TA Document 2002002
AV/C Disc Subunit -
Generic Recordable Video Disc Media Type Specification 1.0

September 11, 2003

Sponsored by:
1394 Trade Association

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

Abstract:
This document describes the Generic Recordable Video Disc media type specific part of the Disc General Subunit Specification.

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

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

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

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

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

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

Editor, 1394 Trade Association
1111 South Main Street, Suite 100
Grapevine, TX 76051
USA

Table of contents

List of figures

List of tables

Change history

Version 1.0 (March 5, 2002)

Original version.

1. Overview

1.1 Purpose

This specification is intended to supplement *AV/C Disc Subunit General, version 1.1, AV/C Digital Interface Command Set Specification 4.0, AV/C Descriptor Mechanism 1.1* and *AV/C Information Block Types 1.0*.

The features specific to the Generic Recordable Video Disc are described in this specification.

The purpose of this specification is to define the basic function of discs that can record AV objects as one virtual media type so that they can be controlled by the Controller with the same method regardless of the types of disc if the target is implemented with this media type.

The Generic Recordable Video Disc media type is characterized by different application profiles. One or more Profile IDs define the requirements for the Generic Recordable Video Disc of a particular type and the subset of AV/C commands implemented in the Generic Recordable Video Disc of that type.

1.2 Scope

The scope of this revision of the standard is to define the requirements for the Player/Recorder Profile. The command set and its operational features are defined in this specification. An application model for the Player/Recorder Profile for a typical Play and Record operation is developed.

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] ISO/IEC 13123:1994, Control and Status Register (CSR) Architecture for Microcomputer Buses[1]

[R2] IEEE Std 1394-1995, Standard for a High Performance Serial Bus[2]

[R3] IEC 61883, Digital Interface for Consumer Electronic Audio/Video Equipment[3]

[R4] TA Document 1999026, AV/C Digital Interface Command Set General, version4.0[4], July 23, 2001

[R5] TA Document 1999045, AV/C Information Block Types Specification, version 1.0[4], July 23, 2001

[R6] TA Document 2001021, AV/C Descriptor Mechanism Specification, version 1.1[4] Draft

[R7] TA Document 2001022, AV/C Disc Subunit General, version 1.1[4] Draft

[1] ISO/IEC [ANSI/IEEE] publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA or from International Electrotechnical Commission, 3 rue de Varembé, Case Postale 131, CH 1211, Genève 20, Switzerland/Suisse.

[2] IEEE publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331 USA.

[3] Document 100C/182/FDIS from: U.S. National Committee of The IEC ANSI, 11 West 42nd Street, 13th Floor, New York, NY 10036 USA. Phone: +1.212.642.4900(Questions) 212.642.4980(Sales). FAX: +1.212.398.0023.

[4] Accepted 1394 Trade Association (AV/C) documents may be ordered through: 1394 Trade Association Regency Plaza Suite 350, 2350 Mission College Blvd., Santa Clara, CA 95054, USA; by contacting taadmin@1394ta.org; by retrieving a PDF-format copy from the 1394 Trade Association web-site at: http://www.1394ta.org/abouttech/specifications/techspec.html.

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.

3.1.7 Controller Read/Write: Each field in a descriptor can either be read/write(R/write/ignore(R/W/I) read/ignore(R/I), or read(R) by an external controller. This qualifier indicates whether the field can be written using WRITE DESCRIPTOR and WRITE INFO BLOCK commands with the partial_replace subfunction.

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 AV Content Object: An Object Entry that describes an AV Track and its contents. There are AV Content Objects for Digital Video, Audio, and Digital Still Images among others. AV Content Objects have an entry_type value in the range of 80_{16} to $8F_{16}$.

3.2.2 AV Frame: A portion of an AV Track defined by the specific data encoding method applied to that AV Track. For example MPEG encoded AV Tracks have I ,P and B frames as defined by MPEG documentation.

3.2.3 AV Track: A collection of recorded data that is described by an AV Content Object Entry. This definition includes video, audio, textual and other types of data. Although sometimes referred to simply as Tracks, the name AV tracks is intended to differentiate AV content from the tracks normally associated with hard disk drive devices.

3.2.4 AV Segment: A partial AV Track defined as the portion of an AV Track between two adjacent positions. AV segments are useful for performance and editing operations on AV Tracks.

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

3.2.6 AV/C: Audio/video control, as in the AV/C Digital Interface Command Set specified by this document.

3.2.7 byte: Eight bits of data.

3.2.8 CSR: A node or unit Control and Status Register, as defined by IEEE Std 1394–1995.

3.2.9 EUI-64: Extended Unique Identifier, 64-bits, as defined by the IEEE. The EUI-64 is a concatenation of the 24-bit company_ID obtained from the IEEE Registration Authority Committee (RAC) and a 40-bit number (typically a silicon serial number) that the vendor identified by company_ID guarantees to be unique for all of its products. The EUI-64 is also known as the node unique ID and is redundantly present in a node's configuration ROM in both the Bus_Info_Block and the Node_Unique_Id leaf.

3.2.10 FCP: Function Control Protocol, as defined by IEC 61883, Digital Interface for Consumer Electronic Audio/Video Equipment[R3].

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

isochronous: A term that indicates the essential characteristic of a time-scale or signal, such that the time intervals between consecutive instances either have the same duration or durations that are integral multiples of the shortest duration. In the context of Serial Bus, "isochronous" is taken to mean a bounded worst-case latency for the transmission of data; physical and logical constraints that introduce jitter preclude the exact definition of "isochronous."

3.2.12 module: The smallest component of physical management, *i.e.*, a replaceable device.

3.2.13 nibble: Four bits of data. A byte is composed of two nibbles.

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

3.2.15 node ID: A 16-bit number, unique within the context of an interconnected group of Serial Buses. The node ID is used to identify both the source and destination of Serial Bus asynchronous data packets. It can identify one single device within the addressable group of Serial Buses (unicast), or it can identify all devices (broadcast).

3.2.16 Object Entry: AV/C descriptor structure type that contains descriptive information. It is commonly used to describe AV-content (a track) that is recorded and stored on the disc device media.

3.2.17 Object List: AV/C descriptor structure type that contains some number of Object Entries.

3.2.18 PCR: Plug Control Register, as defined by IEC 61883, Digital Interface for Consumer Electronic Audio/Video Equipment[R3].

3.2.19 iPCR: Input plug PCR, as defined by IEC 61883.

3.2.20 oPCR: Output plug PCR, as defined by IEC 61883.

3.2.21 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.

3.2.22 quadlet: Four bytes of data.

3.2.23 Serial Bus: The physical interconnections and higher level protocols for the peer-to-peer transport of serial data, as defined by IEEE Std 1394–1995.

3.2.24 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.

3.3 Acronyms and abbreviations

AV/C	Audio Video Control

4. The Generic Recordable Video Disc model

4.1 Device Profiles

The capabilities of disc devices may range from a basic recorder and a player to a complex video and audio editor with features to access an AV content by multiple users simultaneously and reliably. The minimum capabilities of a disc are indicated by a parameter called Profile ID.

This specification applies to a profile known as a Player/Recorder Profile. Future revisions will address other disc devices with different profiles.

Different profiles may be supported simultaneously in a multi-profile Generic Recordable Video Disc. It is not necessary that a profile be a subset of any other profiles.

4.1.1 Player/Recorder Profile

The Player/Recorder Profile supports the functions of recording (storage) and playback of multiple AV tracks using 1394 Isochronous transfers. It does not require an asynchronous connection support. It supports the descriptor model specified in [R7]. The Player/Recorder Profile may not support all the possible fields of a given descriptor. The limitations are described in this specification.

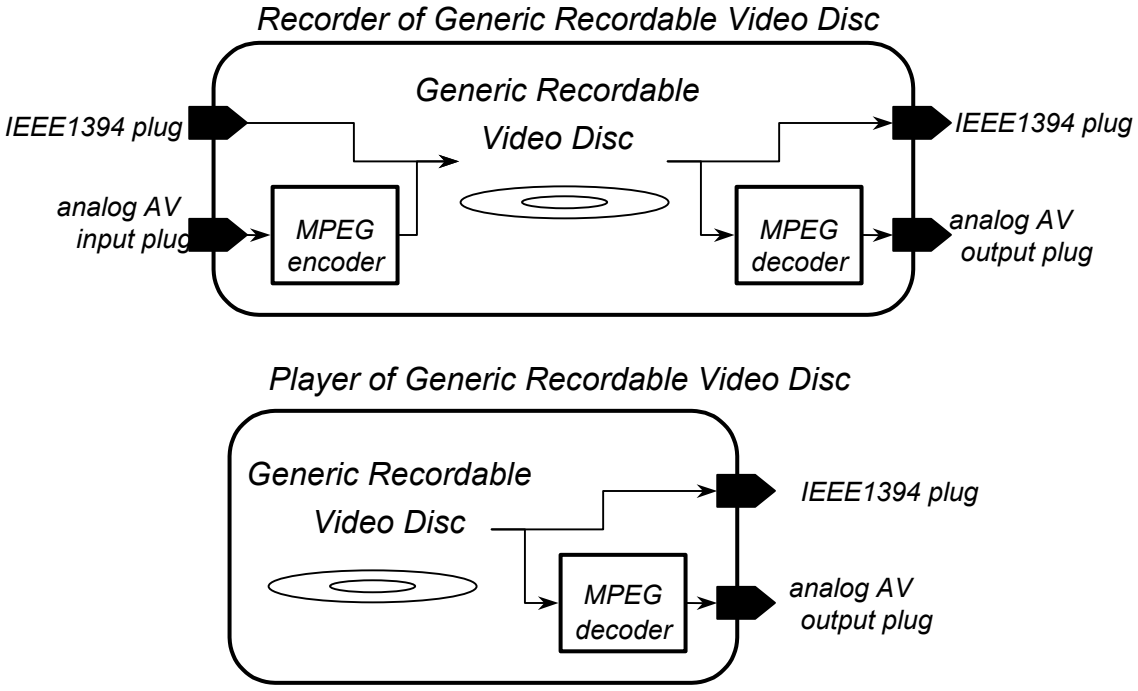

Figure 4.1 – Generic Video Disc player/recorder

The Generic Recordable Video Disc (Player/Recorder Profile) is a digital recorder/player with the following capabilities:

— Play and Record AV tracks. The Generic Recordable Video Disc permits Play and Record operations on a given AV track or tracks. (A device may not have a function of recording.)

— Pause AV tracks. The Generic Recordable Video Disc supports Pause for Play and Record operations. If the record and play operations are active upon the same AV track, the play operation can be paused while recording on the AV track continues.

— Position within AV tracks. The Generic Recordable Video Disc supports search operations to position the playback or recording point in a given AV Track.

The basic operations of the Player/Recorder Profile provide a real-time control of individual AV tracks, plus a random access to individual AV tracks and data files. Stream control includes Stop, Record, Record Pause, Play, Play Pause, Pause Resume, Play Fast Forward, Play Rewind, Search and Simultaneous Record and/or Playback of the same or multiple AV tracks.

4.2 AV Content

The AV content area of the disc device stores all of the data recorded using the AV/C commands (or content that was placed on the disc during manufacture). Included in this area is the descriptor data (object lists, object entries, etc.) The AV content area is a logically contiguous area, and is not divided into multiple AV content areas. Such content area division is not supported by this model.

