TA Document 2003005
AV/C Camera Storage Subunit 2.1

February 4 , 2004

Sponsored by:
1394 Trade Association

Accepted :
1394 Trade Association Board of Directors.

Abstract:
This specification defines a model and command set for Camera Storage.

Keywords:
1394, Digital, Camera, Storage. .

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:

USA
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

The following table shows the change history for this specification.

Version 1.0

Original Version

Version 2.0

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

Content change for Version 2.0

Category	Description
Technical	New format of “long directory entry” for file list is defined.
Technical	Following result code is newly defined for ACCEPTED response. - default position
Technical	Following result codes are newly defined for REJECTED response. - directory exist - not empty - no input - unsupported stream - invalid operation mode - no operation
Technical	Following new commands are newly defined. - CREATE DIRECTORY - FILE SIGNAL FORMAT - FORMAT - PLAY FILE - RECEIVE FILE PARTIAL - RECORD FILE - SEND FILE PARTIAL
Technical	Result code of “invalid parameter” is added to the tables of result codes for RECEIVE FILE, SEND FILE and SEND THUMBNAIL commands.
Technical	Result code of “invalid file type” is removed from the table of result codes for RECEIVE FILE command.
Technical	New *file_info_type* value of “Vendor dependent” is defined for FILE INFO command.
Technical	New field of *list_mode* is added to FILE LIST command.
Technical	New subunit_version_information value of “Version 2.0” is defined for VERSION command.
Technical	New field of *character_set* is added to VOLUME INFO response.
Technical	State machine written in Annex C is modified to support PLAY FILE and RECORD FILE commands.

Version 2.1

Version 2.1 of this document differs from version 2.0 in the following ways:

Content change for Version 2.1

Category	Description
Change for Draft 0.0:1(2003/01/13)	
Technical	Following new profile is defined - limited aynchrous sender
Technical	Following new implementation_profile_id values is defined - limited aynchrous sender
Technical	Enable to use URI as “file_path”
Technical	New camera storage subunit version information is defined for VERSION command
Change for Draft 0.0:2 (2003/02/10)	
Editorial	Modify inadequate explanation 5th paragraph of Profile chapter.
Change for Draft 0.90 (2003/04/30)	
Technical	Following new profiles are defined - aynchrous proxy sender - limited aynchrous proxy sender
Technical	Following new implementation_profile_id values are defined - aynchrous proxy sender - limited aynchrous proxy sender
Technical	Following new Result codes are defined - no proxy function - proxy not available - unsupported protocol
Technical	Following new Result codes are added for SEND FILE command - no proxy function - proxy not available - unsupported protocol
Technical	Following new Result codes are added for SEND FILE PARTIAL command - no proxy function - proxy not available - unsupported protocol
Technical	Following new command is defined. PROXY INFO
Change for Draft 0.91 (2003/05/15)	
Editorial	Correct some editorial errors
Change for TA Candidate 1.0 (2003/09/10)	
Technical	Clarifing the responses’ frame formats and fields’ values when the control command is rejected. - ERASE_FILE control command - FILE_INFO control command - FILE_LIST control command - NUMBER_OF_ENTRIES control command - VOLUME_INFO control command
Technical	Clarifing the value of result field when the fully specified path does not exist. - FILE_LIST control command

Technical	Clarifing the response frames of some STATUS commands while being “suspended” and “aborting”. - RECEIVE_FILE status command - RECEIVE_FILE_PARTIAL status command - SEND_FILE status command - SEND_FILE_PARTIAL status command
Technical	Clarifing the range of “JPEG compressed data portion”
Technical	Clarifing the correct rule between the conflict rules - FILE_LIST control command
Editorial	Correct some editorial errors
Change for BoD Candidate 1.0 (2003/12/17)	
Editorial	Correct some editorial errors

1. Overview

1.1 Purpose

The model and command set described in this document are intended to provide an easy method of accessing the files stored on storage media The files are specified by the file path defined in version1.0 and 2.0 ,additionally ,from this version , the files are enabled to be specified by Uniform Resource Identifier (URI) [R13]. The storage media this document focuses on are removable and non-removable physical media along with external storage referenced by a URI and acting as an Internet proxy.

1.2 Scope

This document describes a command set and subunit model that inputs files to or outputs files from storage media or, acting as an Internet proxy, to output file on the Internet. It is assumed that the file transport layer utilizes Asynchronous Connections and Isochrnous Connection, but other connection methods, such as External Connection or Internal Connection may also be used. The format of isochronous data handled by camera storage subunit is limited to the data defined by IEC-61883 ([R2]-[R6]).

2. References

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

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

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

[R3] IEC-61883-2, Consumer audio/video equipment – Digital interface – Part 2: SD-DVCR data transmission.

[R4] IEC-61883-3, Consumer audio/video equipment – Digital interface – Part 3: HD-DVCR data transmission.

[R5] IEC-61883-4, Consumer audio/video equipment – Digital interface – Part 4: MPEG2-TS data transmission.

[R6] IEC-61883-5, Consumer audio/video equipment – Digital interface – Part 5: SDL-DVCR data transmission.

[R7] TA Document 1998003, AV/C Digital Interface Command Set General Specification, Version 3.0.

[R8] TA Document 2000005, AV/C Compatible Asynchronous Serial Bus Connections, Version 2.0.

[R9] TA Document 1999037, AV/C Command for Management of Enhanced Asynchronous Serial Bus Connections, Version 1.0.

[R10] TA Document 1999031, AV/C Connection and Compatibility Management Specification, Version 1.0.

[R11] JEIDA-49-1998, Exchangeable image file format for digital still cameras (Exif) Version2.1.

[R12] JEIDA-49-2-1999, Design rule for Camera File System (DCF) Version1.0.

[R13] RFC 2396 August 1998, Uniform Resource Identifier (URI)

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.

3.2 Glossary of terms

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

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

3.2.3 DCF basic file: An image file stored directly under a DCF directory, having a DCF file name and the extension “JPG” and having the DCF-stipulated data structure, based on the Exif standard.

3.2.4 DCF basic thumbnail: An Exif thumbnail image included in a DCF basic file.

3.2.5 DCF directory: A directory under the DCF image root directory created in accordance with the DCF directory rules, for storing images.

3.2.6 DCF file: A file with a DCF file name.

3.2.7 DCF file name: A file name assigned in accordance with the DCF file naming conventions.

3.2.8 DCF file number: A four-digit number making up part of the DCF file name.

3.2.9 DCF image root directory: The directory under the root directory, created in accordance with the DCF directory rules.

3.2.10 DCF object: A group of files recorded in accordance with DCF.

3.2.11 DCF thumbnail file: A compressed file for storing the thumbnail image of a DCF extended image file.

3.2.12 Exif standard: Digital still Camera Image File Format Standard, Version 2.1 of the Japan Electronic Industry Development Association (JEIDA).

3.2.13 JPEG standard: ISO/IEC 10918-1 ITU-T Recommendation T81 information technology - Digital compression and coding of continuous-tone still image - Requirements and guidelines.

3.2.14 main image: The primary data of image.

3.2.15 quadlet: Four bytes of data.

3.2.16 receiver: The camera storage subunit that implements the receiver profile. The receiver can input and store file data.

3.2.17 sender: The camera storage subunit that implements the sender profile. The sender can provide contents information and output file data.

3.2.18 still image file: A file that the camera storage subunit can handle as image data.

3.2.19 thumbnail: A small version of the main image, used for indexing.

3.2.20 volume: A unit of storage area physically or logically divided.

3.3 Acronyms and abbreviations

AV/C: Audio Video Control.

DCF: Design rule for Camera File system.

CCM: Connection and Compatibility Management

Exif: Exchangeable image file format for digital still camera

FCP: Function Control Protocol

JPEG: Joint Photographic Expert Group

URI: Uniform Resource Identifier

4. Subunit model

4.1 Camera storage subunit logical model

Figure 4-1 illustrates the logical model of a camera storage subunit.

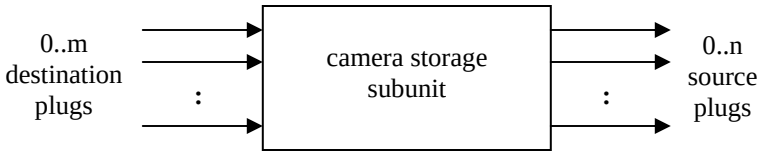


Figure 4-1 – Logical model of a camera storage subunit

The camera storage subunit shall have at least one destination plug or one source plug.

The camera storage subunit may have from zero to m destination plugs. If it supports the sender profile only, then it should have no destination plug.

The camera storage subunit may have from zero to n source plugs to output file data. If it supports the receiver profile only, then it should have no source plug.

Camera storage subunit command frames and response frames are of variable length and may be large. To handle the command and response frames efficiently, the controller and the camera storage subunit should support the transaction of maximum FCP payload size, 512 bytes.

— A controller shall assume that the maximum size of MEDIA INFO response frame is 264 bytes.

— A camera storage subunit supporting asynchronous sender profile shall support more than 44 bytes as FCP payload size, which equals to the size of SEND FILE control command frame to request a DCF file.

The camera storage subunit does not provide file list state information to the controller. Unless one controller prohibits the other controller from accessing the camera storage subunit, it is possible for the other controller to write or erase files. Therefore, because the camera storage subunit does not provide state information indicating whether the file list has been changed by the other controller, each controller should update file list information in a timely manner.

4.2 File system

The camera storage subunit is intended to support the removable or non-removable memory as storage media or, may be acting as an Internet proxy, to support the Internet as a storage media. This specification assumes the storage media is employing hierarchical file.

5. Accessing storage contents

5.1 file_path

The camera storage subunit uses *file_path* to access the specific file.

5.1.1 file_path as file name

The length of *file_path* as file name is variable and is composed of the full path name beginning with the root directory but excluding the drive name, *file_path* uses the "\" character ($5C_{16}$) as a directory separator and is terminated by the NULL character (00_{16}). The length of *file_path* is indicated by *file_path_length*. The length does not include the NULL character.

Consider the directory structure illustrated in Figure 5-1 below:

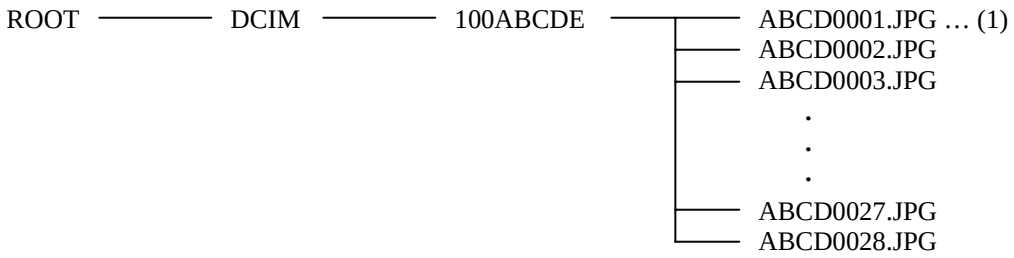


Figure 5-1 – Example of directory structure

file_path_length and *file_path* for the file (1) are specified in Table 5-1 below.

Table 5-1 – Example values of *file_path_length* and *file_path*

field name	value
file_path_length	$1B_{16}$
file_path	"\DCIM\100ABCDE\ABCD0001.JPG"+00_{16}

5.1.2 file_path as Uniform Resource Identifier (URI)

The length of *file_path* as URI is variable and is composed of the absolute (full) URI in layout form beginning with the scheme, *file_path* uses the ":" character ($3A_{16}$) as a scheme and scheme-specific-part separator and is terminated by the NULL character (00_{16}). The scheme-specific-path uses the "generic URI" syntax given as

<scheme>:<scheme-specific-part>

<scheme>://<authority><path>?<query>

The_<path> uses the "/" character ($2F_{16}$) as a separator of hierarchical components. The length of *file_path* is indicated by *file_path_length*. The length does not include the NULL character.

file_path_length and *file_path* for the file (1) are specified in Table 5.2 below.

Example values of *file_path_length* and *file_path*

field name	value
file_path_length	21_{16}
file_path	“http://www.testsite.com/test.html”+00_{16}

The file_path as a URI is only acceptable for the SEND_FILE and SEND_FILE_PARTIAL commands.

5.2 File list

When the storage media is removable and non-removable physical media the controller requests the file list to retrieve the information of the contents stored on the storage. The file list is transferred from the camera storage subunit to the controller in the response frame of the requesting command or using asynchronous connections. The file list consists of multiple directory entries and each directory entry holds file or subdirectory information. There are two types of directory entry format.

5.2.1 Short directory entry

The short directory entry is 20 bytes in size and its format is shown in Figure 5-2 below.

offset	contents
00_{16}	name
:	
07_{16}	
08_{16}	extension
09_{16}	
$0A_{16}$	
$0B_{16}$	attribute_byte
$0C_{16}$	modification_time
$0D_{16}$	
$0E_{16}$	modification_date
$0F_{16}$	
10_{16}	file_size
11_{16}	
12_{16}	
13_{16}	

Figure 5-2 – Structure of short directory entry

The format and meaning of each field in the directory entry is identical to the directory entry of the DOS-FAT file system. For detailed information on the format of each field, please refer to the document addressing the DOS-FAT file system. The file list sent from the camera storage subunit shall not include the directory entries “.” (current directory) and “..” (parent directory) used with the DOS-FAT file system.

The *name* and the *extension* fields indicate the file name. These two fields adhere to the DOS 8.3 file naming convention. If a file's name or extension size is less than allotted field size, then padding bytes shall be used. The value of the padding byte is 20_{16}.

The *attribute byte* field indicates the file's attribute information. This field consists of bit fields as defined in Figure 5-3 below.

bit offset	contents
7 (msb)	implementation_dependent
6	
5	
4	directory
3	implementation_dependent
2	
1	
0 (lsb)	read_only

Figure 5-3 – Structure of attribute byte

The *directory* bit indicates whether the directory entry is a subdirectory. If this bit is 1, then the entry is a subdirectory; otherwise, the entry is a file.

The *read_only* bit indicates whether the file corresponding to the directory entry is read only. If this bit is 1, then the file is read only.

The camera storage subunit may assign any value to the *implementation_dependent* bits. The controller may ignore these bits.

The *modification_time* field indicates the most recent time the file was modified. The value is stored in **little endian** order, with the less significant byte first. The format of time value is shown in Figure 5-4 below.

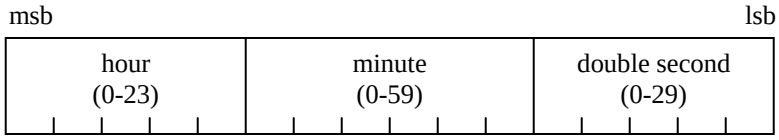


Figure 5-4 – Format of time value

Since there are not enough bits in the time value to express the time to the nearest second, the time is first rounded or truncated to the nearest even second. The resulting seconds field is divided by 2 and written into bits 4-0.

For example, if the modification time is "12:34:56", then the time value is $645C_{16}$ and the *modification_time* field shall have the value of $5C64_{16}$.

The *modification_date* field indicates the most recent date the file was modified. The value is stored in **little endian** order, with the less significant byte first. The format of date value is shown in Figure 5-5 below.

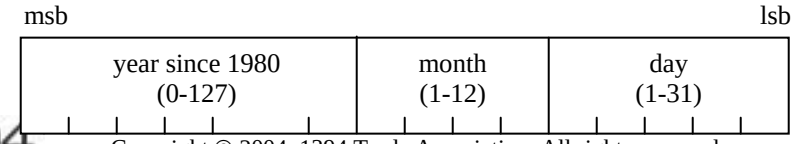


Figure 5-5 – Format of date value

For example, if the modification date is “April 1, 2000”, then the date value is 2881_{16} and the *modification_date* field shall have the value of 8128_{16}.

The *file_size* field indicates the size of the file in bytes. The value is stored in **little endian** order, with the less significant byte first. For example, if the file size is 12345678 bytes ($00BC614E_{16}$ bytes), the *file_size* field shall have the value of $4E61BC00_{16}$.

The camera storage subunit transfers the directory entries of directories and files existing on the specified volume and path. The volume is specified by the *physical_volume_number* and *logical_volume_number* fields. The path is specified by the *request_path* field. For detailed information on the format of the file list request command, please refer to section 7.10.

5.2.2 Long directory entry

The long directory entry is variable length and supports long file name and file size exceeds $FFFFFFFF_{16}$ bytes. The format of long directory entry is shown in Figure 5-6 below.

offset	contents
00_{16}	attribute_byte
01_{16}	modification_time
02_{16}	
03_{16}	modification_date
04_{16}	
05_{16}	file_size
:	
$0C_{16}$	
$0D_{16}$	entry_name_length
$0E_{16}$	entry_name
:	
:	

Figure 5-6 – Structure of directory entry for asynchronous connections

The formats and meanings of *attribute_byte*, *modification_time* and *modification_date* fields are identical to the directory entry format for response frame described in the section 5.2.1.

The meaning of *file_size* field is identical to the directory entry format for response frame described is the section 5.2.1. The field size is 8 bytes and the value is stored in **little endian** order.

The *entry_name_length* field indicates the size of the *entry_name* field in bytes. The length does not include the NULL character.

The *entry_name* field indicates the name of the directory entry. This field is variable length and indicated by ASCII characters terminated by NULL (00_{16}).

5.3 Generation count

The camera storage subunit generates *media_generation_count* to signal the occurrence of a media exchange event. The modification rule for generation count is shown in Table 5-2 below.

Table 5-2 – Modification rule for generation count

	media remove	media insert	file delete/write	bus reset
media_generation_count	unchanged	increased	unchanged	unchanged

If the camera storage subunit supports multiple physical volumes, each physical volume has its own *media_generation_count*. When the camera storage subunit initializes its system, generation count is assigned an initial value. The initial value depends on the subunit implementation. The camera storage subunit may use a fixed initial value such as zero, or may use a random number. If the subunit can maintain the counter value during power-off, by using flash memory or some other method, then that value may be used as an initial value. In the event of a bus reset, the camera storage subunit shall retain the same value before the bus reset.

The size of *media_generation_count* is 1 byte. When a storage media is inserted in the camera storage subunit, this counter value is increased by 1. Because inserting media causes complete modification of the file list, the controller should retrieve the file list again when a new *media_generation_count* value is detected. When the storage media is removed, file access commands shall be rejected. So, because the controller need not maintain a file list, the camera storage subunit shall not update this counter value.

Except for the VOLUME INFO command, when the controller issues a control command with a common frame header (described in section 7.3) it shall assign the current value to the *media_generation_count* field. The camera storage subunit shall examine the field and if the value is not equal to the current value, the command shall be rejected. The camera storage subunit shall set the current value in the response frame.

The range of the count value is 00_{16} to FE_{16}. The value of FF_{16} is used only as a parameter of command frame or response frame of status command as “no meaning”, so the camera storage subunit shall not use this value as a generation counter value. When the value of this counter is FE_{16} and new storage media is inserted, the counter value shall reset to 00_{16}.

5.4 Multi-volume functionality

The camera storage subunit may support multiple physical storage media and each media can be separated into multiple partitions. In this document, the physical media is referred to as the “physical volume” and each partition on the media is referred to as a “logical volume”. By specifying the physical volume and logical volume using *physical_volume_number* and *logical_volume_number* fields in the command frame, the controller can specify a particular area on the storage media. The controller can determine the number of physical volumes and the number of logical volumes on each physical volume by issuing the MEDIA INFO control command described in section 7.13.

Multi-volume is optional. The camera storage subunit and the controller shall support at least one physical volume and one logical volume. When physical volume is comprised of media with multiple partitions, the camera storage subunit may support only one logical volume. The controller may support only a single volume, as one physical volume with one logical volume.

5.4.1 physical_volume_number

The *physical_volume_number* specifies the physical volume in the camera storage subunit. The camera storage subunit shall have at least one physical volume.

The range of *physical_volume_number* is 00_{16} to FE_{16}. The physical volume number assigned to the first volume is 00_{16}. The value of FF_{16} is used only as a parameter of command frame or response frame of status command as “no meaning”, so the camera storage subunit shall not use this value as a physical volume number.

5.4.2 logical_volume_number

The *logical_volume_number* specifies the logical volume on the physical volume specified by the *physical_volume_number* field. When the specified physical volume contains no media, there shall be no logical volume. If the specified physical volume contains media initialized in an unsupported format, the volume shall be counted as a logical volume.

The range of *logical_volume_number* is 00_{16} to FE_{16}. The logical volume number assigned to the first volume is 00_{16}. The value of FF_{16} is used only as a parameter of command frame or response of status command as “no meaning”, so and the camera storage subunit shall not use this value as a logical volume number.

5.5 Result code

When a control command with a common frame header (as described in section 7.3) is accepted or rejected, the camera storage subunit sets the result code to the *result* field in the response frame. Result code is also set in the response frame of camera storage status command (as described in section 7.6). The *result* field in the response frame shall have one of the values in Table 5-3 below:

Table 5-3 – *result* code values in response frame

response type	result	value	meaning
ACCEPTED	success	00_{16}	successful completion
	default position	01_{16}	command is accepted but specified position is invalid
STABLE	not executing	00_{16}	specified control command is not being executed
	executing	01_{16}	specified control command is being executed
	suspended	02_{16}	specified control command is suspended by bus reset
	aborting	03_{16}	specified control command is being aborted
REJECTED	busy	80_{16}	system is busy
	aborted	81_{16}	the command is aborted
	retry	82_{16}	same command is being executed
	no command	83_{16}	there is no corresponding command
	disabled	84_{16}	the command is disabled
	invalid generation count	90_{16}	media_generation_count is invalid
	invalid volume number	91_{16}	volume number is invalid
	invalid subunit plug	92_{16}	source or destination plug number is invalid
	invalid parameter	93_{16}	invalid parameter is used
	no media	$A0_{16}$	storage media doesn't exist
	no file	$A1_{16}$	specified file doesn't exist
	no directory	$A2_{16}$	specified directory doesn't exist
	no space	$A3_{16}$	there is no space to store the file
	file exist	$A4_{16}$	specified file_path already used
	unsupported format	$A5_{16}$	format of specified volume is unsupported
	invalid file type	$A6_{16}$	specified file is not supported
	volume protected	$A7_{16}$	specified volume is write protected
	file protected	$A8_{16}$	specified file is read only
	no thumbnail	$A9_{16}$	there is no thumbnail for the image file
	directory exist	AA_{16}	specified directory_path already exist
	not empty	AB_{16}	the directory to be removed is not empty
	directory protected	AC_{16}	specified directory is write protected
	plug busy	$B0_{16}$	specified plug is busy
	transport error	$B1_{16}$	error occurred data transport layer
	no input	$B2_{16}$	no data is input from specified destination plug
	unsupported stream	$B3_{16}$	input stream is not supported
	invalid operation mode	$C0_{16}$	operation mode is invalid for the specified file
	no operation	$C1_{16}$	there is no corresponding operation
	no proxy function	$D0_{16}$	There is no proxy function
	proxy not available	$D1_{16}$	proxy function is not available
	unsupported protocol	$D2_{16}$	Specified protocol is not supported
	unknown	FE_{16}	an unknown error occurred

If the command is rejected for multiple reasons, the camera storage subunit should return the lowest result code. When the command is rejected and the camera storage subunit cannot return the appropriate result status, the camera storage subunit may return the value FE_{16}.

The possible result codes for each command are summarized in the section devoted to each command. For detailed information, please refer to each section.

6. Profile

This specification defines four types of camera storage subunit profiles. The profile names and values of *implementation_profile_id* are shown in **エラー! 参照元が見つかりません。** below.

Table 6-1 – *implementation_profile_id* values

profile name	value	meaning
asynchronous sender	00_{16}	The subunit can send file data to asynchronous plug.
isochronous sender	01_{16}	The subunit can send file data to isochronous plug.
asynchronous receiver	02_{16}	The subunit can receive file data from asynchronous plug.
isochronous receiver	03_{16}	The subunit can receive file data from isochronous plug.
limited asynchronous sender	04_{16}	The subunit can send file data to asynchronous plug by the send request commands only.
asynchronous proxy sender	05_{16}	The subunit can send file data to asynchronous plug with proxy function.
limited asynchronous proxy sender	06_{16}	The subunit can send file data to asynchronous plug by the send request commands only with proxy function
no info	FF_{16}	no information
Reserved	all others	reserved for future extension

The camera storage subunit supporting the sender profile can provide contents information and can output file data to the subunit plug; the one supporting the receiver profile can input and store file data from the subunit plug. The mandatory command set is determined by the supported profile. For detailed information on support level, please refer to section 7.2. The supported profile is detected by the VERSION command described in section 7.24. The camera storage subunit may support only one profile or it may support multiple profiles.

The AV/C unit that implements the camera storage subunit supporting the asynchronous sender profile and the limited asynchronous sender profile shall implement producer functions of Asynchronous Serial Bus Connections described in [R8], and, in the case of the asynchronous reeiver profile, shall implement consumer functions. To connect the producer and the consumer, the AV/C unit that implements the camera storage subunit supporting at least one of the asynchronous profiles and the controller which handles the AV/C unit should implement the functions defined in [R5].

To connect the source and the destination, the AV/C unit that implements the camera storage subunit supporting at least one of the isochronous profiles and the controller which handles the AV/C unit should implement the CCM functions described in [R10].

The camera storage subunit may disable supported profiles. Consider an AV/C unit that implements a camera subunit, a tape recorder/player subunit and a camera storage subunit. This AV/C unit features the functionality of a digital camcorder and digital still camera. The user can select the operation mode via the mode switch on the unit.The selectable modes are, for example, “Movie rec.”, “Movie play”, “Still rec.”and “Still play”and the camera storage subunit implements asynchronous sender/receiver profiles.

When the user selects “Movie rec.”or ”Movie play” mode, the AV/C unit may accept only commands for the camera subunit or the tape recorder/player subunit. In this case the camera storage subunit is disabled and the commands for the camera storage subunit may be rejected (both profiles are disabled). When the user selects “Still rec” mode, the camera storage subunit may accept camera storage subunit commands defined for the receiver profile and the commands defined for the sender profile may be disabled. In the case of “Still play” mode, the camera storage subunit may disable the receiver profile. When the camera storage subunit receives a disabled command, the REJECTED response with a result code of “disabled”

should be returned. Even if one or both profiles are disabled, the camera storage subunit shall set all implementation_profile_id values that are actually implemented to the response frame when the VERSION status command is received.

Depending on application and system, there are some case which the Controller can accomplish object file information without using file information commands of Camera Storage Subunit.

The print system based on Print By Reference method is an example of such case.

In the print sytem above, printer works as Printer Subunit target and as controller of Camera Storage Subunit at the same time.

The controller of Printer Subunit issues printing request command with printing object file information to printer as a Printer Subunit target.

Camera Storage Subunit Controller in the printer retrieves printing object file information from printing request command .

Then by using SEND_FILE command or SEND_FILE_PARTIAL command set file path with printing object file information, the Controller retrieves printing object file as occasion demands and prints its file.

In fact , the print system above doesn't need file information acquirement commands e.g. FILE_LIST .

So, the aynchronous sender profile is not suitable for the application like the print system above .

Because information acquirement commands are mandatory in its profile, therefore almost unnecessary those commands shall be sopported , too.

So, sender profile, which doesn't request file information acquirement commands to be supported, is newly prepared, and it is named as limited aynchronous sender profile.

Based on limited aynchronous sender profile, it becomes easier to implement Camera Storage Subunit in application or system which controller can accomplish object file information without using file information acquirement commands.

Moreover, limited aynchronous sender profile provides an apparent and easy method of dissembling the files on Camera Storage Subunit .

Camera Storage Subunit can't identify Controller because it has no login mechanism.

So, it is impossible to enable the files to be accessed from only specified Controller and be not accessed from any other Controller.

This lack of features causes problem for application which needs limited file access.

For example, there are some cases which priting object files shall be accessed only from controller in Printer Unit by some reason, e.g. protection copyright.

It is very difficult and almost impossible for controller, excluding one which is acquainted with file information beforehand, to retrieve the files from Camera Storage Subunit which adopts limited aynchronous sender profile and doesn't sopport file information acquirement commands .

The limited aynchronous sender profile is not subset of aynchronous sender profile, because FILE_SEND_PARTIAL is mandatory in limited aynchronous sender profile but it is optional in aynchronous sender profile.

So , it is neeed to put both limited aynchronous sender profile and aynchronous sender profile in one subunit that least common of mandatory commands in both of their profiles shall be supported.

From this version, if the AV unit, which has the camera storage subunit inside, can connect the Internet, the camera storage subunit may act as Internet proxy.

For this case, two sender profiles are defined and added.

One is “aynchronous proxy sender profile”, the other is “limited aynchronous sender profile”.

The aynchronous proxy sender profile is extended the aynchronous sender profile to have proxy function.

The command support level for the aynchronous proxy sender profile is same as the one for aynchronous sender profile except for PROXY INFO command..

The limited aynchronous proxy sender profile is extended the limited aynchronous sender profile to have proxy function.

