**Avid Media Stream Server**



**MEDIA STREAM SERVER**

**PLATFORM**

**PROTOCOL**

**External Reference Specification**

**Version 4.0**

**R& D Lab**

**280 N. Bernardo Ave.**

**Mountain View, CA 94043**

**COMPANY CONFIDENTIAL**

Last Modification: 16 August 2005

Command States

page 1

**1**

**2**

**3**

**TABLE OF CONTENTS**

**LIST OF TABLES................................................................................................................................ 6**

**PREFACE .............................................................................................................................................** **8**

**INTRODUCTION ................................................................................................................................** **9**

3.1

3.2

3.3

3.4

3.5

3.6

3.7

ABSTRACT.......................................................................................................................................  9

PRODUCT IDENTIFICATION ..............................................................................................................  9

AUDIENCE .......................................................................................................................................  9

RELATED DOCUMENTATION..........................................................................................................  10

PURPOSE OF DOCUMENT ...............................................................................................................  10

ACKNOWLEDGMENTS....................................................................................................................  10

TERMS AND DEFINITIONS ..............................................................................................................  10

**4**

**5**

**6**

**7**

**8**

**PRODUCT OVERVIEW................................................................................................................... 13**

4.1  PRODUCT ARCHITECTURE ............................................................................................................. 13

*4.1.1* *Control Ports ........................................................................................................................ 13*

*4.1.2* *Video File System ................................................................................................................. 13*

4.2  SYSTEM OPERATION...................................................................................................................... 13

*4.2.1* *Video Port Selection ............................................................................................................. 13*

*4.2.2* *Video Input Ports.................................................................................................................. 14*

*4.2.3* *Video Output Ports ............................................................................................................... 14*

4.3  TRIMMING FEATURES.................................................................................................................... 14

*4.3.1* *Video Tools........................................................................................................................... 15*

*4.3.2* *Arbitrary Play....................................................................................................................... 15*

**CONTROL PROTOCOL ..................................................................................................................** **17**

5.1  BINARY PROTOCOL ....................................................................................................................... 17

*5.1.1* *RS-422 Physical Protocol..................................................................................................... 17*

*5.1.2* *Binary Protocol Format ....................................................................................................... 17*

5.2  ASCII PROTOCOL.......................................................................................................................... 19

*5.2.1* *ASCII Protocol Responses.................................................................................................... 19*

*5.2.2* *ASCII Command Format ...................................................................................................... 20*

5.3  COMPARISON OF BINARY AND ASCII PROTOCOLS ........................................................................ 20

5.4  COMMAND RESPONSE TIMES ........................................................................................................ 20

5.5  COMMAND DESCRIPTION FORMAT ................................................................................................ 21

**DECODER COMMAND STATES................................................................................................... 23**

**VDCP IMPLEMENTATION ON MSS............................................................................................ 24**

**COMMAND DESCRIPTIONS .........................................................................................................** **28**

8.1  IMMEDIATE COMMANDS ............................................................................................................... 28

*8.1.1* *STOP..................................................................................................................................... 28*

*8.1.2* *RECORD .............................................................................................................................. 28*

*8.1.3* *PLAY..................................................................................................................................... 29*

*8.1.4* *PAUSE.................................................................................................................................. 30*

*8.1.5* *STEP ..................................................................................................................................... 31*

*8.1.6* *CONTINUE .......................................................................................................................... 31*

*8.1.7* *JOG ...................................................................................................................................... 32*

*8.1.8* *SHUTTLE ............................................................................................................................. 32*

page 2

Command States

8.2 PORT CONTROL COMMANDS ................................................................................................ 35

*8.2.1* *OPEN QUERY...................................................................................................................... 35*

*8.2.2* *CLOSE.................................................................................................................................. 35*

*8.2.3* *PORT SELECT ..................................................................................................................... 35*

*8.2.4* *PORT STATUS QUERY........................................................................................................ 36*

*8.2.5* *LAST STATUS QUERY......................................................................................................... 37*

*8.2.6* *ACTIVE ID QUERY.............................................................................................................. 38*

*8.2.7* *POSTION REQUEST QUERY.............................................................................................. 38*

*8.2.8* *POSITION REQUEST BY FRAMES QUERY....................................................................... 39*

8.3 CUEING COMMANDS............................................................................................................... 40

*8.3.1* *RECORD CUE ..................................................................................................................... 40*

*8.3.2* *RECORD CUE WITH DATA................................................................................................ 40*