5. Generic Recordable Video Disc Features and Capabilities

The descriptor structures supported by Player/Recorder Profile are:

- Subunit Identifier
- Subunit Status Descriptor
- Root Contents List

The structures are developed for the Player/Recorder Profile in this revision. Future revisions will address these issues for the additional Generic Recordable Video Disc device profiles.

5.1 Subunit Identifier Descriptor

The structure of the Subunit Identifier is as defined in [R6].

5.1.1 Size value

The *size_of_list_ID*, *size_of_objectID* and the *size_of_object_position* fields are limited as below:

Table 5.1 – the size value

Field name	Value
Size_of_list_ID	02_{16}
Size_of_object_ID	02_{16}
Size_of_object_position	02_{16}

5.1.2 Root_object_list_ID

The number of the *root_object_lists* must be specified as at least 1 for a Player/Recorder Profile. The user is not expected to create the list explicitly. The disc device creates the list on first initialization. The functional characteristics of the subunit are defined in the subunit dependent information field of the Subunit Identifier structure.

Mandatory list ID in the *Root_object_list_ID* field of subunit identifier descriptor follows:

- list ID of root contents list ($30\ 00_{16}$)

5.1.3 Disc subunit dependent information

Table 5.2 – Disc subunit attributes value

Attribute name	Value
has_more_attributes	0_2
supports_copyright	1_2

The *disc_subunit_version* field shall contain a value of $0x12_{16}$. A value of $0x12_{16}$ indicates conformance to [R7].

5.2 Supported_media_type_specification

5.2.1 Generic Recordable Video Disc Implementation_profile_ID

Table 5.3 – Generic Recordable Video Disc Implementation_profile_ID

Implementation_profile_ID	Meaning
10_{16}	Player/Recorder Profile
FF_{16}	Extended Implementation Profiles Exist
All others	Reserved for future specification

5.2.2 Media_type_attributes field

Table 5.4 – media_type_attributes field

Name	Attribute value
has_more_attributes	0_2
Supports_hierarchical_storage	0_2
Supports_two_sided_media	0_2

The *can_record* field indicates the recordable capability of the Generic Recordable Video Disc device.

Table 5.5 – can_record field

can_record field value	Meaning
1_2	This subunit have recording function.
0_2	This subunit does not have recording function.

NOTE —Even if the subunit has a recording function, there is a possibility that it doesn't have a recording function in a particular type of the media. In that case, the field should be set to 1_2, and set 00_2 to recordable bits in the *disc_recordable_information* field in the *list_specific_information* which is in root contents list of loaded media.

5.2.3 Generic Recordable Video Disc type_dependent_information

Address	Length, bytes	External Read/Write	Contents
00 00_{16}	1	R	Generic_recordable_video_disc_version
00 01_{16}	4	R	Maximum_number_of_recordable_objects
00 02_{16}			
00 03_{16}			
00 04_{16}			
00 05_{16}	1	R	Number_of_extended_implementation_profiles (= n)
00 06_{16}	1	R	Extended_implementation_profile_ID[0]
00 07_{16}	1	R	Extended_generic_recordable_video_disc_version[0]
:			:
:			:
:	1	R	Extended_implementation_profile_ID[n-1]
:	1	R	Extended_generic_recordable_video_disc_version[n-1]

Figure 5.1 – Generic_recordable_video_disc_ type_dependent_information

The *Generic_recordable_video_disc_ version* field indicates the version number of the Generic Recordable Video Disc media type specification, to which the *implementation_profile_id* specified for this subunit, if not a multi-profile generic_video_disc_recorder conforms. This field contains an ordinal value, which is increased with each revision of the generic recordable video disc media type specification. If the *implementation_profile_id* is set to FF_{16}, indicating that extended implementation profiles exist, then the generic_video_disc_recorder_version field may also be set to FF_{16}.

Table 5.6 – generic_recordable_video_disc _version

generic_recordable_video_disc _version	Meaning
00_{16}	Version 1.0 of Generic Recordable Video Disc media type specification
All others	Reserved for future version

6. Other descriptors

6.1 Status descriptor

6.1.1 General Disc Subunit Status Area Info Block

This info block is as described in [R7]. It contains the General Disc Subunit Status Area Info Block (88 00_{16}) with the nested Media and Edit Status Info Block (88 04_{16}).

The undo and difference operations may not be supported by a Player/Recorder Profile. If not supported, the *undo_status* and *difference* fields are FF_{16} and 0_2 respectively.

6.1.2 Destination Plug Status Area Info Block

The Destination Plug Status Area Info Block structure is as defined in [R7]. The supported info blocks for Player/Recorder Profile are:

— Plug Status Info Block (88 05_{16})

— Operating Mode Info Block (88 06_{16})

— Plug Configuration Info Block (88 07_{16})

Refer to [R7] for the definition of the structure of this info block. The *AV_object_type* field and *object_and_plug_type specific information* is given in [R7].

— Position Info Block (00 03_{16}).

The structure of the info block is as specified in [R5]. It contains the Position Indicator Info Block (00 02_{16}). The position of current recording point of the AV object will be returned. When root contents list is assigned with the destination plug, the Player/Recorder Profile supports indicator type relative_HMSF_count. It is to be noted that the indicated time may not be frame accurate, as accuracy of position is implementation dependent.

6.1.3 Source Plug Status Area Info. Block

The Source Plug Status Area Info Block structure is as defined in [R7].

The supported info blocks for Player/Recorder Profile are:

— Plug Status Info Block (88 05_{16})

— Operating Mode Info Block (88 06_{16})

— Plug Configuration Info Block (88 07_{16})

Refer to [R7] for the definition of the structure of this info block. The *AV_object_type* field and *object_and_plug_type specific information* is given in [R7].

— Position Info Block (00 03_{16}).

The structure of the Info Block is as specified in [R7]. It contains the Position Indicator Info Block (00 02_{16}). The position of current play point of the AV object will be returned. In case the AV content (AV content object or root contents list) is assigned with the source plug, the Player/Recorder Profile supports indicator type relative_HMSF_count. It is to be noted that the indicated time may not be frame accurate, as accuracy of position is implementation dependent.

6.2 List descriptor length of Object Lists

In the following object lists, the list_descriptor_length field can be set to the value of 0000_{16}(unknown)

Table 6.1 – Object lists which`unknown´setting is allowed

List name	List type
Root Contents List	80_{16}
Root Temporary Contents List	82_{16}
Performance Lists	84_{16}

NOTE —Lists except Root Contents List are optional for Player/Recorder Profile.

6.3 Root contents list

The structure of Root Contents List as described in [R7], the Player/Recorder Profile supports only Root Contents List of list type 80_{16}. The List ID of root contents list is 30 00_{16}.

The structure of the Root Contents Lists is given below:

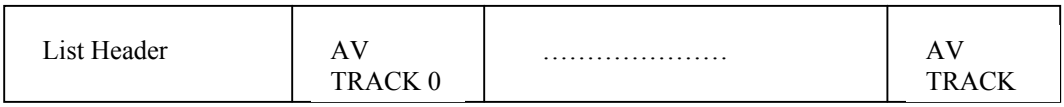

Figure 6.1 – The structure of Root Contents List

Attributes field in header part of root contents list is:

— *has_more_attributes* = 0

— *has_object_ID* = 1

NOTE —*has_child_ID* = 0 (this field is for object entry and not valid in list descriptor.)

The Info Blocks supported for the root contents list are:

— Time Stamp Info Block for descriptor modification (00 07_{16})

The *stamp_type* bit is set to 1_2 and the *time_stamp_data* field is used as 54bit counter.

The *time_stamp_data* field shall not be reset even if a medium is loaded or unloaded.

— Default Playlist Info Block (80 $0B_{16}$)

Set list ID of root contents list.

— Disc Capacity Info Block (80 02_{16})

All *capacity_format_indicator* fields are set to 01_{16} (“bytes”).

— Name Info Block (00 $0B_{16}$).

The Name Info block is a nested info block containing the additional Character Code Info block (00 08_{16}), the Language Code Info Block (00 09_{16}) and the Raw Text Info Block (00 $0A_{16}$).

6.4 Object Descriptor

The object descriptor structure within the Root Contents List as described in [R6]. The Player/Recorder Profile is intended to read and write the Video Content Objects as described in [R7].

Attributes field:

— *has_more_attributes* = 0

NOTE —*has_object_ID* = 0 (this field is for object list and not valid in object entry.)

The supported info blocks for the object descriptors are:

— Video Signal Mode Info Block (88 12_{16}).

— Video Stream Format Subtype Info Block (88 13_{16}).

— Size Indicator Info Block (00 01_{16}).

The supported type value is 00_{16} (indicating HMSF type).

— Time Stamp Info Block (00 04_{16}).

This is the content creation time.

— Program Attribute Info Block (88 14_{16}).

— Name Info Block (00 $0B_{16}$).

The Name Info block is a nested info block containing additional Character Code Info block (00 08_{16}), the Language Code Info Block (00 09_{16}) and the Raw Text Info Block (00 $0A_{16}$).

The Player/Recorder Profile shall be able to reference object entries by position within the list (20_{16}) and by object ID within the list (21_{16}), as described in [R6].

6.4.1 Supported Info Blocks

The following table is a list of info blocks that shall be supported by Player/Recorder Profile.

Table 6.2 – Generic Recordable Video Disc Info Blocks

Info block type	Info block name	The containing Descriptor
00 01_{16}	Size Indicator	Contained in the object descriptor of the Root Contents List
00 02_{16}	Position Indicator	Contained in the Position Info Block
00 03_{16}	Position	Plug Status Area
00 04_{16}	Time Stamp of Content Creation	Contained in the object descriptor of the Root Contents List

Info block type	Info block name	The containing Descriptor
00 07_{16}	Time Stamp of Descriptor Modification	Contained in the list specific information of the Root Contents List
00 08_{16}	Character Code	When present, it is contained in the object descriptor and nested in the Name Info Block
00 09_{16}	Language Code	When present, it is contained in the object descriptor and nested in the Name Info Block
00 $0A_{16}$	Raw Text	Contained in the object descriptor and nested in the Name Info Block
00 $0B_{16}$	Name	Contained in the list specific information and the object descriptor of the Root Contents List
80 02_{16}	Disc Capacity	Contained in the list specific information of the Root Contents List
80 $0B_{16}$	Default Playlist	Contained in the list specific information of the Root Contents List
88 00_{16}	General Disc Subunit Status Area	Disc Subunit Status Descriptor
88 04_{16}	Media and Edit Status	Nested in General Disc Subunit Status area
88 02_{16}	Source Plug Status Area	Contained in Disc Subunit Status Descriptor
88 01_{16}	Destination Plug Status Area	Contained in Disc Subunit Status Descriptor
88 05_{16}	Plug Status	Nested Info block within Source Plug Status Area and Destination Plug Status Area
88 06_{16}	Operating Mode Info	Nested info block within Plug Status info block
88 07_{16}	Plug Configuration	Nested info block within Plug Status info block
88 12_{16}	Video Signal Mode	Nested info block within Video Content Object
88 13_{16}	Video Stream Format Subtype	Nested info block within Video Content Object
88 14_{16}	Program Attribute	Nested info block within Video Content Object

7. Commands

7.1 Mandatory commands (AV/C Disc Subunit)

The following table contains commands that a Player/Recorder Profile shall implement. These commands are a subset of category A commands in [R7]. The command and response structure are as described in [R7].