The command support level for the limited aynchronous proxy sender profile is same as the one for limited aynchronous sender profile except for PROXY INFO command..

7. Camera storage subunit commands

7.1 Camera storage subunit type

The camera storage subunit shall be addressed by a subunit_type value of $0B_{16}$.

7.2 Camera storage subunit commands

Table 7-1 below summarizes the camera storage subunit commands. The support level is differentiated by its implemented profile – asynchronous sender, isochronous sender, asynchronous receiver, isochronous receiver , limited asynchronous sender , asynchronous proxy sender or limited asynchronous proxy sender.

Table 7-1 – Camera storage subunit commands

Opcode	value	defined ctypes															comments
		asynch sender			isoch sender			asynch receiver			iscoch receiver			limited asynch sender			
		C	S	N	C	S	N	C	S	N	C	S	N	C	S	N	
CREATE DIRECTORY	57_{16}	O	O	-	O	O	-	O	O	-	O	O	-	O	O	-	Create new directory or delete existing directory
ERASE FILE	53_{16}	O	O	-	O	O	-	O	O	-	O	O	-	O	O	-	Erase the specified file
FILE INFO	46_{16}	M	M	-	M	M	-	O	O	-	O	O	-	O	O	-	Report the file specific information of the specified file
FILE LIST	43_{16}	M	M	-	M	M	-	O	O	-	O	O	-	O	O	-	Report the file list
FILE SIGNAL FORMAT	47_{16}	-	-	-	M	M	-	-	-	-	-	-	-	-	-	-	Report the available output signal format of the specified file
FORMAT	56_{16}	O	O	-	O	O	-	O	O	-	O	O	-	O	O	-	Format the specified volume
MEDIA INFO	40_{16}	M	M	-	M	M	-	M	M	-	M	M	-	O	O	-	Report the media information
NUMBER OF ENTRIES	42_{16}	M	M	-	M	M	-	O	O	-	O	O	-	O	O	-	Report the number of directory entries in the specified path
NUMBER OF OBJECT FILES	44_{16}	O	O	-	O	O	-	O	O	-	O	O	-	O	O	-	Report the number of files which compose DCF object
OBJECT FILE LIST	45_{16}	O	O	-	O	O	-	O	O	-	O	O	-	O	O	-	Report the file list of the files which compose the DCF object contains a specified file
PLAY FILE	$C3_{16}$	-	-	-	M	M	-	-	-	-	-	-	-	-	-	-	Play specified file and output the stream data to subunit source plug
RECEIVE FILE	52_{16}	-	-	-	-	-	-	M	M	-	-	-	-	-	-	-	Input the file data from subunit destination plug and store in the storage media
RECEIVE FILE PARTIAL	55_{16}	-	-	-	-	-	-	O	O	-	-	-	-	-	-	-	Input the data from subunit destination plug and replace existing file data
RECORD FILE	$C2_{16}$	-	-	-	-	-	-	-	-	-	M	M	-	-	-	-	Input the stream data from subunit destination plug and store in the storage media
SEND FILE	50_{16}	M	M	-	-	-	-	-	-	-	-	-	-	M	M	-	Output the specified file to subunit source plug
SEND FILE PARTIAL	54_{16}	O	O	-	-	-	-	-	-	-	-	-	-	M	M	-	Output the part of specified file to subunit source plug
SEND THUMBNAIL	51_{16}	O	O	-	-	-	-	-	-	-	-	-	-	O	O	-	Output the thumbnail data of the specified file to subunit source plug
VERSION	$B0_{16}$	-	M	-	-	M	-	-	M	-	-	M	-	-	M	-	Report the supported spec version and profile
VOLUME INFO	41_{16}	M	M	-	M	M	-	M	M	-	M	M	-	O	O	-	Report the volume information
PROXY INFO	48_{16}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Report the proxy function information

Opcode	value	defined ctypes						comments
		asych proxy sender			limited asynch proxy sender			
		C	S	N	**C**	**S**	**N**	
CREATE DIRECTORY	57_{16}	O	O	-	O	O	-	Create new directory or delete existing directory
ERASE FILE	53_{16}	O	O	-	O	O	-	Erase the specified file
FILE INFO	46_{16}	M	M	-	O	O	-	Report the file specific information of the specified file
FILE LIST	43_{16}	M	M	-	O	O	-	Report the file list
FILE SIGNAL FORMAT	47_{16}	-	-	-	-	-	-	Report the available output signal format of the specified file
FORMAT	56_{16}	O	O	-	O	O	-	Format the specified volume
MEDIA INFO	40_{16}	M	M	-	O	O	-	Report the media information
NUMBER OF ENTRIES	42_{16}	M	M	-	O	O	-	Report the number of directory entries in the specified path
NUMBER OF OBJECT FILES	44_{16}	O	O	-	O	O	-	Report the number of files which compose DCF object
OBJECT FILE LIST	45_{16}	O	O	-	O	O	-	Report the file list of the files which compose the DCF object contains a specified file
PLAY FILE	$C3_{16}$	-	-	-	-	-	-	Play specified file and output the stream data to subunit source plug
RECEIVE FILE	52_{16}	-	-	-	-	-	-	Input the file data from subunit destination plug and store in the storage media
RECEIVE FILE PARTIAL	55_{16}	-	-	-	-	-	-	Input the data from subunit destination plug and replace existing file data
RECORD FILE	$C2_{16}$	-	-	-	-	-	-	Input the stream data from subunit destination plug and store in the storage media
SEND FILE	50_{16}	M	M	-	M	M	-	Output the specified file to subunit source plug
SEND FILE PARTIAL	54_{16}	O	O	-	M	M	-	Output the part of specified file to subunit source plug
SEND THUMBNAIL	51_{16}	O	O	-	O	O	-	Output the thumbnail data of the specified file to subunit source plug
VERSION	$B0_{16}$	-	M	-	-	M	-	Report the supported spec version and profile
VOLUME INFO	41_{16}	M	M	-	O	O	-	Report the volume information
PROXY INFO	48_{16}	-	M		-	M		Report the proxy function information

In the preceding tables, a dash in the support level column indicates that a command is not defined for the indicated *ctype* value, CONTROL, STATUS, or NOTIFY.

7.3 Common frame header

The control commands defined for the camera storage subunit have the same “common frame header” in the command and response frame. This header format for control command is shown in Figure 7-1 below:

	command format	response format
	msb … lsb	msb … lsb
opcode	Opcode	<=
operand[0]	Subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	media_generation_count	media_generation_count

NOTE “<=” means “same value as command format”.

Figure 7-1 – Common frame header format for control command

The *subfunction* field specifies the operation mode of the control command. If the camera storage subunit receives a command frame with an invalid subfunction value, then the “NOT IMPLEMENTED” response shall be returned to the controller. The control commands except PLAY FILE and RECORD FILE commands have the same subfunction values shown in Table 7-2 below. The subfunction values for PLAY FILE and RECORD FILE commands are shown in each command section.

Table 7-2 – *subfunction* values of control command

subfunction	Value	Meaning
execute	00_{16}	execute the control command
abort	01_{16}	abort the command being executed
resume	02_{16}	resume from bus reset
reserved	all other values	reserved for future extension

The “abort” subfunction is optional. If RECEIVE FILE, RECEIVE FILE PARTIAL, SEND FILE, SEND FILE PARTIAL or SEND THUMBNAIL commands are implemented, the “abort” subfunction for the command shall be implemented. In the case of other commands, the camera storage subunit may not implement the “abort” subfunction. By issuing specific inquiry command, the controller can determine whether or not the ”abort” subfunction is implemented. The format of specific inquiry command is described in the section pertaining to each command.

The “resume” subfunction is used only for the RECEIVE FILE, RECEIVE FILE PARTIAL, SEND FILE, SEND FILE PARTIAL and SEND THUMBNAIL commands. If these commands are implemented, the “resume” subfunction for the command shall be implemented.

The *result* field in the response frame indicates the result status shown in Table 5-3. When the camera storage subunit returns the INTERIM response, this field shall have the same control command value.

The *physical_volume_number* and the *logical_volume_number* fields specify the volume to be accessed. When the controller issues a control command that does not use these fields, the controller shall set the value of these fields to FF_{16}. If the command frame specifies invalid volume number that does not exist in the subunit, the camera storage subunit returns ”REJECTED” or “NOT IMPLEMENTED”response.

1) If the command frame specifies the physical volume number that doesn't exist in the subunit, the "NOT IMPLEMENTED" response should be returned. For example, when the camera storage subunit has two memory slots and *physical_volume_number* is 02_{16} (this value specifies third physical volume), "NOT IMPLEMENTED" should be returned.

2) If the command frame specifies the logical volume number which the camera storage subunit doesn't support, the "NOT IMPLEMENTED" response should be returned. For example, when the camera storage subunit supports only one logical volume (*logical_volume_number* = 0) and specified *logical_volume_number* in the control command is 01_{16} (this value specifies second logical volume), "NOT IMPLEMENTED" should be returned.

3) If the command frame specifies the physical volume number which memory slot has no media, the "REJECTED" response with result code of "no media" should be returned.

4) If the command frame specifies the logical volume number that doesn't exist in the physical volume, the "REJECTED" response with result code of "invalid volume number" should be returned.

The *media_generation_count* field in the command frame specifies the generation counter value.

When the controller issues a control command other than the MEDIA INFO and VOLUME INFO commands, the controller shall set the *media_generation_count* value held by the controller holds. The camera storage subunit will compare this to the current value held by the subunit and if the value is not equal to the value held by the subunit, then the camera storage subunit returns REJECTED response with a result code of "invalid generation count."

When the controller issues MEDIA INFO or VOLUME INFO commands, the controller shall set the value of FF_{16} to the *media_generation_count* field.

The camera storage subunit shall set the current generation counter value to the *media_generation_count* field in the response frame.

Before the controller issues the first control command to accesses the specific volume, the controller should issue VOLUME INFO control command to retrieve the current *media_generation_count* value.

7.4 Rule for control command and INTERIM response

All of the camera storage subunit control commands may return the INTERIM response to the controller. When the camera storage subunit returns INTERIM response, the response frame shall have the same field format and same field values as the control command.

When the controller receives the INTERIM response, the controller shall not retry the command before receiving a final response. The "retry" means sending the control command with the same opcode and same field values as the command issued previously. If the camera storage subunit detects the retried control command, the REJECTED response should be sent to the controller with a result code of "retry". After receiving the INTERIM response, the controller can determine whether or not the specific control command is being executed by issuing the status command. If the controller detects, by issuing a status command, that the control command previously issued has expired before the final response could be received, then the controller may retry the same control command. For detailed information on status command, please refer to section 7.6.

If the camera storage subunit receives a control command while executing another control command (INTERIM response has been sent and final response has not yet been sent), then the camera storage subunit may return the REJECTED response with a result code of "busy". Even if there are multiple

subunit plugs, the camera storage subunit may reject RECEIVE FILE, RECEIVE FILE PARTIAL, SEND FILE, SEND FILE PARTIAL or SEND THUMBNAIL control commands while the subunit is executing another command. The camera storage subunit may accept only one control command at a time; processing multiple commands simultaneously is an optional feature. So the controller shall not assume that the camera storage subunit can execute multiple commands simultaneously. The controller should wait for the final response, and the next control command should be issued after receiving the previous control command's final response. The exception is the control command with the subfunction value of "abort". If the camera storage subunit receives a control command with the subfunction value of "abort," and a control command with the same opcode and field values is being executed, then the control command with the "abort" subfunction should be accepted. If the camera storage subunit supports the "abort" subfunction, the control command with the subfunction value of "abort" should be processed by the camera storage subunit at any time.

7.5 Control command process

This section describes processing of control commands except PLAY FILE and RECORD FILE commands. The processing ofPLAY FILE and RECORD FILE commands are described in each command section. Figure 7-2 illustrates the camera storage subunit control command process.

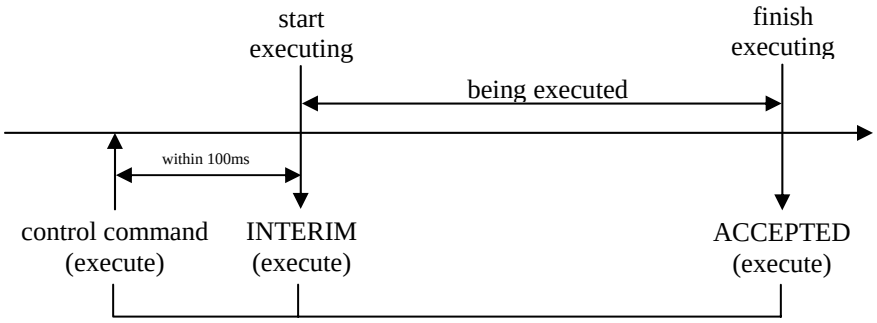


Figure 7-2 – Control command process

The ACCEPTED response shall be sent when execution is complete and when the subunit can accept next control command. If the camera storage subunit can finish executing within 100ms, the camera storage subunit may send ACCEPTED response to the controller without INTERIM response.

The camera storage subunit control command with the subfunction value of "abort" or "resume" consists of the same fields as control command with the subfunction value of "execute." Except for the subfunction field, the field values shall be identical to those of the control command with subfunction value of "execute" issued previously.

When the controller wants the camera storage subunit to abort execution, the controller may issue a control command with the subfunction "abort." Figure 7-3 illustrates the abort process.

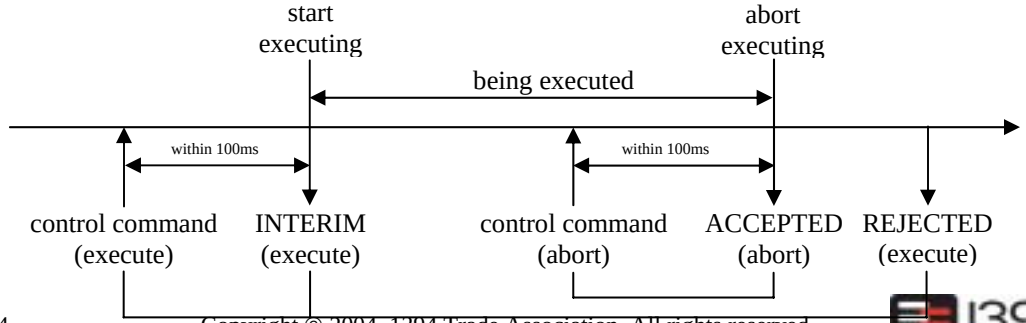


Figure 7-3 – Process of aborting execution

When the camera storage subunit receives the control command with the “abort” subfunction while the subunit is executing a control command with the same opcode and field values (except subfunction), then the subunit should abort execution. Any controller can abort the control command, unless the subunit is reserved by the RESERVE control command described in [R7]; but the controller should not abort an execution requested by another controller. The ACCEPTED response for the “abort” subfunction shall be sent within 100ms, and the REJECTED response with the result code “aborted” for the “execute” subfunction issued previously shall follow. If the camera storage subunit sends the ACCEPTED response for the “abort” subfunction, then the camera storage subunit shall immediately abort the execution of the previously issued control command, and the REJECTED response for the “execute” subfunction shall be sent when the camera storage subunit finishes aborting the process. If the abort subfunction is rejected, then the camera storage subunit shall continue to execute the command. The camera storage subunit shall not send the INTERIM response for a control command with the abort subfunction.

If the camera storage subunit receives an abort request when execution finished but the ACCEPTED response has not yet been sent, then the REJECTED response for the “abort” subfunction should be sent first, followed by the ACCEPTED response for the “execute” subfunction.

When a bus reset occurs, the command executions shall be aborted (except for RECEIVE FILE, SEND FILE and SEND THUMBNAIL control commands). In the cases of RECEIVE FILE, RECEIVE FILE PARTIAL, SEND FILE, SEND FILE PARTIAL and SEND THUMBNAIL control commands, the controller may send the same control command with the subfunction of “resume” after re-establishing the connection. Figure 7-4 illustrates the process for resuming after bus reset.

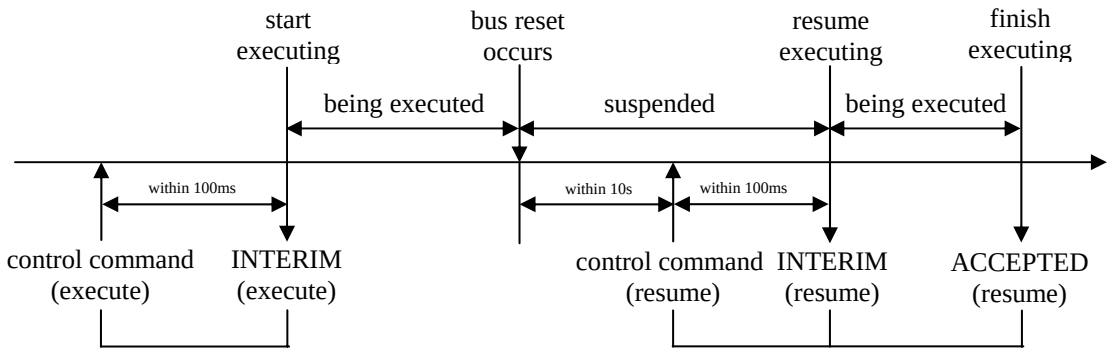


Figure 7-4 – process for resuming after bus reset

The ACCEPTED response of the “resume” subfunction shall be sent to the controller after execution is finished.

In case of RECEIVE FILE, RECEIVE FILE PARTIAL, SEND FILE, SEND FILE PARTIAL and SEND THUMBNAIL control commands, the camera storage subunit should suspend command execution after bus reset and should restart executing upon receiving the control command with the “resume” subfunction. When the camera storage subunit receives the control command with the “resume” subfunction while the subunit is suspending the control command with the same opcode and field values (except subfunction), then the subunit should resume execution. If the control command with the “resume” subfunction is not received within 10 seconds after bus reset, the camera storage subunit shall abort the command execution. For detailed information on command and data flow upon resuming from bus reset, please refer to Annex A.2.

If there is no command corresponding to the control command with the “abort” or “resume” subfunctions, then the camera storage subunit shall send the REJECTED response with a result code of “no command”.

7.6 Status command and specific inquiry command

7.6.1 Status command with the common frame header

The status command is also defined for all commands where the control command frame has a common frame header. There are two types of camera storage subunit status commands. One is for querying command execution status and the other is to query the subunit plug status. The status command except for PLAY FILE and RECORD FILE commands should be used to determine whether the control command that had an INTERIM response is being executed or not. So, the controller should issue the status command except for PLAY FILE and RECORD FILE after the INTERIM response is received and before the final response arrives.

The status command for the commands which control command frame has a common frame header (except for the PLAY FILE, RECORD FILE, RECEIVE FILE, RECEIVE FILE PARTIAL, SEND FILE, SEND FILE PARTIAL and SEND THUMBNAIL commands) is used to query the command execution status. The status command and STABLE response formats are illustrated in Figure 7-5 below. The status commands for the PLAY FILE, RECORD FILE, RECEIVE FILE, RECEIVE FILE PARTIAL, SEND FILE, SEND FILE PARTIAL and SEND THUMBNAIL commands are described in the section devoted to each command.

	command format	response format
	msb … lsb	msb … lsb
opcode	opcode	<=
operand[0]	FF_{16}	result

Figure 7-5 – Status command and STABLE response format

The *result* field in the response frame indicates the status of the command execution. The values for this field are described in section 5.5. If the camera storage subunit is executing the control command have an opcode identical to that of the status command frame, then the *result* field shall have the value of “executing.” When the control command specified by the status command is being aborted (the “abort” subfunction is accepted and the final response for “execute” is not yet sent), this field shall have the value of “aborting”; otherwise, this field shall have the value of “not executing.” Because the camera storage subunit only checks the opcode field, if the response frame has a result code of “executing,” then the control command being processed may be issued by another controller. The response frame with the result code of “executing” sent to the controller does not ensure that the current control command being executed was issued by the controller. Therefore, the controller should essentially manage the status of control command execution by corresponding INTERIM response and final response.

If the optional control command is supported, then the corresponding status command shall be supported, too.

The camera storage subunit should accept the status commands at any time.

7.6.2 Field validation checks and the specific inquiry command

In each control command section of the camera storage subunit specification is a table that describes field validation checks performed by the camera storage subunit on SPECIFIC INQUIRY and CONTROL

command frames. When receiving these frames, the camera storage subunit shall validate the operand and opcode fields indicated by the table to determine whether it implements a specific CONTROL command. The camera storage subunit shall return an IMPLEMENTED or NOT IMPLEMENTED response code as a result of this check.

The following table illustrates an example of field validation checks table. A "√" in the ck (check) column indicates a field that should be validated. A "-" in the column indicates a field that does not need to be validated. For fields that do not need to be checked, a recommended value for a specific inquiry command frame is also given. A controller should set the recommended values in a Specific Inquiry command frame instead of the actual value that would be set in the corresponding control command.

Table 7-3 – Example of field validation checks table

fields	ck	recommended values
field A	√	
field B	-	00_{16}
field C	-	00_{16}

7.7 CREATE DIRECTORY command

The CREATE DIRECTORY command is used to create new directory on the storage media or remove existing directory from the storage media.

7.7.1 CREATE DIRECTORY control command

The formats of the CREATE DIRECTORY control command and response are illustrated in Figure 7-6 below.

	command format								response format							
	msb							lsb	msb							Lsb
opcode	CREATE DIRECTORY (57_{16})								<=							
operand[0]	subfunction								<=							
operand[1]	FF_{16}								result							
operand[2]	physical_volume_number								<=							
operand[3]	logical_volume_number								<=							
operand[4]	media_generation_count								media_generation_count							
operand[5]	create_mode								<=							
operand[6]	reserved								<=							
operand[7]																
operand[8]	directory_path_length								<=							
operand[9]	directory_path								<=							
:																
:																

Figure 7-6 – CREATE DIRECTORY control command and response format

The *create_mode* field specifies the mode of creation. The values of this field are shown in Table 7-4 below.

Table 7-4 – *create_mode* values of the CREATE DIRECTORY command

create_mode	value	meaning
create	00_{16}	Create new directory on the storage media
remove	01_{16}	Remove existing directory from the storage media
reserved	all others	reserved for future extension

The *directory_path_length* field specifies the size of the *directory_path* field in bytes. The length does not include the NULL character.

The *directory_path* field specifies the path name of a directory to be created or removed. The path name is specified by full path name from the root directory. It is specified by ASCII characters terminated by NULL (00_{16}) and begins with the "\" character (root directory).

If *create_mode* is "create" and parent directory of specified *directory_path* doesn't exist, the camera storage subunit may return REJECTED response with the result code of "invalid parameter."

7.7.2 CREATE DIRECTORY field validation checks

The CREATE DIRECTORY field validation checks table is shown in Table 7-5 below.

Table 7-5 – CREATE DIRECTORY field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
create_mode	√	
directory_path_length	-	00_{16}
directory_path	-	00_{16}

7.7.3 Result codes for CREATE DIRECTORY command

The possible result codes for CREATE DIRECTORY command are summarized in Table 7-6 below.

Table 7-6 – Result codes for CREATE DIRECTORY command

ACCEPTED	STABLE	REJECTED
Success	not executing executing	busy retry disabled invalid generation count invalid volume number invalid parameter no media no directory unsupported format volume protected directory exist not empty directory protected unknown
	aborting	aborted no command

If a camera storage subunit implements the “abort” subfunction for CREATE DIRECTORY control command, then “aborting”, “aborted” and “no command” result codes should be supported.

7.8 ERASE FILE command

The ERASE FILE command is used to erase a specific file stored in the camera storage subunit.

7.8.1 ERASE FILE control command

The formats of the ERASE FILE control command and response are illustrated in Figure 7-7 below.

	command format	Response format
	msb lsb	msb lsb
opcode	ERASE FILE (53_{16})	<=
operand[0]	subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	media_generation_count	media_generation_count
operand[5]	reserved	<=
operand[6]		
operand[7]		
operand[8]		
operand[9]		
operand[10]		
operand[11]		
operand[12]	file_path_length	
operand[13]	file_path	
:		
:		

Figure 7-7 – ERASE FILE control command and response format

The *file_path_length* field specifies the size of the *file_path* field in bytes. The length does not include the NULL character.

The *file_path* field specifies the path name of the file to be erased. The path name is specified by ASCII characters terminated by a NULL (00_{16}) and begins with the "\" character (root directory).

When the control command is rejected, each field of the response frame contains the same value of the control command frame except for the *result* and *media_generation_count* field.

7.8.2 ERASE FILE field validation checks

The ERASE FILE field validation checks table is shown in Table 7-7 below.

Table 7-7 – ERASE FILE field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
file_path_length	-	00_{16}
file_path	-	00_{16}

7.8.3 Result codes for ERASE FILE command

The possible result codes for ERASE FILE command are summarized in Table 7-8 below.

Table 7-8 – Result codes for ERASE FILE command

ACCEPTED	STABLE	REJECTED
success	not executing executing	busy retry disabled invalid generation count invalid volume number no media no file unsupported format volume protected file protected unknown
	aborting	aborted no command

If a camera storage subunit implements the "abort" subfunction for ERASE FILE control command, then "aborting", "aborted" and "no command" result codes should be supported.

7.9 FILE INFO command

The FILE INFO command is used to retrieve file specific information.

7.9.1 FILE INFO control command

The formats of the FILE INFO control command and response are illustrated in Figure 7-8 below.

	command format	response format
	msb lsb	msb lsb
opcode	FILE INFO (46_{16})	<=
operand[0]	subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	media_generation_count	media_generation_count
operand[5]	FF FF_{16}	file_info_type
operand[6]		
operand[7]	reserved	<=
operand[8]		
operand[9]		
operand[10]		
operand[11]		
operand[12]	file_path_length	file_specific_information_length
operand[13]	file_path	file_specific_information
:		
:		

Figure 7-8 – FILE INFO control command and response format

The *file_info_type* field in the response frame indicates the type of *file_specific_infotmation*. The values of this field are shown in Table 7-10 below. If the specified file's file info type is not defined for this command or not supported by the camera storage subunit, then the REJECTED response with a result code of "invalid file type" shall be sent.

Table 7-10 – file_info_type values

file_info_type	value	meaning
Exif 2.1	0000_{16}	Exif 2.1 file
Vendor dependent	$FFFF_{16}$	Vendor dependent format file
reserved	all others	reserved for future extension

The *file_path_length* field specifies the size of the *file_path* field in bytes. The length does not include the NULL character.

The *file_path* field specifies the path name of the target file. The path name is specified by ASCII characters terminated by NULL (00_{16}) and begins with the "\" character (root directory).

The *file_specific_information_length* field in the response frame indicates the size of the *file_specific_information* field in the response frame in bytes. If the command is rejected, then this field value should be set to 0, and there would be no *file_specific_information* field in the response frame.

When the control command is rejected, the value of *file_info_type* field is set to $FFFF_{16}$.

7.9.2 file_specific_information format

The camera storage subunit returns information on a specified file by filling the *file_specific_information* field in the response frame. The format of this field is defined for each *file_info_type* value.

7.9.3 file_specific_information for Exif 2.1 file

The format of *file_specific_information* for "Exif 2.1 file" is shown in Figure 7-10 below.

offset	Contents
00_{16}	ImageWidth
01_{16}	
02_{16}	ImageLength
03_{16}	
04_{16}	Compression
05_{16}	
06_{16}	ColorSpace
07_{16}	
08_{16}	DateTime
:	
$1B_{16}$	

Figure 7-10 – Format of *file_specific_information* field for Exif 2.1 file

This information is based on Exif tags explained in reference [R11].

The *ImageWidth* and the *ImageLength* indicate the width and height of a specified image file. If the camera storage subunit cannot determine image size, then these fields shall have the value of zeroes.

The *Compression* indicates the compression scheme applied to the image data. The values of Compression are shown in Table 7-21 below. If the image data is compressed, but not in JPEG format, then "unknown" shall be assigned.

Table 7-2 – *Compression* values for Exif 2.1 file

Compression	value	Meaning
uncompressed	0001_{16}	image is uncompressed
JPEG	0006_{16}	JPEG compression
unknown	$FFFF_{16}$	unknown compression scheme
reserved	all others	reserved for future extension

The *ColorSpace* indicates color space information. The values of *ColorSpace* are shown in Table 7-3 below. If color space other than sRGB is used, Uncalibrated shall be assigned.

Table 7-3 – ColorSpace values for Exif 2.1 file

ColorSpace	value	Meaning
sRGB	0001_{16}	sRGB is used
Uncalibrated	$FFFF_{16}$	other than sRGB
reserved	all others	reserved for future extension

The *DateTime* indicates the date and time a file was changed. This field consists of ASCII characters with a length of 20 bytes, including NULL (00_{16}) for termination. The format is "YYYY:MM:DD HH:MM:SS";

time is shown in 24-hour format and the date and time are separated by one blank character (20_{16}). When the date and time are unknown, all the character spaces except colons (":") may be filled with blank characters.

