Information Block types, and its format is defined in "AV/C Information Block Types Specification Version 1.0"[R8]. In the Music subunit, the label of audio stream shall be described in the *name_info_block* of the *audio_info_block*. The format of *name_info_block* is as follows:

Address	Length, Byte	External Read/Writ	Contents
00 00_{16}	2	R	compound_length
00 01_{16}			
00 02_{16}	2	R	info_block_type = $000B_{16}$(name_info_block)
00 03_{16}			
00 04_{16}	2	R	primary_fields_length
00 05_{16}			
00 06_{16}	1	R	name_data_reference_type = 00_{16}
00 07_{16}	r	R	name_data
:			
:			

Figure 6.7 – Name Info Block

name_data_reference_type: In the Music subunit, the *name_data_reference_type* field shall be set to 00_{16}, and this means that the name data is encoded directly in this structure.

name_data: The *name_data* field consists of from 1 to 3 info blocks. The label of audio stream shall be described in *raw_text_data* in the *raw_text_info_block*. The *raw_text_info_block* is also one of general AV/C Information Block types, and its format is defined in "AV/C Information Block Types Specification Version 1.0"[R8].

raw_text_data: The *raw_text_data* field contains the text bytes. The label of audio stream shall be described in this field separated by a CR (Carriage Return) and LF/NL(Line Feed/New Line) of ASCII code. The example is illustrated below.

Example:

Here is an example of the configuration of a talker's Music subunit. Suppose the talker handles the following stream as one outbound isochronous stream:

- 8 audio streams (sequence [0] to [7] in the isochronous stream)
- 8 multiplexed MIDI streams in 1 sequence (sequence [8] in the isochronous stream)
- using the isochronous stream for audio SYNC (no sequence specified)

Therefore, the talker has 17 music plugs for the isochronous stream and each audio stream and multiplexed MIDI stream have their own label.

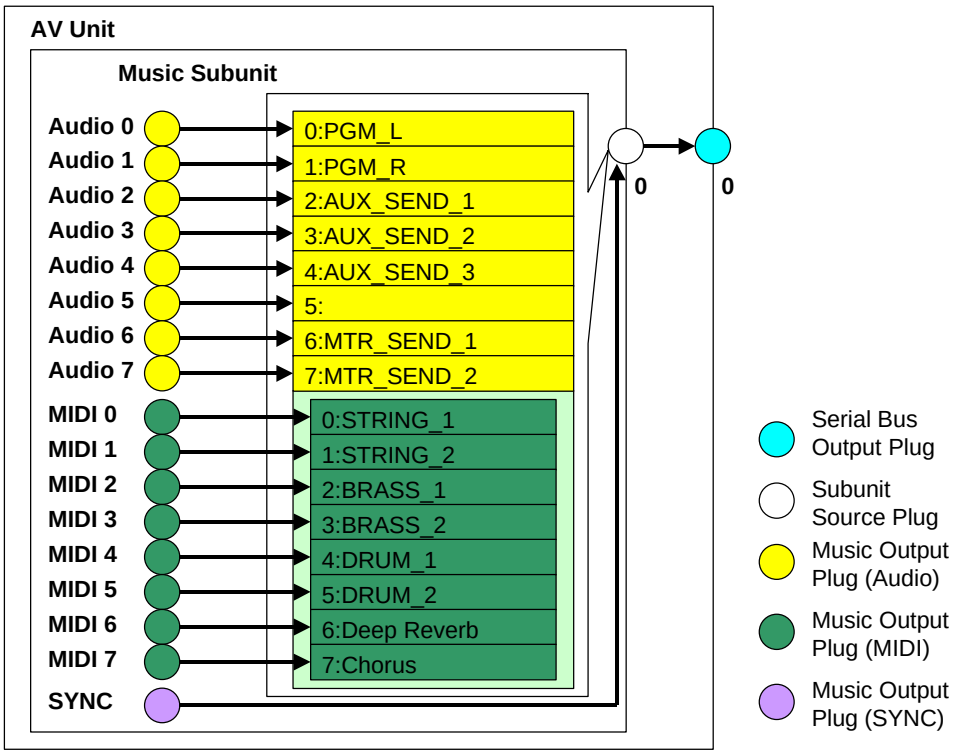


Figure 6.8 – Music Plug Label Example

The relationship between the audio music plug and its label is illustrated as follows:

Table 6.2 – Relationship between audio Music plug and label

Name	Music Output Plug ID	Label
Audio 0	00 00$_{16}$	PGM_L
Audio 1	00 01$_{16}$	PGM_R
Audio 2	00 02$_{16}$	AUX_SEND_1
Audio 3	00 03$_{16}$	AUX_SEND_2
Audio 4	00 04$_{16}$	AUX_SEND_3
Audio 5	00 05$_{16}$	
Audio 6	00 06$_{16}$	MTR_SEND_1
Audio 7	00 07$_{16}$	MTR_SEND_2

In this case, if the *name_info_block* consists of only one *raw_text_info_block*, the *raw_text_data* field shall be as follows:

Table 6.3 – Raw Text Data Field

Raw Text Data
PGM_L(CR)(LF/NL)PGM_R(CR)(LF/NL)AUX_SEND_1(CR)(LF/NL)AUX_SEND_2(CR)(LF/NL)AUX_SEND_3(CR)(LF/NL)(CR)(LF/NL)MTR_SEND_1(CR)(LF)MTR_SEND_2(CR)(LF/NL)

(CR),(LF/NL): The *(CR)* in the above table means Carriage Return. The *(LF/NL)* means Line Feed/New Line. In the *raw_text_data* field, each label is separated by CR and LF/NL. If a music plug does not have a label, then (CR)(LF/NL) follow consecutively.

6.2.3.2 MIDI Info Block (81 04$_{16}$)

The *MIDI_info_block* indicates MIDI stream status connected to the source plug.

The format of *MIDI_info_block* is as follows:

Address	Length, Byte	External Read/Write	Contents
00 00$_{16}$	2	R	compound_length
00 01$_{16}$			
00 02$_{16}$	2	R	info_block_type = 81 04$_{16}$
00 03$_{16}$			
00 04$_{16}$	2	R	primary_fields_length
00 05$_{16}$			
00 06$_{16}$	1	R	number_of_MIDI_streams = n
00 07$_{16}$	p	R	name_info_block[0]
:			
:			:
:	q	R	name_info_block[n-1]
:			
:	r	R	optional info_block
:			

Figure 6.9 – MIDI Info Block

number_of_MIDI_streams: The *number_of_MIDI_streams* field indicates the number of MIDI data streams included in the subunit source plug.

name_info_block[]: The *name_info_block* is one of general AV/C Information Block types, and its format is defined in "AV/C Information Block Types Specification Version 1.0"[R8]. In the Music subunit, the label of each MIDI stream shall be described in the *name_info_block* of the *MIDI info block*. The number in the *name_info_block* is the same as the *number_of_MIDI_streams*. The format of *name_info_block* is illustrated in Figure 6.7.

As with the audio music plug, the label data of the MIDI music plug shall be described in the *raw_text_data* field in the *raw_text_info_block*. As in the music plug label example in Figure 6.8, each MIDI music plug has its own label.

The relationship between a MIDI music plug and its label is illustrated as follows:

Table 6.4 – Relation between MIDI Music plug and label

Name	Music Output Plug ID	Label
MIDI 0	00 00$_{16}$	STRING_1
MIDI 1	00 01$_{16}$	STRING_2
MIDI 2	00 02$_{16}$	BRASS_1
MIDI 3	00 03$_{16}$	BRASS_2
MIDI 4	00 04$_{16}$	DRUM_1
MIDI 5	00 05$_{16}$	DRUM_2
MIDI 6	00 06$_{16}$	Deep_Reverb
MIDI 7	00 07$_{16}$	Chorus

In this case, if the *name_info_block* consists of only one *raw_text_info_block*, the *raw_text_data* field shall be as follows:

Table 6.5 – Raw Text Data Field

Raw Text Data
STRING_1(CR)(LF/NL)STRING_2(CR)(LF/NL)BRASS_1(CR)(LF/NL)BRASS_2(CR)(LF/NL)DRUM_1(CR)(LF/NL)DRUM_2(CR)(LF/NL)Deep_Reverb(CR)(LF/NL)Chorus(CR)(LF/NL)

(CR),(LF/NL): The *(CR)* in the above table means Carriage Return. The *(LF/NL)* means Line Feed/New Line. In the *raw_text_data* field, each label is separated by CR and LF/NL. If a music plug does not have a label, then (CR)(LF/NL) follow consecutively.

6.2.3.3 SMPTE Time Code Info Block (81 05_{16})

Address	Length, Byte	External Read/Writ	Contents
00 00_{16}	2	R	compound_length
00 01_{16}			
00 02_{16}	2	R	info_block_type = 81 05_{16}
00 03_{16}			
00 04_{16}	2	R	primary_fields_length
00 05_{16}			
00 06_{16}	1	R	SMPTE_time_code_activity
00 07_{16}	:	:	reserved
:			

Figure 6.10 – SMPTE Info Block

The *SMPTE_time_code_activity* field contains the following information.

Address	msb							lsb
00 00_{16}	reserved						Tx	Rx

Figure 6.11 – SMPTE Time Code Activity

The following table illustrates the *SMPTE_time_code_activity*.