Table 7.1 – Generic Recordable Video Disc commands (Disc)

Command	Opcode	Defined ctypes			Comments
		C	S	N	
CONFIGURE	$D1_{16}$	X	–	–	Prepare the subunit for a recording or playback operation
ERASE	40_{16}	X*	–	–	Erase the entire AV contents of the disc subunit, just a specified track, or a specified portion of the disc storage space
IMPORT/EXPORT MEDIUM	$C1_{16}$	X	–	–	Put the disc into or remove it from the drive mechanism
PLAY	$C3_{16}$	X	–	–	Begin playing the disc (immediate response)
RECORD (NEW)	$C2_{16}$	X*	–	–	Record a streaming object (audio track, etc.)
SEARCH (Search type is Position and search indicator is relative.)	50_{16}	X	–	–	Performs a search to the specified position on the selected AV track
SET PLUG ASSOCIATION	$D3_{16}$	X	–	–	Associate a List or Object with a Source or Destination Plug
STOP	$C5_{16}$	X	–	–	Stop the current operation

* : this command is not needed to be implemented on a device which does not have recording function.

7.1.1 CONFIGURE

In the Player/Recorder Profile, the CONFIGURE command is used for a plug configuration, and the *subfunction* 00_{16} (Reset to the Default Configuration) and 01_{16} (Set the Configuration) shall be supported.

7.1.2 ERASE

In the Player/Recorder Profile, *erase type* 00_{16} (“Erase All the Contents”) and 01_{16} (“Erase the Specified Object”) shall be supported. Both the descriptor and AV contents are deleted.

This command specifying a video object that is currently recording or playing shall be REJECTED.

This command is not needed to be implemented on a device which does not have recording function.

7.1.3 IMPORT/EXPORT MEDIUM

The operation of this command is as specified in [R7].

7.1.4 PLAY

In the Player/Recorder Profile, the following *subfunctions* shall be supported;

— *Subfunction_1:* 75_{16} (“Normal Play”) and $7D_{16}$ (“Forward Pause”).

7.1.5 RECORD

In the Player/Recorder Profile, the following *subfunctions* shall be supported;

— *Subfunction_1* : 75_{16} (“Record at Normal Speed”) and $7D_{16}$ (“Forward Pause”).

— *Subfunction_2* : 00_{16} (“Record New”).

This command is not needed to be implemented on a device which does not have recording function.

7.1.6 SEARCH - POSITION

In the Player/Recorder Profile, the *subfunction* 00_{16} (“Search Type is Position”) shall be supported. The search indicator value is 00_{16} i.e. search by *Relative_HMSF_count*.

NOTE — The search position may not be frame accurate and only is an estimate of the desired position.

7.1.7 SET PLUG ASSOCIATION

In the Player/Recorder Profile, the *subfunction* 01_{16} (“Set a Specified List/Plug Association”) shall be supported. No default association is supported. The associated List ID descriptor identifier supports the *descriptor_type* 10_{16} (“Object List Descriptor – specified by list ID”), the *descriptor_type* 20_{16} (“Object Entry Descriptor – specified by Position”) and *descriptor_type* 21_{16} (“Object Entry Descriptor – specified by an Object ID”).

7.1.8 STOP

The operation of this command is as specified in [R7]. The position where play operation ends shall be maintained by the disc device at least until the disc device enters a power off condition. This behavior emulates VCR operation.

7.2 Mandatory commands (AV/C General)

The Player/Recorder Profile supports the following additional commands. These commands are a subset of commands in [R6].

Table 7.2 – Generic Recordable Video Disc commands (General)

Command	Opcode	Defined ctypes			Comments
		C	S	N	
OPEN DESCRIPTOR	08_{16}	X	–	–	Gains the right to access the descriptor.
PLUG INFO	02_{16}	–	X	–	Information about serial bus and external plugs.
READ DESCRIPTOR	09_{16}	X	–	–	Read data from the descriptor.
READ INFO BLOCK	06_{16}	X	–	–	Read the specified info block.
WRITE INFO BLOCK	07_{16}	X*	–	–	Write data into specified info block.

* : this command is not needed to be implemented on a device which does not have recording function.

7.2.1 OPEN DESCRIPTOR

The operation of this command is as specified in [R6].

7.2.2 PLUG INFO

The device (Player/Recorder Profile) which has a recording function is required to support one source and one destination plug at a minimum. The device (Player/Recorder Profile) which does not have a recording function is required to support one source plug at a minimum. A disc device may support additional source and destination plug(s) optionally.

7.2.3 READ DESCRIPTOR

The operation of this command is as specified in [R6].

7.2.4 READ INFO BLOCK

The operation of this command is as specified in [R6].

7.2.5 WRITE INFO BLOCK

The operation of this command is as specified in [R6] with the following limitations. It is to be noted that a disc device creates Info Blocks at the time of object creation on “Record” command execution. It is to be noted that data in some of the info blocks can not be determined by the subunit. In these cases, such data may be initialized with values, which may be implementation dependent.

This command is not needed to be implemented on a device which does not have recording function.

Annexes

Annex A: Disc Subunit Commands (normative)

A.1 GENERIC RECORDABLE VIDEO DISC SUPPORT

The GENERIC RECORDABLE VIDEO DISC SUPPORT command is used to show whether the subunit and the currently loaded media can be controlled with the method specified by the Generic Recordable Video Disc Media Type Specification. It can also be used to identify the media format name of disc media currently loaded. This command is a status command

The status command has the following format:

Table A.1 – GENERIC RECORDABLE VIDEO DISC SUPPORT command

	length	ck	msb							lsb	
opcode	1	√	GENERIC RECORDABLE VIDEO DISC SUPPORT ($D7_{16}$)								*common*
operand[0]	1	√	result								*command*
operand[1]	1	√	subfunction_1								*header*
operand[2]	1	√	reserved								*part*
operand[3]	1	√	support_state								*original*
operand[4]	8	√	media_name								*original*
:											
operand[11]											

The fields of the *common_header_part* are as described above.

The *result* field in the response may have one of the following values:

Table A.2 – *result* field in the response

Response frame type	result	result code name	meaning
STABLE	00_{16}	success	Successful completion
	all other values		reserved for future specification
REJECTED	$F2_{16}$	POWER OFF	The disc subunit is powered off.
	$F3_{16}$	SUBUNIT BUSY	busy for another operation.
	FF_{16}	unknown	an unknown error occurred
	all other values		reserved for future specification

The *subfunction_1* field specifies the *implementation_profile_id* of the Generic Recordable Video Disc media type specification. When the target cannot deal with the profile specified in the *implementation_profile_id*, NOT IMPLEMENTED is returned.

The *support_state* field shall be filled with 00_{16} when the controller issues the GENERIC RECORDABLE VIDEO DISC SUPPORT status command.

Table A.3 – support_state

operand offset	msb							lsb
00_{16}	reserved						currently_ loaded_ media_ support	subunit_ support

When the *subunit_support* bit is set to 1 in the response, it means that this subunit has the capability for some particular media to be controlled by the methods which are defined in the Generic Recordable Video Disc Media Type specification with the specified *implementation_profile_id*.

When the *currently_loaded_media_support* bit is set to 1 in the response, it means that the currently loaded media can be controlled by the methods which are defined in the Generic Recordable Video Disc Media Type specification with the specified *implementation_profile_id*.

The *media_name* field shall be filled with FF_{16} when the controller issues the GENERIC RECORDABLE VIDEO DISC SUPPORT status command. In this field in the response, the name of the media are written with minimal ASCII English characters with less than 8 characters. This information can be used for displaying the name of the media to a user.

In the case when media is not loaded or when the target cannot show the name of the media, the target just returns the value of control command frame (filled with FF_{16}). The same applies when currently_loaded_media_support bit is 0_2.

The format of the STABLE response frame is the same as the control command frame, with the *subunit_support, currently_loaded_media_support, media_name*. If the command is REJECTED, then the response frame is exactly the same as the control command frame.

There is no CONTROL command and NOTIFY command for the GENERIC RECORDABLE VIDEO DISC SUPPORT command.

Annex B: New values for existing info block (normative)

B.1 Video Signal Mode Info Block (8812_{16})

This section describes the additional values for the Video Signal Mode Info Block (8812_{16}) which is defined in [R7].

The *video_signal_mode* specifies the type of video signal in effect. Newly defined values for this field are as follows:

Table B.1 – Video signal mode

Value	signal_mode
03_{16}	MPEG2-PS
FF_{16}	unknown

These values can be used in the object entry descriptors for indicating the video signal mode of the contents, and can also be used for a plug configuration purpose.

B.1.1 MPEG2-PS Signal Mode

The *video_signal_mode_specific_information* field for MPEG2-PS signal mode is defined below.

Table B.2 – Mode specific information for MPEG2-PS signal mode

Address Offset	length	msb							lsb
00 00_{16}	1	frame_rate_code				bit_rate_type		reserved	
00 01_{16}	3	(MSB)							
00 02_{16}		video_signal_bit_rate (x 10^3 bit/sec)							
00 03_{16}		(LSB)							

The *frame_rate_code* field indicates the frame rate of MPEG2-PS video stream.

Table B.3 – Frame rate code

Value	frames/second
0_{16}	Forbidden
1_{16}	24000/1001
2_{16}	24
3_{16}	25
4_{16}	30000/1001 (29.97)
5_{16}	30
6_{16}	50
7_{16}	60000/1001
8_{16}	60
F_{16}	No information
all other values	Reserved for future specification

The *bit_rate_type* field specifies the bit rate type of the *video_signal_bit_rate* field. The field can have one of the following table.

Table B.4 – Bit rate type values

bit_rate_type	meaning
00_2	no type is specified.
01_2	average bit rate
10_2	maximum bit rate
11_2	reserved for future extension

The *video_signal_bit_rate* field specifies the bit rate of the video signal recorded by the disc subunit. This field is encoded in BCD format and has a unit of 10^3 bit/sec. The value of FF FF FF_{16} means no information.

B.1.2 Unknown Signal Mode

The *video_signal_mode_specific_information* field for unknown signal mode is defined below.

Table B.5 – Mode specific information for unknown signal mode

Address Offset	length	msb							lsb
00 00_{16}	4	reserved							
00 01_{16}									
00 02_{16}									
00 03_{16}									

Annex C: New media type, list type, object type (normative)

C.1 Supported_media_type

The *supported_media_type* field identifies the type of media. The upper byte indicates the family, while the lower byte specifies more detailed specifications or about its format. The following table illustrates the defined values for the supported media type:

Table C.1 – supported_media_type field

Supported_media_type (MSB)	Value	Supported_media_type (LSB)	Value
Generic Recordable Video Disc	$0D_{16}$	Generic Recordable Video Disc	01_{16}

C.2 Media_type

The *media_type* field indicates the format of the information on this disc. The upper byte indicates the media family, while the lower byte specifies more detailed information. It is encoded as follows:

Table C.2 – media_type field

Media_type (MSB)	Value	Media_type (LSB)	Value
Generic Recordable Video Disc	$0D_{16}$	Generic Recordable Video Disc	01_{16}
		Other	$0E_{16}$
		Reserved for future extension	All other value

The *other* value for *media_type* (the LSB) means that the disc is something other than a recognized AV format. When this value is reported, it indicates that the subunit is able to recognize existence of the disc, but it contains information which is not recognizable to the subunit.

Annex D: Optional commands

The following table contains commands that a Player/Recorder Profile may implement. These commands are a subset of category A commands in [R7]. The command and response structure are as described in [R7].