7.9.3.1 file_specific_information for vendor dependent file

The format of *file_specific_information* for "Vendor dependent file" is shown in Figure 7-2 below.

offset	Contents
00_{16}	company_ID
01_{16}	
02_{16}	
03_{16}	vendor_dependent_data
:	
:	

Figure 7-2 – Format of *file_specific_information* field for vendor dependent file

The *company_ID* field indicates the 24-bit unique ID assigned by IEEE Registration Authority Committee (RAC).

The *vendor_dependent_data* filed indicates the information of the vendor dependent format file. Actual format and meaning are specified by the vendor identified by *company_ID*.

7.9.4 FILE INFO field validation checks

The FILE INFO field validation checks table is shown in Table 7-4 below.

Table 7-4 – FILE INFO field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
file_path_length	-	00_{16}
file_path	-	00_{16}

7.9.5 Result codes for FILE INFO command

The possible result codes for FILE INFO command are summarized in Table 7-5 below.

Table 7-5 – Result codes for FILE INFO command

ACCEPTED	STABLE	REJECTED
success	not executing executing	busy retry disabled invalid generation count invalid volume number no media no file unsupported format invalid file type unknown
	aborting	aborted no command

If the camera storage subunit implements “abort” subfunction for FILE INFO command, then “aborting”, “aborted” and “no command” may return.

7.10 FILE LIST command

The FILE LIST command is used to retrieve the file list. The file list is transferred by response frame of the control command or using asynchronous connections. Please refer to section 5.2 for details about the file list.

7.10.1 FILE LIST control command

The formats of the FILE LIST control command and response are illustrated in Figure 7-3 below. An example of field values and a procedure for querying the file list are provided in Annex A.1. Please refer to the section for more detail.

	command format								response format							
	msb							lsb	msb							lsb
opcode	FILE LIST (43_{16})								<=							
operand[0]	subfunction								<=							
operand[1]	FF_{16}								Result							
operand[2]	physical_volume_number								<=							
operand[3]	logical_volume_number								<=							
operand[4]	media_generation_count								media_generation_count							
operand[5]	file_type								<=							
operand[6]	list_mode						reserved	attribute	<=					<=		<=
operand[7]	start_number								<=							
operand[8]																
operand[9]	0	reserved			number_of_entries				eol	reserved			number_of_entries			
operand[10]	reserved								<=							
operand[11]																
operand[12]	request_path_length								<=							
operand[13]	request_path								file_list							
:																
:																

Figure 7-3 – FILE LIST control command and response format

The *file_type* field specifies the file type of directory entries to be retrieved. The values for this field are shown in Table 7-6 below.

Table 7-6 – *file_type* values of the FILE LIST command

file_type	value	Meaning
all	00_{16}	all type of files and directories
still image	01_{16}	still image file
-	all others	reserved for future extension

If the *file_type* is "all," then the target returns the directory entries of all kind of files and/or directories. If the *file_type* is "still image," then the target returns the directory entries of still image files. The target can handle any type of files as a still image file according to its implementation. At least, the target shall return the directory entries of DCF basic files.

The *attribute* field specifies the directory entry type to be retrieved. This field consists of bit fields as defined in Table 7-7 below.

Table 7-7 – *attribute* field bit assignment for the FILE LIST command

bit	description
bit 0	directory
bit 1(lsb)	file

If bit0 of the *attribute* field is set, the camera storage subunit returns the directory entries of subdirectories. If bit1 of the *attribute* field is set, the camera storage subunit returns the directory entries of files. If both bits are set and *file_type* is "all", the camera storage subunit returns the directory entries of files and subdirectories.

The combinations of *file_type* and *attribute* are limited and have different levels of support. The support levels are shown in Table 7-8 below. By issuing the FILE LIST specific inquiry command, the controller can determine whether or not the specific combination of *file_type* and *attribute* values is supported. If the command frame specifies an unsupported combination, then the camera storage subunit shall return the NOT IMPLEMENTED response. For detailed information on combination usage, please refer to Annex A.1.

Table 7-8 – Support levels by combination of *file_type* and *attribute*

file_type	attribute	support level
all (00_{16})	file (01_{16})	Optional
	directory (02_{16})	mandatory
	file and directory (03_{16})	mandatory
still image (01_{16})	file (01_{16})	mandatory
-	all others	not supported

The *list_mode* field specifies the format of directory entry in returned file list data. The values for this field are shown in Table 7-9 below.

Table 7-9 – *list_mode* values of the FILE LIST command

list_mode	value	Meaning
short entry	0_{16}	Directory entry of file list is defined as Figure 5-2
long entry	1_{16}	Directory entry of file list is defined as Figure 5-6
-	all others	reserved for future extension

When *list_mode* specifies "short entry", then camera storage subunit sets the data of directory entries defined as Figure 5-2 in the *file_list* field in the response frame.

When *list_mode* specifies "long entry", then camera storage subunit sets the data of directory entries defined as Figure 5-6 in the *file_list* field in the response frame.

The *start_number* field specifies the starting position of the directory entries to be retrieved. The value of *start_number* begins at 0. The target sends the directory entries based on *request_path*, *file_type* and *attribute* fields, but the number of directory entries in the response frame is limited, making a paging method necessary. The *start_number* field is used as the paging parameter.

The *eol* (end_of_list) bit in the response frame indicates whether or not the last directory entry in the response frame is the last entry in the directory entry list based on *request_path*. If this bit is 0, then the last entry in the response frame is not the last one in the directory entry list and the other entries follow in the list. If this bit is 1, the last entry in the response frame is the last one in the directory entry list.

The *number_of_entries* field in the command frame specifies the number of directory entries to retrieve.

The *number_of_entries* field in the response frame indicates the actual number of directory entries in the response frame. The *number_of_entries* in the response frame shall be less than or equal to the *number_of_entries* in the command frame. Since each directory entry takes greater than or equal to 20 bytes, the maximum value of *number_of_entries* field is 24.

The *request_path_length* field specifies the size of the *request_path* field in bytes. The length does not include the NULL character.

The *request_path* field specifies the path name of either a directory or a file and the camera storage subunit shall determine whether this field specifies a directory or a file. The format of *request_path* is identical to the *file_path* described in section 5.1. The *request_path* field conforms to the path and file name parameters of the DIR command being used with the DOS file system. If the *request_path* specifies a directory path name and the directory exists, the camera storage subunit returns directory entries within the specified directory. If the *request_path* specifies a file path name and there are files matching the *request_path*, then the camera storage subunit returns the directory entries of those files.

When the controller specifies the file path name, the *request_path* consists of path of directory and file name. The controller may use the wildcard character ("*") for name portion of the file name (corresponding to the *name* field of the directory entry shown in Figure 5-2) or extension portion of the file name (corresponding to the *extension* field of the directory entry shown in Figure 5-2). The function of the wildcard character is the same as in the DOS file system. But, the wildcard is optional and the camera storage subunit may return the REJECTED response with a result code of "invalid parameter" to the controller when the wildcard character is used.

If the fully-specified path does not exist, the camera storage subunit returns the REJECTED response with a result code of "no directory" to the controller even though it is a file path.

The *file_list* field in the response frame is variable in length and holds the directory entries described in section 5.2. The size of each entry is 20 bytes for "short entry" or variable length for "long entry." The number of entries in the *file_list* field is indicated by the *number_of_entries* field in the response frame. If

the specified directory path is found, but there is no entry to be returned, then the control command is accepted, the *number_of_entries* field is set to 0, and the *file_list* field does not exist.

When the control command is rejected, each field of the response frame contains the same value of the control command frame except for the *result* and *media_generation_count* field.

7.10.2 FILE LIST field validation checks

The FILE LIST field validation checks table is shown in Table 7-10 below.

Table 7-10 – FILE LIST field validation checks table

Fields	ck	Recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
file_type	√	
list_mode	√	
attribute	√	
start_number	-	00 00_{16}
number_of_entries	-	00_{16}
request_path_length	-	00_{16}
request_path	-	00_{16}

7.10.3 Result codes for FILE LIST command

The possible result codes for FILE LIST command are summarized in Table 7-11 below.

Table 7-11 – Result code for FILE LIST command

ACCEPTED	STABLE	REJECTED
success	not executing executing	busy retry disabled invalid generation count invalid volume number invalid parameter no media no directory unsupported format unknown
	aborting	aborted no command

If the camera storage subunit implements “abort” subfunction for FILE LIST command, then “aborting”, “aborted” and “no command” may return.

7.11 FILE SIGNAL FORMAT command

The FILE SIGNAL FORMAT command is used to retrieve available output signal format of the specified file.

7.11.1 FILE SIGNAL FORMAT control command

The formats of the FILE SIGNAL FORMAT control command and response are illustrated in Figure 7-12 below.

	command format	response format
	msb lsb	msb Lsb
opcode	FILE SIGNAL FORMAT (47_{16})	<=
operand[0]	Subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	media_generation_count	media_generation_count
operand[5]	Reserved	<=
operand[6]		
operand[7]		
operand[8]	file_path_length	number_of_formats
operand[9]	file_path	signal_formats
:		
:		

Figure 7-12 – FILE SIGNAL FORMAT control command and response format

The *file_path_length* field specifies the size of the *file_path* field in bytes. The length does not include the NULL character.

The *file_path* field specifies the path name of a DCF file. The path name is specified by ASCII characters terminated by NULL (00_{16}) and begins with the "\" character (root directory).

The *number_of_formats* field in the response frame indicates the number of signal formats in the response frame.

The *signal_formats* field in the response frame is variable in length and holds the available output signal formats of the file specified by *file_path* field in the command frame. The number of signal formats in the *signal_formats* field is indicated by the *number_of_formats* field in the response frame. The values of this field is shown in Table 7-12 below.

Table 7-12 – *signal_formats* values for response frame

value	signal mode
00_{16}	SD 525-60
04_{16}	SDL 525-60
08_{16}	HD 1125-60
80_{16}	SD 625-50
84_{16}	SDL 625-50
88_{16}	HD 1250-50
10_{16}	MPEG 25Mbps-60
14_{16}	MPEG 12.5Mbps-60
18_{16}	MPEG 6.25Mbps-60
90_{16}	MPEG 25Mbps-50
94_{16}	MPEG 12.5Mbps-50
98_{16}	MPEG 6.25Mbps-50
01_{16}	D-VHS Digital
FE_{16}	no output

If camera storage can play the specified file but can't output as stream data, then the value of "no output" is set to *signal_formats* field and the value of *number_of_formats* is 01_{16}. If camera storage can't play the specified file, then the value of *number_of_formats* is 00_{16} and *signal_formats* field doesn't exist in the response frame.

7.11.2 FILE SIGNAL FORMAT field validation checks

The FILE SIGNAL FORMAT field validation checks table is shown in Table 7-13 below.

Table 7-13 – FILE SIGNAL FORMAT field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
file_path_length	-	00_{16}
file_path	-	00_{16}

7.11.3 Result codes for FILE SIGNAL FORMAT command

The possible result codes for FILE SIGNAL FORMAT command are summarized in Table 7-14 below.

Table 7-14 – Result codes for FILE SIGNAL FORMAT command

ACCEPTED	STABLE	REJECTED
Success	not executing executing	busy retry disabled invalid generation count invalid volume number invalid parameter no media no file unsupported format unknown
	aborting	aborted no command

If the camera storage subunit implements “abort” subfunction for FILE SIGNAL FORMAT command, then “aborting”, “aborted” and “no command” may return.

7.12 FORMAT command

The FORMAT command is used to format the specified volume.

7.12.1 FORMAT control command

The formats of the FORMAT control command and response are illustrated in Figure 7-13 below.

	command format	response format
	msb … lsb	msb … lsb
opcode	FORMAT (56_{16})	<=
operand[0]	subfunction	<=
operand[1]	FF_{16}	Result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	media_generation_count	media_generation_count
operand[5]	format_type	<=
operand[6]	reserved	<=
operand[7]		
operand[8]		

Figure 7-13 – FORMAT control command and response format

The *format_type* field specifies the type of format. The values of this field is shown in Table 7-15 below.

Table 7-15 – *format_type* values of the FORMAT command

format_type	value	Meaning
default	00_{16}	format the storage media by device default setting
-	all others	reserved for future extension

7.12.2 FORMAT field validation checks

The FOAMAT field validation checks table is shown in Table 7-16 below.

Table 7-16 – FORMAT field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
format_type	√	

7.12.3 Result codes for FORMAT command

The possible result codes for FORMAT command are summarized in Table 7-17 below.

Table 7-17 – Result code for FORMAT command

ACCEPTED	STABLE	REJECTED
Success	not executing executing	busy retry disabled invalid generation count invalid volume number no media volume protected unknown
	aborting	aborted no command

If the camera storage subunit implements “abort” subfunction for FORMAT command, then “aborting”, “aborted” and “no command” may return.

7.13 MEDIA INFO command

The MEDIA INFO command is used to retrieve information about storage media inside the camera storage subunit.

7.13.1 MEDIA INFO control command

The formats of the MEDIA INFO control command and response are illustrated in Figure 7-4 below.

	command format	response format
	msb lsb	msb lsb
opcode	MEDIA INFO (40_{16})	<=
operand[0]	subfunction	<=
operand[1]	FF_{16}	Result
operand[2]	FF FF FF_{16}	<=
operand[3]		
operand[4]		
operand[5]		number_of_physical_volume (n)
operand[6]		number_of_logical_volume[0]
:		:
operand[n+5]		number_of_logical_volume[n-1]

Figure 7-4 – MEDIA INFO control command and response format

The *number_of_physical_volume* field in the response frame indicates the number of physical volumes inside the camera storage subunit. For example, the camera storage subunit has two memory slots, so this field holds the value of 02_{16}. Even if the slot is empty, the slot shall be counted as a physical volume.

The *number_of_logical_volume* fields indicate the number of logical volumes in each physical volume. If a memory slot is empty, then the corresponding field shall hold the value of 00_{16}.

If the camera storage subunit can retrieve the information of specified volume within 100ms, the ACCEPTED response may be returned immediately after the MEDIA INFO command.

7.13.2 MEDIA INFO field validation checks

The MEDIA INFO field validation checks table is shown in Table 7-18 below.

Table 7-18 – MEDIA INFO field validation checks table

fields	ck	recommended values
subfunction	√	

7.13.3 Result codes for MEDIA INFO command

The possible result codes for MEDIA INFO command are summarized in Table 7-19 below.

Table 7-19 – Result code for MEDIA INFO command

ACCEPTED	STABLE	REJECTED
success	not executing executing	busy retry disabled unknown
	aborting	aborted no command

If the camera storage subunit implements “abort” subfunction for MEDIA INFO command, then “aborting”, “aborted” and “no command” may return.

7.14 NUMBER OF ENTRIES command

The NUMBER OF ENTRIES command is used to retrieve the number of directory entries in the specified path.

7.14.1 NUMBER OF ENTRIES control command

The formats of the NUMBER OF ENTRIES control command and response are illustrated in Figure 7-5 below.

	command format		response format	
	msb lsb		msb Lsb	
opcode	NUMBER OF ENTRIES (42_{16})		<=	
operand[0]	Subfunction		<=	
operand[1]	FF_{16}		Result	
operand[2]	physical_volume_number		<=	
operand[3]	logical_volume_number		<=	
operand[4]	media_generation_count		media_generation_count	
operand[5]	file_type		<=	
operand[6]	reserved	attribute	<=	<=
operand[7]	FF FF_{16}		number_of_entries	
operand[8]				
operand[9]	Reserved			
operand[10]				
operand[11]				
operand[12]	request_path_length			
operand[13]	request_path			
:				
:				

Figure 7-5 – NUMBER OF ENTRIES control command and response format

The *file_type* field specifies the file type of directory entries to be counted. The values and the meanings are identical to those of the FILE LIST control command described in section 7.10.1.

The *attribute* field consists of bit fields and specifies the directory entry type to be counted. The bit assignment of this field and meanings are identical to those of the FILE LIST control command described in section 7.10.1.

The *request_path_length* field specifies the size of the *request_path* field in bytes. The length does not include the NULL character.

The *request_path* field specifies the path name of a directory or file. The format and usage of this field are identical to those of the FILE LIST control command described in section 7.10.1.

The *number_of_entries* field in the response frame indicates the number of directory entries based on the *request_path*, *file_type* and *attribute* fields.

When the control command is rejected, each field of the response frame contains the same value of the control command frame except for the *result* and *media_generation_count* field.

7.14.2 NUMBER OF ENTRIES field validation checks

The NUMBER OF ENTRIES field validation checks table is shown in Table 7-20 below.

Table 7-20 – NUMBER OF ENTRIES field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
file_type	√	
attribute	√	
request_path_length	-	00_{16}
request_path	-	00_{16}

7.14.3 Result codes for NUMBER OF ENTRIES command

The possible result codes for NUMBER OF ENTRIES command are summarized in Table 7-21 below.

Table 7-21 – Result code for NUMBER OF ENTRIES command

ACCEPTED	STABLE	REJECTED
Success	not executing executing	busy retry disabled invalid generation count invalid volume number no media no directory unsupported format unknown
	aborting	aborted no command

If the camera storage subunit implements “abort” subfunction for NUMBER OF ENTRIES command, then “aborting”, “aborted” and “no command” may return.

7.15 NUMBER OF OBJECT FILES command

The NUMBER OF OBJECT FILES command is used to retrieve the number of files composing a DCF object.

7.15.1 NUMBER OF OBJECT FILES control command

The formats of the NUMBER OF OBJECT FILES control command and response are illustrated in Figure 7-6 below.

	command format	response format
	msb lsb	msb lsb
opcode	NUMBER OF OBJECT FILES (44_{16})	<=
operand[0]	subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	media_generation_count	media_generation_count
operand[5]	reserved	<=
operand[6]		
operand[7]	FF FF_{16}	number_of_files
operand[8]		
operand[9]	reserved	
operand[10]		
operand[11]		
operand[12]	file_path_length	
operand[13]	file_path	
:		
:		

Figure 7-6 – NUMBER OF OBJECT FILES control command and response format

The *file_path_length* field specifies the size of the *file_path* field in bytes. The length does not include the NULL character.

The *file_path* field specifies the path name of a DCF file. The path name is specified by ASCII characters terminated by NULL (00_{16}) and begins with the "\" character (root directory). The camera storage subunit returns the number of files which compose the DCF object include the file specified by this path name.

The *number_of_files* field in the response frame indicates the number of files which compose the DCF object. This value shall include the file specified by *file_path* field.

When the control command is rejected, each field of the response frame contains the same value of the control command frame except for the result and *media_generation* field.

7.15.2 NUMBER OF OBJECT FILES field validation checks

The NUMBER OF OBJECT FILES field validation checks table is shown in Table 7-22 below.

Table 7-22 – NUMBER OF OBJECT FILES field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
Media_generation_count	-	00_{16}
file_path_length	-	00_{16}
file_path	-	00_{16}

7.15.3 Result codes for NUMBER OF OBJECT FILES command

The possible result codes for NUMBER OF OBJECT FILES command are summarized in Table 7-23 below.

Table 7-23 – Result codes for NUMBER OF OBJECT FILES command

ACCEPTED	STABLE	REJECTED
Success	not executing executing	busy retry disabled invalid generation count invalid volume number no media no file unsupported format unknown
	aborting	aborted no command

If the camera storage subunit implements “abort” subfunction for NUMBER OF OBJECT FILES command, then “aborting”, “aborted” and “no command” may return.

7.16 OBJECT FILE LIST command

The OBJECT FILE LIST command is used to retrieve the file list of the files that compose the DCF object containing the specified file. The file list is transferred by response frame of the control command.

7.16.1 OBJECT FILE LIST control command

The formats of the OBJECT FILE LIST control command and response are illustrated in Figure 7-16 below.

	command format								response format							
	msb							lsb	msb							lsb
opcode	OBJECT FILE LIST (45_{16})								<=							
operand[0]	subfunction								<=							
operand[1]	FF_{16}								result							
operand[2]	physical_volume_number								<=							
operand[3]	logical_volume_number								<=							
operand[4]	media_generation_count								media_generation_count							
operand[5]	reserved								<=							
operand[6]																
operand[7]	start_number								<=							
operand[8]																
operand[9]	0	reserved				number_of_entries			eol	reserved				Number_of_entries		
operand[10]	reserved								<=							
operand[11]																
operand[12]	file_path_length								<=							
operand[13]	file_path								file_list							
:																
:																

Figure 7-16 – OBJECT FILE LIST control command and response format

The *start_number* field specifies the starting position of the directory entry in the directory entry list of target DCF object files. Usage of this field is identical to that of the FILE LIST command.

The *eol* (end_of_list) bit in the response frame indicates whether the last directory entry in the response frame is the last entry in the directory entry list of the requested DCF object. If this bit is 0, then the last entry in the response frame is not the last one in the directory entry list and the other entries follow in the list. If this bit is 1, then the last entry in the response frame is the last one in the directory entry list.

The *number_of_entries* field in the command frame specifies the number of directory entries to retrieve. The *number_of_entries* field in the response frame indicates the actual number of directory entries in the response frame. The actual number of directory entries in the response frame may not be equal to the *number_of_entries* in the command frame, but it shall be less than or equal to the *number_of_entries* in the command frame.

The *file_path_length* field specifies the size of the *file_path* field in bytes. The length does not include the NULL character.

The *file_path* field specifies the path name of a DCF file. The path name is specified by ASCII characters terminated by NULL (00_{16}) and begins with the "\" character (root directory). The camera storage subunit returns the file list of the files that compose the DCF object and include the file specified by this path name.

The *file_list* field in the response frame is variable in length and holds the directory entries described in section 5.2. The size of each entry is 20 bytes. The number of entries in the *file_list* field is indicated by the *number_of_entries* field in the response frame. The file list of a DCF object shall include the file specified by the *file_path* field.

7.16.2 OBJECT FILE LIST field validation checks

The OBJECT FILE LIST field validation checks table is shown in Table 7-24 below.

Table 7-24 – OBJECT FILE LIST field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
start_number	-	00 00_{16}
number_of_entries	-	00_{16}
file_path_length	-	00_{16}
file_path	-	00_{16}

7.16.3 Result codes for OBJECT FILE LIST command

The possible result codes for OBJECT FILE LIST command are summarized in Table 7-23 below.

Table 7-25 – Result codes for OBJECT FILE LIST command

ACCEPTED	STABLE	REJECTED
success	not executing executing	Busy retry disabled invalid generation count invalid volume number invalid parameter no media no file unsupported format unknown
	aborting	Aborted no command

If the camera storage subunit implements “abort” subfunction for OBJECT FILE LIST command, then “aborting”, “aborted” and “no command” may return.

7.17 PLAY FILE command

The PLAY FILE command is used to request the camera storage subunit to play the specified file and output the stream data to the specified source plug.

7.17.1 PLAY FILE control command

There are two types of the format of PLAY FILE control command.

7.17.1.1 PLAY FILE control command with file_path

This type of control command specifies a path and starting position of target file to be played. The formats of the PLAY FILE control command with *file_path* is illustrated in Figure 7-7 below.

	command format								response format							
	msb							lsb	msb							lsb
opcode	PLAY FILE ($C3_{16}$)								<=							
operand[0]	Subfunction								<=							
operand[1]	FF_{16}								result							
operand[2]	physical_volume_number								<=							
operand[3]	logical_volume_number								<=							
operand[4]	Media_generation_count								media_generation_count							
operand[5]	start_position								<=							
operand[6]																
operand[7]																
operand[8]																
operand[9]	Duration								<=							
operand[10]																
operand[11]																
operand[12]																
operand[13]	source_plug								source_plug							
operand[14]	start_position_unit							duration_unit	<=							<=
operand[15]	Reserved								<=							
operand[16]	file_path_length								<=							
operand[17]	file_path								<=							
:																
:																

Figure 7-7 – PLAY FILE control command and response format with *file_path* field

The *subfunction* field specifies the playback mode of PLAY FILE command. The values of this field are shown in Table 7-26 below.

Table 7-26 – *subfunction* values of the PLAY FILE command

playback mode	value	support level	description
STOP	00_{16}	M	Stop playback operation
NEXT FRAME	30_{16}	O	Playback the next sequential frame or field
SLOWEST FORWARD	31_{16}	O	Playback at a special effect speed described in detail below
SLOW FORWARD 6	32_{16}	O	
SLOW FORWARD 5	33_{16}	O	
SLOW FORWARD 4	34_{16}	O	
SLOW FORWARD 3	35_{16}	O	
SLOW FORWARD 2	36_{16}	O	
SLOW FORWARD 1	37_{16}	O	
X1	38_{16}	O	Playback at normal speed
FAST FORWARD 1	39_{16}	O	Playback at a special effect speed described in detail below
FAST FORWARD 2	$3A_{16}$	O	
FAST FORWARD 3	$3B_{16}$	O	
FAST FORWARD 4	$3C_{16}$	O	
FAST FORWARD 5	$3D_{16}$	O	
FAST FORWARD 6	$3E_{16}$	O	
FASTEST FORWARD	$3F_{16}$	O	
PREVIOUS FRAME	40_{16}	O	Playback the previous sequential frame or field
SLOWEST REVERSE	41_{16}	O	Playback in reverse at a special effect speed described in detail below
SLOW REVERSE 6	42_{16}	O	
SLOW REVERSE 5	43_{16}	O	
SLOW REVERSE 4	44_{16}	O	
SLOW REVERSE 3	45_{16}	O	
SLOW REVERSE 2	46_{16}	O	
SLOW REVERSE 1	47_{16}	O	
X1 REVERSE	48_{16}	O	Playback at normal speed in reverse
FAST REVERSE 1	49_{16}	O	Playback in reverse at a special effect speed described in detail below
FAST REVERSE 2	$4A_{16}$	O	
FAST REVERSE 3	$4B_{16}$	O	
FAST REVERSE 4	$4C_{16}$	O	
FAST REVERSE 5	$4D_{16}$	O	
FAST REVERSE 6	$4E_{16}$	O	
FASTEST REVERSE	$4F_{16}$	O	
REVERSE	65_{16}	O	Playback at normal speed in reverse
FORWARD	75_{16}	M	Playback at normal speed
PAUSE	7D	O	Pause in playback

The subunit support level for PLAY FILE command varies according to the playback mode requested.

Speed variations in either a forward or reverse playback direction are collectively referred to as trick play modes. All trick play modes are optional. However, if trick play modes are supported for the specified file, a camera storage subunit shall implement them as follows:

There are four groups of trick play modes: slow forward, fast forward, slow reverse and fast reverse. A camera storage subunit may implement each group independently.

a) If a camera storage subunit implements a trick play group, it shall implement the basic playback option, i.e., either SLOWEST or FASTEST in the direction implemented. PLAY FILE control commands with a *subfunction* that specifies a SLOW *n* or FAST *n* playback mode may be rejected by the camera storage subunit as NOT IMPLEMENTED. Optionally, the camera storage subunit may accept all of the SLOW *n* or FAST *n* playback modes and interpret as SLOWEST or FASTEST.

b) If a camera storage subunit implements more than one speed within a trick play group, it shall recognize all of the SLOW *n* or FAST *n* playback modes as well as the SLOWEST and FASTEST playback mode. A camera storage subunit is not required to implement all seven possible playback speeds; it is required only to map all possible playback modes within the trick play group to the speeds it does support. The actual speeds encoded by the playback modes shall be subject to one of the following restrictions, as appropriate:

SLOWEST <= SLOW 6 <= SLOW 5 <= SLOW 4 <= SLOW 3 <= SLOW 2 <= SLOW 1 <= X1

X1 <= FAST 1 <= FAST 2 <= FAST 3 <= FAST 4 <= FAST 5 <= FAST 6 <= FASTEST

If a camera storage implements trick play modes but can not play the specified file in specified implemented trick play mode, the camera storage subunit shall reject the command with result code of "invalid operation mode".

The *start_position_unit* field and *start_position* field specify the starting point from the top of the file to be played. The values of *start_position_unit* field are shown in Table 7-27 below.

Table 7-27 – *start_position_unit* values

start_position_unit	value	support level
timecode_1	00_{16}	O
timecode_2	01_{16}	O
current position	08_{16}	M
top	09_{16}	M
end	$0A_{16}$	O
reserved	all others	-

The format of *start_position* field for timecode_1 is shown in Figure 7-8 below.

offset	Contents							
	msb							lsb
00_{16}	Frame							
01_{16}	Second							
02_{16}	Minute							
03_{16}	Hour							

Figure 7-8 – Format of *start_position* field for timecode_1

The format of *start_position* field for Timecode_2 is shown in Figure 7-9 below.

offset	Contents							
	msb							lsb
00_{16}	Millisecond							
01_{16}			Second					
02_{16}	Minute							
03_{16}	Hour							

Figure 7-9 – Format of *start_position* field for timecode_2

If the controller wants to play the file from current position, top of the file or end of the file, the controller can set "current position"(08_{16}), "top"(09_{16}) or "end"($0A_{16}$) to *start_position_unit* and *start_position* field is ignored by camera storage subunit.