Table 6.6 – SMPTE Time Code Activity

SMPTE Time Code Activity	Meaning
xxxx xxx1	**Rx:** If this bit is set to 1, then this subunit receives the SMPTE time code data.
xxxx xx1x	**Tx:** If this bit is set to 1, then this subunit transmits the SMPTE time code data.
all others	reserved for future specification

6.2.3.4 Sample Count Info Block(81 06_{16})

Address	Length, Byte	External Read/Writ	Contents
00 00_{16}	2	R	compound_length
00 01_{16}			
00 02_{16}	2	R	info_block_type = 81 06_{16}
00 03_{16}			
00 04_{16}	2	R	primary_fields_length
00 05_{16}			
00 06_{16}	1	R	sample_count_activity
00 07_{16}	:	:	reserved
:			

Figure 6.12 – Sample Count Info Block

The *sample_count_activity* field contains the following information.

Address	msb							lsb
00 00_{16}	reserved						Tx	Rx

Figure 6.13 – Sample Count Activity

The following table illustrates the *sample_count_activity*.

Table 6.7 – Sample Count Activity

Sample Count Activity	Meaning
xxxx xxx1	**Rx:** If this bit is set to 1, then this subunit receives the Sample Count data.
xxxx xx1x	**Tx:** If this bit is set to 1, then this subunit transmits the Sample Count data.
all others	reserved for future specification

6.2.3.5 Audio SYNC Info Block (81 07$_{16}$)

Address	Length, Byte	External Read/Writ	Contents
00 00$_{16}$	2	R	compound_length
00 01$_{16}$			
00 02$_{16}$	2	R	info_block_type = 81 07$_{16}$
00 03$_{16}$			
00 04$_{16}$	2	R	primary_fields_length
00 05$_{16}$			
00 06$_{16}$	1	R	audio_SYNC_activity
00 07$_{16}$	:	:	reserved
:			

Figure 6.14 – Audio SYNC Count Info Block

The *audio_SYNC_activity* field contains the following information.

Address	msb							lsb
00 00$_{16}$	reserved						Ex	Bus

Figure 6.15 – Audio SYNC Activity

The following table illustrates the *audio_SYNC_activity.*

Table 6.8 – Audio SYNC Activity

Audio SYNC Activity	Meaning
xxxx xxx1	**Bus:** If this bit is set to 1, then this subunit receives the audio SYNC from 1394 bus.
xxxx xx1x	**Ex:** If this bit is set to 1, then this subunit receives the audio SYNC from sync. source.
all others	reserved for future specification

7. Music Subunit Commands

This chapter describes subunit commands developed for Music subunit.

The table below summarizes the subunit commands for Music subunit.

Table 7.1– Music Subunit Commands

Opcode	Value	Support Level (by ctype)			Comments
		C	S	N	
DESTINATION PLUG CONFIGURE	40_{16}	M	M	-	Make a destination Music subunit set up a connection between specified Music input plug and a destination subunit plug, and inquire about the current status of the connection.
SOURCE PLUG CONFIGURE	41_{16}	-	O	-	Inquire about the current status of the connection between specified Music output plug and a source subunit plug.
DESTINATION CONFIGURATIONS	42_{16}	-	O	-	Request a sequence location of an isochronous stream handled by the specified destination subunit plug.
SOURCE CONFIGURATIONS	43_{16}	-	O	-	Request a sequence location of an isochronous stream handled by the specified source subunit plug.
MUSIC PLUG INFO	$C0_{16}$	-	M	-	Information about the music input and output plug of each music plug type in the subunit.
CURRENT CAPABILITY	$C1_{16}$	-	O	-	Information about the currently available music input or output plug format.

C: Control, S: Status, N: Notify

M = Mandatory, R = Recommended, O = Optional

It is recommended to implement either the SOURCE PLUG CONFIGURE status command or the SOURCE CONFIGURATIONS status command.

7.1 DESTINATION PLUG CONFIGURE Command

The purpose of the DESTINATION PLUG CONFIGURE control command is to connect or disconnect multiple destination subunit plugs and Music input plugs on a target. One operation to connect or disconnect is specified as one subcommand. One DESTINATION PLUG CONFIGURE control command can contain up to 72 subcommands. Subcommands shall be executed sequentially in order.

The DESTINATION PLUG CONFIGURE command also has status command type. The DESTINATION PLUG CONFIGURE status command is used to inquire about the connection of Music input plug. This command should be issued to a listener node.

7.1.1 DESTINATION PLUG CONFIGURE control command

The format of the DESTINATION PLUG CONFIGURE control command frame is shown in the figure below.

	length	msb							lsb
opcode	1	DESTINATION PLUG CONFIGURE(40_{16})							
operand[0]	1	number_of_subcommands = n							
operand[1]	1	result_status = FF_{16}							
operand[2]	1	number_of_completed_subcommands = FF_{16}							
operand[3]	7	subcommand[0]							
:									
:		:							
:	7	subcommand [n-1]							
:									

Figure 7.1 – DESTINATION PLUG CONFIGURE control command frame

This command is to connect or disconnect multiple destination subunit plugs and Music input plugs and is issued to a listener node.

7.1.1.1 Field Definitions

number_of_subcommands: The *number_of_subcommands* field indicates the number of subcommands to be executed on the target. Due to the limitation of the AV/C command length, the maximum value of the *number_of_subcommands* field shall be 72 (72 subcommands: 510 bytes).

result_status: In the case of a command frame, the *result_status* field shall be set to FF_{16}.

number_of_completed_subcommands: In the case of a command frame, the *number_of_completed_subcommands* field shall be set to FF_{16}.

subcommands: The format of the *subcommand* field is described below.

address offset	length	msb							lsb
00_{16}	1	subfunction							
01_{16}	1	music_plug_type							
02_{16}	1	music_plug_ID(MSB)							
03_{16}	1	music_plug_ID(LSB)							
04_{16}	1	subunit_plug_ID							
05_{16}	1	stream_position[0]							
06_{16}	1	stream_position[1]							

Figure 7.2 – Subcommand

subfunction: The *subfunction* field determines the operation to be performed by the target, as defined by the table below.

Table 7.2 – Subfunction

Value	Meaning
00_{16}	CONNECT
01_{16}	CHANGE_CONNECTION
02_{16}	DISCONNECT
03_{16}	DISCONNECT_ALL
04_{16}	DEFAULT_CONFIGURE
all others	reserved for future specification

CONNECT (00_{16}): The CONNECT subfunction makes the destination subunit connect a specified input music plug to a subunit destination plug and to a corresponding stream(sequence) in the AM824 packet. If the specified Music input plug has already connected to a destination subunit plug, the subcommand shall return "music_plug already connected" in the result_status field.

CHANGE_CONNECTION (01_{16}): The CHANGE_CONNECTION subfunction makes the destination subunit execute the CONNECT subfunction and the DISCONNECT subfunction with one subcommand. If the specified Music input plug has been connected to a destination subunit plug, the target should first disconnect them and establish a new connection between the specified Music input plug and the specified subunit plug.

In contrast to the CONNECT subfunction, when the specified input plug has already connected to a subunit plug, the target should change the connection instead of returning an error.

The controller should issue this command only for a music plug connected by its own CONNECT subcommand. In the case of establishing a new connection, the controller should use the CONNECT subcommand.

DISCONNECT (02_{16}): The DISCONNECT subfunction makes the destination subunit disconnect the input music plug from the subunit plug. In this case, the *stream_position* field shall be set to FF_{16}. When the specified music plug is not connected to the specified subunit plug, the target shall return an OK *result_status* (this is not an error).

DISCONNECT_ALL (03_{16}): The DISCONNECT_ALL subfunction makes the destination subunit disconnect all of the input music plugs from the connected subunit plug. The *music_plug_type*, *music_plug_ID*, *subunit_plug_ID* and *stream_position* field shall be FF_{16} for the DISCONNECT_ALL subfunction.

DEFAULT_CONFIGURE (04_{16}): The DEFAULT_CONFIGURE subfunction makes the target subunit reset all music plugs to the default configuration. All fields that indicate parameters shall be set to FF_{16}.

When the subfunction is set to values other than the values above, the target shall stop execution and return the "undefined subfunction" *result_status* in the response frame.

music_plug_type, music_plug_ID: The *music_plug_type* and *music_plug_ID* fields indicate the Music input plug address together. When DISCONNECT_ALL or DEFAULT_CONFIGURE subfunction is specified, those fields shall be set to FF_{16}.

music_plug_type: The *music_plug_type* field indicates a kind of data type managed by the music plug. The value for the *music_plug_type* field is defined as the table below.

Table 7.3 – Music Plug Type

Value	Meaning
00_{16}	Audio
01_{16}	MIDI
02_{16}	SMPTE time code
03_{16}	Sample Count
80_{16}	Audio SYNC
FF_{16}	reserved
all others	reserved for future specification

music_plug_ID: The *music_plug_ID* field indicates the music plug identification number. The specified music plug shall be an input music plug. For all values of the *music_plug_type*, from 0 to 65534, a *music_plug_ID* number is allocated.

Table 7.4 – Music Input Plug

Value	Meaning
0000_{16} – $FFFE_{16}$	Music input plug0 – 65534
$FFFF_{16}$	reserved for each field.

When the specified Music input plug does not exist on the target, the target shall return "*music_plug* does not exist" in the *result_status* field as an error.

subunit_plug_ID: The *subunit_plug_ID* field indicates a subunit plug number to be connected or disconnected.