*8.3.3* *TIME DELAY RECORD CUE .............................................................................................. 40*

*8.3.4* *TIME DELAY RECORD CUE WITH DATA ........................................................................ 41*

*8.3.5* *PLAY CUE............................................................................................................................ 42*

*8.3.6* *PLAY CUE WITH DATA ...................................................................................................... 43*

*8.3.7* *PLAY CUE WITH FRAMES ................................................................................................. 44*

8.4 MATERIAL MANAGEMENT COMMANDS ........................................................................................ 46

*8.4.1* *MARK ................................................................................................................................... 46*

*8.4.2* *COPY.................................................................................................................................... 47*

*8.4.3* *LOUTH COPY...................................................................................................................... 47*

*8.4.4* *COPY TERMINATE ............................................................................................................. 48*

*8.4.5* *NEW COPY .......................................................................................................................... 48*

*8.4.6* *ID RENAME ......................................................................................................................... 48*

*8.4.7* *SET START OF CLIP ........................................................................................................... 49*

*8.4.8* *ID DELETE .......................................................................................................................... 49*

*8.4.9* *DELETE PROTECT ............................................................................................................. 50*

*8.4.10* *DELETE UNPROTECT........................................................................................................ 50*

*8.4.11* *DISK CLEAR ........................................................................................................................ 50*

8.5 LIST QUERIES ............................................................................................................................ 51

*3.6.1* *LIST QUERY ........................................................................................................................ 51*

*3.3.2* *NEXT QUERY....................................................................................................................... 52*

*8.5.1* *IDS ADDED TO MAIN QUERY........................................................................................... 53*

*8.5.2* *IDS ADDED QUERY............................................................................................................ 54*

*8.5.3* *IDS DELETED QUERY........................................................................................................ 55*

8.6 ID CHARACTERISTIC QUERIES ............................................................................................. 57

*8.6.1* *SIZE QUERY ........................................................................................................................ 57*

*3.3.13* *ID REQUEST QUERY.......................................................................................................... 57*

*8.6.2* *ID REQUEST DISK QUERY ................................................................................................ 58*

*8.6.3* *ID INFO RETRIEVAL QUERY............................................................................................. 59*

*8.6.4* *ID BYTE COUNT QUERY.................................................................................................... 61*

8.7 USER INFO COMMANDS.......................................................................................................... 62

*8.7.1* *USER INFO OPEN............................................................................................................... 62*

*8.7.2* *USER INFO CLOSE............................................................................................................. 62*

*8.7.3* *USER INFO ADD................................................................................................................. 63*

*8.7.4* *USER INFO MODIFY .......................................................................................................... 63*

*8.7.5* *USER INFO DELETE .......................................................................................................... 63*

*8.7.6* *USER INFO QUERY ............................................................................................................ 64*

*8.7.7* *USER INFO LIST ................................................................................................................. 64*

8.8 ENCODER CONFIGURATION COMMANDS.......................................................................... 65

*8.8.1* *PRESET ................................................................................................................................ 65*

*8.8.2* *SYSTEM BIT RATE .............................................................................................................. 65*

*8.8.3* *SYSTEM BIT RATE QUERY................................................................................................. 65*

*8.8.4* *VIDEO STANDARD ............................................................................................................. 65*

*8.8.5* *VIDEO STANDARD QUERY ............................................................................................... 66*

*8.8.6* *VERTICAL LINE INFO ........................................................................................................ 66*

Command States

page 3

*8.8.7* *VERTICAL LINE INFO QUERY .......................................................................................... 67*

*8.8.8* *OUTPUT PROGRAM ID...................................................................................................... 67*

*8.8.9* *OUTPUT PROGRAM ID QUERY........................................................................................ 67*

*8.8.10* *DROP FRAME ..................................................................................................................... 67*

*8.8.11* *DROP FRAME QUERY........................................................................................................ 68*

*8.8.12* *ENCODER AUTO STOP...................................................................................................... 68*

*8.8.13* *ENCODER AUTO STOP QUERY ........................................................................................ 68*

*8.8.14* *MEDIA PARAMETERS ........................................................................................................ 69*

*8.8.15* *MEDIA PARAMETERS QUERY .......................................................................................... 69*

*8.8.16* *AUDIO FORMAT ................................................................................................................. 69*

*8.8.17* *AUDIO FORMAT QUERY ................................................................................................... 70*

*8.8.18* *AUDIO LEVEL QUERY ....................................................................................................... 70*

*8.8.19* *COMPRESSION SETTINGS QUERY................................................................................... 71*