If the *start_position* field specifies the position exceed the length of the file or specifies impossible position, the camera storage subunit has following options:

1) reject the command with result code of "invalid parameter"

2) accept the command with result code of "default position" and start playing from top of the file

The *duration_unit* field and *duration* field specify the duration of playing specified file. The values of *duration_unit* field are shown in Table 7-28 below.

Table 7-28 – *duration_unit* values

duration_unit	value	support level
timecode_1	00_{16}	O
timecode_2	01_{16}	O
unspecify	$0F_{16}$	M
reserved	all others	-

The formats of *duration* field for timecode_1 and timecode_2 are same as *start_position* field for timecode_1 and timecode_2.

If the controller specifies "unspecify"($0F_{16}$), *duration* field is ignored by camera storage subunit and the duration of playing depends on the implementation of the camera storage subunit.

If specified file has duration of the time and specified *duration* exceeds the length of specified file, camera storage subunit may stop playing at the end of specified file, or if there is NEXT file to play, the subunit may start playing the NEXT file.

The *source_plug* field in the command frame specifies which subunit source plug will be used to output the stream data. The values of *source_plug* field in the command frame are shown in Table 7-29 below.

Table 7-29 – *source_plug* values for command frame

value	meaning
00_{16} – $1E_{16}$	source plug 0 - 30
FE_{16}	no output
FF_{16}	any available source plug
all others	reserved

The controller may specify "any available source plug" (FF_{16}). When the controller want the camera storage subunit to play the file but not to output stream data, the controller may specify "no output"(FE_{16}) .

The *source_plug* field in the response frame indicates actual subunit source plug which outputs the stream data. The values of *source_plug* field in the response frame are shown in Table 7-30 below.

Table 7-30 – *source_plug* values for response frame

value	meaning
00_{16} – $1E_{16}$	source plug 0 - 30
FE_{16}	no output
all others	reserved

If the camera storage subunit can play specified file but can't output the stream data from specified subunit plug, then the camera storage subunit returns the ACCEPTED response with *source_plug* value of "no

output." For example, if the controller issues the PLAY FILE control command specifying MPEG4 file and the camera storage subunit doesn't implement the mechanism to output MPEG4 stream data to subunit plug and doesn't implement conversion mechanism to other implemented stream format, the camera storage subunit starts playing the specified MPEG4 file but no stream data is output to source plug. In case that the controller sets "no output" to *source_plug* filed in the command frame, the camera storage subunit also set "no output" to *source_plug* filed in the response frame.

Camera storage subunit may play plural files using plural source plugs simultaneously. But camera storage subunit can play only one file that is not output from subunit source plug. In other words, if camera storage subunit has n source plugs, the subunit may play maximum of n+1 files simultaneously; n files are output from subunit source plug from 0 through n-1 and 1 file is not output from subunit source plug. If the controller specifies "any available destination plug" and n+1 files are being played at that time, the camera storage subunit rejects the command with result code of "plug busy."

The *file_path_length* field specifies the size of the *file_path* in bytes. The length does not include the NULL character.

The *file_path* specifies the path name of the target file. The path name is specified by ASCII characters terminated by NULL (00_{16}) and begins with the "\" character (root directory).

Figure 7-10 illustrates the process of PLAY FILE control command.

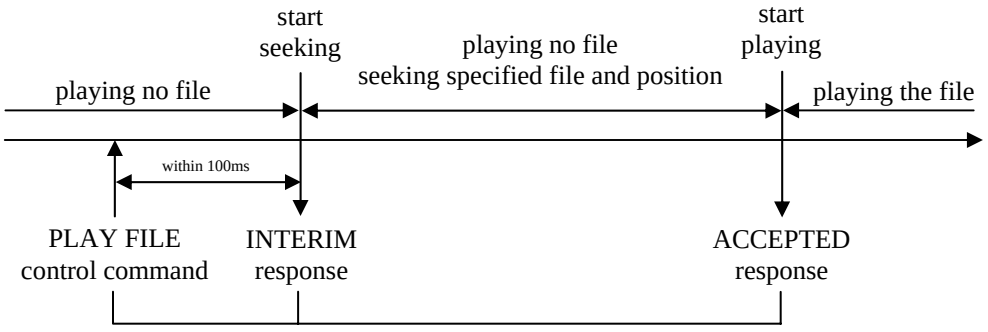


Figure 7-10 – Process of PLAY FILE control command

The camera storage subunit returns the final response to the controller when the subunit starts playing. After issuing final response, the camera storage subunit continues to play the file in specified playback mode until it finishes playing specified file or receives another PLAY FILE control command.

Figure 7-11 illustrates the process of PLAY FILE control command while playing.

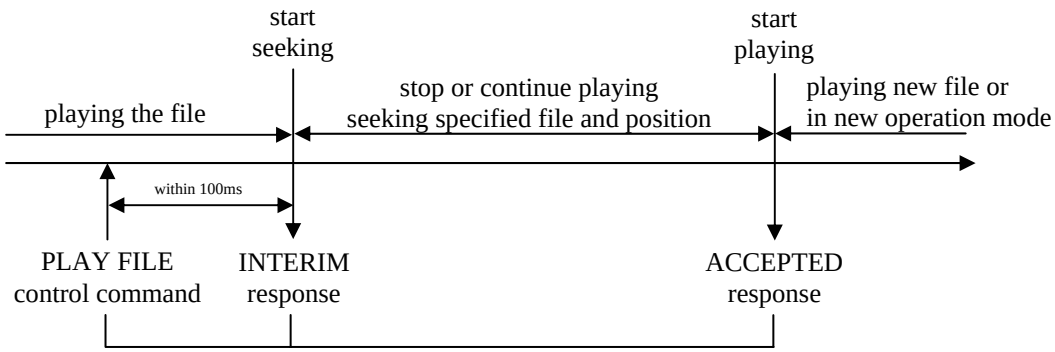


Figure 7-11 – Process of PLAY FILE control command while playing

When the camera storage subunit receives PLAY FILE control command while playing a file, the subunit issues INTERIM response within 100ms. Then start seeking specified file and position and when the subunit starts playing new file or in new operation mode, the camera storage subunit issues final response to the controller. While seeking the file or position, the camera storage subunit should continue to play previous file in previous operation mode to keep outputting stream data.

When the camera storage subunit receives PLAY FILE control command while preparing for another PLAY FILE control command, the camera storage subunit has following options:

1) reject the later command with result code of "busy"

2) accept the later command and reject the former command with result code of "aborted"

Figure 7-12 illustrates the process of PLAY FILE control command while preparing previous PLAY FILE command.

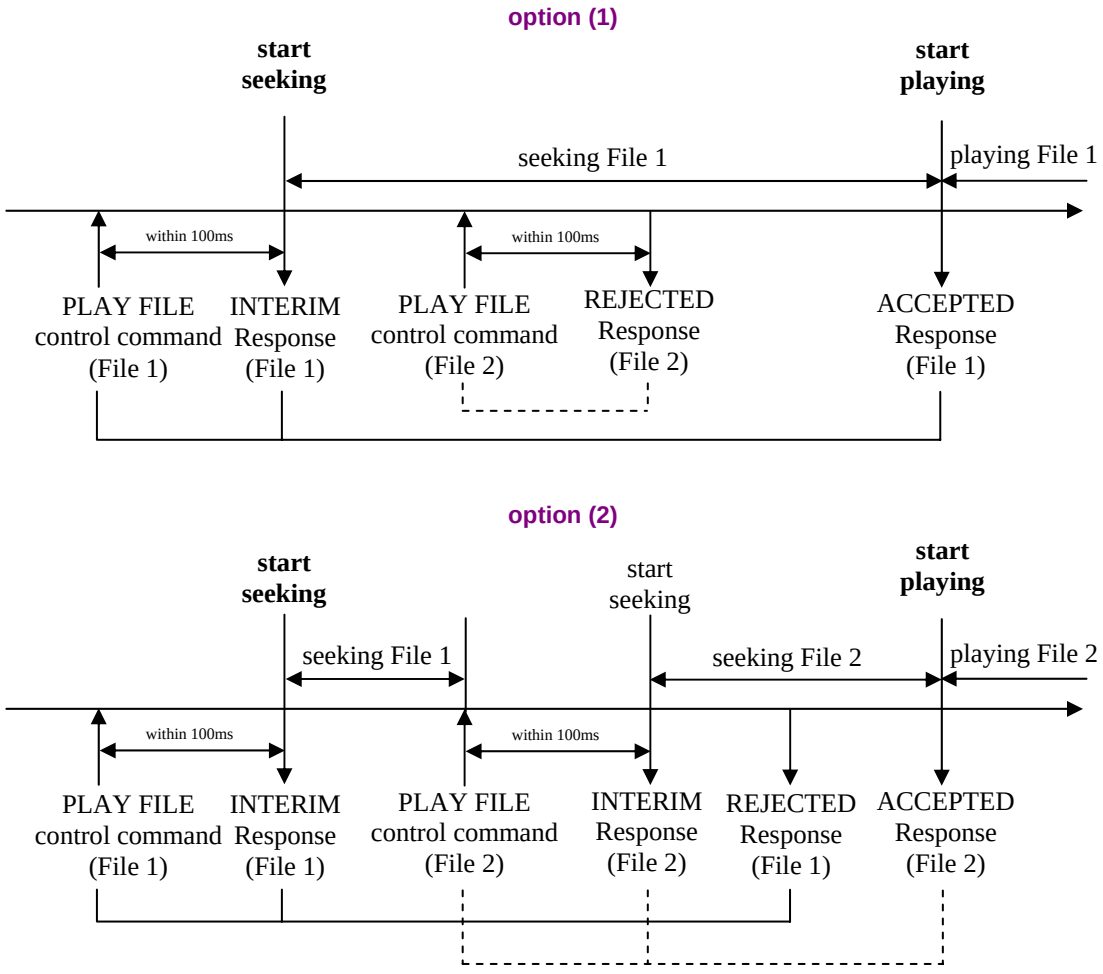


Figure 7-12 – Process of PLAY FILE control command while preparing

7.17.1.2 PLAY FILE control command without file_path

This type of control command requests camera storage subunit to change the playing mode of the file being played or to start playing the current selected file from current position. The formats of the PLAY FILE control command with file_path is illustrated in Figure 7-13 below.

	command format	response format
	msb lsb	msb lsb
opcode	PLAY FILE ($C3_{16}$)	<=
operand[0]	Subfunction	<=
operand[1]	FF_{16}	result
operand[2]	source_plug	source_plug
operand[3]	FF FF_{16}	<=
operand[4]		

Figure 7-13 – PLAY FILE control command and response format without *file_path* field

The meaning of *subfunction* field is same as PLAY FILE control command with *file_path*.

The *source_plug* field specifies the subunit source plug associated with the file to be played. The values are same as PLAY FILE control command with *file_path* described before.

If camera storage subunit receives this type of command frame when the subunit is playing a file, the subunit changes the playing mode of current file according to specified subfunction. If camera storage subunit receives this type of command when the subunit is not playing file, the subunit may reject the command with result code of “no operation” or accept the command and start playing a file and output stream data using specified source plug. If command is accepted, the controller can retrieve the information of the file by using PLAY FILE status command.

If camera storage subunit receives this type of command, duration of playing a file depends on the implementation of the subunit. The behaviour of camera storage subunit is same as when *duration_unit* field of PLAY FILE control command with *file_path* is set to “unspecify.”

7.17.2 PLAY FILE status command

The status command and STABLE response format for PLAY FILE command are illustrated in Figure 7-14 below.

	command format	response format
	msb lsb	msb Lsb
opcode	PLAY FILE ($C3_{16}$)	<=
operand[0]	source_plug	Subfunction
operand[1]		FF_{16}
operand[2]		physical_volume_number
operand[3]		logical_volume_number
operand[4]		media_generation_count
operand[5]		source_plug
operand[6]		Reserved
operand[7]		
operand[8]		file_path_length
operand[9]		file_path
:		
:		

Figure 7-14 – PLAY FILE status command and STABLE response format

The *source_plug* field specifies the source plug that is used to output the stream data. The values of this field are same as *source_plug* field in the response frame of PLAY FILE control command with file_path and are shown in Table 7-30. If controller want to know the information about a file being played but not output, the controller shall set the value of “no output” to this field.

The *subfunction* filed indicates the operation mode of playback. The values of this field are same as control command.

If the value of “STOP” is set to this field, then and the fields after operand[0] don’t exist in the response frame. There are two cases that value of “STOP” is set to this field:

1) Camera storage subunit is not playing a file using specified source plug.

2) The value of “no output” is specified in the *source_plug* field and camera storage subunit is not playing a file without outputting stream data of the file.

If the camera storage subunit is playing a file and outputting stream data using specified source plug or, “no output” is specified in the *source_plug* field and camera storage subunit is playing a file without outputting stream data of the file, then the response frame has following fields.

The *physical_volume_number*, *logical_volume_number* and *media_generation_count* fields indicate the information of physical volume number, logical volume number and media generation count of the storage media that stores the file being played back.

The *file_path_length* field indicates the size of the *file_path* in bytes. The length does not include the NULL character.

The *file_path* indicates the path name of the file that is being played. The path name is specified by ASCII characters terminated by NULL (00_{16}) and begins with the “\” character (root directory).

7.17.3 PLAY FILE field validation checks

The PLAY FILE field validation checks table is shown in Table 7-31 below.

Table 7-31 – PLAY FILE field validation checks table

Fields	ck	recommended values
Subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
start_position	-	00 00 00 00_{16}
Duration	-	00 00 00 00_{16}
source_plug	√	
start_position_unit	√	
duration_unit	√	
file_path_length	-	00_{16}
file_path	-	00_{16}

7.17.4 Result codes for PLAY FILE command

The possible result codes for PLAY FILE command are summarized in Table 7-32 below.

Table 7-32 – Result codes for PLAY FILE command

ACCEPTED	STABLE	REJECTED
success default position		busy aborted disabled invalid generation count invalid volume number invalid subunit plug invalid parameter no media no file invalid file type invalid operation mode no operation unknown

7.18 RECEIVE FILE command

The RECEIVE FILE command is used to input the file data received from the subunit destination plug and store it in the storage media.

7.18.1 RECEIVE FILE control command

The formats of the RECEIVE FILE control command is illustrated in Figure 7-15 below.

	command format	response format
	msb lsb	msb lsb
opcode	RECEIVE FILE (52_{16})	<=
operand[0]	Subfunction	<=
operand[1]	FF_{16}	Result
operand[2]	physical_volume_number	physical_volume_number
operand[3]	logical_volume_number	logical_volume_number
operand[4]	media_generation_count	media_generation_count
operand[5]	receive_size	<=
operand[6]		
operand[7]		
operand[8]		
operand[9]	destination_plug	<=
operand[10]	receive_mode	<=
operand[11]	file_type	<=
operand[12]	file_path_length	new_file_path_length
operand[13]	file_path	new_file_path
:		
:		

Figure 7-15 – RECEIVE FILE control command and response format

The *physical_volume_number* and the *logical_volume_number* fields specify the volume where the received file is stored. The controller can specify "any available volume" by using the value of FF_{16}. When the value of FF_{16} is specified, the camera storage subunit shall select a volume from the available volumes and shall set the selected volume number to the response frame. If the *physical_volume_number* is FF_{16} the *media_generation_count* field of the command frame shall be ignored and the current generation counter value of the selected physical volume shall be set to the response frame.

The *destination_plug* field specifies which subunit destination plug will be used to receive the file data. The controller shall not specify "any available destination plug" (FF_{16}). Before issuing this command, the controller shall establish connection between the camera storage subunit destination plug and the unit plug or subunit source plug of another subunit in the same unit. The controller shall specify the actual plug number used by the connection. If the specified destination plug is being used when the RECEIVE FILE command is received, then the camera storage subunit shall return the REJECTED response with the result code of "busy."

The *receive_mode* field specifies the operation mode when there is already a file with the same path name indicated by *file_path* field on the specified volume. The values of this field are shown in Table 7-33 below. The camera storage subunit that supports the RECEIVE FILE control command shall implement all operation modes. In the case that there is no file which has same path name indicated by *file_path*, the command is accepted regardless of the *receive_mode* value.

Table 7-33 – *receive_mode* values of the RECEIVE FILE command

receive_mode	value	operation mode
protect	00_{16}	the command is rejected with result code of "file exist"
over_write	01_{16}	the existing file is replaced to received file
rename	02_{16}	the received file is stored using another path name

The *file_type* field specifies the type of the file to be received. When the controller specifies only file extension by *file_path* field ("*.???" is used), the controller shall specify one of the valid values shown in Table 7-34 below. The value of FF_{16} is not valid and is used when full path name is specified by *file_path* field. If the controller specifies full path name, then the value of this field shall be FF_{16}.

Table 7-34 – *file_type* values of the RECEIVE FILE command

file_type	value	Meaning
DCF	00_{16}	DCF file
unspecified	FF_{16}	no information about file type
reserved	all others	Reserved for future extension

If *file_type* is “DCF”, then the camera storage subunit should store the received file in the DCF directory using DCF file name, and specified file extension should be used for the DCF file name.

The *file_path_length* field specifies the size of the *file_path* field in bytes. The length does not include the NULL character.

The *new_ file_path_length* field in the response frame indicates the size of the *new_file_path* field in bytes. The length does not include the NULL character.

The *receive_size* field specifies the size of the data to be stored in bytes. If the controller does not know the size or the size exceeds $FFFFFFFF_{16}$, then the controller shall set the field to zero and the camera storage subunit should receive the data.

The *file_path* field specifies the path name of the received file. When *file_type* is “DCF”, the controller shall specify “*.???” (“???” part is extension of file name). For example, the controller may use “*.JPG” for *file_path*. When *file_type* is “unspecified”, the controller shall specify the full path name of the file.

The *new_file_path* field in the response frame indicates the path name of the stored file. If the command is rejected, then the value of this field shall be NULL (00_{16}) and the *new_file_path_length* field shall be set to 00_{16}.

If the controller specifies the full path name, then the camera storage subunit uses the specified *file_path* and the new file is created in the location that indicated by *file_path*. In this case, the *new_file_path* field in the response frame holds same value as the *file_path* field. By specifying the value according to DCF naming rule, the controller can write DCF files as one DCF object.

When a directory specified in the *file_path* field does not exist, there are three options to be exercised. Depending on its implementation, the camera storage subunit has the following options:

(1) reject the command with result code of “no directory”

(2) accept the command and create the specified directory, then store the file in the directory

(3) accept the command and store the file using other path name

In case of (2) the value of *new_file_path* field in the response frame is the same as the *file_path* in the command frame. In case of (3) the value of *new_file_path* depends on the implementation of the camera storage subunit.

If the controller specifies “*.???” as *file_path*, the path name of the received file depends on the implementation of the camera storage subunit, but the file extension shall be used for the path name of the stored file. The controller can determine the path name by checking the *new_file_path* field in the response frame.

The relation between *file_path, new_file_path, receive_mode, file_type* and response type is summarized in Table 7-35 below. Suppose the file “\DCIM\100ABCDE\ABCD0001.JPG” exists and the file “\DCIM\100ABCDE\WXYZ0002.JPG” does not exist on the specified volume:

Table 7-35 – Relation between *file_path* and *new_file_path* for RECEIVE FILE command

file_path	receive_ mode	file_type	new_file_path	response type
"*.JPG"	don't care	DCF	depend on the camera storage subunit but DCF file name is used for the path name and file extension is "JPG"	ACCEPTED
"\DCIM\100ABCDE\ABCD0001.JPG"	protect	unspecified	NULL (00_{16})	REJECTED
	over_write	unspecified	"\DCIM\100ABCDE\ABCD0001.JPG"	ACCEPTED
	rename	unspecified	depend on the camera storage subunit	ACCEPTED
"\DCIM\100ABCDE\WXYZ0002.JPG"	don't care	unspecified	"\DCIM\100ABCDE\WXYZ0002.JPG"	ACCEPTED

NOTE "don't care" means "any value is to be specified".

7.18.2 RECEIVE FILE status command

The status command and STABLE response format for RECEIVE FILE command are illustrated in Figure 7-16 below.

	command format	response format
	msb lsb	msb lsb
opcode	RECEIVE FILE (52_{16})	<=
operand[0]	destination_plug	Subfunction
operand[1]		Result
operand[2]		physical_volume_number
operand[3]		logical_volume_number
operand[4]		media_generation_count
operand[5]		received_size
operand[6]		
operand[7]		
operand[8]		
operand[9]		destination_plug
operand[10]		receive_mode
operand[11]		file_type
operand[12]		file_path_length
operand[13]		file_path
:		
:		

Figure 7-16 – RECEIVE FILE status command and STABLE response format

The *destination_plug* field in the command frame specifies the subunit destination plug to query the status. If the specified plug does not exist on the camera storage subunit, then the NOT IMPLEMENTED shall be returned. In this case, the format and field values of the response frame are the same as the command frame.

If the camera storage subunit receives the RECEIVE FILE status command while executing /suspending/aborting the RECEIVE FILE control command that is using the specified destination plug, then the camera storage subunit shall copy the field values of the executing control command to the status response frame (except for the *result* and *received_size* field). If the camera storage subunit is not executing the control command, then the response frame contains only *subfunction*, *result*, *physical_volume_number*, *logical_volume_number* and *media_generation_count* fields and these fields except the *result* field shall have the value of FF_{16}.

The *result* field in the response frame indicates the status of the command execution. The values for this field are described in section 5.5. If the RECEIVE FILE control command that is using the specified destination plug is being executed, then the value of "executing" shall be set. If the RECEIVE FILE control command that is using the specified destination plug is suspended by bus reset, then the value of "suspended" shall be set. If the RECEIVE FILE control command that is using the specified destination plug is being aborted (ACCEPTED response for abort subfunction has been sent and REJECTED response for execute or resume subfunction has not been sent yet), then the value of "aborting" shall be set. Otherwise, the value of "not executing" shall be set.

The *received_size* field in the response frame indicates the approximate size (in bytes) of data already received by the subunit. The size may be the exact received size when the status command is received, a unit of the segment buffer of Asynchronous Connections, etc. If the camera storage subunit cannot determine the size, then the value of $FFFFFFFF_{16}$ shall be set whenever the RECEIVE FILE control command is being executed. When the size exceeds $FFFFFFFF_{16}$, the value of $FFFFFFFF_{16}$ should be set.

7.18.3 RECEIVE FILE field validation checks

The RECEIVE FILE field validation checks table is shown in Table 7-36 below.

Table 7-36 – RECEIVE FILE field validation checks table

Fields	ck	recommended values
Subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
receive_size	-	00 00 00 00_{16}
destination_plug	√	
receive_mode	√	
file_type	√	
file_path_length	-	00_{16}
file_path	-	00_{16}

7.18.4 Result codes for RECEIVE FILE command

The possible result codes for RECEIVE FILE command are summarized in Table 7-37 below.

Table 7-37 – Result codes for RECEIVE FILE command

ACCEPTED	STABLE	REJECTED
success	not executing executing suspended aborting	busy aborted retry no command disabled invalid generation count invalid volume number invalid subunit plug invalid parameter no media no directory no space file exist unsupported format volume protected file protected plug busy transport error unknown

7.19 RECEIVE FILE PARTIAL command

The RECEIVE FILE PARTIAL command is used to input the data received from the subunit destination plug and replace the data of specified file.

7.19.1 RECEIVE FILE PARTIAL control command

The formats of the RECEIVE FILE PARTIAL control command is illustrated in Figure 7-17 below.

	Command format	response format
	msb lsb	msb lsb
opcode	RECEIVE FILE PARTIAL (55_{16})	<=
operand[0]	subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	physical_volume_number
operand[3]	logical_volume_number	logical_volume_number
operand[4]	media_generation_count	media_generation_count
operand[5] : operand[12]	start_offset	<=
operand[13] : operand[20]	receive_size	received_size
operand[21]	Destination_plug	<=
operand[22]	close_mode	<=
operand[23]	reserved	<=
operand[24]	file_path_length	file_path_length
operand[25] : :	file_path	file_path

Figure 7-17 – RECEIVE FILE PARTIAL control command and response format

The meaning and function of each field except *start_offset*, *receive_size*, *received_size* and *close_mode* are identical to those of the RECEIVE FILE command.

The *start_offset* field specifies the starting byte offset of the data in the specified file to be replaced. If the *start_offset* field specifies the byte offset exceed the byte size of specified file, the camera storage subunit rejects the command with result code of “invalid parameter”.

The *receive_size* field specifies the byte size of the data to be received. The controller shall specify the actual byte size of the data to be received. If *receive_size* field indicates zero, the camera storage subunit rejects the command with result code of “invalid parameter”.

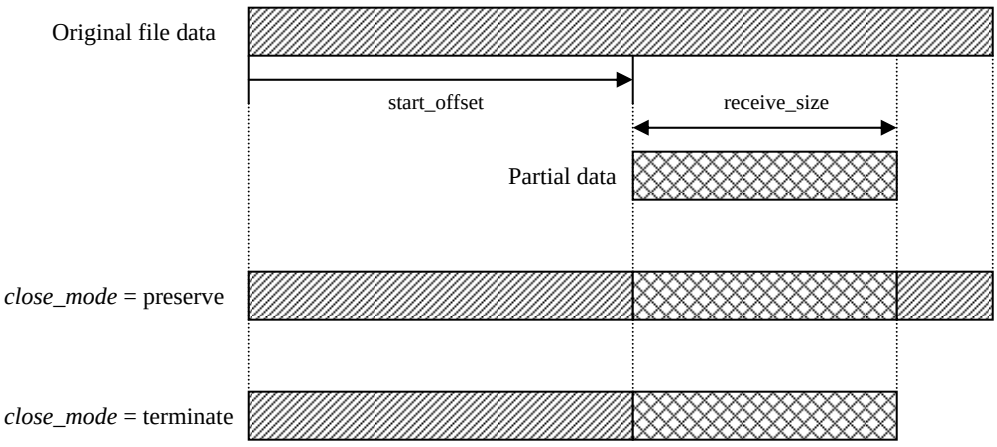
The *received_size* field indicates the byte size of the data actually received.

When the camera storage subunit receives the RECEIVE FILE PARTIAL control command, the subunit opens the specified file and seek the offset point. Then the subunit starts receiving the data and overwrite the data. When the subunit finishes receiving the specified amount of data, the subunit issues the final response to the controller. The *received_size* field in the final response indicates the actual amount of received data.

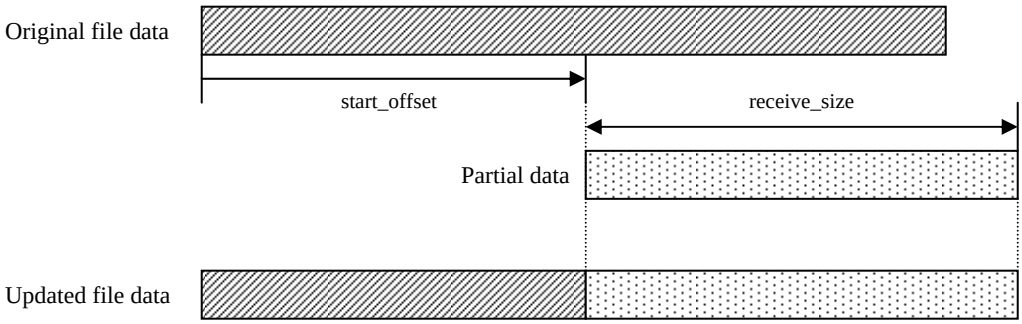
The *close_mode* filed specifies the process after receiving file data. This field is ignored when the sum of *start_offset* and *receive_size* exceeds the size of original file. In this case, after the camera storage subunit overwrites the last data of the original file, the subunit continues to receive data from destination plug and append the data to the file. The values of *close_mode* field and meanings are shown in Table 7-38 and Figure 7-18 below.

Table 7-38 – *close_mode* values of the RECEIVE FILE PARTIAL command

close_mode	Value	meaning
preserve	00_{16}	The data after received data is preserved
terminate	01_{16}	The file is terminated at the end of received data
reserved	all others	Reserved for future extension



(1) The sum of *start_offset* and *receive_size* doesn't exceed the size of original file



(2) The sum of *start_offset* and *receive_size* exceeds the size of original file

Figure 7-18 – Relation between *close_mode* and updated file structure

The *file_path_length* field in the command frame specifies the size of the *file_path* field in the command frame in bytes. The length does not include the NULL character.

The *file_path* field in the command frame specifies the path name of the file to be received. When the value of the *start_offset* field is zero, the controller may specify "*.???" ("???" part is extension of file name). For example, the controller may use "*.JPG" for *file_path*. When wildcard character ("*") is used in the *file_path* field and the value of the *start_offset* field is zero, then the camera storage subunit create a new file. The path name of the created file depends on the implementation of the camera storage subunit, but the file extension shall be used for the path name of the created file.

If *start_offset* is not zero and the file specified by *file_path* field doesn't exist, the command is rejected with the result code of "no file." If *start_offset* is zero and the file specified by *file_path* field doesn't exist, the camera storage subunit has following options:

(1) reject the command with result code of "no file"

(2) accept the command and create the specified file

The *file_path_length* field in the response frame indicates the size of the *file_path* field in the response frame in bytes. The length doesn't include the NULL character.