Table 7.5 – Subunit Plug ID

Value	Meaning
00_{16} – $1E_{16}$	destination plug[0] – [30]
$1F_{16}$ – FE_{16}	reserved
FF_{16}	no plug

When the subfunction is DISCONNECT or DISCONNECT_ALL or DEFAULT_CONFIGURE, the *subunit_plug_ID* field shall be set to FF_{16}.

When the specified *subunit_plug* does not exist, the target shall return "*subunit_plug* does not exist" in the *result_status* field as an error.

stream_position: The *stream_position* field indicates the stream position (the AM824 packet sequence number) corresponding to the music plug. There are three formats defined according to the plug type.

address offset	length	msb							lsb
00_{16}	1	stream_number(0 – 255)							
01_{16}	1	FF_{16}							

Figure 7.3 – Format 0 (e.g. audio/SMPTE time code/Sample Count)

address offset	length	msb							lsb
00_{16}	1	stream_number(0 – 255)							
01_{16}	1	multiplex_index(0 – 7)							

Figure 7.4 – Format 1 (e.g. MIDI)

address offset	length	msb							lsb
00_{16}	1	FF_{16}							
01_{16}	1	FF_{16}							

Figure 7.5 – Format 2 (e.g. audio SYNC)

When the subfunction field is DISCONNECT or DISCONNECT_ALL or DEFAULT_CONFIGURE, both *stream_number* and *mutilpex_index* shall be set to FF_{16}.

7.1.1.2 DESTINATION PLUG CONFIGURE control command response

The format of the DESTINATION PLUG CONFIGURE control response frame is shown in the figure below.

	length	msb							lsb
opcode	1	DESTINATION PLUG CONFIGURE(40_{16})							
operand[0]	1	number_of_subcommands = n							
operand[1]	1	result_status							
operand[2]	1	number_of_completed_subcommands							
operand[3]	7	subcommand[0]							
*									
:		:							
:	7	subcommand [n-1]							
:									

Figure 7.6 – DESTINATION PLUG CONFIGURE control command response frame

If the target fails to execute the subcommand, it shall stop executing the subcommand and return the ACCEPTED response. In this case, the target shall report the cause of the failure in the *result_status* field and the location of the failed subcommand in the *number_of_completed_subcommands* respectively.

If the format of the command frame has an error (such as a different number of subcommand than reported in *number_of_subcommands*), no subcommand is executed and the REJECTED response shall be returned.

number_of_subcommands: The *number_of_subcommands* field has the same value as the DESTINATION PLUG CONFIGURE control command.

result_status: In an ACCEPTED response, the *result_status* field shall be set the value as a result of command described below. If the listener fails to configure the destination plug, the *number_of_completed_subcommands* field in the response frame shall be set to the number of processes that succeeded.

Definition of value is described below.

Table 7.6 – Result Status Value

Value	Meaning
00_{16}	OK
01_{16}	unknown subfunction
02_{16}	unknown music_plug_type
03_{16}	music_plug does not exist
04_{16}	subunit_plug does not exist
05_{16}	music_plug already connected
all others	reserved for future specification

The "OK" *result_status* means that all the subcommands have been executed successfully.

The "unknown subfunction" *result_status* means that the subfunction specified in the failed subcommand is not defined.

The "unknown *music_plug_type*" *result_status* means that the music plug type specified in the failed subcommand is illegal.

The "*music_plug* does not exist" *result_status* means that the Music input plug ID specified in the failed subcommand does not exist on the target.

The "*subunit_plug* does not exist" *result_status* means that the subunit plug specified in the failed subcommand does not exist on the target.

The "*music_plug* already connected" *result_status* means that the specified Music input plug and the specified subunit plug have been already connected.

In a REJECTED response, the *result_status* field shall be set to FF_{16}.

number_of_completed_subcommands: In an ACCEPTED response, the *number_of_completed_subcommands* field indicates the number of successful subcommands. When the *result_status* field is "OK", the *number_of_completed_subcommands* field shall be equal to the *number_of_subcommands* field. When the *result_status* field is not "OK", the command has failed at the [*number_of_completed_subcommands*] procedure. The *result_status* field explains the cause of the failure.

In a REJECTED response, the *number_of_completed_subcommands* field shall be set to 00_{16}.

subcommand: The format of the *subcommand* field is shown in Figure 7.2.

7.1.1.3 DESTINATION PLUG CONFIGURE control command and response field values

The following table shows the field values in the DESTINATION PLUG CONFIGURE control command and response frames.

Table 7.7 – Result status, number of completed subfunctions

Field	Command	Response		
		ACCEPTED	REJECTED	INTERIM
result_status	FF16	result_status_value	FF16	---
number_of_completed_subcommands	FF16	XX16	0016	---

7.1.2 DESTINATION PLUG CONFIGURE status command

The DESTINATION PLUG CONFIGURE status command is used to inquire about the destination subunit plug connected to the specified Music input plug and the stream index of the sequence managed by the specified Music input plug. The format of the DESTINATION PLUG CONFIGURE status command frame is shown in the figure below.

	length	msb							lsb
opcode	1	DESTINATION PLUG CONFIGURE(4016)							
operand[0]	1	number_of_subcommands = n							
operand[1]	1	FF16							
operand[2]	1	number_of_completed_subcommands = FF16							
operand[3]	7	subcommand[0]							
:									
:		:							
:	7	subcommand [n-1]							
:									

Figure 7.7 – DESTINATION PLUG CONFIGURE status command frame

7.1.2.1 Field Definitions

number_of_subcommands: The *number_of_subcommands* field specifies the number of subcommands contained in the command frame. As the length of one AV/C command frame is limited to 512 bytes and the length of one subcommand is 7 bytes, the maximum number of this field shall be 72 (4816).

number_of_completed_subcommands: In the case of a command frame, *number_of_completed_subcommands* field shall be set to FF16.

subcommand: The format of the *subcommand* field is shown in the figure below.

address offset	length	msb							lsb
00_{16}	1	result_status							
01_{16}	1	music_plug_type							
02_{16}	1	music_plug_ID(MSB)							
03_{16}	1	music_plug_ID(LSB)							
04_{16}	1	subunit_plug_ID							
05_{16}	1	stream_position[0]							
06_{16}	1	stream_position[1]							

Figure 7.8 – Subcommand

This command is used to inquire about one music plug status in conjunction with *result_status*, *music_plug_type*, *music_plug_ID*, *subunit_plug_ID* and *stream_position.* Unlike control command/response, this command will be used to query all of the designated music plugs and return a *result_status* to each reference.

result_status: The *result_status* field shall be FF_{16} in status command frame.

music_plug_type, music_plug_ID: The *music_plug_type* and *music_plug_ID* field indicates the address of Music input plug to be queried. The definition of those fields is the same as the DESTINATION PLUG CONFIGURE control command.

subunit_plug_ID, stream_position: The *subunit_plug_ID* and the *stream_position* field shall be set to FF_{16} in the DESTINATION PLUG CONFIGURE status command.

7.1.2.2 DESTINATION PLUG CONFIGURE status command response

The format of the DESTINATION PLUG CONFIGURE status response frame is shown in the figure below.

	length	msb							lsb
opcode	1	DESTINATION PLUG CONFIGURE(40_{16})							
operand[0]	1	number_of_subcommands = n							
operand[1]	1	FF_{16}							
operand[2]	1	number_of_completed_subcommands							
operand[3]	7	subcommand[0]							
:									
:		:							
:	7	subcommand [n-1]							
:									

Figure 7.9 – DESTINATION PLUG CONFIGURE status command response frame

number_of_subcommands: The *number_of_subcommands* field shall be the same as the DESTINATION PLUG CONFIGURE status command.

number_of_completed_subcommands: This field indicates the number of acquired music plug. This field shall be updated in the response frame according to each query. (*result_status* is "OK" or no connection)

In REJECTED response, the *number_of_completed_subcommands* field shall be set to FF_{16}.

subcommand: The format of the subcommand field is the same as the DESTINATION PLUG CONFIGURE status command frame.(Refer to Figure 7.8.)

result_status: The *result_status* field indicates the connection information of each Music input plug.

The encoding of the *result_status* field is shown in the table below.

Table 7.8 – Result Status Value

Value	Meaning
00_{16}	OK
01_{16}	no connection
02_{16}	unknown music_plug_type
03_{16}	music_plug does not exist
all others	reserved for future specification

The "OK" *result_status* means that the subcommand has been executed successfully.

The "no connection" *result_status* means that the specified music plug has no connection.

The "unknown *music_plug_type*" *result_status* means that the specified music plug type is illegal.

The "*music_plug* does not exist" *result_status* means that the Music input plug does not exist on the target.

music_plug_type, music_plug_ID: These fields indicate the address of Music input plug to be queried. Detail is the same as the control command.

subunit_plug_ID, stream_position: The *subunit_plug_ID* and the *stream_position* indicate the destination subunit plug and stream position that is connected to the specified music plug. The format of the *stream_position* field is the same as the DESTINATION PLUG CONFIGURE control command. The *stream_position* field should be set FF_{16} except when the *result_status* field is "OK".

7.1.2.3 DESTINATION PLUG CONFIGURE status command and response field values

The following table shows the field values in the DESTINATION PLUG CONFIGURE status command and response frames.