Table D.1 – Player/Recorder Profile optional commands (Disc Subunit commands)

Command	Opcode	Defined ctypes			Comments
		C	S	N	
COMBINE	41_{16}	X	–	–	Combine two tracks into a single track
CREATE AV TRACK	$D5_{16}$	X	–	–	This command is used to create a track of a given size.
DISC STATUS	$D0_{16}$	–	–	X	Request notification when the status of the subunit changes
DIVIDE	42_{16}	X	–	–	Separate a specified track into two tracks
INCREMENT OBJECT POSITION NUMBER	51_{16}	X	–	–	Divide a track while recording
MONITOR	$C6_{16}$	X	–	–	Listen to what is being recorded
RECORD (loop)	$C2_{16}$	X	–	–	Records on the selected AV track in loop mode.
REHEARSAL	$C7_{16}$	X	–	–	Playback a few positions continuously
SEARCH (Search type is Relative and search indicator is relative)	50_{16}	X	–	–	Performs a search to the specified position on the selected AV track
UNDO	44_{16}	X	–	–	Undo the most recent editing operation(s)

D.1 COMBINE

In a Player/Recorder Profile, the COMBINE command is optional. The operation of this command is as specified in [R7].

D.2 CREATE AV TRACK

In a Player/Recorder Profile, the CREATE AV TRACK command is optional. When the CREATE AV TRACK is supported it will create both a new AV track and a new AV Content Object descriptor. Right after an AV track is created by the CREATE AV TRACK command, the AV track is not recorded. When the controller issues a command to set a play back position in the not recorded part in the AV track, the subunit may return a REJECTED response. When the controller issues a command to set a recording position in the not recorded part except the beginning position of the not recorded part, the subunit also may return a REJECTED response.

D.3 DISC STATUS

In a Player/Recorder Profile, the DISC STATUS command is optional. When the DISC STATUS is supported, at least *subfunction* of 00_{16} ("Full Status") is required. This will cause a notification to occur whenever the status descriptor changes.

D.4 DIVIDE

In a Player/Recorder Profile, the DIVIDE command is optional. The operation of this command is as specified in [R7].

D.5 INCREMENT OBJECT POSITION NUMBER

In a Player/Recorder Profile, the INCREMENT OBJECT POSITION NUMBER command is optional. The operation of this command is as specified in [R7]. In a Player/Recorder Profile, this command is used to create new object when recording is going on.

D.6 MONITOR

In a Player/Recorder Profile, the MONITOR command is optional. When MONITOR is supported, at least *subfunction* 60_{16} ("Monitoring Off"), 70_{16} ("Monitoring On") are required.

D.7 RECORD(LOOP)

In a Player/Recorder Profile, the loop *subfunction* is optionally supported.

Subfunction1 : 75_{16} ("Record at Normal Speed") and $7D_{16}$ ("Forward Pause").

Subfunction2 : 06_{16} ("Record Loop").

D.8 REHEARSAL

In a Player/Recorder Profile, the REHEARSAL command is optional. The operation of this command is as specified in [R7].

D.9 SEARCH – RELATIVE

In a Player/Recorder Profile, the SEARCH command with subfunction 10_{16} (Search Type is Relative) is optional. The search indicator value is 00_{16} i.e. search by *Relative_HMSF_count*.

NOTE — The search position may not be frame accurate and only is an estimate of the desired position.

D.10 UNDO

In a Player/Recorder Profile, the UNDO command is optional. When UNDO is supported, the *undo_status* and *difference* fields take on appropriate values based on implementation. If supported, UNDO will be implemented for at least the ERASE and WRITE INFO BLOCK commands.

Annex E: Descriptor examples (Informative)

This section describes examples of descriptors of the target. Note that the target does not always have the structures written in this section.

E.1 Subunit Identifier Descriptor examples

E.1.1 disc_subunit_dependent_information examples

E.1.1.1 Example 1

When the target only performs actions based on the Generic Recordable Video Disc media type (GRVD) specification, the disc_subunit_dependent_information is as follows:

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	disc_subunit_dependent_info_fields_length = XX XX_{16}
$00\ 01_{16}$			
$00\ 02_{16}$	1	R	attributes = 0000 $000X_2$
$00\ 03_{16}$	1	R	disc_subunit_version = 12_{16}
$00\ 04_{16}$	1	R	number_of_supported_media_types = 01_{16}
$00\ 05_{16}$			supported_media_type_specification[0] (GRVD)
:			
:			
:			

Figure E.1 – example of disc_subunit_dependent_information fields 1

E.1.1.2 Example 2

When the target is able to perform actions based on both media type specifications for 'Disc A' and the Generic Recordable Video Disc (GRVD), the disc_subunit_dependent_information is as follows:

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	disc_subunit_dependent_info_fields_length = XX XX_{16}
$00\ 01_{16}$			
$00\ 02_{16}$	1	R	attributes = 0000 $000X_2$
$00\ 03_{16}$	1	R	disc_subunit_version = 12_{16}
$00\ 04_{16}$	1	R	number_of_supported_media_types = 02_{16}
$00\ 05_{16}$			supported_media_type_specification[0] (GRVD)
:			
:			
:			
:			supported_media_type_specification[1] (Disc 'A')
:			
:			
:			

Figure E.2 – example of disc_subunit_dependent_information fields 2

E.1.2 supported_media_type_specification fields examples

This is the example of the supported_media_type_specification fields for the Generic Recordable Video Disc.

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	Supported_media_type = $0D\ 01_{16}$ (Generic Recordable Video Disc)
$00\ 01_{16}$			
$00\ 02_{16}$	1	R	Implementation_profile_ID = 10_{16}
$00\ 03_{16}$	1	R	media_type_attributes = $0000\ 000R_2$
$00\ 04_{16}$	2	R	Type_dependent_length = $00\ 06_{16}$
$00\ 05_{16}$			
$00\ 06_{16}$	1	R	generic_recordable_video_disc_version = 00_{16}
$00\ 07_{16}$	4	R	Maximum_number_of_recordable_objects = $XX\ XX\ XX\ XX_{16}$
$00\ 08_{16}$			
$00\ 09_{16}$			
$00\ 0A_{16}$			
$00\ 0B_{16}$	1	R	Number_of_extended_implementation_profiles = 0

Figure E.3 – example of supported_media_type_specification fields

In this example, only one *Implementatiion_profile_ID*(10_{16}) is defined, so the extended implementation profiles are not used.
If a target does not have a recording function, the *can_record* flag (R bit) is set to 0_2.
If a target has a recording function, the *can_record* flag (R bit) is set to 1_2.

E.2 contents descriptor examples

This section shows examples of contents descriptor.

E.2.1 root contents list

Following figure is an example of root contents list;

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$ $00\ 01_{16}$	2	R	descriptor length = XX XX_{16}
$00\ 02_{16}$	1	R	list_type = 80_{16}
$00\ 03_{16}$	1	R	Attributes = $0001\ 1000_2$
$00\ 04_{16}$ $00\ 05_{16}$	2	R	size_of_list_specific_information = XX XX_{16}
$00\ 06_{16}$ $00\ 07_{16}$	2	R	non_info_block_fields_length = $00\ 04_{16}$
$00\ 08_{16}$	1	R	disc_subunit_list_attributes = $0000\ 00XX_2$
$00\ 09_{16}$ $00\ 0A_{16}$	2	R	Media_type = XX XX_{16}
$00\ 0B_{16}$	1	R	disc_recordable_information = $XX01\ 0000_2$
:	***Time Stamp Info Block for descriptor modification***		
:	***Default playlist info block***		
:	***Disc_capacity_info_block***		
:	***Name_info_block***		
: :	2	R	Number_of_entries = n
:	***Object entry entry descriptor[0]*** : ***Object entry entry descriptor[n-1]***		

Figure E.4 – root contents list

NOTE —When RECORD NEW operation is started, new object entry is added.

There are 2 ways to describe the media_type field.

1) When the target only performs actions based on the Generic Recordable Video Disc media type (GRVD) specification, the media_type field is set to $0D\ 01_{16}$ (Generic Recordable Video Disc).

2) When the target is able to perform actions based on both media type specifications for 'Disc A' and the Generic Recordable Video Disc (GRVD), the media_type field is set to the value which indicates 'Disc A'.

Address	Length, bytes	External Read/Write	Contents
00 00_{16} 00 01_{16}	2	R	Compound_length = 00 $0B_{16}$
00 02_{16} 00 03_{16}	2	R	Info_block_type = 00 07_{16} (Time Stamp Info Block)
00 04_{16} 00 05_{16}	2	R	Primary_fields_length = 00 07_{16}
00 06_{16} : 00 $0C_{16}$	7	R	*a *b Time_stamp_data (54bit unsigned integer counter type)

*a : *valid* field
*b : *stamp_type* field = 1_2 (binary counter)

Figure E.5 – Time Stamp Info Block for descriptor modification

In the case of the root contents list, *valid* field = 1_2 (“valid”).

Address	Length, bytes	External Read/Write	Contents
00 00_{16} 00 01_{16}	2	R	Compound_ length = 00 06_{16}
00 02_{16} 00 03_{16}	2	R	Info_Block_type = 80 $0B_{16}$
00 04_{16} 00 05_{16}	2	R	Primary_fields_length = 00 02_{16}
00 06_{16} 00 07_{16}	2	R	Default_playlist_ID = 30 00_{16} (list ID of root contents list.)

Figure E.6 – Default playlist info block

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	Compound_ length = $00\ 1A_{16}$
$00\ 01_{16}$			
$00\ 02_{16}$	2	R	Info_Block_type = $80\ 02_{16}$
$00\ 03_{16}$			
$00\ 04_{16}$	2	R	Primary_fields_length = $00\ 16_{16}$
$00\ 05_{16}$			
$00\ 06_{16}$	1	R	Capacity_format_indicator = 01h (“bytes“)
$00\ 07_{16}$	2	R	disc_total_playback_capacity_length = a
$00\ 08_{16}$			
$00\ 09_{16}$	a	R	disc_total_playback_capacity = XX XX XX XX XX_{16}
:			
:			
:	2	R	disc_maximum_recording_capacity_length = a
:			
:	a	R	disc_maximum_recording_capacity = XX XX XX XX XX_{16}
:			
:			
:	2	R	disc_remaining_recording_capacity_length = a
:			
:	a	R	disc_remaining_recording_capacity = XX XX XX XX XX_{16}
:			
:			

Figure E.7 – Disc_capacity_info_block

Address	Length, bytes	External Read/Write	Contents
00 00_{16} 00 01_{16}	2	R	Compound_length = XX XX_{16}
00 02_{16} 00 03_{16}	2	R	info_block_type = 00 $0B_{16}$ (name Info Block)
00 04_{16} 00 05_{16}	2	R	Primary_fields_length = XX XX_{16}
00 06_{16}	1	R	Name_data_reference_type = 00_{16}
00 07_{16}	1	R	Name_data_attributes = 0000 0011_2
00 08_{16} 00 09_{16}	2	RWI	Maximum_number_of_characters = XX XX_{16}
	Character code info block		
00 $0A_{16}$ 00 $0B_{16}$	2	R	Compound_length = 00 06_{16}
00 $0C_{16}$ 00 $0D_{16}$	2	R	Info_block_type = 00 08_{16} (character code Info Block)
00 $0E_{16}$ 00 $0F_{16}$	2	R	Primary_fields_length = 00 02_{16}
00 10_{16}	1	RWI	Character code type = XX_{16}
00 11_{16}	1	RWI	Character code type specific = XX_{16}
	Language_code_info_block		
00 12_{16} 00 13_{16}	2	R	Compound_length = 00 07_{16}
00 14_{16} 00 15_{16}	2	R	Info_block_type = 00 09_{16} (language code Info Block)
00 16_{16} 00 17_{16}	2	R	Primary_fields_length = 00 03_{16}
00 18_{16}	1	RWI	Language code type = XX_{16}
00 19_{16} 00 $1A_{16}$	2	RWI	Language code type specific = XX XX_{16}
	Raw_text_info_block		
00 $1B_{16}$ 00 $1C_{16}$	2	R	Compound_length = 00 04_{16} + a
00 $1D_{16}$ 00 $1E_{16}$	2	R	Info_block_type = 00 $0A_{16}$ (raw text Info Block)
00 $1F_{16}$ 00 20_{16}	2	R	Primary_fields_length = a
00 21_{16} :	a	RWI	Raw_text_data

Figure E.8 – Name info block

The *character_code_info_block* and *Language_code_info_block* :

Case 1:
If a target supports only one specific character code and language code, it sets the *character_code_info_block* and the *language_code_info_block* as appropriate values and the controller cannot modify these fields.