The *file_path* field in the response frame indicates the path name of the file that data is written. If the *file_path* field in the command frame specifies full path name of the file, then the *file_path* field in the response frame contains same path name as command frame. If the *file_path* field in the command frame specifies the path name using wildcard character, then the *file_path* field in the response frame indicates the path name of created file.

The RECEIVE FILE PARTIAL control command can be used to modify file data. An example of typical sequence of camera storage command is shown in Figure 7-19 below.

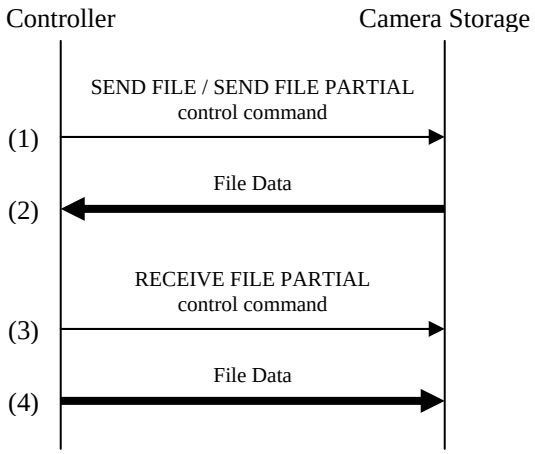


Figure 7-19 – Typical sequence to update file data using RECEIVE FILE PARTIAL command

At first, controller issues SEND FILE or SEND FILE PARTIAL control command and receive file data. The controller checks the received file data, for example, such as header data, and then issues RECEIVE FILE PARTIAL control command and send modified file data. If another controller modifies same target file between (2) and (3), the target file structure may be broken by updating file data using RECEIVE FILE PARTIAL control command. Therefore to accomplish updating file data safely, camera storage subunit should implement RESERVE control command. If the controller detects the camera storage subunit supports RESERVE control command, then the controller should issue commands according to following sequence.

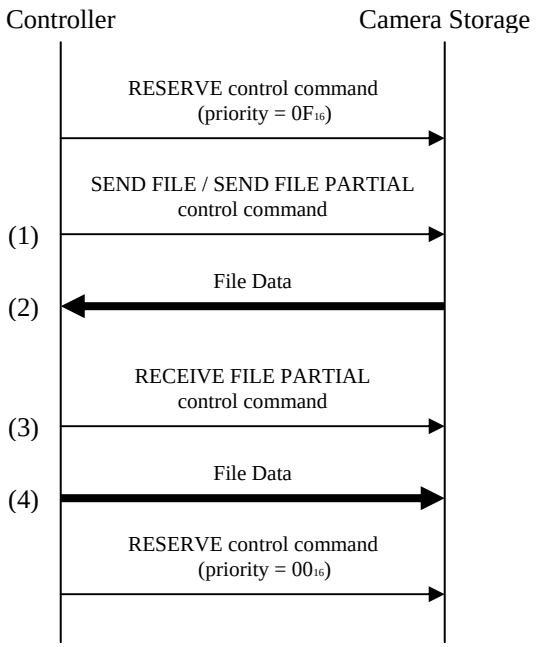


Figure 7-20 – Preferable sequence to update file data using RECEIVE FILE PARTIAL command

Before issuing SEND FILE or SEND FILE PARTIAL control command to check the file data, controller issues RESERVE control command to camera storage subunit. The priority field of RESERVE control command shall be set to $0F_{16}$ to exclude control commands from another controller. Then the controller issues SEND FILE or SEND FILE PARTIAL control command to obtain the file data. After updating the file data by using RECEIVE FILE PARTIAL control command, the controller issues RESERVE control command with priority field set to 00_{16} to cancel exclusion.

7.19.2 RECEIVE FILE PARTIAL status command

The status command and STABLE response format for RECEIVE FILE PARTIAL command are illustrated in Figure 7-21 below.

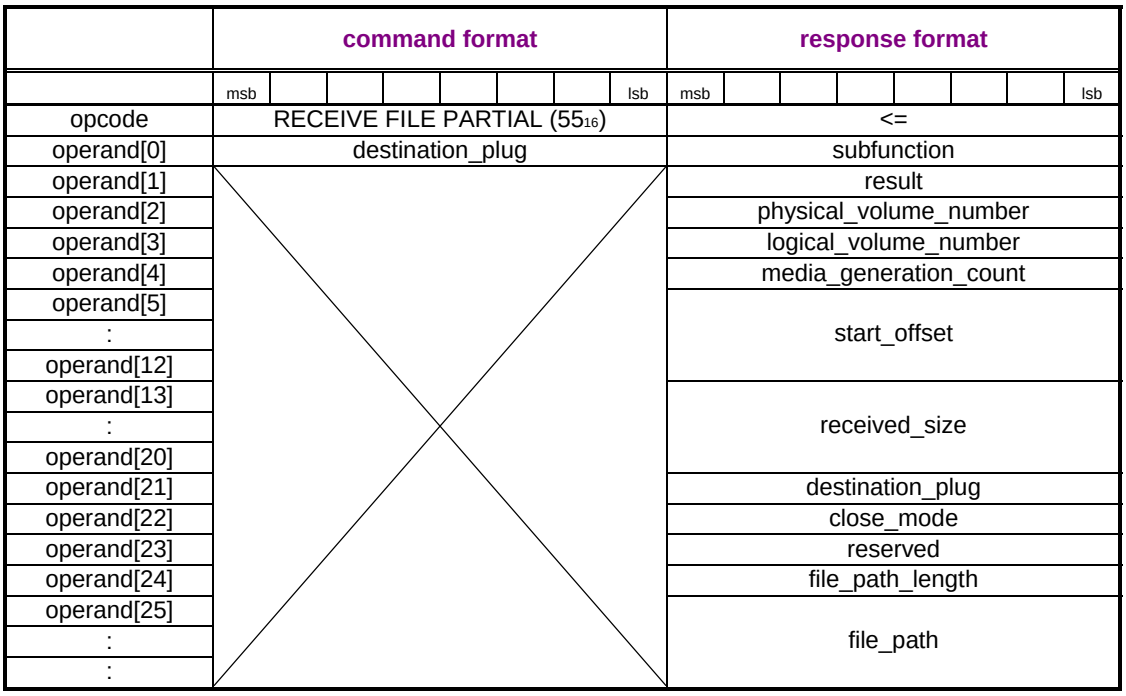


Figure 7-21 – RECEIVE FILE PARTIAL status command and STABLE response format

The *destination_plug* field in the command frame specifies the subunit destination plug to query the status. If the specified plug does not exist on the camera storage subunit, then the NOT IMPLEMENTED shall be returned. In this case, the format and field values of the response frame are the same as the command frame.

If the camera storage subunit receives the RECEIVE FILE PARTIAL status command while executing /suspending/aborting the RECEIVE FILE PARTIAL control command that is using the specified destination plug, then the camera storage subunit shall copy the field values of the executing control command to the status response frame (except for the *result* and *received_size* field). If the camera storage subunit is not executing the control command, then the response frame contains only *subfunction*, *result*, *physical_volume_number*, *logical_volume_number* and *media_generation_count* fields and these fields except the *result* field shall have the value of FF_{16}.

The *result* field in the response frame indicates the status of the command execution. The values for this field are described in section 5.5. If the RECEIVE FILE PARTIAL control command that is using the specified destination plug is being executed, then the value of “executing” shall be set. If the RECEIVE FILE PARTIAL control command that is using the specified destination plug is suspended by bus reset, then the value of “suspended” shall be set. If the RECEIVE FILE PARTIAL control command that is using the specified destination plug is being aborted (ACCEPTED response for abort subfunction has been sent and REJECTED response for execute or resume subfunction has not been sent yet), then the value of “aborting” shall be set. Otherwise, the value of “not executing” shall be set.

The *received_size* field in the response frame indicates the approximate size (in bytes) of data already received by the subunit. The size may be the exact received size when the status command is received, a unit of the segment buffer of Asynchronous Connections, etc. If the camera storage subunit cannot determine the size, then the value of $FFFFFFFF_{16}$ shall be set whenever the RECEIVE FILE PARTIAL control command is being executed.

7.19.3 RECEIVE FILE PARTIAL field validation checks

The RECEIVE FILE PARTIAL field validation checks table is shown in Table 7-39 below.

Table 7-39 – RECEIVE FILE PARTIAL field validation checks table

Fields	ck	recommended values
Subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
start_offset	-	00 00 00 00 00 00 00 00_{16}
receive_size	-	00 00 00 00 00 00 00 00_{16}
destination_plug	√	
close_mode	√	
file_path_length	-	00_{16}
file_path	-	00_{16}

7.19.4 Result codes for RECEIVE FILE PARTIAL command

The possible result codes for RECEIVE FILE PARTIAL command are summarized in Table 7-40 below.

Table 7-40 – Result codes for RECEIVE FILE PARTIAL command

ACCEPTED	STABLE	REJECTED
success	not executing executing suspended aborting	busy aborted retry no command disabled invalid generation count invalid volume number invalid subunit plug invalid parameter no media no file no space unsupported format volume protected file protected plug busy transport error unknown

7.20 RECORD FILE command

The RECORD FILE command is used to request the camera storage subunit to input stream data from specified destination plug and record the data as a file.

7.20.1 RECORD FILE control command

There are two types of the format of RECORD FILE control command.

7.20.1.1 RECORD FILE control command with file_path

The formats of the RECORD FILE control command with file_path is illustrated in Figure 7-22 below.

	command format	response format
	msb lsb	msb lsb
opcode	RECORD FILE ($C2_{16}$)	<=
operand[0]	Subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	Media_generation_count	media_generation_count
operand[5]	Reserved	<=
operand[6]		
operand[7]		
operand[8]		
operand[9]		
operand[10]		
operand[11]		
operand[12]		
operand[13]	destination_plug	destination_plug
operand[14]	signal_format	recording_signal_format
operand[15]	record_mode	<=
operand[16]	file_path_length	recording_file_path_length
operand[17]	file_path	recording_file_path
:		
:		

Figure 7-22 – RECORD FILE control command and response format with *file_path* field

The *subfunction* field specifies the recording mode of RECORD FILE command. The values of this field are shown in Table 7-41 below.

Table 7-41 – *subfunction* values of the RECORD FILE command

recording mode	value	support level	description
STOP	00_{16}	M	Stop recording operation
RECORD	75_{16}	M	Record at normal speed
RECORD PAUSE	$7D_{16}$	O	Pause in recording

The subunit support level for RECORD FILE command varies according to the recording mode requested.

When *subfunction* field specifies “STOP” or “RECORD PAUSE”, *record_mode*, *file_path_length* and *file_path* fields are ignored and the values of *new_file_path_length* and *new_file_path* fields in the response frame are same as the command frame.

The *destination_plug* field in the command frame specifies which subunit destination plug will be used to input the stream data. The controller may specify “any available destination plug” (FF_{16}). If the controller specifies “any available destination plug” and all subunit destination plugs are being used to record files at that time, the camera storage subunit rejects the command with result code of “plug busy.”

The *destination_plug* field in the response frame indicates actual subunit destination plug which inputs the stream data.

The *signal_format* field in the command frame specifies the signal format to be recorded. The values of this field are shown in Table 7-42 below.

Table 7-42 – *signal_format* values for RECORD FILE command

Value	signal mode
00_{16}	SD 525-60
04_{16}	SDL 525-60
08_{16}	HD 1125-60
80_{16}	SD 625-50
84_{16}	SDL 625-50
88_{16}	HD 1250-50
10_{16}	MPEG 25Mbps-60
14_{16}	MPEG 12.5Mbps-60
18_{16}	MPEG 6.25Mbps-60
90_{16}	MPEG 25Mbps-50
94_{16}	MPEG 12.5Mbps-50
98_{16}	MPEG 6.25Mbps-50
01_{16}	D-VHS Digital
FF_{16}	any available format

If camera storage subunit can't record the data in specified format, the camera storage subunit should record in other supported format. Camera storage subunit shall support "any available format" and all other formats are optional.

The *recording_signal_format* field in the response frame indicates the actual signal format being recorded. The values of this field shall be one of the values in Table 7-42 except "any available format."

The *record_mode* field specifies the mode of recording and has meaning only when there is already a file with the same path name indicated by *file_path* field on the specified volume. The values of this field is shown in Table 7-43 below.

Table 7-43 – *record_mode* values

record_mode	Value	operation mode
protect	00_{16}	the command is rejected with result code of "file exist"
over_write	01_{16}	the existing file is replaced to received stream
rename	02_{16}	the received stream is stored using another path name
append	03_{16}	the received stream is appended to existing file

The camera storage subunit that supports the RECORD FILE control command shall implement "protect", "over_write" and "rename." In the case that there is no file which has same path name indicated by *file_path*, the command is accepted regardless of the *record_mode* value.

The *file_path_length* field specifies the size of the *file_path* in bytes. The length does not include the NULL character.

The *file_path* specifies the path name of the target file. The path name is specified by ASCII characters terminated by NULL (00_{16}) and begins with the "\" character (root directory).

The *file_path* field specifies the path name of the recorded file. The controller shall specify the full path name of the file or otherwise specify "*" (wildcard character).

The *recording_file_path* field in the response frame indicates the path name of the file that is being recorded. If the command is rejected, then the value of this field shall be NULL (00_{16}) and the *recording_file_path_length* field shall be set to 00_{16}.

If the controller specifies the full path name, then the camera storage subunit uses the specified *file_path* and if specified file doesn't exist, the new file is created in the location that indicated by *file_path*. In this case, the *recording_file_path* field in the response frame holds same value as the *file_path* field.

When a directory specified in the *file_path* field does not exist, there are three options to be exercised. Depending on its implementation, the camera storage subunit has the following options:

(1) reject the command with result code of "no directory"

(2) accept the command and create the specified directory, then record the file in the directory

(3) accept the command and record the file using other path name

In case of (2) the value of *recording_file_path* field in the response frame is the same as the *file_path* in the command frame. In case of (3) the value of *recording_file_path* depends on the implementation of the camera storage subunit.

If the controller specifies "*" as *file_path*, the path name of the received file depends on the format of input stream data and implementation of the camera storage subunit. The controller can determine the path name by checking the *recording_file_path* field in the response frame.

The relation between *file_path*, *recording_file_path*, *record_mode* and response type is summarized in Table 7-44 below. Suppose the file "\MOVIE\MOVIE001.MPG" exists and the file "\MOVIE\MOVIE002.MPG" does not exist on the specified volume:

Table 7-44 – Relation between *file_path* and *recording_file_path* for RECORD FILE command

file_path	record_mode	recording_file_path	response type (result)
"*"	don't care	depend on the camera storage subunit	ACCEPTED
"\MOVIE\MOVIE001.MPG"	protect	NULL (00_{16})	REJECTED (file exist)
	over_write	"\MOVIE\MOVIE001.MPG"	ACCEPTED
	rename	depend on the camera storage subunit	ACCEPTED
	append	"\MOVIE\MOVIE001.MPG"	ACCEPTED
"\MOVIE\MOVIE002.MPG"	don't care	"\MOVIE\MOVIE002.MPG"	ACCEPTED

NOTE "don't care" means "any value is to be specified"

Figure 7-23 illustrates the process of RECORD FILE control command.

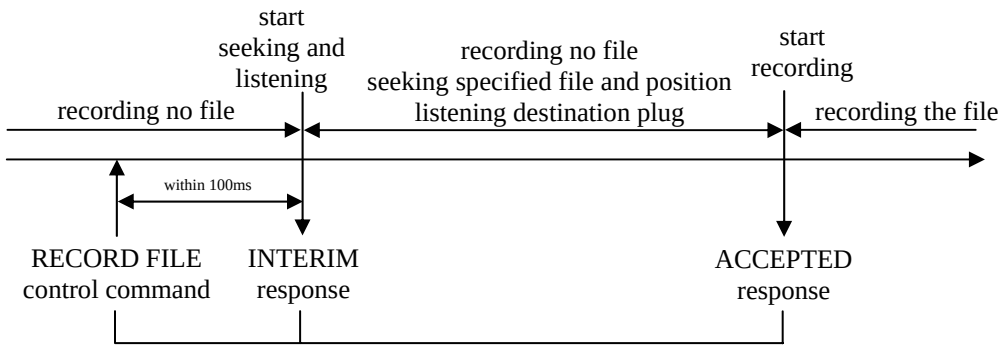


Figure 7-23 – Process of RECORD FILE control command

The camera storage subunit returns the final response to the controller when the subunit starts recording. After issuing final response, the camera storage subunit continues to record the file until input stream data stops or receives another RECORD FILE control command. If camera storage subunit can't receive stream data from specified destination plug within 5 second after issuing INTERIM response, the camera storage subunit may reject the command with result code of "no input." If camera storage subunit doesn't support the format of input stream data, the camera storage subunit may reject the command with result code of "unsupported stream."

Figure 7-24 illustrates the process of RECORD FILE control command while recording.

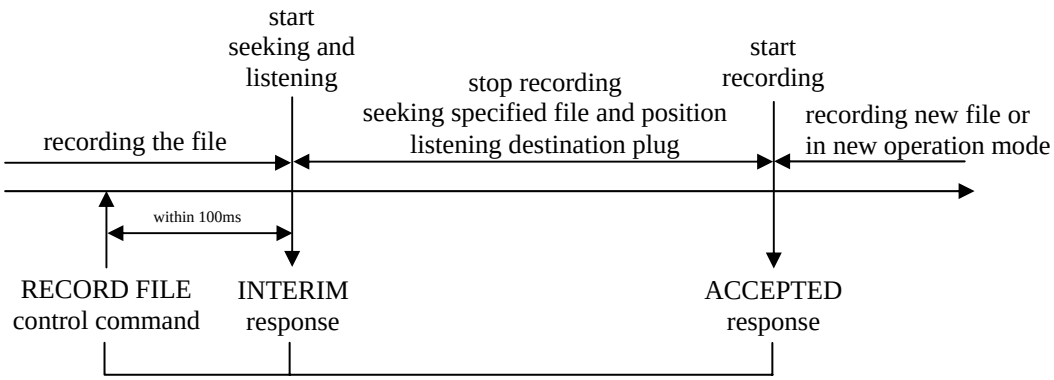


Figure 7-24 – Process of RECORD FILE control command while playing

When the camera storage subunit receives RECORD FILE control command while recording a file, the subunit issues INTERIM response within 100ms. Then start seeking specified file and position and when the subunit starts recording new file or in new operation mode, the camera storage subunit issues final response to the controller.

When the camera storage subunit receives RECORD FILE control command while preparing for another RECORD FILE control command, the camera storage subunit has following options:

1) reject the later command with result code of "busy"

2) accept the later command and reject the former command with result code of "aborted"

Figure 7-25 illustrates the process of RECORD FILE control command while preparing previous RECORD FILE command.

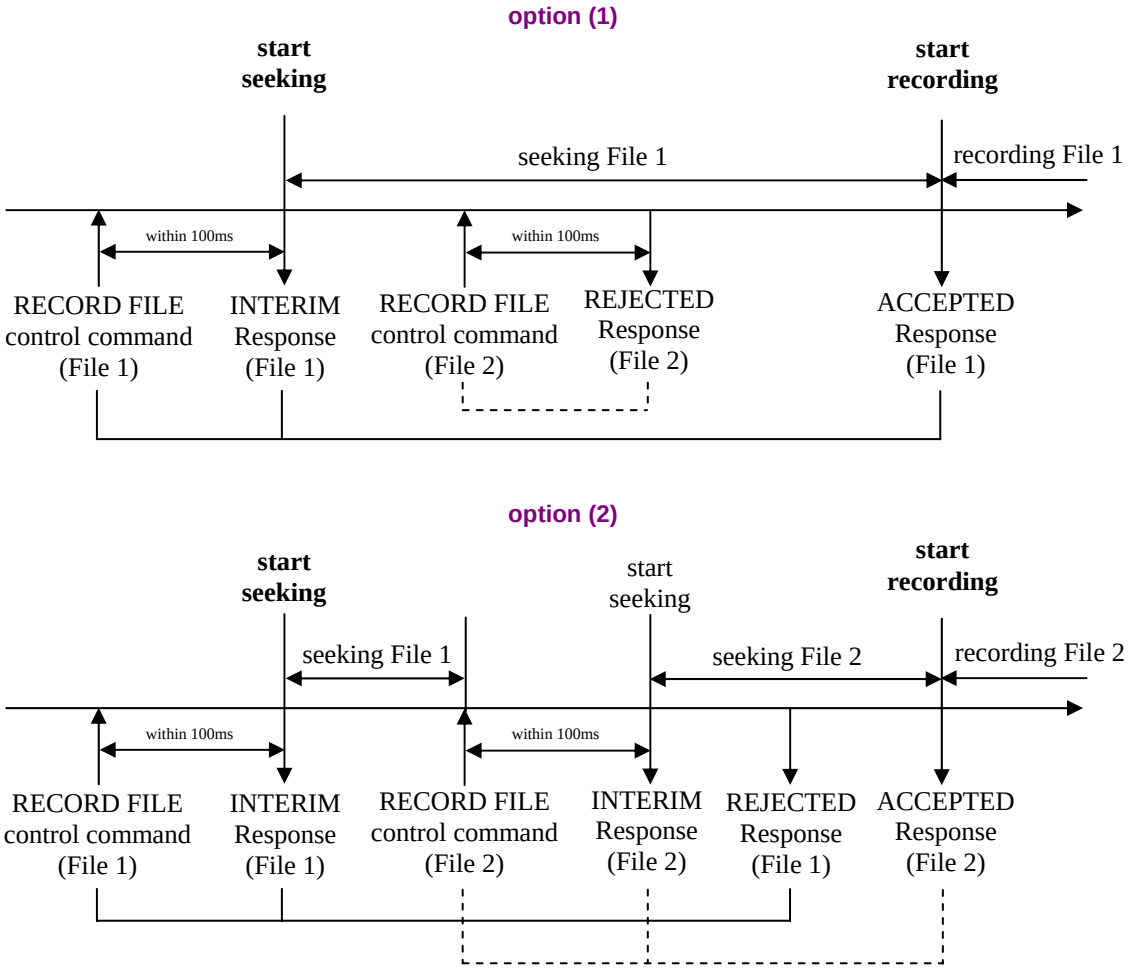


Figure 7-25 – Process of RECORD FILE control command while preparing

7.20.1.2 RECORD FILE control command without file_path

This type of control command requests camera storage subunit to start recording the stream data to current selected file from current position. The formats of the RECORD FILE control command with file_path is illustrated in Figure 7-26 below.

	command format	response format
	msb lsb	msb lsb
opcode	RECORD FILE ($C2_{16}$)	<=
operand[0]	subfunction	<=
operand[1]	FF_{16}	result
operand[2]	destination_plug	destination_plug
operand[3]	FF FF_{16}	<=
operand[4]		

Figure 7-26 – RECORD FILE control command and response format without *file_path* field

The meaning of *subfunction* field is same as RECORD FILE control command with *file_path*.

The *destination_plug* field specifies the subunit destination plug associated with the file to be recorded. The values are same as RECORD FILE control command with *file_path* described before.

When camera storage subunit receives this type of command frame, the subunit changes the recording mode of current file according to specified subfunction. If camera storage subunit receives this type of command when the subunit is not recording file, the subunit may reject the command with result code of "no operation" or accept the command and create a new file and start recording input stream data from default subunit destination plug. If command is accepted, the controller can retrieve the information of the file and subunit destination plug by using RECORD FILE status command.

If recording is paused by subfunction of "RECORD PAUSE" and recording is resumed again by subfunction of "RECORD", new stream data should be recorded in the same file before recording is paused.

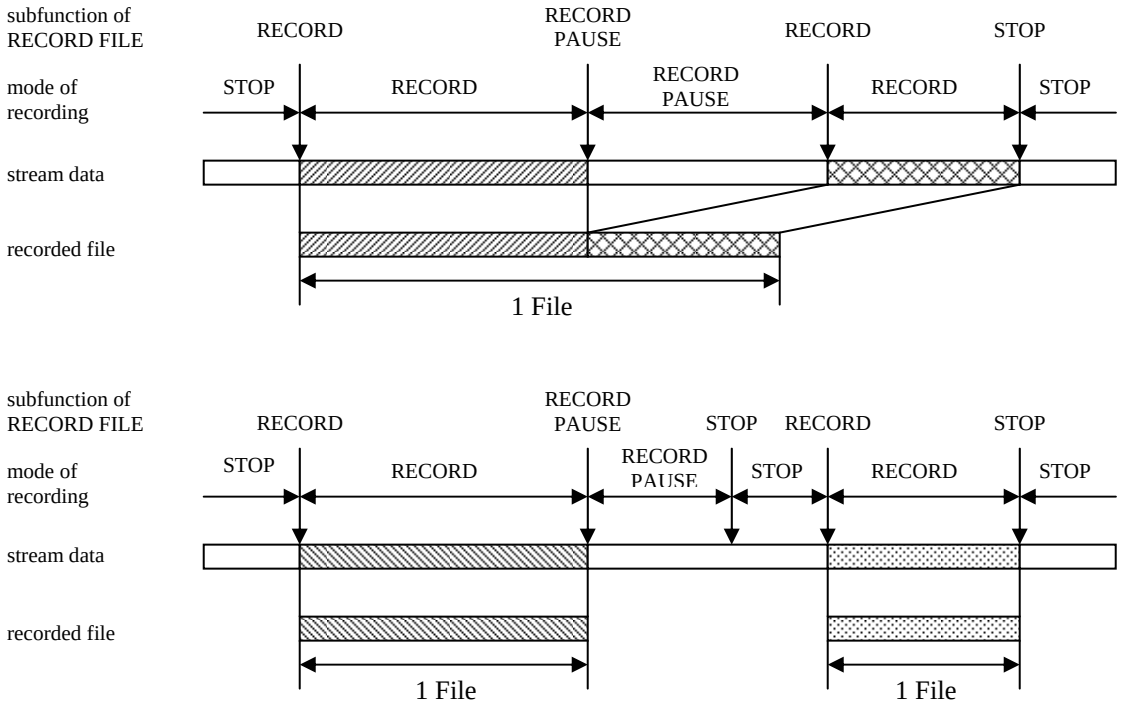


Figure 7-27 – Relation between recording mode and preferable recorded file structure

7.20.2 RECORD FILE status command

The status command and STABLE response format for RECORD FILE command are illustrated in Figure 7-28 below.

	command format	response format
	msb … lsb	msb … lsb
opcode	RECORD FILE ($C2_{16}$)	<=
operand[0]	destination_plug	subfunction
operand[1]		FF_{16}
operand[2]		physical_volume_number
operand[3]		logical_volume_number
operand[4]		media_generation_count
operand[5]		destination_plug
operand[6]		recording_signal_format
operand[7]		reserved
operand[8]		recording_file_path_length
operand[9]		recording_file_path
:		
:		

Figure 7-28 – RECORD FILE status command and STABLE response format

The *destination_plug* field specifies the destination plug that is used to input the stream data.

The *subfunction* filed indicates the operation mode of recording. The values of this field are same as control command. If the camera storage subunit is not recording a file using specified destination plug, then the value of “STOP” shall be set in this field and the fields after operand[0] don’t exist in the response frame. If the recording mode of camera storage subunit is “RECORD” or “RECORD PAUSE” using specified destination plug, then the response frame has following fields.

The *physical_volume_number*, *logical_volume_number* and *media_generation_count* fields indicate the information of physical volume number, logical volume number and media generation count of the storage media that stores the file being recorded.

The *recording_signal_format* field indicates the signal format being recorded.

The *recording_file_path_length* field indicates the size of the *recording_file_path* in bytes. The length does not include the NULL character.

The *recording_file_path* field indicates the path name of the file that is being recorded. The path name is specified by ASCII characters terminated by NULL (00_{16}) and begins with the “\” character (root directory).

7.20.3 RECORD FILE field validation checks

The RECORD FILE field validation checks table is shown in Table 7-45 below.

Table 7-45 – RECORD FILE field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
destination_plug	√	
signal_format	√	
record_mode	√	
file_path_length	-	00_{16}
file_path	-	00_{16}

7.20.4 Result codes for RECORD FILE command

The possible result codes for RECORD FILE command are summarized in Table 7-46 below.

Table 7-46 – Result codes for RECORD FILE command

ACCEPTED	STABLE	REJECTED
success		busy aborted disabled invalid generation count invalid volume number invalid subunit plug invalid parameter no media no file no directory no space file exist unsupported format volume protected file protected plug busy transport error no input unsupported stream invalid operation mode no operation unknown

7.21 SEND FILE command

The SEND FILE command is used to request the camera storage subunit to send the data to the specified source plug.