Table 7.9 – Field values in the DESTINATION PLUG CONFIGURE status command

Fields	Command	Response		
		INTERIM	REJECTED	STABLE
number_of_subcommands	number of subcommands	←	←	←
result_status	FF16	←	←	Result Status
number_of_completed_subcommands	FF16	←	←	number of completed subcommands

To query a Music input plug connected to the specified destination subunit plug with the specified stream position, the fields in the subcommand should be set as follows:

Table 7.10 – Field values in the DESTINATION PLUG CONFIGURE status command (to query Music input plug)

Fields	Command	Response		
		INTERIM	REJECTED	STABLE
result_status	FF16	←	←	Result Status
music_plug_type	Plug Type	←	←	←
music_plug_ID	FFFF16	←	←	Music Plug ID
subunit_plug_ID	Subunit Plug ID	←	←	←
stream_position[0]	Stream Position	←	←	←
stream_position[1]	Stream Position	←	←	←

To query a destination subunit plug connected to the specified Music input plug, the fields in the subcommand should be set as follows:

Table 7.11 – Field values in the DESTINATION PLUG CONFIGURE status command (to query subunit plug and stream position)

Fields	Command	Response		
		INTERIM	REJECTED	STABLE
result_status	FF16	←	←	Result Status
music_plug_type	Plug Type	←	←	←
music_plug_ID	Music Plug ID	←	←	←
subunit_plug_ID	FF16	←	←	Subunit Plug ID
stream_position[0]	FF16	←	←	Stream Position
stream_position[1]	FF16	←	←	Stream Position[1]

[1] The meaning of the *stream_position* field varies according to the *music_plug_type* field.

7.2 SOURCE PLUG CONFIGURE Command

The SOURCE PLUG CONFIGURE command has only status command type. The SOURCE PLUG CONFIGURE status command is used to inquire about the connection of Music output plug. This command should be issued to a talker (source node). One SOURCE PLUG CONFIGURE control command can contain up to 72 subcommands. Subcommands shall be executed sequentially in order.

7.2.1 SOURCE PLUG CONFIGURE status command

The SOURCE PLUG CONFIGURE status command is used to inquire about the source subunit plug connected to the specified Music output plug and the stream index of the sequence managed by the specified Music output plug. The format of the SOURCE PLUG CONFIGURE status command frame is shown in the figure below.

	length	msb							lsb
opcode	1	SOURCE PLUG CONFIGURE(41_{16})							
operand[0]	1	number_of_subcommands = n							
operand[1]	1	FF_{16}							
operand[2]	1	number_of_completed_subcommands = FF_{16}							
operand[3]	7	subcommand[0]							
:									
:		:							
:	7	subcommand [n-1]							
:									

Figure 7.10 – SOURCE PLUG CONFIGURE status command frame

7.2.1.1 Field Definitions

number_of_subcommands: The *number_of_subcommands* field specifies the number of subcommands contained in the command frame. As the length of one AV/C command frame is limited to 512 bytes and the length of one subcommand is 7 bytes, the maximum number of this field shall be 72 (48_{16}).

number_of_completed_subcommands: In the case of a command frame, the *number_of_completed_subcommands* field shall be set to FF_{16}.

subcommands: The format of the *subcommand* field is shown in Figure 7.2.

The format of the SOURCE PLUG CONFIGURE status response frame shall be the same as the corresponding status command.

7.2.1.2 SOURCE PLUG CONFIGURE status command and response field values

The following table shows the field values in the SOURCE PLUG CONFIGURE status command and response frames.

Table 7.12 – Field values in the SOURCE PLUG CONFIGURE status command

Fields	Command	Response		
		INTERIM	REJECTED	STABLE
number_of_subcommands	number of subcommands	←	←	←
result_status	FF16	←	←	Result Status
number_of_completed_subcommands	FF16	←	←	number of completed subcommands

To query a Music output plug connected to the specified source subunit plug with the specified stream position, the fields in the subcommand shall be set as follows:

Table 7.13 – Field values in the SOURCE PLUG CONFIGURE status command (To query Music output plug)

Fields	Command	Response		
		INTERIM	REJECTED	STABLE
result_status	FF16	←	←	Result Status
music_plug_type	Plug Type	←	←	←
music_plug_ID	FFFF16	←	←	Music Plug ID
subunit_plug_ID	Subunit Plug ID	←	←	←
stream_position[0]	Stream Position	←	←	←
stream_position[1]	Stream Position	←	←	←

To query a source subunit plug connected to the specified Music output plug, the fields in the subcommand shall be set as follows:

Table 7.14 – Field values in the SOURCE PLUG CONFIGURE status command (To query subunit plug and stream position)

Fields	Command	Response		
		INTERIM	REJECTED	STABLE
result_status	FF16	←	←	Result Status
music_plug_type	Plug Type	←	←	←
music_plug_ID	Music Plug ID	←	←	←
subunit_plug_ID	FF16	←	←	Subunit Plug ID
stream_position[0]	FF16	←	←	Stream Position
stream_position[1]	FF16	←	←	Stream Position[2]

7.3 DESTINATION CONFIGURATIONS Command

7.3.1 DESTINATION CONFIGURATIONS status command

The DESTINATION CONFIGURATIONS status command is used to acquire the configuration of audio and music data streams managed by the subunit destination plug ID. The format of the DESTINATION CONFIGURATIONS status command frame is shown in the figure below.

	length	msb							lsb
opcode	1	DESTINATION CONFIGURATIONS(4216)							
operand[0]	1	subunit_plug_ID							

Figure 7.11 – DESTINATION CONFIGURATIONS status command frame

7.3.1.1 Field Definitions

subunit_plug_ID: The *subunit_plug_ID* field specifies the subunit destination plug in the Music subunit.

7.3.1.2 DESTINATION CONFIGURATIONS status command response

The format of the DESTINATION CONFIGURATIONS status response is defined in the figure below.

[2] The meaning of the *stream_position* field varies according to the *music_plug_type* field.

	length	msb							lsb
opcode	1	DESTINATION CONFIGURATIONS(42_{16})							
operand[0]	1	subunit_plug_ID							
operand[1]	2	start_of_music_plug_ID = n							
operand[2]									
operand[3]	2	end_of_music_plug_ID = m							
operand[4]									
operand[5]	5	music_plug_info[n]							
:									
:		:							
:	5	music_plug_info[m]							
:									

Figure 7.12 – DESTINATION CONFIGURATIONS status command response frame

subunit_plug_ID: The *subunit_plug_ID* field specifies the subunit destination plug in the Music subunit. This field shall be the same as the corresponding DESTINATION CONFIGURATIONS status command frame.

start_of_music_plug_ID: The *start_of_music_plug_ID* field indicates the music plug ID at the head of the *music_plug_info*.

end_of_music_plug_ID: The *end_of_music_plug_ID* field indicates the music plug ID at the end of the *music_plug_info*.

music_plug_info: The format of the *music_plug_info* field is defined as follows:

address offset	length	msb							lsb
00_{16}	1	music_plug_type							
01_{16}	1	music_plug_ID(MSB)							
02_{16}	1	music_plug_ID(LSB)							
03_{16}	1	stream_position[0]							
04_{16}	1	stream_position[1]							

Figure 7.13 – Music Plug Info

music_plug_type: The *music_plug_type* field is defined in Table 7.3.

music_plug_ID: The *music_plug_ID* field is defined in Table 7.4.

stream_position: The *stream_position* field is defined in Figure 7.3, Figure 7.4 and Figure 7.5.

7.3.1.3 Rules

Each *music_plug_info* describes each sequence in the isochronous stream managed by the specified subunit plug. The order of the *music_plug_info* shall comply with the location of the corresponding sequence.

If a stream is not assigned to a certain Music input plug, the *music_plug_info* field corresponding to the sequence can be omitted.

Example:

Here is an example of the configuration of a listener's Music subunit. Suppose the music plugs on the listener are configured as below, The figure shows that:

- The serial bus input plug [0] is connected to the destination subunit plug [0] of the Music subunit.
- The serial bus input plug [1] is connected to the destination subunit plug [1] of the Music subunit.
- Both isochronous streams received by those serial bus input plugs contain 8 audio sequences and 1 MIDI sequence.
- The MIDI sequence contains one stream in index 0.