For example, the Japanese model for BS-digital has the following values:

Character code type field: 08_{16} ("Japanese EUC")
Character code type specific field : 01_{16} ("Japanese EUC for BS digital")
Language code type field = 01_{16} ("ISO 639")
Language code type specific field = $6A61_{16}$ ("ja")

Case 2:
If a target has preferred a character code and a language code, it sets the *character_code_info_block* and the *language_code_info_block* as appropriate values and the controller can modify these fields.

Case 3:
If a target cannot set the appropriate values, it sets the *Character code type* field, the *Character code type specific* field, the language_code_type field and the language_code_type specific field to FF_{16} as initial values and the controller can modify these fields.

The initial value of the *primary_fields_length* field in the raw_text_info_block is 0.

E.2.2 Video content object entry

Address	Length, bytes	External Read/Write	Contents
00 00_{16} 00 01_{16}	2	R	Descriptor length = XX XX_{16}
00 02_{16}	1	R	Entry_type = 83_{16} (Video Content object)
00 03_{16}	1	R	Attributes = 0000 1000_2
00 04_{16} 00 05_{16}	2	R	Object ID
00 06_{16} 00 07_{16}	2	R	Size_of_entry_specific_information = XX XX_{16}
Entry_specific_information			
00 08_{16} 00 09_{16}	2	R	Non_info_block_fields_length = 00 01_{16}
00 $0A_{16}$	1	R	Disc_subunit_object_attributes = 0000 $00XX_2$
:	***Video signal mode info block***		
:	***Video Stream Format Subtype Info Block***		
:	***Size Indicator Info Block***		
:	***Time Stamp Info Block***		
:	***Program Attribute Info Block***		
:	***Name info block***		
:	***(Other optional info blocks)***		

Figure E.9 – Video contents object entry example

The following info blocks are mandatory for the Player/Recorder Profile and described in this order.

1) Video Signal Mode Info Block

2) Video Stream Format Subtype Info Block

3) Size Indicator Info Block

4) Time Stamp Info Block

5) Program Attribute Info Block

6) Name Info Block

Other info blocks are optional, and described in the *optional info blocks* area.

When “RECORD NEW” operation is started, a new video object entry is added at the end of root contents list with all mandatory info blocks (and optional info blocks, if they exist.).

NOTE —The player/recorder profile does not support methods to add new info blocks, thus when new object entry is created, all necessary info blocks are provided within new entry.

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	Compound_length = $00\ 09_{16}$
$00\ 01_{16}$			
$00\ 02_{16}$	2	R	Info_block_type = $88\ 12_{16}$ (Video Signal Mode Info Block)
$00\ 03_{16}$			
$00\ 04_{16}$	2	R	Primary_fields_length = $00\ 05_{16}$
$00\ 05_{16}$			
$00\ 06_{16}$	1	R	Video_signal_mode = XX_{16}
$00\ 07_{16}$	1	R	Frame_rate_code & reserved bits = $XXXX\ XXXX_2$
$00\ 08_{16}$	3	R	Video_signal_bit_rate = $XX\ XX\ XX_{16}$
$00\ 09_{16}$			
$00\ 0A_{16}$			

Figure E.10 – Video signal mode info block

All fields in the *video_signal_mode_info_block* are maintained by the target. When recording, the plug configuration of destination plug is reflected to this info block.

Address	Length, bytes	External Read/Write	Contents
00 00_{16} 00 01_{16}	2	R	Compound_length = 00 22_{16}
00 02_{16} 00 03_{16}	2	R	Info_block_type = 88 13_{16} (Video Stream Format Subtype Info Block)
00 04_{16} 00 05_{16}	2	R	Primary_fields_length = 00 $1E_{16}$
00 06_{16}	1	RWI	Validity_flags = XXXX $XX00_2$
00 07_{16} : 00 $0A_{16}$	4	RWI	Format_identifier = XX XX XX XX_{16}
00 $0B_{16}$: 00 13_{16}	9	RWI	Network_information = XX XX XX XX XX XX XX XX XX_{16}
00 14_{16} : 00 23_{16}	16	RWI	Stream_format_name = XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX_{16}

Figure E.11 – Video Stream Format Subtype Info Block

The information in the *video_stream_format_subtype_info_block* is usually revised by a controller. When a new video content object is recorded, it is strongly recommended for the controller to set an effective value in this info block.

Some disc media may not deal with the information for this info block. In this case, if the command to write from the controller is ACCEPTED, the target may not update the values of some fields in this info block.

When the target cannot set appropriate values for this info block, it can use the following initial values:

Validity_flags **field**

The target sets the initial value to 00_{16}. (all flags are set to 0)

Format Identifier **field**

The target sets the initial value to all FF_{16}.

Network information **field**

The target sets the initial value to all FF_{16}.

Stream_format_name **field**

The target sets the initial value to all FF_{16}.

Address	Length, bytes	External Read/Write	Contents
00 00_{16}	2	R	Compound_length = 00 $0A_6$
00 01_{16}			
00 02_{16}	2	R	Info_block_type = 00 01_{16} (size indicator Info Block)
00 03_{16}			
00 04_{16}	2	R	Primary_fields_length = 00 06_{16}
00 05_{16}			
00 06_{16}	1	R	size_indicator_type = 00_{16} (“H_M_S_F”)
00 07_{16}	2	R	Hour = XX XX_{16}
00 08_{16}			
00 09_{16}	1	R	Minutes = XX_{16}
00 $0A_{16}$	1	R	Seconds = XX_{16}
00 $0B_{16}$	1	R	Frames = XX_{16}

Figure E.12 – Size Indicator Info Block (H_M_S_F)

The value of *size_indicator_info_block* (H_M_S_F) is set by the target. Accuracy of size information is subunit dependent.

“no information” values for *Hour, Minutes, Seconds and Frames* fields are all FF_{16}

Address	Length, bytes	External Read/Write	Contents
00 00_{16}	2	R	Compound_length = 00 $0B_{16}$
00 01_{16}			
00 02_{16}	2	R	Info_block_type = 00 04_{16} (Time Stamp Info Block (content creation time))
00 03_{16}			
00 04_{16}	2	R	Primary_fields_length = 00 07_{16}
00 05_{16}			
00 06_{16}	2	RWI	Year = XX XX_{16}
00 07_{16}			
00 08_{16}	1	RWI	Month = XX_{16}
00 09_{16}	1	RWI	Day = XX_{16}
00 $0A_{16}$	1	RWI	Hours = XX_{16}
00 $0B_{16}$	1	RWI	Minutes = XX_{16}
00 $0C_{16}$	1	RWI	Seconds = XX_{16}

Figure E.13 – Time Stamp Info Block (Content_creation_date_and_time)

The value of the *time_stamp_info_block(content creation time)* is set by a controller.

If a target can maintain this info block, initial values of this info block is appropriate date and time. Otherwise the target sets the initial values to all FF_{16}.

Address	Length, bytes	External Read/Write	Contents
00 00_{16}	2	R	Compound_length = XX XX_{16}
00 01_{16}			
00 02_{16}	2	R	Info_block_type = 88 14_{16} (Program Attributes Info Block)
00 03_{16}			
00 04_{16}	2	R	Primary_fields_length = XX XX_{16}
00 05_{16}			
	Program starting date & time		
00 06_{16}	2	RWI	Year = XX XX_{16}
00 07_{16}			
00 08_{16}	1	RWI	Month = XX_{16}
00 09_{16}	1	RWI	Day = XX_{16}
00 $0A_{16}$	1	RWI	Hour = XX_{16}
00 $0B_{16}$	1	RWI	Minute = XX_{16}
00 $0C_{16}$	1	RWI	Second = XX_{16}
	Program duration		
00 $0D_{16}$	2	RWI	Hours = XX XX_{16}
00 $0E_{16}$			
00 $0F_{16}$	1	RWI	Minutes= XX_{16}
00 10_{16}	1	RWI	Seconds= XX_{16}
00 11_{16}	1	RWI	***Parental rating*** = XXXX $XXXX_2$
00 12_{16}	1	RWI	***Program modes*** = XXXX $XXXX_2$
	Service name		
00 13_{16}	1	RWI	Character code type = XX_{16}
00 14_{16}	1	RWI	Character code type specific = XX_{16}
00 15_{16}	1	RWI	Maximum service name length = XX_{16}
00 16_{16}	1	RWI	Service name length = y
	y	RWI	(Service name)

Figure E.14 – Program Attribute Info Block

Some disc media may not deal with the information for this info block. In this case, if the command to write from the controller is ACCEPTED, the target may not update the values of some fields in this info block.

When the target cannot set appropriate values for this info block, it can use the following initial values:

Program starting date & time **field:**

The target sets the initial value to all FF_{16}.

Program duration **field**

The target sets the initial value to all FF_{16}.

Parental rating **field**

The target sets the initial value of rating field to 1_2. (no restriction by parental rating)

***Program mode* field & *HD/SD* field**

The target sets the initial value to11_2. ("No information")

***REPLAYED* field**

The target sets the initial value to 11_2 ("no information").

***Reserved* Field**

The *Reserved* field is for future extension, and set to 0000_2.

***Service name* field**

Case 1:
If a target supports only one specific character code, it sets *Character code type* field and *Character code type specific* field as appropriate values and the controller cannot modify these fields.
For example, the Japanese model for BS-digital has the initial values as follows:

Character code type field: 08_{16} ("Japanese EUC")
Character code type specific field: 01_{16} ("Japanese EUC for BS digital")

Case 2:
If a target has preferred character code, it sets the *Character code type* field and the *Character code type specific* field as appropriate values and the controller can modify these fields.

Case 3:
If a target cannot set the appropriate values, it sets the *Character code type* field and the *Character code type specific* field to FF_{16} as initial values and the controller can modify these fields.

The *Maximum service name length* field is maintained by the target. A controller cannot modify this field. Write commands on this field are accepted, but not affected.

The target sets the initial value of *Service name length* field to 00_{16}.