7.21.1 SEND FILE control command

The formats of the SEND FILE control command is illustrated in Figure 7-29 below.

	command format	response format
	msb Lsb	msb lsb
opcode	SEND FILE (50_{16})	<=
operand[0]	Subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	media_generation_count	media_generation_count
operand[5]	FF FF FF FF_{16}	<=
operand[6]		
operand[7]		
operand[8]		
operand[9]	source_plug	<=
operand[10]	Reserved	<=
operand[11]		
operand[12]	file_path_length	<=
operand[13]	file_path	<=
:		
:		

Figure 7-29 – SEND FILE control command and response format

The *source_plug* field specifies which subunit source plug will be used to send the file data. The controller shall not specify "any available source plug" (FF_{16}). Before issuing this command, the controller shall establish connection between camera storage subunit source plug and the unit plug or subunit destination plug of another subunit in the same unit. The controller shall specify the actual plug number to be used by the connection.

If the specified source plug is being used when the SEND FILE command is received, then the camera storage subunit shall return the REJECTED response with the result code of "busy." If the unit plug or subunit destination plug that is connected with the specified camera storage subunit source plug can't receive the specified file, then the camera storage subunit returns the REJECTED response with result code of "invalid file type." For example, if the controller issues the SEND FILE control command specifying audio file and the connected unit plug is configured to output NTSC signal and there is no conversion mechanism from specified audio file to NTSC signal, then the command is rejected with the result code of "invalid file type."

The *file_path_length* field specifies the size of the *file_path* in bytes. The length does not include the NULL character.

The *file_path* specifies the path name of the target file. The *file path* may be a path name or a URI as specified in Section 5.1.. If a URI is set as *file path* the physical volume number filed , logical volume number filed and media generation count field shall be ignored.

7.21.2 SEND FILE status command

The status command and STABLE response format for the SEND FILE command are illustrated in Figure 7-30 below.

	command format	response format
	msb lsb	msb lsb
opcode	SEND FILE (50_{16})	<=
operand[0]	source_plug	subfunction
operand[1]		result
operand[2]		physical_volume_number
operand[3]		logical_volume_number
operand[4]		media_generation_count
operand[5]		sent_size
operand[6]		
operand[7]		
operand[8]		
operand[9]		source_plug
operand[10]		reserved
operand[11]		
operand[12]		file_path_length
operand[13]		file_path
:		
:		

Figure 7-30 – SEND FILE status command and STABLE response format

The *source_plug* field in the command frame specifies the subunit source plug to query for status. If the specified plug does not exist on the camera storage subunit, then the NOT IMPLEMENTED shall be returned. In this case, the format and field values of the response frame are the same as the command frame.

If the camera storage subunit receives the SEND FILE status command while executing /suspending/aborting the SEND FILE control command that is using the specified source plug, then the camera storage subunit shall copy the field values of the executing control command to the status response frame (except for the *result* and *sent_size* field). If the camera storage subunit is not executing the control command, then the response frame contains only *subfunction*, *result*, *physical_volume_number*, *logical_volume_number* and *media_generation_count* fields, and all fields except the *result* field shall have the value of FF_{16}.

The *result* field in the response frame indicates command execution status. The values for this field are described in section 5.5. If the SEND FILE control command that is using the specified source plug is being executed, then the value of “executing” shall be set. If the SEND FILE control command that is using the specified source plug is suspended by bus reset, then the value of “suspended” shall be set. If the SEND FILE control command that is using the specified source plug is being aborted (ACCEPTED response for abort subfunction has been sent and REJECTED response for execute or resume subfunction has not been sent yet), then the value of “aborting” shall be set; otherwise, the value of “not executing” shall be set.

The *sent_size* field in the response frame indicates the approximate size (in bytes) of data already sent from the subunit. The size may be the exact sent size when the status command was received, a unit of the segment buffer of Asynchronous Connections, etc. If the camera storage subunit cannot determine the size, the value of $FFFFFFFF_{16}$ shall be set whenever the SEND FILE control command is being executed. When the size exceeds $FFFFFFFF_{16}$, the value of $FFFFFFFF_{16}$ should be set.

7.21.3 SEND FILE field validation checks

The SEND FILE field validation checks table is shown in Table 7-47 below.

Table 7-47 – SEND FILE field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
source_plug	√	
file_path_length	-	00_{16}
file_path	-	00_{16}

7.21.4 Result codes for SEND FILE command

The possible result codes for SEND FILE command are summarized in Table 7-48 below.

Table 7-48 – Result codes for SEND FILE command

ACCEPTED	STABLE	REJECTED
success	not executing executing suspended aborting	busy aborted retry no command disabled invalid generation count invalid volume number invalid subunit plug invalid parameter no media no file unsupported format plug busy transport error no proxy function proxy not available unsupported protocol unknown

7.22 SEND FILE PARTIAL command

The SEND FILE PARTIAL command is used to request the camera storage subunit to send part of data in the specified file to the specified source plug.

7.22.1 SEND FILE PARTIAL control command

The formats of the SEND FILE PARTIAL control command is illustrated in Figure 7-31 below.

	command format	response format
	msb lsb	msb lsb
opcode	SEND FILE PARTIAL (54_{16})	<=
operand[0]	Subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	media_generation_count	media_generation_count
operand[5]	start_offset	<=
:		
operand[12]		
operand[13]	send_size	sent_size
:		
operand[20]		
operand[21]	source_plug	<=
operand[22]	Reserved	<=
operand[23]		
operand[24]	file_path_length	<=
operand[25]	file_path	<=
:		
:		

Figure 7-31 – SEND FILE PARTIAL control command and response format

The meaning and function of each field except *start_offset, send_size* and *sent_size* are identical to those of the SEND FILE command.

The *start_offset* field specifies the starting byte offset of the data in the specified file to be sent. If the *start_offset* field specifies the byte offset exceed the byte size of specified file, the camera storage subunit reject the command with result code of “invalid parameter.”

The *send_size* field specifies the byte size of the data to be sent.

The *sent_size* field indicates the byte size of the data actually sent.

When the camera storage subunit receives the SEND FILE PARTIAL control command, the subunit opens the specified file and seek the offset point. Then the subunit starts sending the data and when the subunit finishes sending the specified amount of data or reaches to the end of the file, the subunit issues the final response to the controller. The *sent_size* field in the final response indicates the actual amount of sent data.

The *file_path* specifies the path name of the target file. The *file path* may be a path name or a URI as specified in Section 5.1.. If a URI is set as *file path* the physical volume number filed , logical volume number filed and media generation count field shall be ignored.

7.22.2 SEND FILE PARTIAL status command

The status command and STABLE response format for the SEND FILE PARTIAL command are illustrated in Figure 7-32 below.

	command format	response format
	msb lsb	msb lsb
opcode	SEND FILE PARTIAL (54_{16})	<=
operand[0]	source_plug	subfunction
operand[1]		result
operand[2]		physical_volume_number
operand[3]		logical_volume_number
operand[4]		media_generation_count
operand[5]		start_offset
:		
operand[12]		
operand[13]		sent_size
:		
operand[20]		
operand[21]		source_plug
operand[22]		reserved
operand[23]		
operand[24]		file_path_length
operand[25]		file_path
:		
:		

Figure 7-32 – SEND FILE PARTIAL status command and STABLE response format

The *source_plug* field in the command frame specifies the subunit source plug to query for status. If the specified plug does not exist on the camera storage subunit, then the NOT IMPLEMENTED shall be returned. In this case, the format and field values of the response frame are the same as the command frame.

If the camera storage subunit receives the SEND FILE PARTIAL status command while executing /suspending /aborting the SEND FILE PARTIAL control command that is using the specified source plug, then the camera storage subunit shall copy the field values of the executing control command to the status response frame (except for the *result* and *sent_size* field). If the camera storage subunit is not executing the control command, then the response frame contains only *subfunction*, *result*, *physical_volume_number*, *logical_volume_number* and *media_generation_count* fields, and all fields except the *result* field shall have the value of FF_{16}.

The *result* field in the response frame indicates command execution status. The values for this field are described in section 5.5. If the SEND FILE PARTIAL control command that is using the specified source plug is being executed, then the value of “executing” shall be set. If the SEND FILE PARTIAL control command that is using the specified source plug is suspended by bus reset, then the value of “suspended” shall be set. If the SEND FILE PARTIAL control command that is using the specified source plug is being aborted (ACCEPTED response for abort subfunction has been sent and REJECTED response for execute or resume subfunction has not been sent yet), then the value of “aborting” shall be set; otherwise, the value of “not executing” shall be set.

The *sent_size* field in the response frame indicates the approximate size (in bytes) of data already sent from the subunit. The size may be the exact sent size when the status command was received, a unit of the segment buffer of Asynchronous Connections, etc. If the camera storage subunit cannot determine the size, the value of $FFFFFFFF_{16}$ shall be set whenever the SEND FILE control command is being executed.

7.22.3 SEND FILE PARTIAL field validation checks

The SEND FILE PARTIAL field validation checks table is shown in Table 7-49 below.

Table 7-49 – SEND FILE PARTIAL field validation checks table

Fields	ck	recommended values
Subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
start_offset	-	00 00 00 00 00 00 00 00_{16}
send_size	-	00 00 00 00 00 00 00 00_{16}
source_plug	√	
file_path_length	-	00_{16}
file_path	-	00_{16}

7.22.4 Result codes for SEND FILE PARTIAL command

The possible result codes for SEND FILE PARTIAL command are summarized in Table 7-50 below.

Table 7-50 – Result codes for SEND FILE PARTIAL command

ACCEPTED	STABLE	REJECTED
success	not executing executing suspended aborting	busy aborted retry no command disabled invalid generation count invalid volume number invalid subunit plug invalid parameter no media no file unsupported format plug busy transport error no proxy function proxy not available unsupported protocol unknown

7.23 SEND THUMBNAIL command

The SEND THUMBNAIL command is used to trigger sending the thumbnail data of a specified file such as still image file, movie file, etc.

7.23.1 SEND THUMBNAIL control command

The formats of the SEND THUMBNAIL control command and response are illustrated in Figure 7-33 below. The meaning and function of each field are identical to those of the SEND FILE command. If

camera storage subunit doesn't support thumbnail data of specified file, then the command shall be rejected with the result code of "invalid file type".

	command format	response format
	msb lsb	msb lsb
opcode	SEND THUMBNAIL (51_{16})	<=
operand[0]	subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	Media_generation_count	media_generation_count
operand[5]	FF FF FF FF_{16}	<=
operand[6]		
operand[7]		
operand[8]		
operand[9]	source_plug	<=
operand[10]	reserved	<=
operand[11]		
operand[12]	file_path_length	<=
operand[13]	file_path	<=
:		
:		

Figure 7-33 – SEND THUMBNAIL control command and response format

The structure of thumbnail data sent from the camera storage subunit is compatible with DCF basic thumbnail data explained in reference [R12]. If the file specified by *file_path* is an Exif file, then the camera storage subunit extracts the JPEG compressed DCF basic thumbnail data and sends the data. If the DCF thumbnail file of the specified file exists, then the camera storage subunit extracts JPEG compressed thumbnail data from the thumbnail file and sends the data. In both cases, the camera storage subunit shall send only the JPEG compressed data portion(from SOI to EOI). When an Exif info header exists in the thumbnail file, the header shall be discarded..

If specified file is not DCF file and includes thumbnail data or thumbnail file for specified file exists, camera storage subunit extracts thumbnail data and , if necessary, converts the image format to compatible with DCF basic thumbnail, then sends the data.

If there is no thumbnail data or thumbnail file for a specified image file, the REJECTED response with the result code of "no thumbnail" shall be returned

7.23.2 SEND THUMBNAIL status command

The status command and STABLE response format for the SEND FILE command is illustrated in Figure 7-34 below. The usage of each field is identical to that of the SEND FILE status command described in section 7.21.2.

	command format	response format
	msb lsb	msb lsb
opcode	SEND THUMBNAIL (51_{16})	<=
operand[0]	source_plug	subfunction
operand[1]		result
operand[2]		physical_volume_number
operand[3]		logical_volume_number
operand[4]		media_generation_count
operand[5]		sent_size
operand[6]		
operand[7]		
operand[8]		
operand[9]		source_plug
Operand[10]		reserved
Operand[11]		
Operand[12]		file_path_length
Operand[13]		file_path
:		
:		

Figure 7-34 – SEND THUMBNAIL status command and STABLE response format

7.23.3 SEND THUMBNAIL field validation checks

The SEND THUMBNAIL field validation checks table is shown in Table 7-51 below.

Table 7-51 – SEND THUMBNAIL field validation checks table

fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	
media_generation_count	-	00_{16}
source_plug	√	
file_path_length	-	00_{16}
file_path	-	00_{16}

7.23.4 Result codes for SEND THUMBNAIL command

The possible result codes for SEND THUMBNAIL command are summarized in Table 7-52 below.

Table 7-52 – Result codes for SEND THUMBNAIL command

ACCEPTED	STABLE	REJECTED
success	not executing executing suspended aborting	busy aborted retry no command disabled invalid generation count invalid volume number invalid subunit plug invalid parameter no media no file unsupported format invalid file type no thumbnail plug busy transport error unknown

7.24 VERSION command

The VERSION status command can be used to obtain the version information of the subunit specification. It can be used to obtain the latest specification version information, or to inquire whether or not a specific version number is supported.

7.24.1 Obtain the latest version information

The VERSION status command can be used to obtain the latest version of the subunit specification. The formats of the command and STABLE response frames are illustrated in Figure 7-35 below.

	command format	response format
	msb | | | | | | | lsb	msb | | | | | | | lsb
opcode	VERSION ($B0_{16}$)	<=
operand[0]	FF_{16}	subunit_version_information
operand[1]	FF_{16}	subunit_version_information_dependent_field
:	FF .. FF_{16}	
operand[32]	FF_{16}	

Figure 7-35 – VERSION status command and response format for the latest version

The *subunit_version_information* field in the response frame indicates the latest version number of the subunit specification it implements. The definition of this field depends on each subunit specification. For AV/C Camera Storage Subunit Specification, the value of this field is defined in Table 7-53 below.

Table 7-53 – *subunit_version_information* field definitions

subunit_version_information	values	meaning
Version 1.0	10_{16}	Version 1.0 of the camera storage subunit specification
Version 2.0	20_{16}	Version 2.0 of the camera storage subunit specification
Version 2.1	21_{16}	Version 2.1 of the camera storage subunit specification
no info	FF_{16}	no information
reserved	all others	reserved for future extension

The *subunit_version_information_dependent_field* indicates the list of *implementation_profile_id* defined in the specification that is specified by the *subunit_version_information* filed. This field can include up to 32 profiles. If the camera storage subunit implements fewer than 32 profiles, then they are terminated by a byte with the value FF_{16}. The value of entries past the terminating FF_{16} is indeterminate and shall be ignored by any controller that requests profile information. The values of *implementation_profile_id* are defined in Table 6.1エラー! 参照元が見つかりません。.

7.24.2 Obtain the support level of the specified version

The VERSION status command can also be used for obtaining the support level of the specified version of the subunit specification. The format of the command and STABLE response frames are illustrated in Figure 7-36 below.

	command format								response format							
	msb							lsb	msb							lsb
opcode	VERSION ($B0_{16}$)								<=							
operand[0]	subunit_version_information								<=							
operand[1]	FF_{16}								subunit_version_information_dependent_field							
:	FF .. FF_{16}															
operand[32]	FF_{16}															

Figure 7-36 – VERSION status command and response format for the specified version

If the subunit is designed to support the specification specified by the *subunit_version_information* field value, then it returns the STABLE response with the *subunit_version_information_dependent_field* set to indicate its support information. The definitions of the *subunit_version_information* and *subunit_version_information_dependent_field* are the same as previous section 7.24.1.

If the subunit does not support the specification specified by the *subunit_version_information* field value, then it returns the NOT IMPLEMENTED response.

7.25 VOLUME INFO command

The VOLUME INFO command is used to retrieve the information of the specified volume.

7.25.1 VOLUME INFO control command

The formats of the VOLUME INFO control command and response are illustrated in Figure 7-37 below.

	command format	response format
	msb lsb	msb lsb
opcode	VOLUME INFO (41_{16})	<=
operand[0]	subfunction	<=
operand[1]	FF_{16}	result
operand[2]	physical_volume_number	<=
operand[3]	logical_volume_number	<=
operand[4]	FF_{16}	media_generation_count
operand[5]	FF_{16}	attribute
operand[6]	00_{16}	character_set
operand[7]	reserved	<=
operand[8]		
operand[9]	FF FF FF FF FF FF FF FF_{16}	maximum_capacity
:		
operand[16]		
operand[17]	FF FF FF FF FF FF FF FF_{16}	free_space
:		
operand[24]		

Figure 7-37 – VOLUME INFO control command and response format

The *attribute* field in the response frame indicates the status attribute of the specified volume. The *attribute* field consists of bit fields as defined in Table 7-54 below. If the camera storage subunit cannot determine a particular status, then the corresponding field shall have the value of zero.

Table 7-54 – Bit assignment of the *attribute* field of the VOLUME INFO command

bit offset	description
bit 0 (msb)	reserved for future specification
bit 1	
bit 2	
bit 3	
bit 4	
bit 5	
bit 6	write_protected
bit 7 (lsb)	read_only_media

If *read_only_media* bit is set, then the specified volume has read only media.

If *write_protected* bit is set, then the specified volume is write protected.

The value of the reserved bits shall be zero.

The *character_set* field in the response frame indicates the character set used by the specified volume. The file paths of the files on the volume are described using the character set. The values of this field are shown in Table 7-55 below.

Table 7-55 – *character_set* values of the VOLUME INFO command

character_set	value	description
unspecified	00_{16}	character set is unspecified
UTF8	01_{16}	UTF8 of unicode
UTF16	02_{16}	UTF16 of unicode
reserved	all others	reserved for future specification

If camera storage subunit cannot determine the character set used by the specified volume, the value of "unspecified" may be set.

The *maximum_capacity* field in the response frame indicates the maximum capacity (in bytes) of the specified volume.

The *free_space* field in the response frame indicates the remaining free space (in bytes) of the specified volume.

When the control command is rejected, each field of the response frame contains the same value of the control command frame except for the *result* and *media_generation_count* field.

7.25.2 VOLUME INFO field validation checks

The VOLUME INFO field validation checks table is shown in Table 7-56 below.

Table 7-56 – VOLUME INFO field validation checks table

Fields	ck	recommended values
subfunction	√	
physical_volume_number	√	
logical_volume_number	√	

7.25.3 Result codes for VOLUME INFO command

The possible result codes for VOLUME INFO command are summarized in Table 7-57 below.

Table 7-57 – Result code for VOLUME INFO command

ACCEPTED	STABLE	REJECTED
success	not executing executing	busy retry disabled invalid volume number no media unsupported format unknown
	aborting	aborted no command

If the camera storage subunit implements "abort" subfunction for VOLUME INFO command, then "aborting", "aborted" and "no command" may return.

7.26 PROXY INFO command

The PROXY INFO command is used to obtain the information of the proxy function.

7.26.1 PROXY INFO status command

The PROXY INFO status command can be used to obtain the information of the proxy function. The formats of the command and STABLE response frames are illustrated in Figure 7-49 below.

	command format							
	msb							lsb
opcode	PROXY INFO (48_{16})							
operand[0]	Subfunction							
operand[1]	Result							
operand[2]	FF_{16} ….FF_{16}							
:								
operand[32]								

Figure 7-389 – PROXY INFO status command format

The *subfunction* field be taken by the target as defined in the Table 7-58 below.

Table 7-59 – subfunction field definitions of PROXY INFO status command

Symbol	value	Meaning
query_protocol	00_{16}	Get the supports protocol for proxy service
Reserved	all others	reserved for future specification

7.26.2 Obtain the supported protocol

The PROXY INFO status command can be used to obtain the information of the proxy function. The formats of the command and STABLE response frames are illustrated in Figure 7-50 below.

	command format	response format
	msb lsb	msb lsb
opcode	PROXY INFO (48_{16})	<=
operand[0]	query_protocol	<=
operand[1]	FF_{16}	Result
operand[2]	FF_{16}	supported_protocol_ID[0]
operand[3]	FF_{16}	supported_protocol_ID[1]
:	:	:
operand[32]	FF_{16}	supported_protocol_ID[30]

Figure 7-39 – VOLUME INFO status command and response format

The value of *supported_protocol_ID* is defined as follows.

Table 7-60 – *supported_protocol_ID* values of the VOLUME INFO command

value	supported_protocol_ID	remarks
00_{16}	http	Mandatory protocol
01_{16}	https	Optional
02_{16}	ftp	Optional
03_{16}	ftps	Optional
FF_{16}	no information	Optional
all others	reserved	reserved for future specification

Http is mandatory protocol . So *supported_protocol_ID[0]* filed should be set 00_{16}.

7.26.3 Result codes of PROXY INFO command

The possible result codes for PROXY INFO command are summarized Table 7-61 below.

Table 7-62 – Result code for PROXY INFO command

ACCEPTED	STABLE	REJECTED
Success	not executing executing	busy retry disabled unknown
	aborting	aborted no command

.

Annexes

Annex A. Command and data flow (normative)

A.1 Standard communication sequences

This section contains examples of standard communication sequences held between the controller and the camera storage subunit. The order of the commands is roughly illustrated in below.

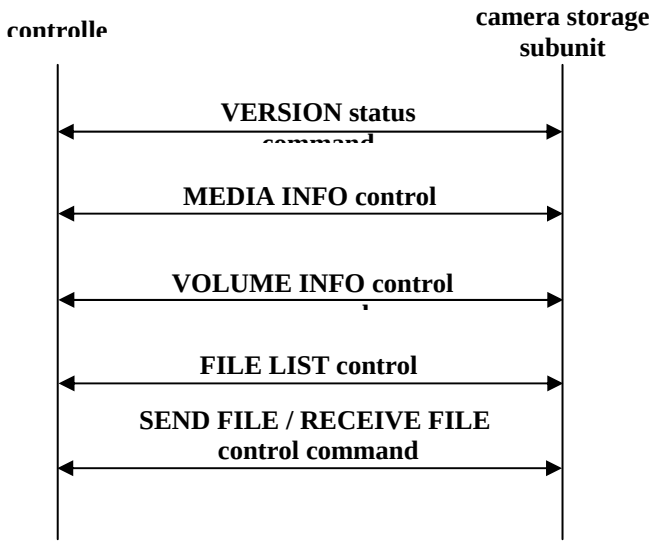


Figure A-1 – Example of command order in the standard communication sequence

A.1.1 Subunit discovery and acquisition of subunit information

To detect the camera storage subunit, for example, the controller may issue the VERSION status command to all nodes. If the node implements the camera storage subunit, then the controller receives the STABLE response and the camera storage subunit is detected. The STABLE response includes subunit version and profile information regarding the camera storage subunit. The example of communication sequence is illustrated in Figure A-2 below.

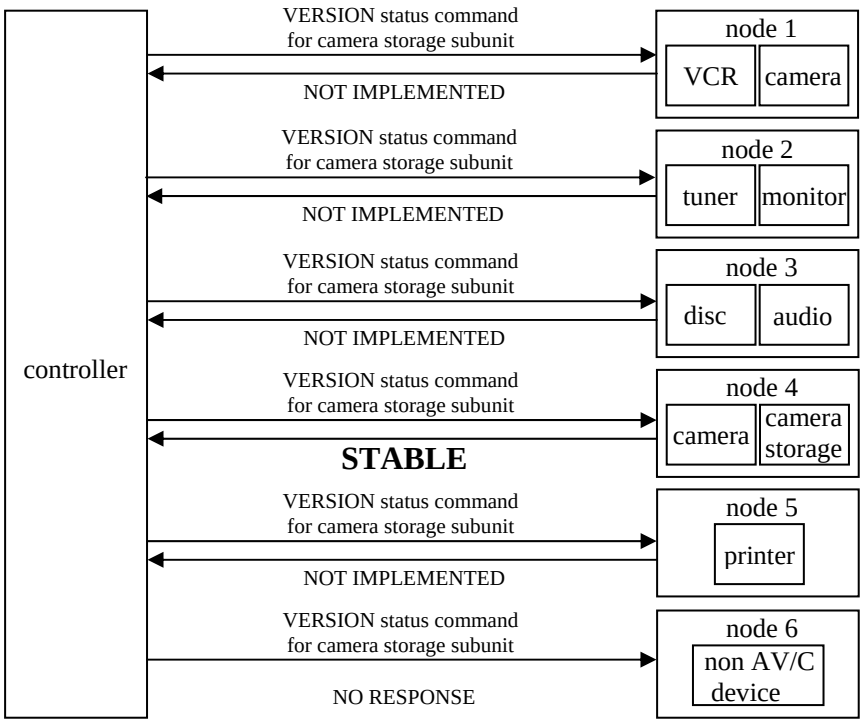


Figure A-2 – Example of communication sequence to detect camera storage subunit

The controller may also use the SUBUNIT INFO status command described in [R7] to detect the camera storage subunit. In this case, the controller may issue the VERSION status command to the detected camera storage subunit in order to check the implemented subunit version and profile information.

When multiple camera storage subunits are detected on the 1394 bus, the selection method depends on the implementation of the controller. The method is beyond the scope of this specification.

The controller may configure its application by retrieved subunit version and profile if necessary.

A.1.2 Contents discovery and file data transfer

The controller supporting multiple volumes should issue the MEDIA INFO control command to retrieve the number of volumes. If the controller application supports only one physical volume and only one logical volume, then the controller may skip this process.

After retrieving the information, the controller application selects and accesses one of the volumes.

A.1.2.1 Sequence under normal condition

First, before the controller accesses a specific volume, the controller should issue the VOLUME INFO control command to retrieve the current *media_generation_count* value. The controller should set the retrieved value to following command frame to access the volume.

Then the controller issues the FILE LIST control command to retrieve the file list. For detailed sequence of issuing the FILE LIST control command, please refer to Annex B.

After retrieving the file list or directory structure by using the FILE LIST control command, the controller issues the SEND FILE / RECEIVE FILE control command to transfer the file data.

If the controller assumes that there is a specific information file on the volume, then the controller may issue the SEND FILE control command without the FILE LIST control command. For example, some digital cameras support DPOF (Digital Print Order Format) file. This file provides information for printing, including the full path name of the files to be printed. The path name of DPOF file is fixed ("/MISC/AUTPRINT.MRK") and the controller may issue the SEND FILE control command with this path name. Then the controller analyzes the DPOF file and may issue the SEND FILE control command to retrieve image file data using the full path name described in the DPOF file. In this sequence, the controller does not need to issue the FILE LIST control command.

A.1.2.2 Sequence after detection of media change

When storage media (physical volume) is inserted, the camera storage subunit increases the value of *media_generation_count* by one. If the control command is rejected and value of *result* field is "invalid generation count," then the physical volume has exchanged. In this case, the REJECTED response frame includes new *media_generation_count* value. So, the controller should retrieve a new file list by way of the FILE LIST control command using a new *media_generation_count* value. Because the physical volume has changed, the controller should issue the MEDIA INFO control command and VOLUME INFO control command if necessary. Figure A-3 below illustrates the example of sequence after detection of media change.

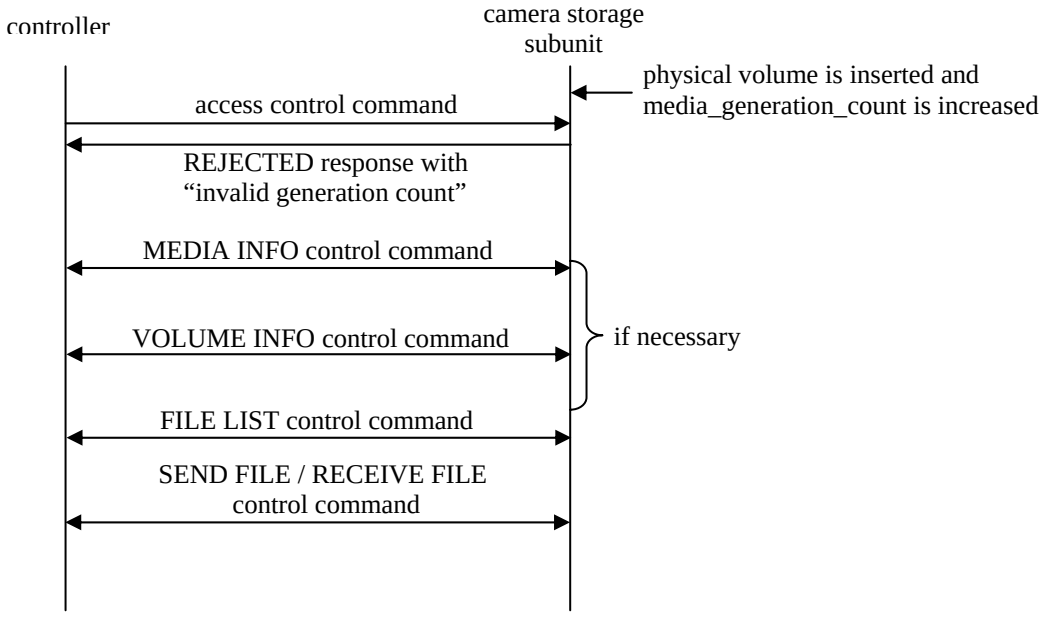


Figure A-3 – Example of sequence after detection of media change

A.2 Data transfer sequences

This section describes the command and data flow sequences of a normal data transferring process and a resuming process after bus reset.