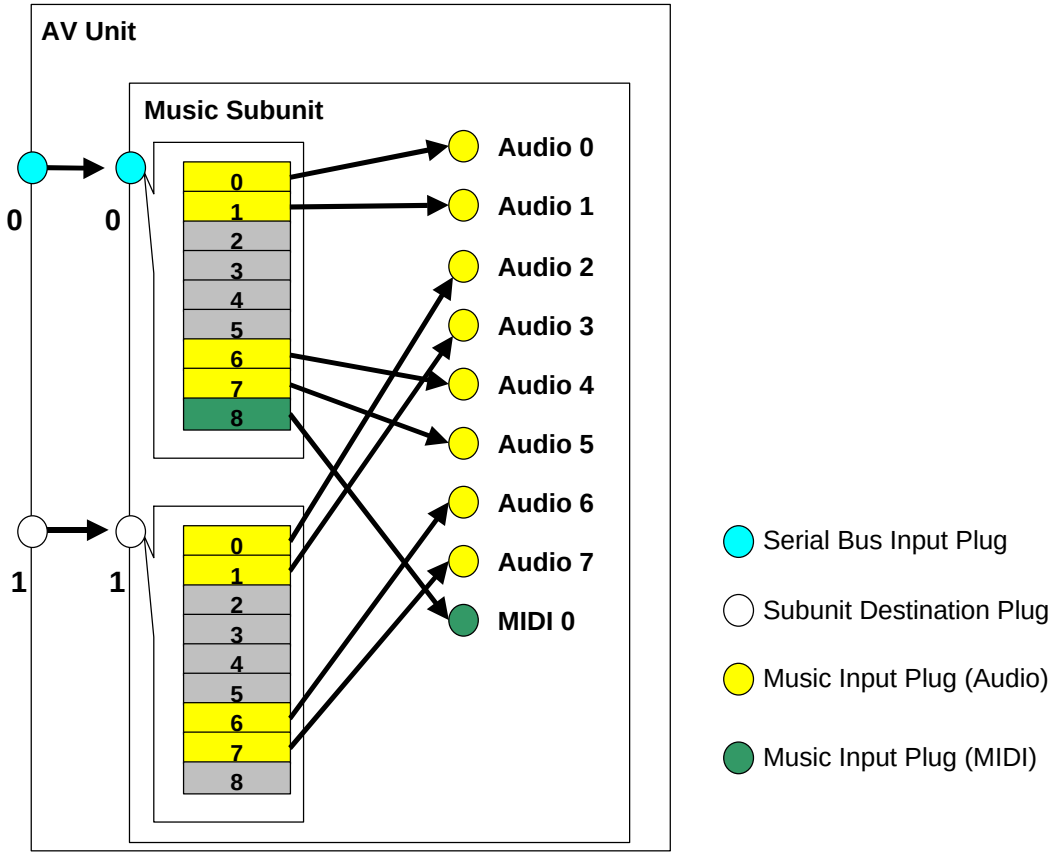


Figure 7.14 – Listener Configuration Example

For subunit plug = 0, the *music_plug_info* fields in the DESTINATION CONFIGURATIONS status response frame should be as follows:

Table 7.15 – Destination Configuration on Subunit Plug 0

	music_plug_type	music_plug_ID	stream_position[0]	stream_position[1]
sequence[0]	Audio (00_{16})	00 00_{16}	00_{16}	FF_{16}
sequence[1]	Audio (00_{16})	00 01_{16}	01_{16}	FF_{16}
sequence[6]	Audio (00_{16})	00 04_{16}	06_{16}	FF_{16}
sequence[7]	Audio (00_{16})	00 05_{16}	07_{16}	FF_{16}
sequence[8][0]	MIDI (01_{16})	00 00_{16}	08_{16}	00_{16}

For subunit plug = 1, the *music_plug_info* fields in the DESTINATION CONFIGURATIONS status response frame should be as follows:

Table 7.16 – Destination Configuration on Subunit Plug 1

	music_plug_type	music_plug_ID	stream_position[0]	stream_position[1]
sequence[0]	Audio (00_{16})	00 02_{16}	00_{16}	FF_{16}
sequence[1]	Audio (00_{16})	00 03_{16}	01_{16}	FF_{16}
sequence[6]	Audio (00_{16})	00 06_{16}	06_{16}	FF_{16}
sequence[7]	Audio (00_{16})	00 07_{16}	07_{16}	FF_{16}

7.4 SOURCE CONFIGURATIONS command

7.4.1 SOURCE CONFIGURATIONS status command

The SOURCE CONFIGURATIONS status command is used to acquire the configuration of audio and music data streams managed by the subunit source plug ID. The format of the SOURCE CONFIGURATIONS status command frame is shown in the figure below.

	length	msb							lsb
opcode	1	SOURCE CONFIGURATIONS(43_{16})							
operand[0]	1	subunit_plug_ID							

Figure 7.15 – SOURCE CONFIGURATIONS status command frame

7.4.1.1 Field Definitions

subunit_plug_ID: The *subunit_plug_ID* field specifies the subunit source plug in the Music subunit.

7.4.1.2 SOURCE CONFIGURATIONS status command response

	length	msb							lsb
opcode	1	SOURCE CONFIGURATIONS(43_{16})							
operand[0]	1	subunit_plug_ID							
operand[1]	2	start_of_music_plug_ID = n							
operand[2]									
operand[3]	2	end_of_music_plug_ID = m							
operand[4]									
operand[5]	5	music_plug_info[n]							
:									
:		:							
:	5	music_plug_info[m]							
:									

Figure 7.16 – SOURCE CONFIGURATIONS status command response frame

subunit_plug_ID: The *subunit_plug_ID* field specifies the subunit source plug in the Music subunit. The value shall be the same as the corresponding SOURCE CONFIGURATIONS status command.

start_of_music_plug_ID: The *start_of_music_plug_ID* field indicates the music plug ID at the head of the *music_plug_info*.

end_of_music_plug_ID: The *end_of_music_plug_ID* field indicates the music plug ID at the end of the *music_plug_info*.

music_plug_info: The *music_plug_info* field describes an audio and music data sequence contained in the isochronous stream managed by the specified subunit plug.

The format of the *music_plug_info* field is defined in Figure 7.13.

Note:

The sequence managing MIDI message may include up to 8 multiplexed MIDI sequences. The *music_plug_info* field shall describe such MIDI sequences in order of the multiplexed index. The multiplexed index is described in the *stream_position [1]* field for the MIDI music_plug_type.

Example:

Here is an example of the configuration of a talker's Music subunit. Suppose the talker handles the following stream as one outbound isochronous stream:

- 8 audio streams (sequence [0] to [7] in the isochronous stream)
- 8 multiplexed MIDI streams in 1 sequence (sequence [8] in the isochronous stream)
- using the isochronous stream for audio SYNC (no sequence specified)

Therefore, the talker has 17 music plugs for the isochronous stream and should return the following parameters for the *music_plug_info* fields in the SOURCE CONFIGURATIONS status response.

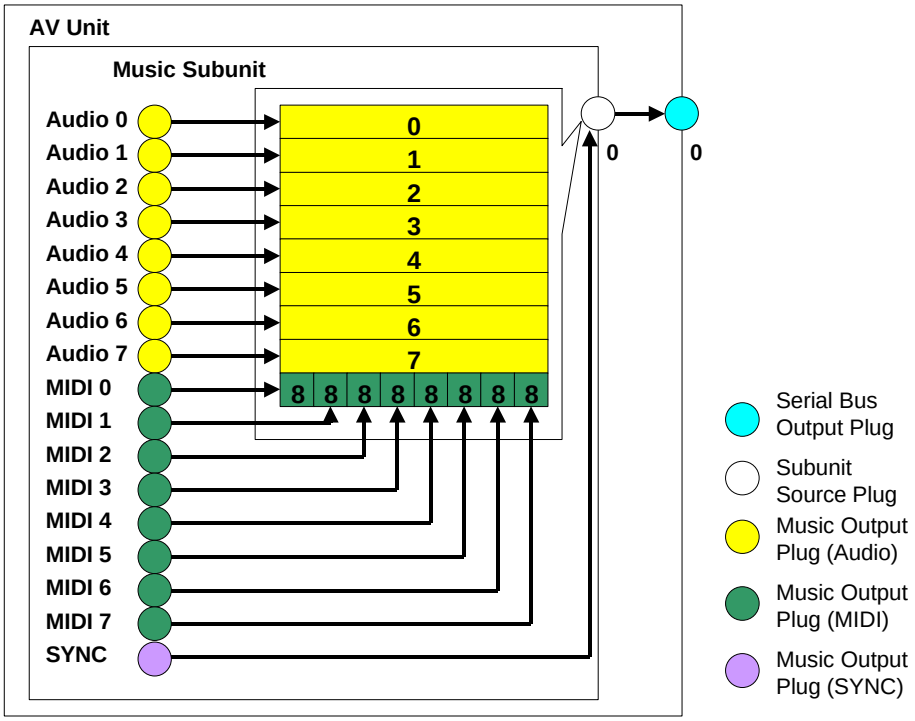


Figure 7.17 – Source Configuration Example

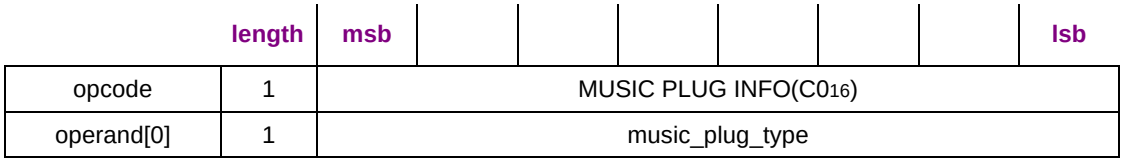
Table 7.17 – Source Configuration of Source Subunit Plug 0

	music_plug_type	music_plug_ID	stream_position[0]	stream_position[1]
music_plug_info[0]	Audio (00_{16})	00 00_{16}	00_{16}	FF_{16}
music_plug_info[1]	Audio (00_{16})	00 01_{16}	01_{16}	FF_{16}
music_plug_info[2]	Audio (00_{16})	00 02_{16}	02_{16}	FF_{16}
music_plug_info[3]	Audio (00_{16})	00 03_{16}	03_{16}	FF_{16}
music_plug_info[4]	Audio (00_{16})	00 04_{16}	04_{16}	FF_{16}
music_plug_info[5]	Audio (00_{16})	00 05_{16}	05_{16}	FF_{16}
music_plug_info[6]	Audio (00_{16})	00 06_{16}	06_{16}	FF_{16}
music_plug_info[7]	Audio (00_{16})	00 07_{16}	07_{16}	FF_{16}
music_plug_info[8]	MIDI (01_{16})	00 00_{16}	08_{16}	00_{16}
music_plug_info[9]	MIDI (01_{16})	00 01_{16}	08_{16}	01_{16}
music_plug_info[10]	MIDI (01_{16})	00 02_{16}	08_{16}	02_{16}
music_plug_info[11]	MIDI (01_{16})	00 03_{16}	08_{16}	03_{16}
music_plug_info[12]	MIDI (01_{16})	00 04_{16}	08_{16}	04_{16}
music_plug_info[13]	MIDI (01_{16})	00 05_{16}	08_{16}	05_{16}
music_plug_info[14]	MIDI (01_{16})	00 06_{16}	08_{16}	06_{16}
music_plug_info[15]	MIDI (01_{16})	00 07_{16}	08_{16}	07_{16}
music_plug_info[16]	SYNC (80_{16})	00 00_{16}	FF_{16}	FF_{16}