Address	Length, bytes	External Read/Write	Contents
00 00_{16} 00 01_{16}	2	R	Compound_length = 00 XX_{16}
00 02_{16} 00 03_{16}	2	R	info_block_type = 00 $0B_{16}$ (name Info Block)
00 04_{16} 00 05_{16}	2	R	Primary_fields_length = 00 XX_{16}
00 06_{16}	1	R	Name_data_reference_type = 00_{16}
00 07_{16}	1	R	Name_data_attributes = 0000 0011_2
00 08_{16} 00 09_{16}	2	RWI	Maximum_number_of_characters = XX XX_{16}
	Character code info block		
00 $0A_{16}$ 00 $0B_{16}$	2	R	Compound_length = 00 06_{16}
00 $0C_{16}$ 00 $0D_{16}$	2	R	Info_block_type = 00 08_{16} (character code Info Block)
00 $0E_{16}$ 00 $0F_{16}$	2	R	Primary_fields_length = 00 02_{16}
00 10_{16}	1	RWI	Character code type = XX_{16}
00 11_{16}	1	RWI	Character code type specific = XX_{16}
	Language_code_info_block		
00 12_{16} 00 13_{16}	2	R	Compound_length = 00 07_{16}
00 14_{16} 00 15_{16}	2	R	Info_block_type = 00 09_{16} (language code Info Block)
00 16_{16} 00 17_{16}	2	R	Primary_fields_length = 00 03_{16}
00 18_{16}	1	RWI	Language code type = XX_{16}
00 19_{16} 00 $1A_{16}$	2	RWI	Language code type specific = XX XX_{16}
	Raw_text_info_block		
00 $1B_{16}$ 00 $1C_{16}$	2	R	Compound_length = 00 04_{16} + a
00 $1D_{16}$ 00 $1E_{16}$	2	R	Info_block_type = 00 $0A_{16}$ (raw text Info Block)
00 $1F_{16}$ 00 20_{16}	2	R	Primary_fields_length = a
00 21_{16} :	a	RWI	(Raw_text_data)

Figure E.15 – Name info block

The *character_code_info_block* and *language_code_info_block* : .

Case 1:
If a target supports only one specific character code and language code, it sets the *character_code_info_block* and the *language_code_info_block* as appropriate values and the controller cannot modify these fields.

For example, the Japanese model for BS-digital has the initial values as follows:

Character code type field: 08_{16} ("Japanese EUC")
Character code type specific field: 01_{16} ("Japanese EUC for BS digital")
Language code type field = 01_{16} (ISO 639)
Language code type specific field = $6A61_{16}$ ("ja")

Case 2:
If a target has preferred a character code and a language code, it sets the *character_code_info_block* and the *language_code_info_block* as appropriate values and the controller can modify these fields.

Case 3:
If a target cannot set the appropriate values, it sets the *Character code type* field, the *Character code type specific* field, the language_code_type field and the language_code_type specific field to FF_{16} as initial values and the controller can modify these fields.

The initial value of the *primary_fields_length* field in the *raw_text_info_block* is 0.

E.3 Status descriptor

This section shows examples of the Status Descriptor structure.

E.3.1 General_disc_subunit_status_info_block

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	compound_length = $00\ 0D_{16}$
$00\ 01_{16}$			
$00\ 02_{16}$	2	R	info_block_type = $88\ 00_{16}$
$00\ 03_{16}$			
$00\ 04_{16}$	2	R	primary_fields_length = $00\ 09_{16}$
$00\ 05_{16}$			
$00\ 06_{16}$			***Media_and_edit_status_info_block***
:			

Figure E.16 – General_disc_subunit_status_info_block

Address Offset	Msb							lsb
00_{16}	Compound_length =$00\ 07_{16}$							
01_{16}								
02_{16}	info_block_type = $88\ 04_{16}$ (media_and_edit_status infoblock)							
03_{16}								
04_{16}	primary_fields_length = $00\ 03_{16}$							
05_{16}								
06_{16}	Disc_in_drive = XX_2		error_condition = XX_2			Reserved = 0000_2		
07_{16}	undo_status = XX_{16}							
08_{16}	Difference = 0_2	Auto_update = 1_2	Reserved = $00\ 0000_2$					

Figure E.17 – Media_and_edit_status_info_block

The *disc_in_drive* field indicates media existence. The target maintains this field corresponding to existence of disc medium.

NOTE —In the above example, this subunit does not support temporary contents list structure, so that the *difference* field is set as 0_2("no difference") and the *auto_update* field is set as 1_2 ("not support temporary contents list and update always occur"). If a subunit support temporary contents list structure, the *difference* field and the *auto_update* field indicate condition of temporary contents list.

E.3.2 Destination Plug Status Area Info Block

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	Descriptor length field = XX XX_{16}
$00\ 01_{16}$			
$00\ 02_{16}$	2	R	Info_block_type = 88 01_{16}
$00\ 03_{16}$			
$00\ 04_{16}$	2	R	primary_fields_length
$00\ 05_{16}$			
$00\ 06_{16}$	1	R	Number_of_destination_plugs = n
$00\ 07_{16}$			***Plug_status_info_block[0]***
:			***:***
			plug_status_info_block[n-1]

Figure E.18 – Destination Plug Status Area Info Block

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	Descriptor length field = XX XX_{16}
$00\ 01_{16}$			
$00\ 02_{16}$	2	R	Info_block_type = 88 05_{16}
$00\ 03_{16}$			
$00\ 04_{16}$	2	R	Primary_fields_length = XX XX_{16}
$00\ 05_{16}$			
$00\ 06_{16}$	1	R	Plug_number
Nested structures			
$00\ 07_{16}$	***Operating Mode Info Block (88 06_{16})***		
:			
:	***Plug Configuration Info Block (88 07_{16})***		
:			
:	***Position Info Block (00 03_{16})***		
:			

Figure E.19 – destination plug Plug_status_info_block

Address	Length, bytes	External Read/Write	Contents
00 00_{16}	2	R	descriptor length field = 00 07_{16}
00 01_{16}			
00 02_{16}	2	R	info_block_type = 88 06_{16}
00 03_{16}			
00 04_{16}	2	R	primary_fields_length = 00 03_{16}
00 05_{16}			
00 06_{16}	1	R	operating_mode
00 07_{16}	2	R	operating_mode_specific_information
00 08_{16}			(FF_{16} pad bytes if necessary)

Figure E.20 – Operating Mode Info Block

Size of the *operating_mode_specific_information* field is 2 bytes and the target may pad FF_{16} if necessary.

NOTE — A value of FF_{16}("SUSPENDED") in the *operating_mode* field indicates that the plug is not available.

Address	Length, bytes	External Read/Write	Contents
00 00_{16}	2	R	descriptor length field = XX XX_{16}
00 01_{16}			
00 02_{16}	2	R	info_block_type = 88 07_{16}
00 03_{16}			
00 04_{16}	2	R	primary_fields_length = XX XX_{16}
00 05_{16}			
00 06_{16}	1	R	AV_object_type = XX_{16}
:			***Object_and_plug_type_specific_information***
:			

Figure E.21 – Plug Configuration Info Block

When a destination plug is configured for MPEG2-TS object, the *AV_object_type* field is set to 83_{16}(Video Content object) and the *object_and_plug_type_specific_information* is fixed length structure (6 bytes) as follows:

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	1	R	*a | Reserved = $000\ 0000_2$
$00\ 01_{16}$	1	R	Video_signal_mode = 01_{16} (MPEG2-TS)
$00\ 02_{16}$	1	R	Frame_rate | Reserved
$00\ 03_{16}$	3	R	video_signal_bit_rate (x 10^3 bit/sec)
$00\ 04_{16}$			
$00\ 05_{16}$			

*a Increment_position_number bit = 0_2

Figure E.22 – Object_and_plug_type_specific_information for MPEG2-TS object

In the above example, increment position number function is not supported, so that the *increment_position_number* field is set to 0_2.

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	Descriptor length field = $00\ 14_{16}$
$00\ 01_{16}$			
$00\ 02_{16}$	2	R	Info_block_type = $00\ 03_{16}$
$00\ 03_{16}$			
$00\ 04_{16}$	2	R	Primary_fields_length = $00\ 10_{16}$
$00\ 05_{16}$			
			List descriptor reference
$00\ 06_{16}$	1	R	Descriptor_type = 10_{16}
$00\ 07_{16}$	2	R	list ID = $30\ 00_{16}$ (root contents list)
$00\ 08_{16}$			
			Nested position indicator info block
$00\ 09_{16}$	2	R	descriptor length field = $00\ 0B_{16}$
$00\ 0A_{16}$			
$00\ 0B_{16}$	2	R	info_block_type = $00\ 02_{16}$
$00\ 0C_{16}$			
$00\ 0D_{16}$	2	R	primary_fields_length = $00\ 07_{16}$
$00\ 0E_{16}$			
$00\ 0F_{16}$	1	R	Indicator_type = 00_{16} (relative HMSF)
$00\ 10_{16}$	2	R	object_position_number
$00\ 11_{16}$			
$00\ 12_{16}$	1	R	+/- | Hours
$00\ 13_{16}$	1	R	Minutes
$00\ 14_{16}$	1	R	Seconds
$00\ 15_{16}$	1	R	Frames

Figure E.23 – Position Info Block for destination plug

In above example, the root contents list is assigned to destination plug.

The recording position in an object is equal to or greater than 00:00:00:00, so that *"+/-"* field is always set to 0_2.

When any list is not associated to destination plug,

list ID field in List descriptor reference is set to FF FF_{16}.

Nested position indicator info block

object_position_number field is set to FF FF_{16},

Hours, Minutes, *Seconds*, *Frames* fields are set to 00_{16}.

E.3.3 Source Plug Status Area Info Block

Address	Length, bytes	External Read/Write	Contents
00 00_{16} 00 01_{16}	2	R	Descriptor length field = XX XX_{16}
00 02_{16} 00 03_{16}	2	R	Info_block_type = 88 02_{16}
00 04_{16} 00 05_{16}	2	R	primary_fields_length
00 06_{16}	1	R	Number_of_destination_plugs = n
00 07_{16} :			***Plug_status_info_block[0]*** ***:*** ***plug_status_info_block[n-1]***

Figure E.24 – Source Plug Status Area Info Block

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	Descriptor length field = XX XX_{16}
$00\ 01_{16}$			
$00\ 02_{16}$	2	R	Info_block_type = $88\ 05_{16}$
$00\ 03_{16}$			
$00\ 04_{16}$	2	R	Primary_fields_length = XX XX_{16}
$00\ 05_{16}$			
$00\ 06_{16}$	1	R	Plug_number
Nested structures			
$00\ 07_{16}$	***Operating Mode Info Block ($88\ 06_{16}$)***		
:			
:	***Plug Configuration Info Block ($88\ 07_{16}$)***		
:			
:	***Position Info Block ($00\ 03_{16}$)***		
:			

Figure E.25 – source plug Plug_status_info_block

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$	2	R	descriptor length field = $00\ 07_{16}$
$00\ 01_{16}$			
$00\ 02_{16}$	2	R	info_block_type = $88\ 06_{16}$
$00\ 03_{16}$			
$00\ 04_{16}$	2	R	primary_fields_length = $00\ 03_{16}$
$00\ 05_{16}$			
$00\ 06_{16}$	1	R	operating_mode
$00\ 07_{16}$	2	R	operating_mode_specific_information
$00\ 08_{16}$			(FF_{16} pad bytes if necessary)

Figure E.26 – Operating Mode Info Block

The size of *operating_mode_specific_information* field is 2 bytes and the target player/recorder may pad FF_{16} if necessary.

NOTE — A value of FF_{16}("SUSPENDED") in the *operating_mode* field indicates that the plug is not available.