A.2.1 Normal sequence for sending

After retrieving the file list, the controller issues the SEND FILE or SEND THUMBNAIL commands to request data transfer. The controller shall setup the connection between the camera storage subunit source plug and the destination before issuing the trigger command. Figure A-4 below illustrates the sequence undertaken between the controller and the camera storage subunit. This figure illustrates the example of the controller as consumer also.

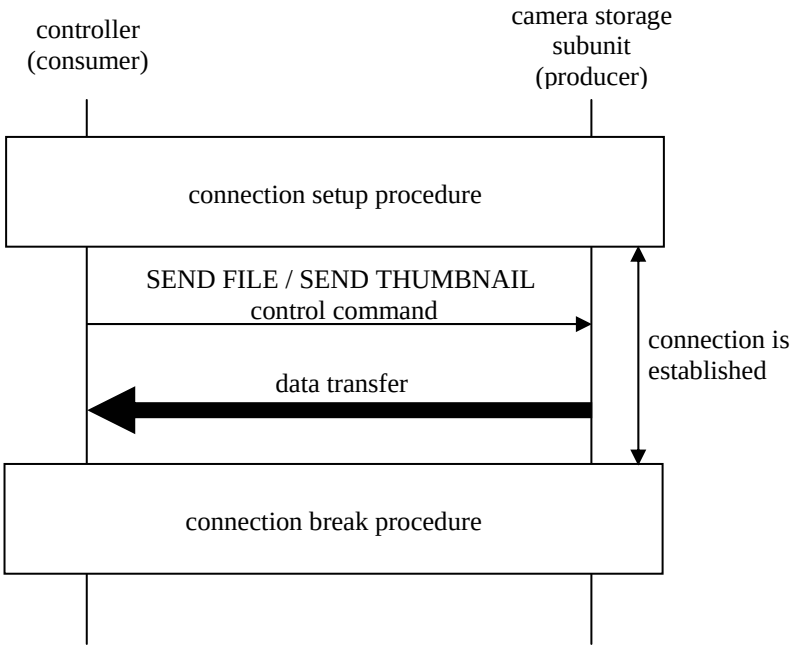


Figure A-4 – Example of normal sequence for sending

While connection is established, the controller may issue multiple SEND FILE or SEND THUMBNAIL commands to retrieve multiple data items. When data transfer is complete, the controller may break the connection.

A.2.2 Resuming sequence for sending after bus reset

When bus reset occurs while data is being transferred, the controller should execute the resuming process. After detecting bus reset, the controller shall first resume connection and then issue a trigger command with the subfunction “resume.” The following figure illustrates the resuming process sequence.

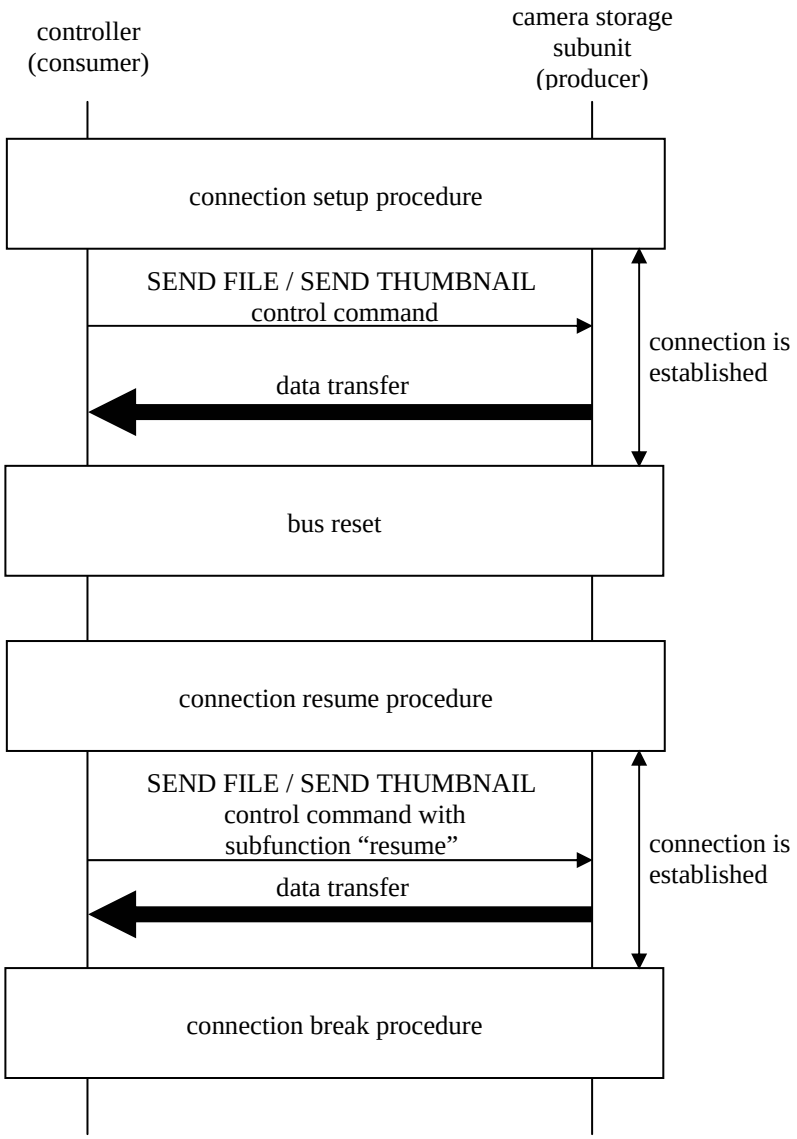


Figure A-5 – Example of resuming sequence for sending after bus reset

The camera storage subunit shall not restart sending before receiving trigger command with subfunction “resume”. If trigger command arrives before connection is resumed, the command shall be rejected.

A.2.3 Normal sequence for receiving

After retrieving the volume information, the controller issues RECEIVE FILE commands to request receiving the file data. The controller shall setup the connection between the camera storage subunit destination plug and source of the file before issuing trigger command. Figure A-6 illustrates the sequence between controller and camera storage subunit. This figure shows the example that the controller is also producer.

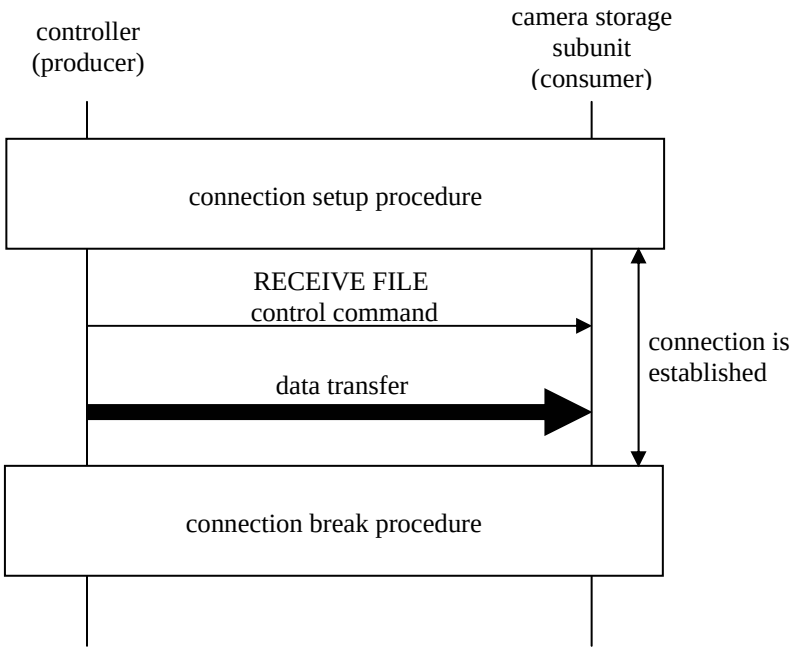


Figure A-6 – Example of normal sequence for receiving

While connection is established, the controller may issue plural RECIEVE FILE commands to send plural file data. After completion of file data transfer, the controller may break the connection.

In the case where data arrives after connection is setup but before the trigger command arrives, the camera storage subunit shall discard the data. So, when the controller serves as both consumer and producer, the controller shall issue the commands in the following order.

1) connection setup commands to consumer and producer
2) trigger command to consumer
3) trigger command to producer after receiving response from consumer

A.2.4 Resuming sequence for receiving after bus reset

When bus reset occurs during data transfer, the controller should execute the resuming process. After detecting bus reset, the controller shall first resume connection and then issue the trigger command with the subfunction “resume.” Figure A-7 illustrates the sequence of a resuming process.

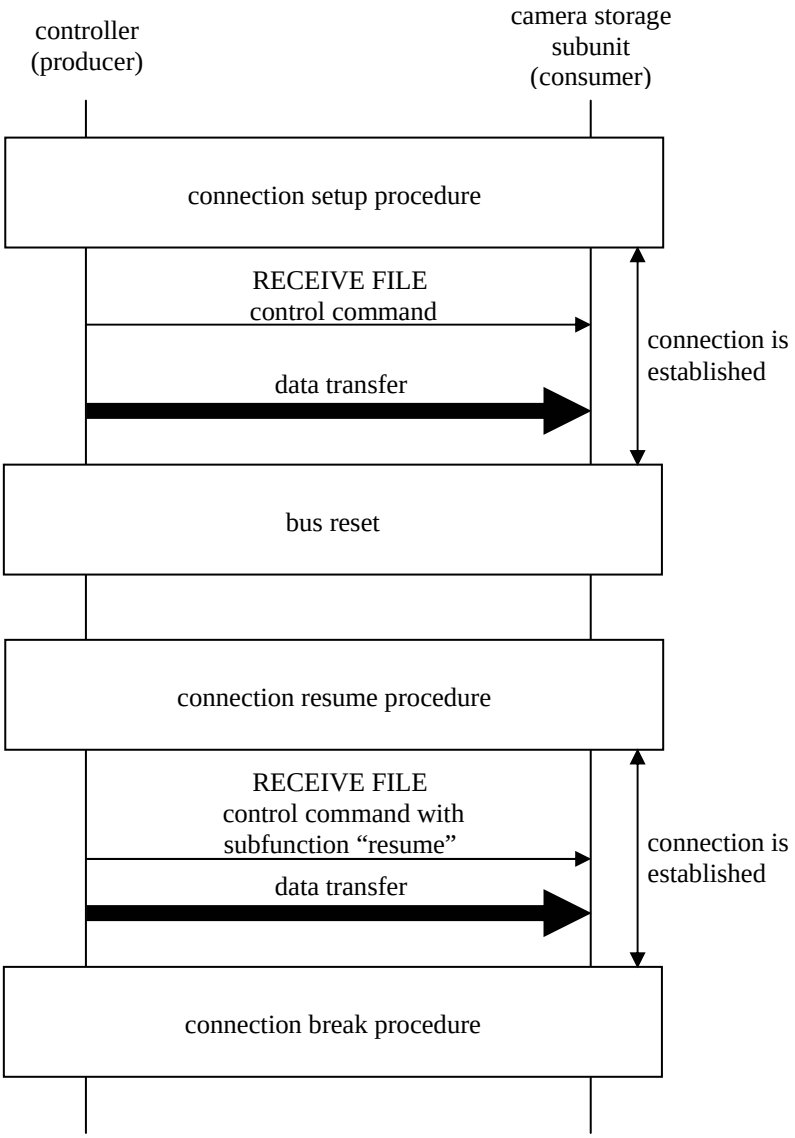


Figure A-7 – Example of resuming sequence for receiving after bus reset

If the RECEIVE FILE command with the subfunction “resume” arrives before connection is resumed, then the command shall be rejected. In the case where data arrives after connection is resumed but before the trigger command arrives, the camera storage subunit shall discard the data and shall return the REJECTED response when the trigger command with a “resume” subfunction arrives. So, when the controller serves as both consumer and producer, the controller shall resume in the following order.

4) resume connection between consumer and producer
5) issue trigger command with subfunction “resume” to consumer
6) issue trigger command with subfunction “resume” to producer after receiving response from consumer

Annex B. Example of procedure (informative)

B.1 Procedure for querying file list

This section describes the command and data flow for the file list querying procedure.

Suppose the directory structure shown in Figure B-1 below:

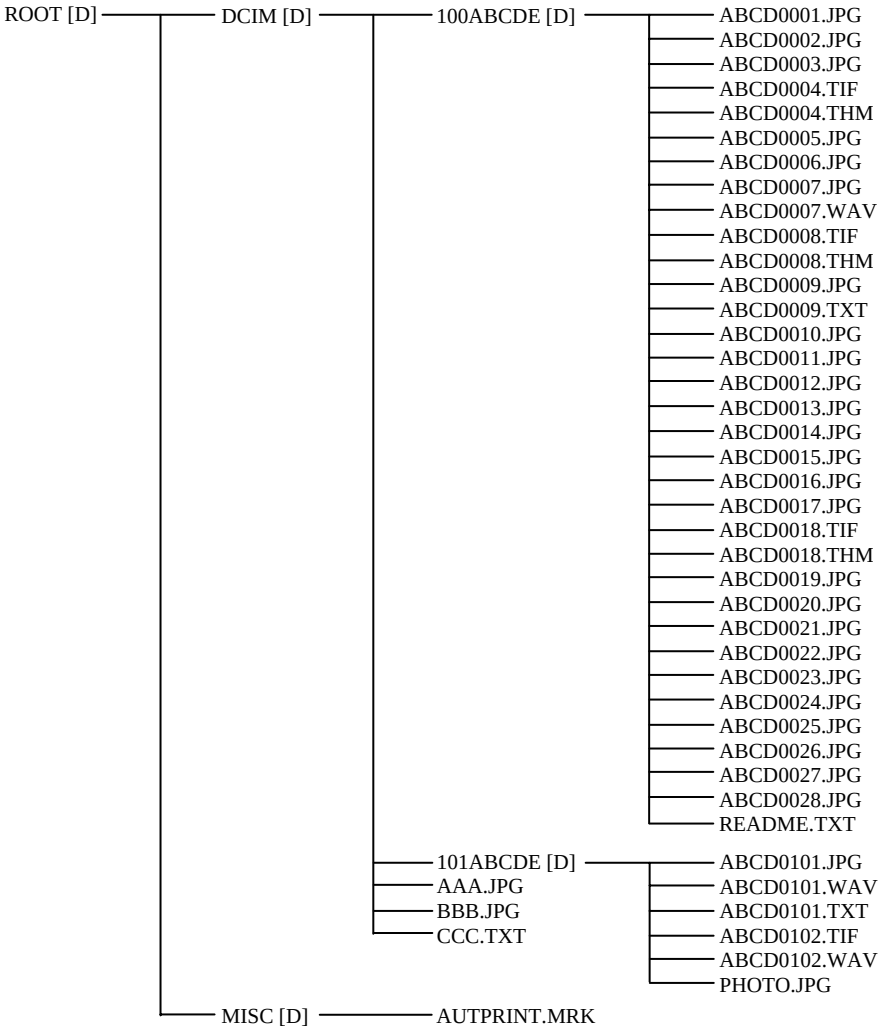


Figure B-1 – Example of directory structure

The entries marked “[D]” are directories and the others are files. In the following examples, the camera storage subunit handles files of extension “JPG” or “TIF” as image files.

On “/DCIM/100ABCDE” directory, there are 28 image files (25 JPG files and 3 TIF files), and on “/DCIM/101ABCDE” directory, there are 3 image files (2 JPG files and 1 TIF file).

B.1.1 Procedure for querying file lists of still image files on DCF directories

This clause explains the procedure for querying file lists of still image files on DCF directories.

First, the controller sets the following values for each field and issues the FILE LIST command to retrieve the directory entries of the DCF directory.

Table B-1 – FILE LIST command values for the DCF directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	02_{16} (directory)
start_number	0000_{16}
number_of_entries	18_{16}
request_path_length	05_{16}
request_path	“\DCIM”+00_{16}

The camera storage subunit receives the command and since the start_number is 0000_{16}, the camera storage subunit returns directory entries of subdirectories from the top of the entry list. There is no “DCIM” file in the root directory and a “DCIM” subdirectory exists, so the camera storage subunit determines that this *request_path* specifies the directory path and returns the directory entries of subdirectories existing in the specified directory. In this case, 2 subdirectories exist in the "\DCIM" directory, so the camera storage subunit returns 2 directory entries. No other entry follows, so the camera storage subunit shall set the *end_of_list* field value to 1.

Table B-2 – FILE LIST response values for DCF directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	02_{16} (directory)
start_number	0000_{16}
end_of_list	1_{2}
number_of_entries	02_{16}

The controller receives the directory entries of the DCF directories, then sets the following values to each field and issues the FILE LIST command to retrieve directory entries of still image files in the first DCF directory.

Table B-3 – First FILE LIST command values for still image in the first DCF directory

field name	value
file_type	01_{16} (still image)
list_mode	0_{16} (short entry)
attribute	01_{16} (file)
start_number	0000_{16}
number_of_entries	18_{16}
request_path_length	$0E_{16}$
request_path	“\DCIM\100ABCDE”+00_{16}

The camera storage subunit receives the command and since the start_number is 0000_{16}, the camera storage subunit returns 24 directory entries from the top of the entry list of image files. There is no “100ABCDE” file in “\DCIM” directory and the “100ABCDE” subdirectory exists, so the camera storage subunit determines that this *request_path* specifies the directory path and returns the directory entries of still image files existing in “\DCIM\100ABCDE” directory. In this case, the camera storage subunit returns 24

directory entries, but other entries exist following 24th entry. So, the camera storage subunit shall set the value the *end_of_list* field to 0.

Table B-4 – First FILE LIST response values for still image in the first DCF directory

field name	value
file_type	01_{16} (still image)
list_mode	0_{16} (short entry)
attribute	01_{16} (file)
start_number	0000_{16}
end_of_list	0_{2}
number_of_entries	18_{16}

The controller receives the directory entry list and by checking the *end_of_list* field, recognizes that following entries exist. Then the controller sets the following values to each field and issues the FILE_LIST command to retrieve the next directory entries.

Table B-5 – Second FILE LIST command values for still image in the first DCF directory

field name	value
file_type	01_{16} (still image)
list_mode	0_{16} (short entry)
attribute	01_{16} (file)
start_number	0018_{16}
number_of_entries	18_{16}
request_path_length	$0E_{16}$
request_path	“\DCIM\100ABCDE”+00_{16}

The camera storage subunit receives the command and since the start_number is 0018_{16}, the camera storage subunit returns the directory entries from the 25th in the entry list on, based on *request_path*. In this case, there are 4 remaining directory entries, so the camera storage subunit returns 4 directory entries from the 25th entry in the list of image files. Since there are no more entries, the camera storage subunit shall set the value of the *end_of_list* field to 1.

Table B-6 – Second FILE LIST response values for still image in the first DCF directory

field name	value
file_type	01_{16} (still image)
list_mode	0_{16} (short entry)
attribute	01_{16} (file)
start_number	0018_{16}
end_of_list	1_{2}
number_of_entries	04_{16}

Then the controller sets the following values to each field and issues the FILE LIST command to retrieve directory entries of still image files on the second DCF directory.

Table B-7 – FILE LIST command values for still image in the second DCF directory

field	value
file_type	01_{16} (still image)
list_mode	0_{16} (short entry)
attribute	01_{16} (file)
start_number	0000_{16}
number_of_entries	18_{16}
request_path_length	$0E_{16}$
request_path	"\DCIM\101ABCDE"+00_{16}

The camera storage subunit receives the command and since the start_number is 0000_{16}, the camera storage subunit returns directory entries from the top of the entry list of image files. There is no "101ABCDE" file in "\DCIM" directory and the "101ABCDE" subdirectory exists, so the camera storage subunit determines that this *request_path* specifies the directory path and returns the directory entries of still image files existing in the "\DCIM\101ABCDE" directory. The controller requests 24 entries, but there are only 3 image files. So, the camera storage subunit returns only 3 directory entries and the camera storage subunit shall set the value of the *end_of_list* field to 1.

Table B-8 – FILE LIST response values for still image in the second DCF directory

field	value
file_type	01_{16} (still image)
list_mode	0_{16} (short entry)
attribute	01_{16} (file)
start_number	0000_{16}
end_of_list	1_2
number_of_entries	03_{16}

In this example, the file list includes directory entries of JPG files and TIF files. The file types included in the file list depend on the implementation of the camera storage subunit and there may be entries that are not supported by the controller. So, the controller shall select the directory entries from retrieved file list.

The *start_number* does not need to specify the next entry from the previous retrieval command. The controller may set the *start_number* field to any value. When the directory entry specified by the *start_number* field does not exist, the camera storage subunit shall return the REJECTED response to the controller with the result code of "invalid parameter".

B.1.2 Procedure for querying the file list of all entries

This section explains the procedure for querying the file list of all entries.

First, the controller sets the following values to each field and issues the FILE LIST command to retrieve the directory entries of files and subdirectories in the root directory.

Table B-9 – FILE LIST command values for root directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	0000_{16}
number_of_entries	18_{16}
request_path_length	01_{16}
request_path	"\"+00_{16}

The camera storage subunit receives the command and since the *start_number* is 0000_{16}, the camera storage subunit returns the directory entries of subdirectories and files from the top of the entry list. In this case, 2 subdirectories and no files exist in the root directory, so the camera storage subunit returns 2 directory entries of subdirectories. No other entry follows, so the camera storage subunit shall set the value of the *end_of_list* field to 1.

Table B-10 – FILE LIST response values for root directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	0000_{16}
end_of_list	1_{2}
number_of_entries	02_{16}

The controller receives the directory entries and, by examining the *attribute* field of each entry, recognizes that there are 2 directory entries of subdirectories and no entries of files. Then the controller sets the following values to each field and issues the FILE LIST command to retrieve directory entries of subdirectories and files in "/DCIM" directory.

Table B-11 – FILE LIST command values for "/DCIM" directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	0000_{16}
number_of_entries	18_{16}
request_path_length	05_{16}
request_path	"\DCIM"+00_{16}

The camera storage subunit receives the command and since the *start_number* is 0000_{16}, the camera storage subunit returns directory entries of subdirectories and files from the top of the entry list. There is no "DCIM" file in the root directory and the "DCIM" subdirectory exists, so the camera storage subunit determines that this *request_path* specifies the directory path and returns the directory entries of subdirectories and files existing in the specified directory. In this case, 2 subdirectories and 3 files exist in "\DCIM" directory, so the camera storage subunit returns 5 directory entries. No other entry follows, so the camera storage subunit shall set the value of the *end_of_list* field to 1.

Table B-12 – FILE LIST response values for "/DCIM" directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and files)
start_number	0000_{16}
end_of_list	1_2
number_of_entries	05_{16}

The controller receives the directory entries and, by examining the *attribute* field of each entry, recognizes there are 2 directory entries of subdirectories and 3 entries of files. Then the controller sets the following values to each field and issues the FILE LIST command to retrieve directory entries of subdirectories and files in "/DCIM/100ABCDE" directory.

Table B-13 – First FILE LIST command values for "\DCIM\100ABCDE" directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	0000_{16}
number_of_entries	18_{16}
request_path_length	$0E_{16}$
request_path	"\DCIM\100ABCDE"+00_{16}

The camera storage subunit receives the command and since the start_number is 0000_{16}, the camera storage subunit returns 24 directory entries from the top of the entry list. There is no "100ABCDE" file in "\DCIM" directory and the "100ABCDE" subdirectory exists, so the camera storage subunit determines that this *request_path* specifies the directory path and returns the directory entries of subdirectories and files existing in "\DCIM\100ABCDE" directory. In this case, the camera storage subunit returns 24 directory entries, but other entries exist following 24th entry. So, the camera storage subunit shall set the value of the *end_of_list* field to 0.

Table B-14 – First FILE LIST response values for "\DCIM\100ABCDE" directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	0000_{16}
end_of_list	0_2
number_of_entries	18_{16}

The controller receives the directory entry list and, by checking the *end_of_list* field, recognizes that following entries exist. Then the controller sets the following values to each field and issues the FILE_LIST command to retrieve the next directory entries.

Table B-15 – Second FILE LIST command values for "\DCIM\100ABCDE" directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	0018_{16}
number_of_entries	18_{16}
request_path_length	$0E_{16}$
request_path	"\DCIM\100ABCDE"+00_{16}

The camera storage subunit receives the command and since the start_number is 0018_{16}, the camera storage subunit returns the directory entries from the 25th entry in the entry list, based on the *request_path*. In this case, the number of the rest of the directory entries is 10, so the camera storage subunit returns 10 directory entries from the 25th entry in the list of directories and files. Since there are no more entries, the camera storage subunit shall set the value of the *end_of_list* field to 1.

Table B-16 – Second FILE LIST response values for "\DCIM\100ABCDE" directory

field name	value
file_type	01_{16} (still image)
list_mode	0_{16} (short entry)
attribute	01_{16} (file)
start_number	0018_{16}
end_of_list	1_2
number_of_entries	$0A_{16}$

The controller retrieves the file list and, by checking the *eol* bit, recognizes that there are no more entries. Then the controller recognizes that there are no subdirectories in the "\DCIM\100ABCDE" directory by examining the *attribute* field of each directory entry.

Then the controller sets the following values to each field and issues the FILE LIST command to retrieve the directory entries of directories and files in the "/DCIM/101ABCDE" directory.

Table B-17 – FILE LIST command values for "\DCIM\101ABCDE" directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	0000_{16}
number_of_entries	18_{16}
request_path_length	$0E_{16}$
request_path	"\DCIM\101ABCDE"+00_{16}

The camera storage subunit receives the command and since the start_number is 0000_{16}, the camera storage subunit returns directory entries from the top of the entry list of directories and files. There is no "101ABCDE" file in "\DCIM" directory and the "101ABCDE" subdirectory exists, so the camera storage subunit determines that this *request_path* specifies the directory path and returns the directory entries of the directories and files existing in the "\DCIM\101ABCDE" directory. The controller requests 24 entries, but there are only 6 files. So, the camera storage subunit returns only 6 directory entries and shall set the value of the *end_of_list* field to 1.

Table B-18 – FILE LIST response values for "\DCIM\101ABCDE" directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	0000_{16}
end_of_list	1_2
number_of_entries	06_{16}

Then the controller sets the following values to each field and issues the FILE LIST command to retrieve the directory entries of directories and files in the "/MISC" directory.

Table B-19 – FILE LIST command values for "\MISC" directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	0000_{16}
number_of_entries	18_{16}
request_path_length	05_{16}
request_path	"\MISC"+00_{16}

The camera storage subunit receives the command and since the start_number is 0000_{16}, the camera storage subunit returns the directory entries from the top of the entry list of directories and files. There is no "MISC" file in the root directory and the "MISC" subdirectory exists, so the camera storage subunit determines that this *request_path* specifies the directory path and returns the directory entries of the directories and files existing in the "\MISC" directory. The controller requests 24 entries, but there is only 1 file. So, the camera storage subunit returns only 1 directory entry and shall set the value of the *end_of_list* field to 1.

Table B-20 – FILE LIST response values for "\MISC" directory

field name	value
file_type	00_{16} (all)
list_mode	0_{16} (short entry)
attribute	03_{16} (directory and file)
start_number	00_{16}
end_of_list	1_2
number_of_entries	01_{16}

The subdirectory order in which entries are queried is open. The controller can query in the order of "/MISC", "/DCIM", "/DCIM/101ABCDE," and then "/DCIM/101ABCDE". The camera storage subunit shall accept any order.

Annex C. Camera storage subunit state machine (informative)

C.1 Code definitions

All of the functions and definitions in this section use the syntax specified in Table C-1 and Table C-3 below.