7.5 MUSIC PLUG INFO command

7.5.1 MUSIC PLUG INFO status command

The MUSIC PLUG INFO status command is used to inquire about the number of music plugs on one of the Music subunit determined by the AV/C address contained in the AV/C frame. The format of MUSIC PLUG INFO command frame is shown by the figure below.



	length	msb							lsb
opcode	1	MUSIC PLUG INFO($C0_{16}$)							
operand[0]	1	music_plug_type							

Figure 7.18 – MUSIC PLUG INFO status command frame

7.5.1.1 Field Definition

music_plug_type: The *music_plug_type* field is almost the same as Table 7.3, but 1 parameter is different. The following table illustrates the *music_plug_type* of MUSIC PLUG INFO status command.

Table 7.18 – Music Plug Type

Value	Meaning
00_{16}	Audio
01_{16}	MIDI
02_{16}	SMPTE time code
03_{16}	Sample Count
80_{16}	Audio SYNC
FF_{16}	all kind of plugs
all others	reserved for future specification

7.5.1.2 MUSIC PLUG INFO status command response

The MUSIC PLUG INFO status command response is shown below.

	length	msb							lsb
opcode	1	MUSIC PLUG INFO($C0_{16}$)							
operand[0]	1	FF_{16}							
operand[1]	1	number_of_music_plug_type_info = n							
operand[2]	5	music_plug_type_info[0]							
:									
:		:							
:	5	music_plug_type_info[n-1]							
:									

Figure 7.19 – MUSIC PLUG INFO status command response format

music_plug_type_info: The content of the *music_plug_type_info* field is shown below.

	length	msb							lsb
operand[x]	1	music_plug_type							
operand[x+1]	2	number_of_music_input_plug							
operand[x+2]									
operand[x+3]	2	number_of_music_output_plug							
operand[x+4]									

Figure 7.20 – Music Plug Type Info Format

music_plug_type: The *music_plug_type* field is defined in Table 7.3.

number_of_music_input_plug: In the *music_plug_type_info* format, operand[x+1] and operand[x+2] shall indicate the number of Music input plugs in the subunit.

number_of_music_output_plug: In the *music_plug_type_info* format, operand[x+3] and operand[x+4] shall indicate the number of Music output plugs in the subunit.

7.6 CURRENT CAPABILITY command

7.6.1 CURRENT CAPABILITY status command

The CURRENT CAPABILITY status command configures the music input or output plug format that is currently used. The format of CURRENT CAPABILITY command frame is shown below.

	length	msb							lsb
opcode	1	CURRENT CAPABILITY($C1_{16}$)							
operand[0]	1	music_plug_direction							
operand[1]	1	music_plug_type							
operand[2]	1	FF_{16}							
operand[3]	2	start_of_music_plug_ID = n							
operand[4]									
operand[5]	2	end_of_music_plug_ID = m							
operand[6]									

Figure 7.21 – CURRENT CAPABILITY status command format

7.6.1.1 Field Definitions

music_plug_direction: The *music_plug_direction* field indicates the direction of music plugs. The encoding of the *music_plug_direction* field is shown in the table below.

Table 7.19 – Music Plug Direction

Value	Meaning
00_{16}	music_input_plug
01_{16}	music_output_plug
all others	reserved for future specification

music_plug_type: The *music_plug_type* field is defined in Table 7.3.

start_of_music_plug_ID: The *start_of_music_plug_ID* field indicates the music plug ID at the head of the *music_plug_format_info*.

end_of_music_plug_ID: The *end_of_music_plug_ID* field indicates the music plug ID at the end of the *music_plug_format_info*.

7.6.1.2 CURRENT CAPABILITY status command response

The format of CURRENT CAPABILITY command response is shown below.

	length	msb							lsb
opcode	1	CURRENT CAPABILITY($C1_{16}$)							
operand[0]	1	music_plug_direction							
operand[1]	1	music_plug_type							
operand[2]	1	music_plug_attribute							
operand[3]	2	start_of_music_plug_ID = n							
operand[4]									
operand[5]	2	end_of_music_plug_ID = m							
operand[6]									
operand[2]	3	music_plug_format_info[n]							
:									
:		:							
:	3	music_plug_format_info[m]							
:									

Figure 7.22 – CURRENT CAPABILITY status command response format

music_plug_attribute: The *music_plug_attribute* field indicates the configuration method of music plugs. The encoding of the *music_plub_attribute* field is shown in the table below.

Table 7.20 – Music Plug Attribute

Value	Meaning
00_{16}	**simple:** Specified music plugs have quite same *music_plug_type*.
01_{16}	**compound:** Specified music plugs have each *music_plug_type*.
all others	reserved for future specification

music_plug_format_info: The content of music_plug_format_info field is shown below.

	length	msb							lsb
operand[x]	2	music_plug_ID							
operand[x+1]									
operand[x+2]	2	format_info							
operand[x+3]									

Figure 7.23 – Music Plug Format Info

format_info: If the *music_plug_type* is audio, the encoding of the *format_info* field is the same as *available_format_info* field in the *audio_capability* field of *music_subunit_specific_information.* The format is shown in the Figure 5.9, Table 5.7 and Table 5.8.

If the *music_plug_type* is MIDI, the *format_info* field is shown in the figure below.

	length	msb							lsb
operand[x]	1	MIDI_version				MIDI_revision			
operand[x+1]	1	MIDI_adaptation_layer_version							

Figure 7.24 – MIDI Format Info

MIDI_version, MIDI_revision: The *MIDI_version* field and the *MIDI_revision* field are defined in Figure 5.11.

MIDI_adaptation_layer_version: The *MIDI_adaptation_layer_version* field is defined in Table 5.9.

If the *music_plug_type* is SMPTE time code, the *format_info* field is shown in the figure below.

	length	msb							lsb
operand[x]	1	reserved						Tx	Rx
operand[x+1]	1	reserved							

Figure 7.25 – SMPTE Time Code Format Info

The following table illustrates the *SMPTE_time_code_format_info.*

Table 7.21 – SMPTE Time Code Format Info

SMPTE Time Code Format Information	Meaning
xxxx xxx1	**Rx:** If this bit is set to 1, then this subunit can receive the SMPTE time code data.
xxxx xx1x	**Tx:** If this bit is set to 1, then this subunit can transmit the SMPTE time code data.
all others	reserved for future specification

If the *music_plug_type* is Sample Count, the *format_info* field is shown in the figure below.

	length	msb							lsb
operand[x]	1	reserved						Tx	Rx
operand[x+1]	1	reserved							

Figure 7.26 – Sample Count Format Info

The following table illustrates the s*ample_count_format_info.*

Table 7.22 – Sample Count Format Info

Sample Count Format Info	Meaning
xxxx xxx1	**Rx:** If this bit is set to 1, then this subunit can receive the Sample Count data.
xxxx xx1x	**Tx:** If this bit is set to 1, then this subunit can transmit the Sample Count data.
all others	reserved for future specification

If the *music_plug_type* is audio SYNC the *format_info* field is shown in the figure below.

	length	msb							lsb
operand[x]	1	reserved						Ex	Bus
operand[x+1]	1	reserved							

Figure 7.27 – Audio SYNC Format Info

The following table illustrates the *audio_SYNC_format_info* field.

Table 7.23 – Audio SYNC Format Info

Audio SYNC Format Info	Meaning
xxxx xxx1	**Bus:** If this bit is set to 1, then this subunit can receive the audio SYNC from 1394 bus.
xxxx xx1x	**Ex:** If this bit is set to 1, then this subunit can receive the audio SYNC from sync. source.
all others	reserved for future specification

Annexes

Annex A: Connection Establishment Scenario

This chapter describes an example of procedures to establish a connection between a Music output plug on a talker node and a Music input plug on a listener node. Suppose that a controller is provided the following parameters:

- for talker: talker's node ID (or GUID), Music output plug type and Music output plug ID
- for listener: listener's node ID (or GUID), Music input plug type and Music input plug ID

And the controller is able to establish a connection by issuing AV/C commands to both the talker and the listener.

This scenario assumes that the controller, talker and listener node implement mandatory commands (and some optional commands if necessary) of the "AV/C General Specification 4.0"[R6] and the "AV/C Connection and Compatibility Management Specification 1.0"[R9].