Address	Length, bytes	External Read/Write	Contents
$00\ 00_{16}$ $00\ 01_{16}$	2	R	descriptor length field = XX XX_{16}
$00\ 02_{16}$ $00\ 03_{16}$	2	R	info_block_type = $88\ 07_{16}$
$00\ 04_{16}$ $00\ 05_{16}$	2	R	primary_fields_length = XX XX_{16}
$00\ 06_{16}$	1	R	AV_object_type = XX_{16}
$00\ 07_{16}$: :			***Object_and_plug_type_specific_information***

Figure E.27 – Plug Configuration Info Block

The object_and_plug_type_specific_information for video object is as follows:

Address	Length, bytes	External Read/Write	Contents	
$00\ 01_{16}$	1	R	*b	Reserved = $000\ 0000_2$

*b *audio_mute* bit = 0_2 (not effect)

Figure E.28 – Object_and_plug_type_specific_information for video object

In above example, the audio mute function is not supported, so that the *audio_mute* field is set to 0_2.

Address	Length, bytes	External Read/Write	Contents
00 00$_{16}$	2	R	Descriptor length field = 00 17$_{16}$
00 01$_{16}$			
00 02$_{16}$	2	R	Info_block_type = 00 03$_{16}$
00 03$_{16}$			
00 04$_{16}$	2	R	Primary_fields_length = 00 13$_{16}$
00 05$_{16}$			
List descriptor reference			
00 06$_{16}$	1	R	Descriptor_type = 21$_{16}$
00 07$_{16}$	2	R	List ID of contents list (root contents list or root performance list)
00 08$_{16}$			
00 09$_{16}$	1	R	List_type
00 0A$_{16}$	2	R	Object_ID of a content object
00 0B$_{16}$			
	Nested position indicator info block		
00 0C$_{16}$	2	R	Descriptor length field = 00 0B$_{16}$
00 0D$_{16}$			
00 0E$_{16}$	2	R	Info_block_type = 00 02$_{16}$
00 0F$_{16}$			
00 10$_{16}$	2	R	Primary_fields_length = 00 07$_{16}$
00 11$_{16}$			
00 12$_{16}$	1	R	Indicator_type = 00$_{16}$ (relative HMSF)
00 13$_{16}$	2	R	Object_position_number = 00 00$_{16}$
00 14$_{16}$			
00 15$_{16}$	1	R	+/- | Hours
00 16$_{16}$	1	R	Minutes
00 17$_{16}$	1	R	Seconds
00 18$_{16}$	1	R	Frames

Figure E.29 – Position Info Block for video contents in source plug status

In above example, a video content object is assigned to source plug.

The recording position in a video content object is equal to or greater than 00:00:00:00, so that "+/-" field is always set to 0_2.

When any list or object is not associated to source plug,

list ID field in List descriptor reference is set to FF FF$_{16}$.

Nested position indicator info block:

object_position_number field is set to FF FF$_{16}$,

Hours, Minutes, *Seconds*, *Frames* fields are set to 00$_{16}$.

Annex F: Generic Recordable Video Disc Use Examples (Informative)

This section describes examples of control sequences from the controller. Note that the target does not always perform actions which are written in this section.

F.1 Descriptor Access Examples

F.1.1 Method to read the Subunit Identifier Descriptor or the Status Descriptor

The following method is applied for reading the Subunit Identifier Descriptor and Status Descriptor.

Reading procedure is as follows:

1) The controller issues the OPEN DESCRIPTOR command with the following parameters:

   descriptor identifier : indicate the objective descriptor.

   Subunit Identifier Descriptor : *descriptor_type* = 00_{16}

   Status Descriptor : *descriptor_type* = 80_{16}, *reference_method* = 00_{16} (full descriptor reference)

   subfunction : READ OPEN or WRITE OPEN

2) The controller issues the READ DESCRIPTOR command with the following parameters:

   descriptor identifier : indicate the objective descriptor.

   data_length = size that controller can receive.

   (example : Async. buffer size of controller - overhead of READ DESCRIPTOR command)

   address = 00 00_{16}

3) The target returns appropriate part of specified descriptor.

   There are three cases of ACCEPTED response.

   Case 1 : *read_result_status* = 10_{16}

   The READ request can be handled with no problem.

   Case 2 : *read_result_status* = 12_{16}

   The READ request started in valid data space, but went beyond the end of valid data space.

   data_length = The actual number of bytes read.

   In the above two cases, “read whole descriptor” sequence is finished, and the controller has been received the whole descriptor contents.

   Case 3 : *read_result_status* = 11_{16}

   The READ request was only partially satisfied due to the data transfer capacity limitations.

   data_length = The actual number of bytes read.

   The controller must issue additional READ command(s) to get all of the desired data until read_result_status = 12_{16} or 10_{16}

   Additional READ DESCRPTOR command(s) parameter is set as follows:

   data_length = size that controller can receive.

address = address of preceding READ DESCRIPTOR command
+ data_length value of preceding READ DESCRIPTOR command.

4) The controller issues the OPEN DESCRIPTOR command with the following parameters:

descriptor identifier : indicate the objective descriptor.

subfunction : CLOSE

F.1.2 Method to read an object entry descriptor

This method is applied for reading an object entry descriptor in the list descriptor.

Reading procedure as follows:

1) The controller issues the OPEN DESCRIPTOR command with the following parameters:

descriptor identifier : indicate the object list descriptor, which include objective object entry.

descriptor_type = 10_{16} (Object list descriptor - specified by list ID)

subfunction : READ OPEN or WRITE OPEN

2) The controller issues the READ DESCRIPTOR command with the following parameters:

descriptor identifier : indicate the objective object entry.

descriptor_type = 20_{16} (Object entry descriptor - specified by object position)

or

descriptor_type = 21_{16} (Object entry descriptor - specified by an object ID)

data_length = size that controller can receive.

(example : Async. buffer size of controller - overhead of READ DESCRIPTOR command)

address = 00 00_{16} (top of descriptor)

3) The target returns the appropriate part of specified descriptor.

There are three cases of ACCEPTED response.

Case 1 : *read_result_status* = 10_{16}

The READ request can be handled with no problem.

Case 2 : *read_result_status* = 12_{16}

The READ request started in valid data space, but went beyond the end of valid data space.

data_length = The actual number of bytes read.

In the above two cases, “read an object entry” sequence is finished, and the controller has been received whole object entry contents.

Case 3 : *read_result_status* = 11_{16}

The READ request was only partially satisfied due to the data transfer capacity limitations.

data_length = The actual number of bytes read.

The controller must issue additional READ command(s) to get all of the desired data until *read_result_status* = 12_{16} or 10_{16}

Additional READ DESCRPTOR command(s) parameter is set to following:

data_length = size that controller can receive.

address = address of preceding READ DESCRIPTOR command
+ data_length value of preceding READ DESCRIPTOR command.

4) The controller issues OPEN DESCRIPTOR command with the following parameters:

descriptor identifier : indicate the object list descriptor which include objective object entry.

subfunction : CLOSE

F.1.3 Method to read the header part of the root contents list descriptor

When the READ DESCRIPTOR command with the descriptor identifier specified root contents list, the target may return the header part of root contents list. The controller can read object entries with specifying each entry.

Header part of the root contents list contains the following fields:

list_type

attributes

size_of_list_specific_information

list_specific_information

number_of_entries

Reading sequence for the list descriptor is as follows:

1) The controller issues the OPEN DESCRIPTOR command, which specifies root contents list.

Parameters of the OPEN DESCRIPTOR command are the following :

descriptor identifier : indicate the objective list.

descriptor_type =10_{16} (Object list descriptor - specified by list ID)

descriptor_type_specific_reference = 3000_{16} (list ID of the root contents list)

subfunction : READ OPEN

2) The controller issues the READ DESCRIPTOR command with the following parameters :

descriptor identifier : indicate the objective list.

descriptor_type =10_{16} (Object list descriptor - specified by list ID)

descriptor_type_specific_reference = 3000_{16} (list ID of the root contents list)

data_length = size that controller can receive.

address = 0002_{16} (address of *list_type* field)

3) The target sends header part of the root contents list with RESPONSE frame.

RESPONSE procedure is same as “Method to read whole descriptor”, thus controller may need to issue more READ DESCRIPTOR command(s).

NOTE — In some implementations, it may be very difficult to maintain the *descriptor_length* field of the root contents list, thus the above sequence excludes *descriptor_length* field.

4) The controller issues the OPEN DESCRIPTOR command, which specifies the root contents list.

Parameters of the OPEN DESCRIPTOR command are the following:

descriptor identifier : indicate the root contents list.

descriptor_type =10_{16} (Object list descriptor - specified by list ID)

descriptor_type_specific_reference = 3000_{16} (list ID of the root contents list)

subfunction : CLOSE

F.1.4 Method to read the whole root contents list descriptor

Reading whole root contents list is:

1) The controller issues the OPEN DESCRIPTOR command, which specifies the root contents list.

   The parameters of the OPEN DESCRIPTOR command are as follows:

   descriptor identifier : indicate the objective list.

   descriptor_type =10_{16} (Object list descriptor - specified by list ID)

   descriptor_type_specific_reference = 3000_{16} (list ID of the root contents list)

   subfunction : READ OPEN

2) Read header part of the root contents list. The method is same as part 2) & 3) of "Method to read header part of root contents list descriptor."

3) Parse header part and look for the number of the object entries from the *number_of_entries* field.

4) Read each object entry. The method is same as part 2) & 3) of " Method to read an object entry descriptor". In this situation, descriptor_type of descriptor identifier for the READ DESCRIPTOR command is 20_{16} (Object entry descriptor - specified by object position)

5) The controller issues the OPEN DESCRIPTOR command with the following parameters;

   descriptor identifier : indicate the objective list.

   descriptor_type =10_{16} (Object list descriptor - specified by list ID)

   descriptor_type_specific_reference = 3000_{16} (list ID of the root contents list)

   subfunction : CLOSE

F.1.5 Method to read a info block in the Subunit Identifier Descriptor or the Status Descriptor

For a Player/Recorder Profile, reading the primary field of an info block is guaranteed.

This method is applied for reading info blocks in the Subunit Identifier Descriptor or Status Descriptor.

Reading procedure is as follows:

1) The controller issues the OPEN DESCRIPTOR command with the following parameters ;

   descriptor identifier : indicate the descriptor which contains objective info block.

   Subfunction : READ OPEN

2) The controller issues the READ INFO BLOCK command with the following parameters ;

   info_block_reference_path : indicate objective info block.

   descriptor_type =30_{16} (Information block - specified by its type and instance count)

data_length = size that controller can receive.

(example : Async. buffer size of controller - overhead of READ INFO BLOCK command)

address = 00 00_{16}

3) The target returns appropriate part of specified descriptor.

There are three cases of ACCEPTED response.

Case 1 : *read_result_status* = 10_{16}

The READ request can be handled with no problem.

Case 2 : *read_result_status* = 12_{16}

The READ request started in valid data space, but went beyond the end of valid data space

data_length = The actual number of bytes read.

Above two cases, "read info block" sequence is finished, and the controller has been received contents in the primary field of the info block.

Case 3 : *read_result_status* = 11_{16}

The READ request was only partially satisfied due to the data transfer capacity limitations.

data_length = The actual number of bytes read.