Table C-1 – State machine code definitions

```
#define RESP_TIMEOUT_LIMIT  appropriate_value_in_milisecond
// When camera storage subunit receives AV/C control command and processing of the command takes
// longer time than this value, the camera storage subunit issues INTERIM response for the command.

typedef struct {
      Byte ctrlID;
      Byte ctype;
      Byte subunit_type_and_id;
      Byte opcode;
      Byte operand[509];
} cmdFrame;

typedef struct {
      Byte ctrlID;
      Byte response;
      Byte subunit_type_and_id;
      Byte opcode;
      Byte operand[509];
} respFrame;

typedef struct {
      Byte subfunc;                 // subfunction filed
      Byte result;                  // result field
      Byte phy_vol_num;             // physical_volume_number field
      Byte log_vol_num;             // logical_volume_number field
      Byte media_gen_count;         // media_generation_count field
} cmnHeader;

cmdFrame csCmd;
cmdFrame csCurCmd;
respFrame csResp;
cmnHeader *pcsCmdHdr = (cmnHeader*) csCmd.operand;
cmnHeader *pcsCurCmdHdr = (cmnHeader*) csCurCmd.operand;
cmnHeader *pcsRespHdr = (cmnHeader*) csResp.operand;
bool cmdFlag;       // to indicate if cs command has arrived
bool interimFlag;   // to indicate if INTERIM response has already sent
bool brFlag;        // to indicate if bus reset has occurred
bool endFlag;       // to indicate if execution of command has already finished

Byte media_gen_count;  // media generation count of the camera storage subunit.
```

Table C-2 – State machine code definitions (Contd.)

```
void initialize(void)  // do the power-reset initialization.
{
      media_gen_count = 0;
}

bool avc_cmd(cmdFrame *cmd);
// If the command for the camera storage subunit is arrives, this function sets
// the command frame data to the buffer pointed by cmd and returns TRUE.

void avc_resp(respFrame resp);
// Generate AV/C response frame to the controller.

bool process_type(Byte opcode);
// Judge if the process of the control command of specified opcode takes more than 100 msec.
// If the process of the command takes more than 100msec,  this function returns TRUE.

void copy_cmd_frame(cmdFrame *dest, cmdFrame *src);
// Copy the src structure to the dest.

bool cmd_cmp(cmdFrame cmd1, cmdFrame cmd2, bool subfunc);
// Compare command frame data. If frame data of cmd1 and cmd2 are same, TRUE is returned.
// When subfunc is FALSE, this function compares frame data except subfunction field.

void execute_command(cmdFrame cmd, respFrame *resp, bool block);
// Execute the command frame specified by cmd structure data and set the response frame data to the buffer
// pointed by resp.
// procType specifies the mode of execution. When FALSE is specified, this function creates
// new thread for the command and returns immediately. When TRUE is specified,
// this function starts execution and returns when the process is finished.

void abort_command(cmdFrame cmd);
// Abort the execution of command correspond to cmd structure data.

void suspend_command(cmdFrame cmd);
// Suspend the execution of command correspond to cmd structure data.

bool end_of_execution(cmdFrame cmd);
// Return TRUE when following conditions match :
// (1) execute_command() had been called with same cmd value and block value of FALSE.
// (2) execution of the command invoked by (1) has finished
// (3) this function is not called yet since the execution has finished.

bool bus_reset_event();
// Return TRUE when this function is called first after bus reset has occurred.

bool transfer_frame(Byte opcode) // return TRUE if the command is frame transfer command.
{
      return((opcode == SEND FILE) || (opcode == SEND FILE PARTIAL) ||
            (opcode == SEND THUMBNAIL) ||
            (opcode == RECEIVE FILE) || (opcode == RECEIVE FILE PARTIAL));
}
```

Table C-3 – State machine code definitions (Contd.)

```
void reset_timer();
// Reset and start timer.

bool timeout(int time);
// Return TRUE if timer value exceed the value specified by time. The unit of time is millisecond.

Byte play_mode();
// Return current playing mode.

Byte record_mode();
// Return current recording mode.
```

C.2 Camera storage state machine for single task model

Figure C-1 through Figure C-7 shown below describe the camera storage subunit state transitions of command execution for single task model. In this figure, the camera storage subunit is assumed to accept only one control command at a time and there are more assumption as follows :

1) While camera storage subunit is sending or receiving asynchronous frame data, the camera storage subunit can't play or record file data.
2) While camera storage subunit is playing or recording a file, the camera storage subunit can't send or receive asynchronous frame data.
3) Camera storage subunit can't play and record at same time and can play or record only one file at a time.

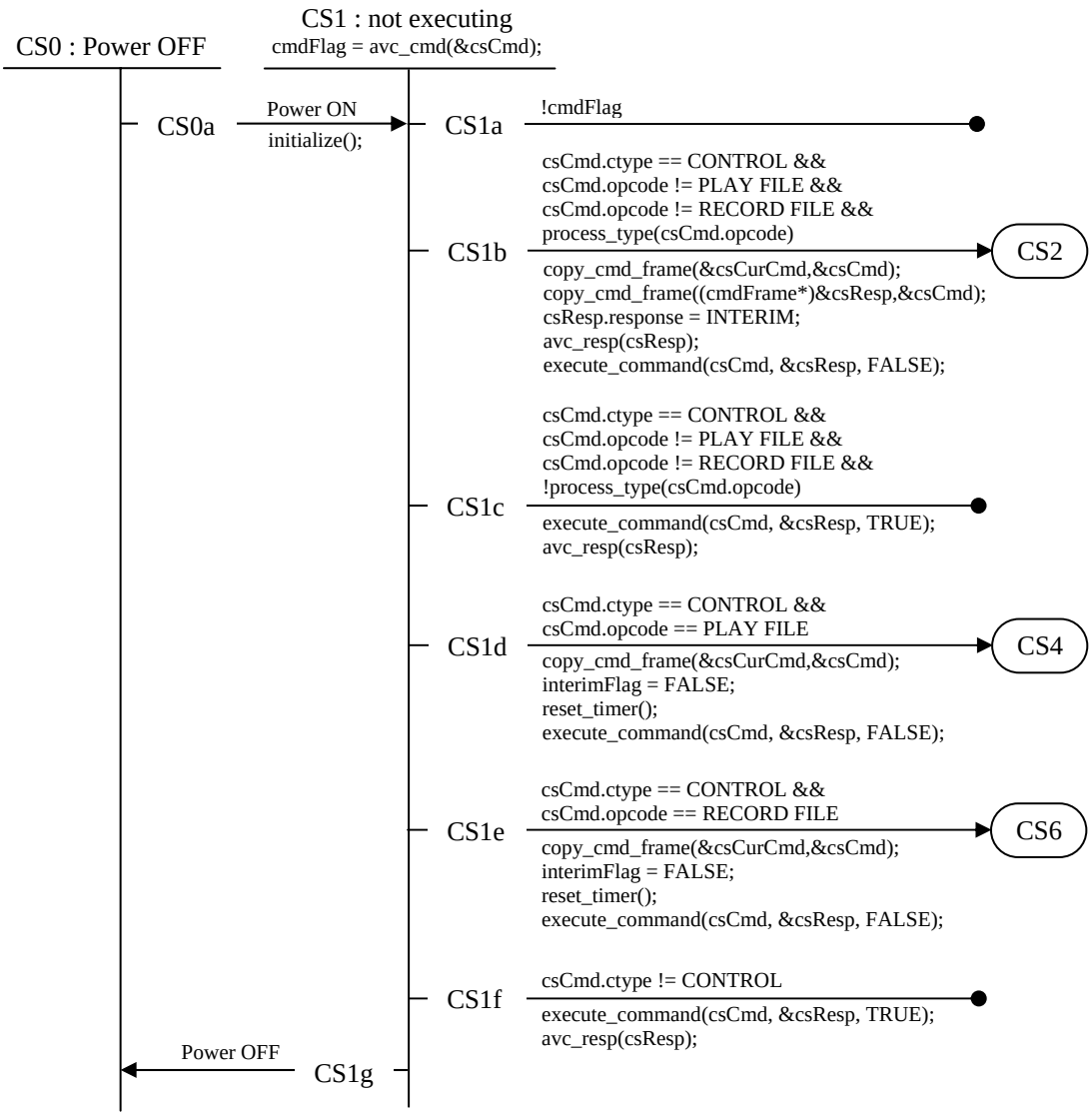


Figure C-1 – State machine for single task model

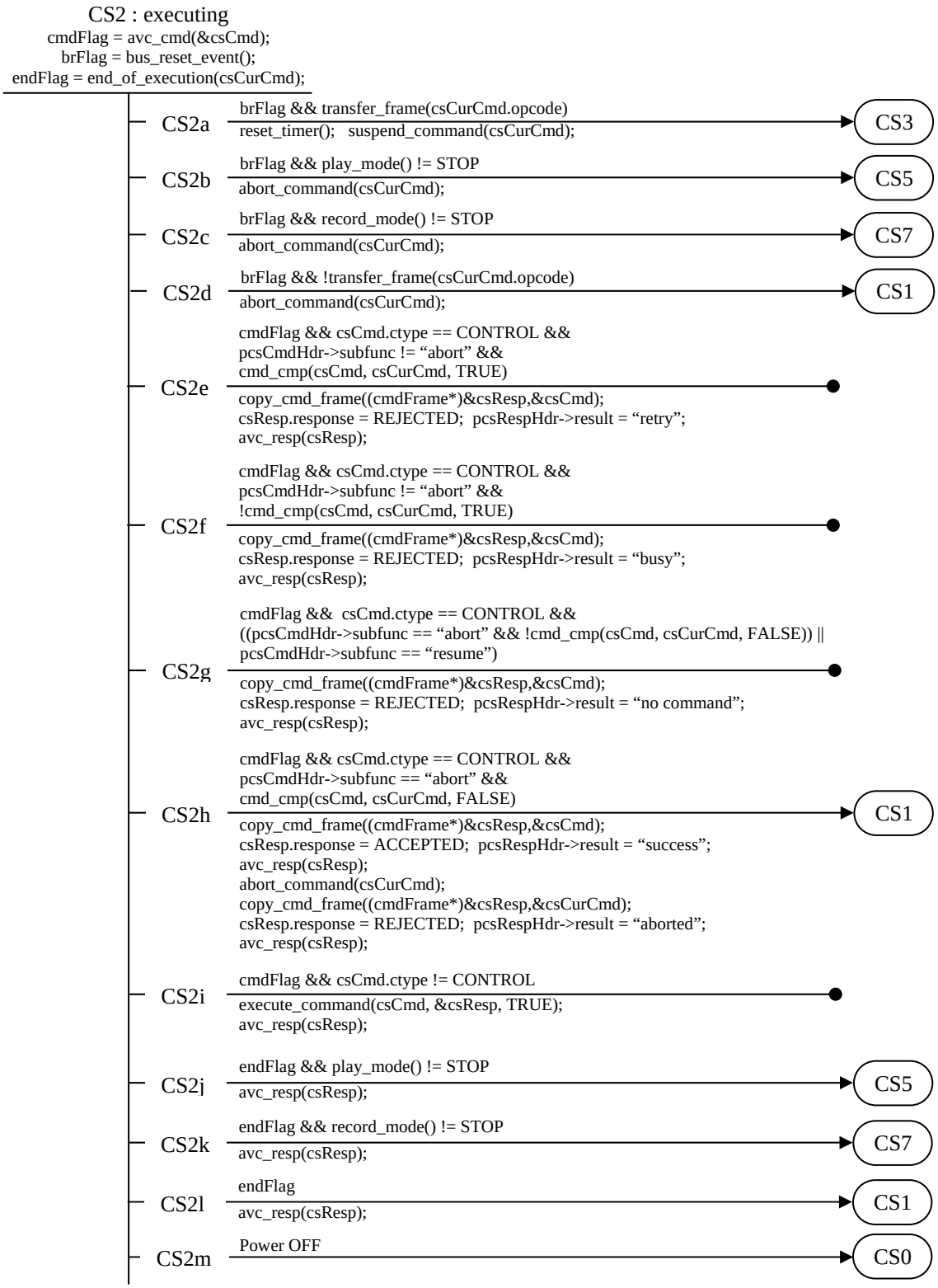


Figure C-2 – State machine for single task model (Contd.)

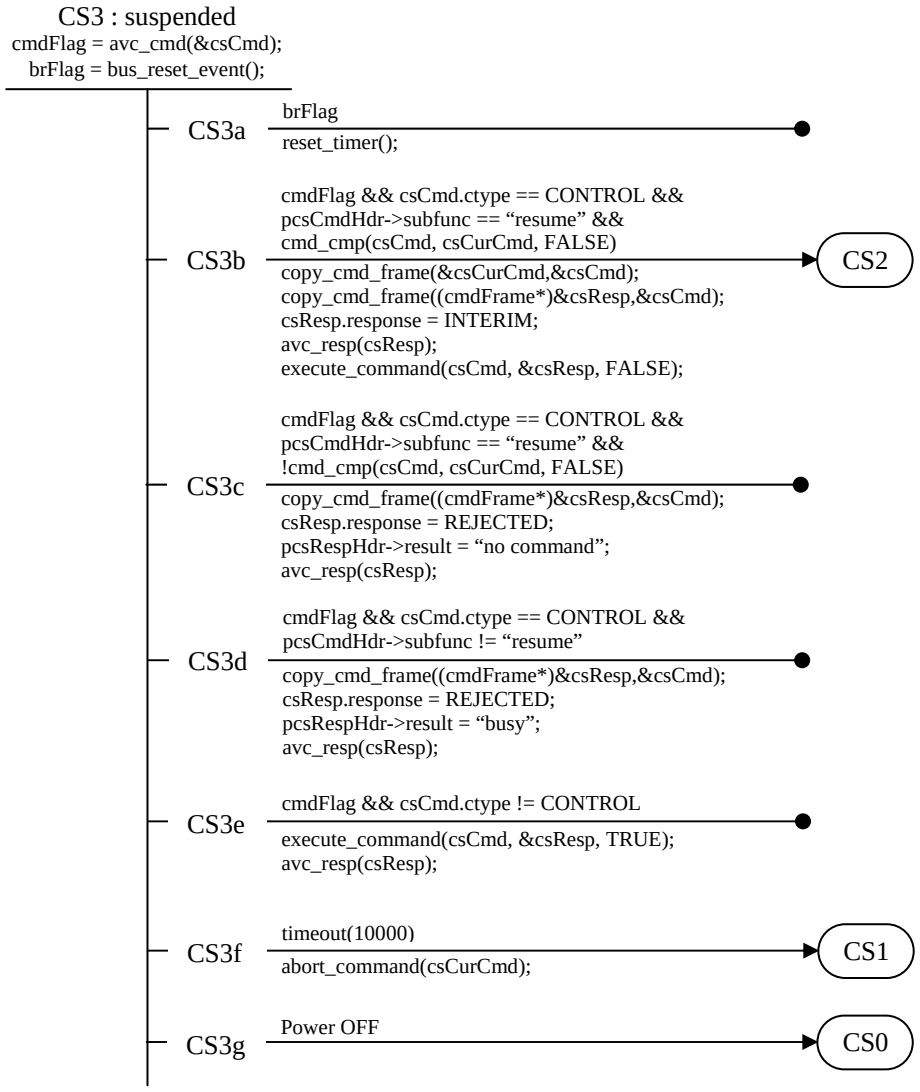


Figure C-3 – State machine for single task model (Contd.)

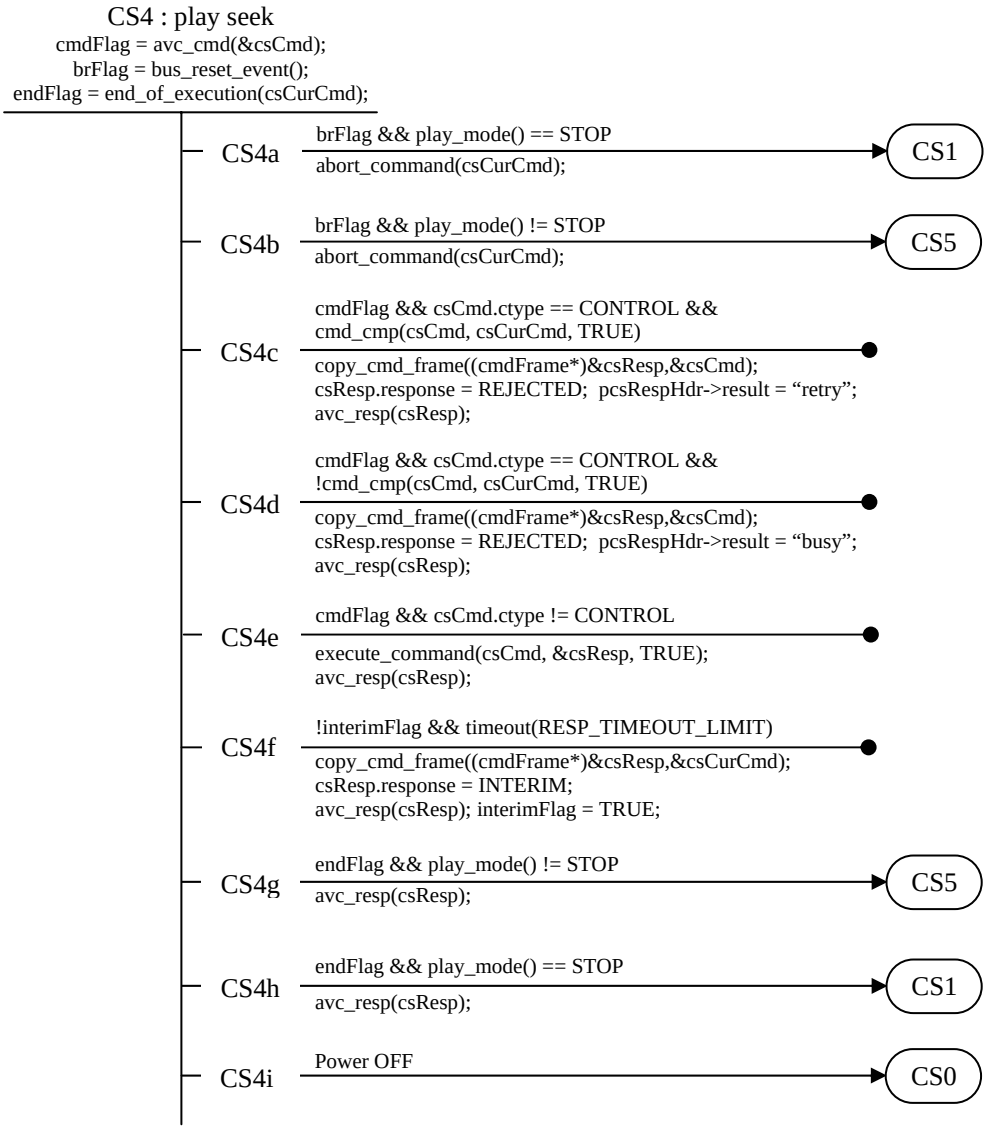


Figure C-4 – State machine for single task model (Contd.)

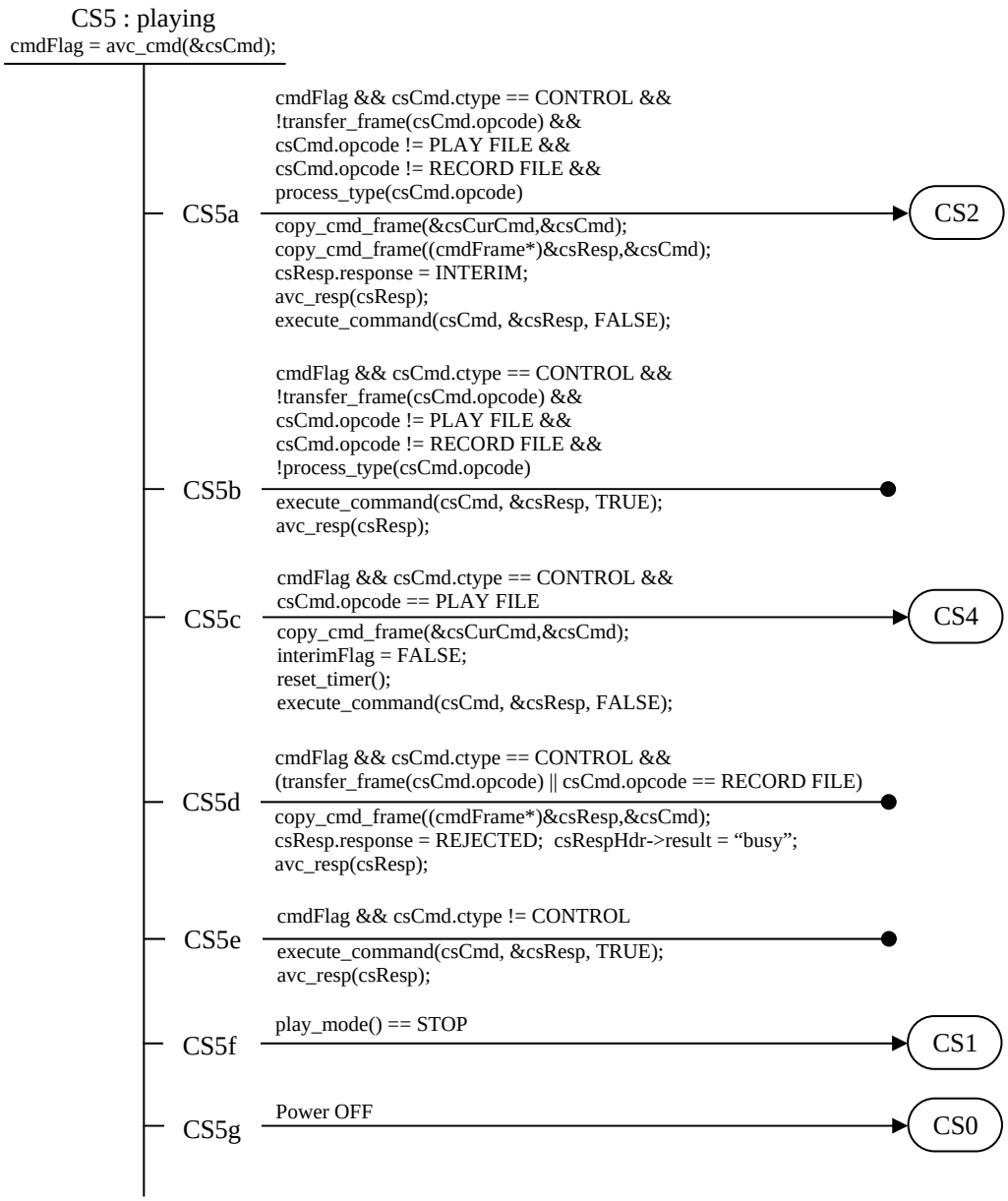


Figure C-5 – State machine for single task model (Contd.)

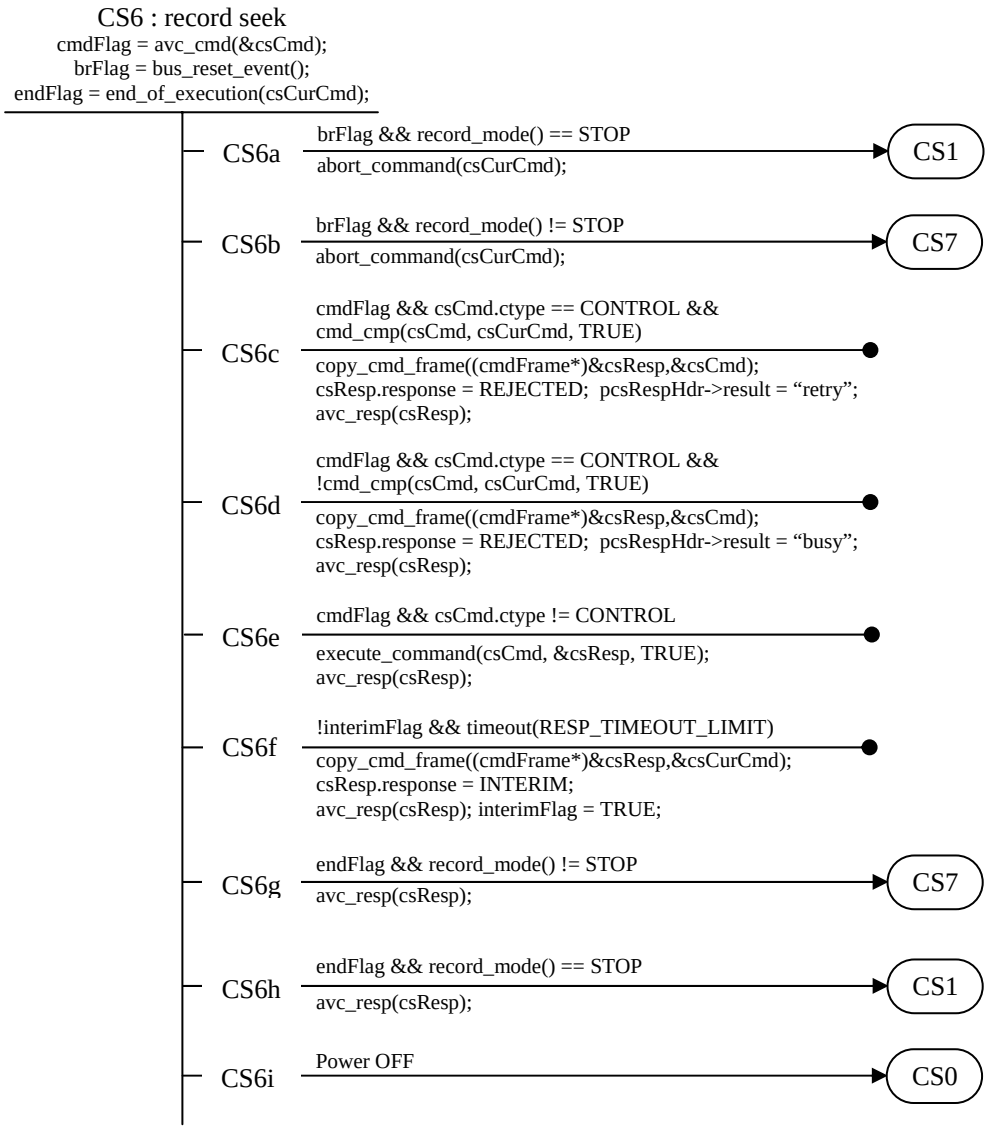


Figure C-6 – State machine for single task model (Contd.)

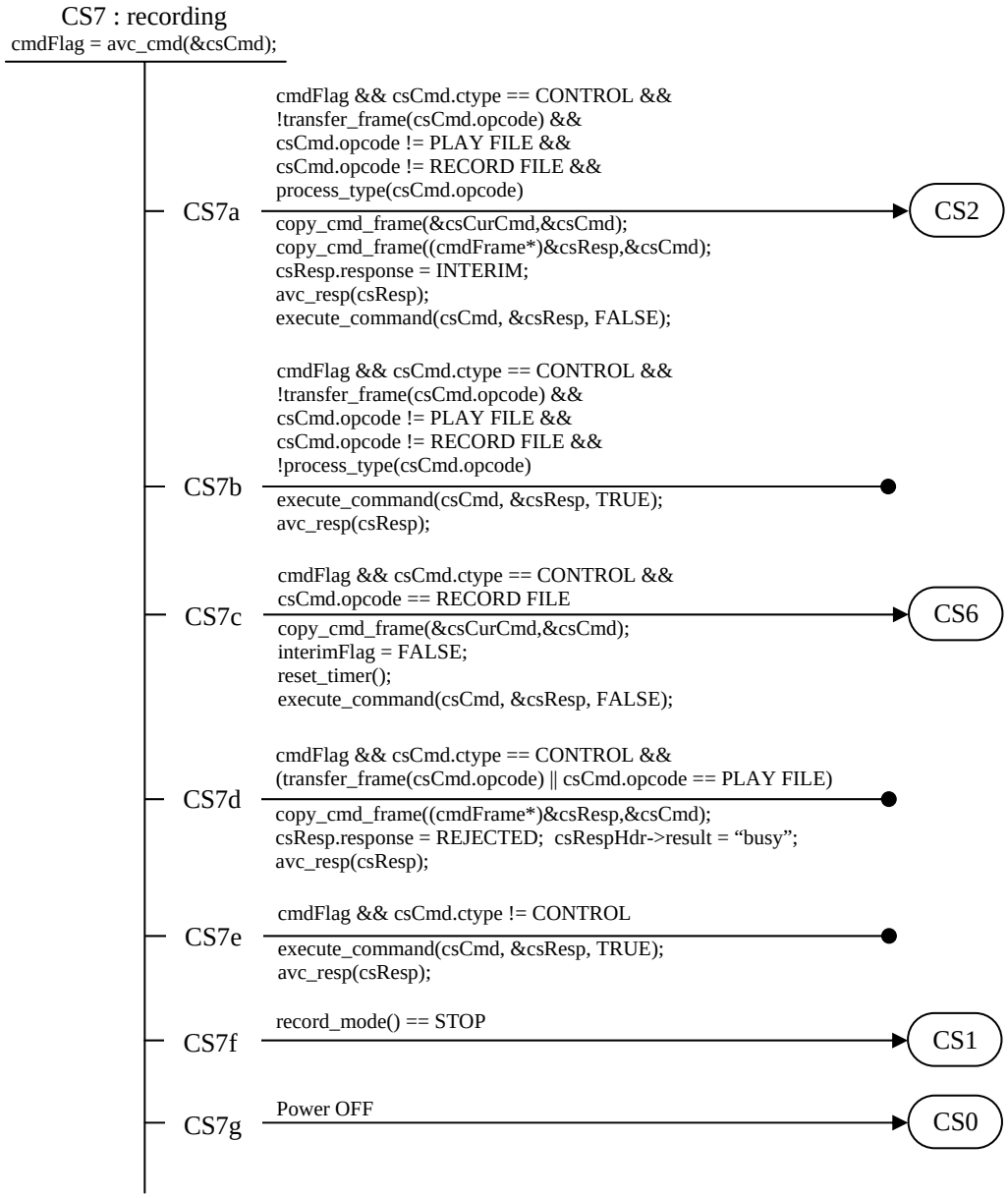


Figure C-7 – State machine for single task model (Contd.)