A.1 (Procedure 1) Acquisition of the stream information for Music output plug

A.1.1 Using SOURCE PLUG CONFIGURE status command

The controller issues the SOURCE PLUG CONFIGURE status command to the talker to find out which subunit source plug is connected to the given Music output plug. The response of the command indicates the subunit source plug ID connected to the Music output plug and the stream index.

The following figure shows the command format that the controller should issue for this purpose.

	length	msb							lsb
opcode	1	SOURCE PLUG CONFIGURE(41_{16})							
operand[0]	1	number_of_subcommands = n							
operand[1]	1	result_status = FF_{16}							
operand[2]	1	number_of_completed_subcommands = FF_{16}							
operand[3]	1	result_status = FF_{16}							
operand[4]	1	music_plug_type							
operand[5]	1	music_plug_ID							
operand[6]	1								
operand[7]	1	subunit_plug_ID = FF_{16}							
operand[8]	1	stream_position[0] = FF_{16}							
operand[9]	1	stream_position[1] = FF_{16}							

Figure A.7.1 – SOURCE PLUG CONFIGURE status command frame

music_plug_type: The *music_plug_type* field indicates the Music output plug type provided for the controller.

music_plug_ID: The *music_plug_ID* field indicates the Music output plug type provided for the controller.

subunit_plug_ID, stream_position: The *subunit_plug_ID* and *stream_position* field shall be set to FF_{16} in the SOURCE PLUG CONFIGURE status command.

The status response indicated in the following figure should be returned for the status command.

	length	msb							lsb
opcode	1	SOURCE PLUG CONFIGURE(41_{16})							
operand[0]	1	number_of_subcommands = n							
operand[1]	1	result_status = FF_{16}							
operand[2]	1	number_of_completed_subcommands = FF_{16}							
operand[3]	1	result_status = FF_{16}							
operand[4]	1	music_plug_type							
operand[5]	1	music_plug_ID							
operand[6]	1								
operand[7]	1	subunit_plug_ID							
operand[8]	1	stream_position[0]							
operand[9]	1	stream_position[1]							

Figure A.1.2 – SOURCE CONFIGURE status response

music_plug_type: The *music_plug_type* field shall be the same as the status command.

music_plug_ID: The *music_plug_ID* field shall be the same as the status command.

subunit_plug_ID: The *subunit_plug_ID* field indicates the identifier of the subunit source plug connected to the Music output plug specified by the *music_plug_type* and the *music_plug_ID* field.

stream_position: The *stream_position* field indicates the location of the audio or music data stream handled by the Music output plug in an isochronous stream. The audio or music data stream is multiplexed in the isochronous stream and sent out through the subunit source plug (indicated by the *subunit_plug_ID* field) and the output serial bus plug.

Note: This concept is described in the "AV/C Audio and Music Data Protocol 1.0"[R10]. Please refer to the document for detailed specification.

A.1.2 Using SOURCE CONFIGURATIONS status command

In a typical controller application, a controller may collect all the target information first and provide it to the user so that the user can select the proper one among the provided alternatives. For that purpose, the controller may issue the SOURCE CONFIGURATIONS status command to inquire the configuration of the talker's subunit source plugs.

Before issuing the SOURCE CONFIGURATIONS status command, the controller may issue the PLUG INFO status command to the talker's Music subunit to acquire the number of subunit source plugs in the Music subunit. (Therefore, the controller should implement the PLUG INFO status command in this case).

	length	msb							lsb
opcode	1	PLUG INFO (02_{16})							
operand[0]	1	subfunction = 00_{16} (for units and subunits)							
operand[1]	4	all FF_{16}							
:									
operand[4]									

Figure A.1.3 – PLUG INFO status command frame

The following PLUG INFO status response may be returned.

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

Figure A.1.4 – PLUG INFO status response frame

destination_plugs: The *destination_plugs* field indicates the number of destination subunit plugs on the target Music subunit.

source_plugs: The *source_plugs* field indicates the number of subunit source plugs on the target Music subunit.

The controller may issue the SOURCE CONFIGURATIONS status command for each subunit source plug and find the subunit source plug connected to the specified Music output plug.

	length	msb							lsb
opcode	1	SOURCE CONFIGURATIONS(43_{16})							
operand[0]	1	subunit_plug_ID							

Figure A.1.5 – SOURCE CONFIGURATIONS status command frame

After this procedure, the controller can find the connection (1) (connection between the Music output plug and the subunit source plug) illustrated in the figure below:

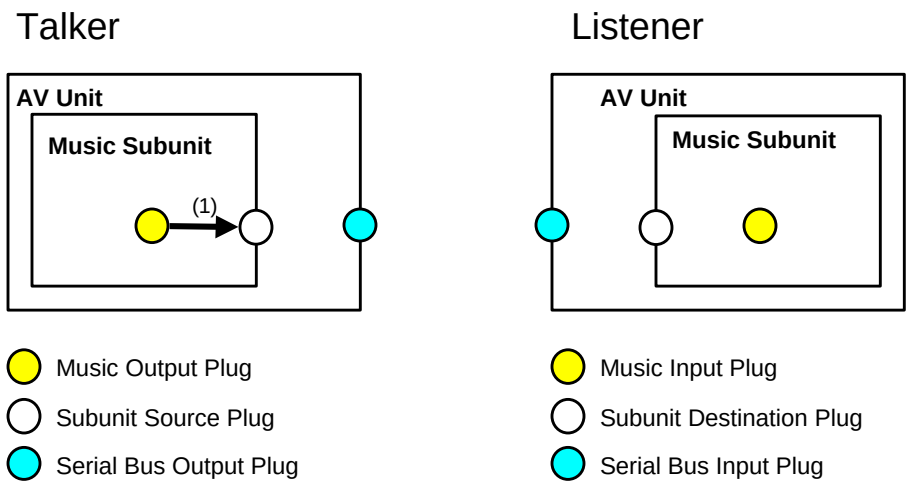


Figure A.1.6 – Connection between Music Output Plug and Source Subunit Plug

A.2 (Procedure 2) Connection between source subunit plug and serial bus output plug

The controller issues the SIGNAL SOURCE control command to the talker to establish a connection between the subunit source plug and the serial bus output plug. In the control command, subunit source plug is specified as "signal_source", and the serial bus output plug is specified as "signal_destination".

The SIGNAL SOURCE command is defined in the "AV/C Connection and Compatibility Management Specification 1.0"[R9]. Please refer to the document for detailed specification.

	length	msb							lsb
opcode	1	SIGNAL SOURCE ($1A_{16}$)							
operand[0]	1	reserved						F_{16}	
operand[1]	1	subunit_type = Music Subunit ($0C_{16}$)					subunit_ID		
operand[2]	1	source_plug_ID							
operand[3]	1	FF_{16} (unit output plug)							
operand[4]	1	destination_plug_ID = Any available Serial Bus oPCR ($7F_{16}$)							

Figure A.1.7 – SIGNAL SOURCE control command frame

signal_source (operand[1] and operand[2]):

The subunit source plug in the music subunit is specified as the signal source.

subunit_type: The *subunit_type* field indicates the music subunit.

subunit_ID: The *subunit_ID* field indicates the identifier of the music subunit. The value depends on the implementation.

source_plug_ID: The *source_plug_ID* field indicates the identifier of the subunit source plug connected to the specified Music output plug. This value must have been acquired in the Procedure 1.

signal_destination (operand[3] and operand[4]):

The serial bus output plug is specified as the signal destination. Therefore, as the signal destination is a unit output plug, the value of the operand [3] should be FF_{16}.

destination_plug_ID: Since the talker determines the identifier of the serial bus output plug, the controller should put $7F_{16}$ (any available serial bus output plug) into the *destination_plug_ID* field.

The connection (1) and (2) shown in the figure below has been established so far as.

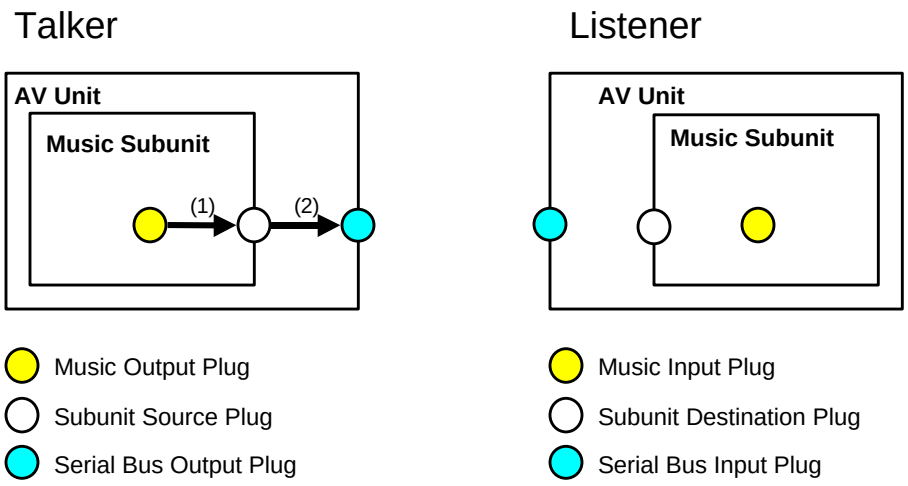


Figure A.1.8 – Connection between Music Output Plug and Serial Bus Output Plug

A.3 (Procedure 3) Acquisition of the connection between Music Input Plug and Destination Subunit Plug

The controller issues the DESTINATION PLUG CONFIGURE status command to the listener to find out which subunit destination plug is connected to the provided Music input plug. The response for the command indicates the subunit destination plug ID connected to the Music input plug and the stream index.

	length	msb							lsb
opcode	1	DESTINATION PLUG CONFIGURE(40_{16})							
operand[0]	1	number_of_subcommands = 1							
operand[1]	1	result_status = FF_{16}							
operand[2]	1	number_of_completed_subcommands = FF_{16}							
operand[3]	1	result_status = FF_{16}							
operand[4]	1	music_plug_type = audio (00_{16})							
operand[5]	1	music_plug_ID							
operand[6]	1								
operand[7]	1	subunit_plug_ID = FF_{16}							
operand[8]	1	stream_position[0] = FF_{16}							
operand[9]	1	stream_position[1] = FF_{16}							

Figure A.1.9 – DESTINATION PLUG CONFIGURE status command

number_of_subcommands: The *number_of_subcommands* field indicates the number of subcommands contained in the command. In this case, the value 1 should be set to the field.

music_plug_ID: The *music_plug_ID* field indicates the identifier of the Music input plug to be queried.