The controller must issue the additional READ command(s) to get all of the desired data until *read_result_status* = 12_{16} or 10_{16}

Additional READ INFO BLOCK command(s) parameter is set as follows:

data_length = size that controller can receive.

address = address of preceding READ INFO BLOCK command

+ *data_length* value of preceding READ INFO BLOCK command.

4) The controller issues the OPEN DESCRIPTOR command with the following parameters:

descriptor identifier : indicate the descriptor which contains objective info block.

Subfunction : CLOSE

F.1.6 Method to read a info block in an object entry

For a Player/Recorder Profile, reading the primary field of an info block is guaranteed.

Reading procedure is as follows:

1) The controller issues the OPEN DESCRIPTOR command with the following parameters :

descriptor identifier : indicate the object list descriptor which contains the object entry with objective info block.

Subfunction : READ OPEN or WRITE OPEN

2) The controller issues the READ INFO BLOCK command with the following parameters :

info_block_reference_path : indicate objective info block.

descriptor_type =30_{16} (Information block - specified by its type and instance count)

data_length = size that controller can receive.

(example : Async. buffer size of controller - overhead of READ INFO BLOCK command)

address = 00 00_{16}

3) The target returns appropriate part of specified descriptor.

There are three cases of the ACCEPTED response.

Case 1 : *read_result_status* = 10_{16}

The READ request can be handled with no problem.

Case 2 : *read_result_status* = 12_{16}

The READ request started in valid data space, but went beyond the end of valid data space

data_length = The actual number of bytes read.

Above two cases, "read info block" sequence is finished, and the controller has been received contents in primary field of info block.

Case 3 : *read_result_status* = 11_{16}

The READ request was only partially satisfied due to the data transfer capacity limitations.

data_length = The actual number of bytes read.

The controller must issue additional READ command(s) to get all of the desired data until the *read_result_status* = 12_{16} or 10_{16}

Additional READ INFO BLOCK command(s) parameter is set as follows:

data_length = size that controller can receive.

address = address of preceding READ INFO BLOCK command

+ *data_length* value of preceding READ INFO BLOCK command.

4) The controller issues OPEN DESCRIPTOR command with the following parameters ;

descriptor identifier : indicate the object list descriptor, which contains the object entry with objective info block.

Subfunction : CLOSE

F.1.7 Method to write info block

All modifiable fields in descriptors of the Player/Recorder Profile are composed of info blocks.

When using the WRITE INFO BLOCK command, it is strongly recommended to modify whole primary field at once. The *group_tag* function is optional for the Player/Recorder Profile, thus the controller should write primary field of info block at one WRITE INFO BLOCK transaction.

Write sequence is as follows:

1) The controller issues the OPEN DESCRIPTOR command to open root contents list. Parameters of the OPEN DESCRIPTOR command are as follows:

descriptor identifier : indicate the object list descriptor which contains the object entry with objective info block..

Subfunction = 03_{16} ("WRITE OPEN")

2) If the controller does not know the actual size of the info block, the controller issues the READ INFO BLOCK command.

3) The controller issues the WRITE INFO BLOCK command. The parameters follow:

info_block_reference_path : point to objective info block.

group_tag = 00_{16} ("immediate")

replacement_data_length = length of *replacement_info_block_data* field.

address = 00 00_{16} (top of primary field)

original_data_length : length of original primary field.

replacement_info_block_data : the new data to be written into the info block.

4) The controller issues the OPEN DESCRIPTOR command with the following parameters ;

descriptor identifier : indicate the object list descriptor which contains the object entry with objective info block..

Subfunction : CLOSE

F.2 Player/Recorder Profile control example

F.2.1 Method to find Generic Recordable Video Disc (Player/Recorder Profile)

Assumption: the controller has already discovered the disc subunit.

<Sequence>

Issue the GENERIC RECORDABLE VIDEO DISC SUPPORT command with implementation_profile_id = 10_{16}.

If the subunit_support bit of the support_state field in the ACCEPTED response frame is 1, it indicates that the target is supporting the Player/Recorder Profile.

F.2.2 Method to check availability of currently loaded disc and know the name of the disc

1) **Check whether medium is existing in drive or not.**
Check the *disc_in_drive* field of media_and_edit_status_info_block, which is in the primary fields of the general_disc_subunit_status_area_info_block in the Status Descriptor.

If the *disc_in_drive* value is 01_2 , disc is loaded.

2) **Check availability of currently loaded disc and know the name of the loaded disc**
Issue the GENERIC RECORDABLE VIDEO DISC SUPPORT command with implementation_profile_id = 10_{16}.
If the currently_loaded_media_support bit of the support_state field in the ACCEPTED response frame is 1, it indicates that the currently loaded disc on the target is supporting the Player/Recorder Profile.
The media_name field describes the name of the media which is currently loaded on the target.

F.2.3 Method to check medium change or contents change.

Medium change:

As described in above section, medium change is detected by watching the *disc_in_drive* field.

Contents or medium change;

When

- Any change/update is applied on contents in a disc.
  (i.e. record new object, delete object, divide object, combine objects, change object's name, etc.)
- unload medium
- load medium

the *time_stamp_data* value of the Time Stamp Info Block in header part of root contents list is updated.

So that controller can detect modification of contents on a disc by polling the *time_stamp_data*.

F.2.4 Method to check remaining capacity

Remaining capacity is specified in the *disc_remaining_recording_capacity* field of the disc_capacity_info_block, which is located in the Root Contents List *optional_info_block_area*, thus the controller can know remaining capacity using the READ INFO BLOCK command.

In the case of the Player/Recorder Profile, the format of the *disc_remaining_recording_capacity* field is "bytes."

F.2.5 Method to get list of contents

Read header part of root contents list and memorize time stamp info block.

NOTE — When root contents list is updated, the *time_stamp_info_block* in the Root Contents List *list_specific_information* is updated too. Before the controller does something which affects the contents, (ERASE, write attribute, etc.), the controller should check if the time stamp value is not changed.

Read each object entry in root contents list.

NOTE —

A) If there are other type object entries than video object, a controller should check the *entry_type* field of object entry to work without problem.

B) If an object has a child list, a controller should check the *has_child_ID* flag in attribute field of object entry to work without problem.

Above two restrictions (A & B) are to take steps to meet the future extension of the Generic Recordable Video Disc specification.

F.2.6 Method to RECORD specific contents on destination plug #j

Setup destination plug #j

Use the CONFIGURE command to set the *video_signal_mode* in video object, the *object_and_plug_type_specific_information* field of the *plug_configuration_info_block*, which is located in the *plug_status_info_block* in the *destination_plug_status_area_info_block* of the Disc Subunit Status Descriptor.

Issue the SET PLUG ASSOCIATION command to associate root contents list with destination plug #j.

Issue the RECORD command to destination plug #j, parameters are followings;

destination_plug : destination plug #j

subfunction_1 : 75_{16} (“forward”)

subfunction_2 : 00_{16} (“new”)

NOTE —As soon as accepting the RECORD command, the target creates a new video object entry with info blocks in the root contents list.

Write attributes of new video object by using the WRITE INFO BLOCK commands.

NOTE —During recording, size information in video object entry may be inaccurate. If the controller wants to know the size of the video object during recording, the controller should read the status of destination plug.

F.2.7 Method to “RECORD PAUSE”

— When changing the *rec_state* from “RECORD forward” to “RECORD pause”

Issue the RECORD command to destination plug #j. The parameters are as follows:

destination_plug : destination plug #j

subfunction_1 : $7D_{16}$ (“forward pause”)

subfunction_2 : 00_{16} (“new”)

— When changing the *rec_state* from “RECORD pause” to “RECORD forward”

Issue the RECORD command to destination plug #j. The parameters are as follows:

destination_plug : destination plug #j

subfunction_1 : 75_{16} (“forward”)

subfunction_2 : 00_{16} (“new”)

NOTE — Changing *rec_state* does not cause to create new video object. The RECORD operation is on going.

F.2.8 Method to STOP record operation on destination plug #j

Issue STOP command specifying destination plug #j.

NOTE — When STOP command is ACCEPTED, the value of the *size_indicator_info_block* in the video object indicate the size of the video contents recorded.

F.2.9 Method to PLAY specific object on source plug #i at the beginning

1) Setup source plug #i
2) Issue the SET PLUG ASSOCIATION command to associate specific object with source plug #i
3) Issue the SEARCH command to the beginning point (00:00:00.00)

NOTE — Player/Recorder Profile supports relative HMSF format on SEARCH command.

4) Issue the PLAY command with “forward X1” to source plug #i

F.2.10 Method to “PLAY PAUSE”

— When changing *play_state* from “PLAY forward” to “PLAY pause”.

Issue the PLAY command to source plug #j. The parameters are as follows:

source_plug : source plug #j

subfunction_1 : $7D_{16}$ (“forward pause”)

— When changing *play_state* from “PLAY pause” to “PLAY forward”.

Issue the PLAY command to source plug #i. The parameters are as follows:

source_plug : source plug #i

subfunction_1 : 75_{16} (“forward”)

F.2.11 Method to SEARCH specific position.

Issue the SEARCH command. The parameters are as follows:

search_type : 00_{16} (position)

indicator_type : 00_{16} (relative HMSF count)

NOTE — After SEARCH operation is ended, operating_mode of plug is resume to the same operating_mode before the SEARCH command was issued.

F.2.12 Method to STOP PLAY operation on source plug #i

Issue the STOP command specifying source plug #i.

NOTE — The value of a current pointer is maintained as it is positioned when the operation stopped.

F.2.13 Method to ERASE specific video object

Issue the ERASE command specifying the video object.

F.2.14 Method to change plug association

Issue the SET PLUG ASSOCIATION command.

NOTE —SET PLUG ASSOCIATION command may be REJECTED when *operating_mode* is not STOP.

F.2.15 Method to "RECORD LOOP" specific AV track from the beginning (Optional)

(1) Issue the RECORD command to destination plug #j, with parameters as follows:

subfunction_1 : 75_{16} ("forward")

subfunction_2 : 06_{16} ("loop")

start_position : beginning position.
indicator_type : 00_{16} ("relative_HMSF_count")
indicator_specific field : 00 00 00 00_{16} (beginning position)

(2) Write attribute of video object by using the WRITE INFO BLOCK commands.

NOTE — When the plug configuration that is set up for RECORD loop command is different from the plug configuration for creating this AV track, the AV-HDD can reject RECORD loop command.

F.2.16 Method to "RECORD LOOP" specific AV track from the current recording position (Optional)

(1) Issue the RECORD command to destination plug #j, with parameters as follows:

subfunction_1 : 75_{16} ("forward")

subfunction_2 : 06_{16} ("loop")

start_positon : unspecified.
indicator_type : 00_{16} ("relative_HMSF_count")
indicator_specific field : FF FF FF FF_{16} (current recording position)

(2) Write attribute of video object by using WRITE INFO BLOCK commands.

NOTE — When the plug configuration that is set up for the RECORD loop command is different from the plug configuration for creating this AV track, the target can reject the RECORD loop command.

F.2.17 Method to PLAY specific AV track in loop recording (Optional)

(1) Setup source plug #i

(2) Issue the SET PLUG ASSOCIATION control command to associate specific AV object with source plug #i

(3) Issue the SEARCH control command to search the start position in the recorded part of the AV track that is in loop recording

(4) Issue the PLAY control command to source plug #i

Note 1. When the controller issues the SEARCH control command that specifies the position in the not recorded part of the AV track, the SEARCH control command may be rejected.

Note 2. When the current play back position catches up the current recording position, the target may change play back mode to follow the current recording position.