If the Music input plug is connected with a subunit destination plug, the identifier of the subunit destination plug is returned in the *subunit_plug_ID* field. In this case, the controller should issue the INPUT SELECT control command to connect the subunit destination plug with a certain serial bus input plug according to procedure 4 (see A.4).

If the Music input plug is not connected with any subunit destination plug, the controller should issue the INPUT SELECT control command to connect an arbitrary subunit destination plug and an arbitrary serial bus input plug according to procedure 4 (see A.4). After the establishment, the controller should issue the DESTINATION PLUG CONFIGURE control command to connect the Music input plug with the subunit destination plug according to procedure 5 (see A.5).

A.4 (Procedure 4) Connection between Input Serial Bus Plug and Destination Subunit Plug

The controller issues the INPUT SELECT control command to the AV unit on the listener node to establish a connection between music subunits on the talker and the listener node.

The INPUT SELECT command is defined in the “AV/C Connection and Compatibility Management Specification 1.0”[R9]. Please refer to the document for detailed specification.

	length	msb							lsb
opcode	1	INPUT SELECT ($1B_{16}$)							
operand[0]	1	subfunction = CONNECT (00_{16})							
operand[1]	1	reserved					F_{16}		
operand[2]	1	node_ID							
operand[3]	1								
operand[4]	1	output_plug							
operand[5]	1	input_plug = FF_{16}							
operand[6]	1	subunit_type = Music Subunit ($0C_{16}$)					subunit_ID		
operand[7]	1	destination_plug_ID							
operand[8]	1	reserved							

Figure A.1.10 – INPUT SELECT control command

node_ID: The *node_ID* field indicates the node ID of the talker node.

output_plug: The *output_plug* field indicates the serial bus output plug ID on the talker node to be connected to the input plug. This value must have acquired by procedure 2.

input_plug: The *input_plug* field indicates the listener's serial bus input plug to be connected. Since the listener determines the identifier of the serial bus input plug, the controller should set the field to FF_{16}.

signal_destination (operand[6] and operand[7]):

subunit_type: The *subunit_type* field indicates the Music subunit.

subunit_ID: The *subunit_ID* field indicates the identifier of the Music subunit. The value depends on the implementation.

destination_plug_ID: The *destination_plug_ID* field indicates the identifier of the subunit destination plug on the listener node. A controller may choose not to select any specific plug and rely on the listener node to choose instead. The controller may set FF_{16} to the *destination_plug_ID* field.

If the control command succeeds, the control response shown in the figure below should be returned.

	length	msb							lsb
opcode	1	INPUT SELECT ($1B_{16}$)							
operand[0]	1	subfunction = CONNECT (00_{16})							
operand[1]	1	reserved							result_status = no_error(0)
operand[2]	1	node_ID							
operand[3]	1								
operand[4]	1	output_plug							
operand[5]	1	input_plug = Serial Bus iPCR							
operand[6]	1	subunit_type = Music Subunit ($0C_{16}$)							subunit_ID
operand[7]	1	destination_plug_ID							
operand[8]	1	reserved							

Figure A.1.11 – INPUT SELECT control response frame

node_ID, output_plug: The *node_ID* and *output_plug* field shall be the same as the control command.

input_plug: The identifier of the serial bus input plug selected by the listener is returned.

So far, the connections (1), (2), (3) and (4) shown in the figure below have been established.

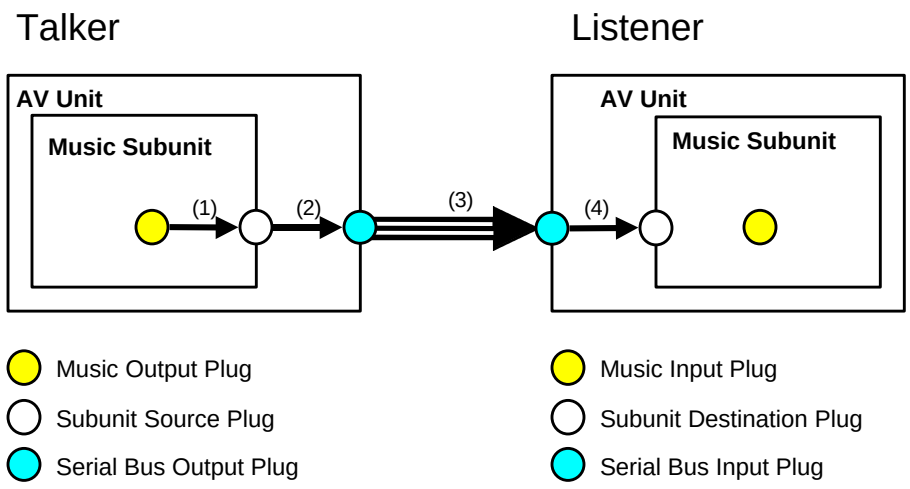


Figure A.1.12 – Connection between Music Output Plug and Destination Subunit Plug

A.5 (Procedure 5) Connecting Input Music Plug and Destination Subunit Plug

The controller should issue the DESTINATION PLUG CONFIGURE control command to the Music subunit on the listener to connect the Music input plug and the subunit destination plug.

	length	msb							lsb
opcode	1	DESTINATION PLUG CONFIGURE(40_{16})							
operand[0]	1	number_of_subcommands = 1							
operand[1]	1	result_status = FF_{16}							
operand[2]	1	number_of_completed_subcommands = FF_{16}							
operand[3]	1	subcommand[0] = CONNECT							
operand[4]	1	music_plug_type							
operand[5]	1	music_plug_ID							
operand[6]	1								
operand[7]	1	subunit_plug_ID							
operand[8]	1	stream_position[0]							
operand[9]	1	stream_position[1]							

Figure A.1.13 – DESTINATION PLUG CONFIGURE control command

music_plug_type, music_plug_ID: The *music_plug_type* and *music_plug_ID* field indicates the Music input plug provided to the controller. The meaning of those fields is defined in Table 7-3 and Table 7-4.

subunit_plug_ID: The *subunit_plug_ID* field indicates the identifier of the subunit destination plug to be connected to the Music input plug. The identifier must have been acquired by procedure 4.

stream_position: The *stream_position* field indicates the location of the audio or music data stream. The Music input plug should receive the stream from the isochronous stream handled by subunit destination plug. The audio or music data stream is multiplexed in the isochronous stream and received through the subunit destination plug (indicated by the *subunit_plug_ID* field) and the serial bus input plug.

As shown in the figure below, the connection from the Music output plug to the Music input plug has been established.

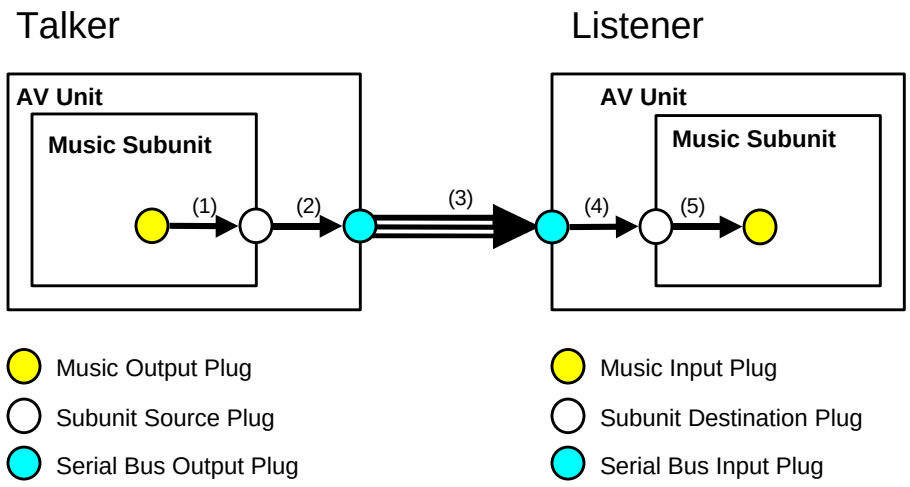


Figure A.1.14 – Connection between Music Output Plug and Music Input Plug