*8.8.20* *ENABLE VITC CAPTURE ................................................................................................... 72*

*8.8.21* *PREROLL ............................................................................................................................. 72*

*8.8.22* *ENCODER CONFIGURATION ........................................................................................... 73*

8.9 DECODER CONFIGURATION COMMANDS.......................................................................... 74

*8.9.1* *SET GPI................................................................................................................................ 74*

*8.9.2* *SUBCARRIER ADJUST........................................................................................................ 74*

*8.9.3* *SUBCARRIER ADJUST QUERY.......................................................................................... 74*

*8.9.4* *HORIZONTAL SYNC TIMING............................................................................................. 74*

*8.9.5* *HORIZONTAL SYNC TIMING QUERY ............................................................................... 75*

*8.9.6* *EMBEDDED AUDIO ........................................................................................................... 75*

*8.9.7* *EMBEDDED AUDIO QUERY ............................................................................................. 75*

*8.9.8* *GENLOCK MODE ............................................................................................................... 76*

*8.9.9* *GENLOCK MODE QUERY ................................................................................................. 76*

*8.9.10* *DECORDER CONFIGURATION......................................................................................... 76*

*8.9.11* *DECORDER CONFIGURATION QUERY........................................................................... 77*

*8.9.12* *AUDIO CUE......................................................................................................................... 77*

*8.9.13* *AUDIO CUE QUERY ........................................................................................................... 77*

*8.9.14* *DECODER SETTINGS QUERY ........................................................................................... 78*

*8.9.15* *CHROMA GAIN REFERENCE ............................................................................................ 78*

*8.9.16* *DECODER AUDIO GAIN.................................................................................................... 79*

*8.9.17* *UPCONVERSION MODE .................................................................................................... 79*

*8.9.18* *DOWNCONVERSION MODE.............................................................................................. 80*

*8.9.19* *SET BACKGROUND COLOR.............................................................................................. 80*

*8.9.20* *SET CONVERSION COLOR ................................................................................................ 80*

8.10 SYSTEM CONTROL................................................................................................................... 82

*8.10.1* *DEVICE TYPE QUERY........................................................................................................ 82*

*8.10.2* *PROTOCOL ......................................................................................................................... 82*

*8.10.3* *PROTOCOL QUERY............................................................................................................ 82*

*8.10.4* *SOFTWARE REVISION QUERY.......................................................................................... 82*

*8.10.5* *DISK STORAGE SIZE QUERY ............................................................................................ 83*

*8.10.6* *MINIMUM FREE SPACE .................................................................................................... 83*

*8.10.7* *MINIMUM FREE SPACE QUERY....................................................................................... 83*

*8.10.8* *PARAMETER LIMITS QUERY ............................................................................................ 83*

*8.10.9* *PARAMETERS SUPPORTED QUERY ................................................................................ 84*

*8.10.10* *SYSTEM STATUS QUERY ............................................................................................... 84*

8.11 COMPOSITE-TO-D1 UTILITIES ............................................................................................... 88

*8.11.1* *COMPOSITE-TO-D1 BOARD CONFIGURATION............................................................. 88*

*8.11.2* *AUDIO GAIN ....................................................................................................................... 88*

8.12 VIDEO FILE SYSTEM CHECK.................................................................................................. 89

*8.12.1* *CHECK VIDEO FILE SYSTEM ........................................................................................... 89*

*8.12.2* *VIDEO FILE SYSTEM CHECK IN PROGRESS .................................................................. 89*

*8.12.3* *NUMBER OF DAMAGED VIDEO FILES ........................................................................... 89*

*8.12.4* *VIDEO FILE FIX.................................................................................................................. 90*

page 4

Command States

**9**

*8.12.5* *FINISH CHECK VIDEO FILES ........................................................................................... 90*

*8.12.6* *VIDEO FILESYSTEM PLAY ONLY ..................................................................................... 90*

*8.12.7* *ABORT CHECK VIDEO FILES ........................................................................................... 90*

8.13  MISCELLANEOUS UTILITIES ................................................................................................. 91

*8.13.1* *OPERATION MODE ............................................................................................................ 91*

*8.13.2* *PLAY WHILE TRANSFER MODE ....................................................................................... 91*

*8.13.3* *RIPPLE THROUGH DELETE MODE................................................................................. 91*

*8.13.4* *NODE LIST .......................................................................................................................... 92*

*8.13.5* *LONG DISK SIZE QUERY................................................................................................... 92*

*8.13.6* *DECODER RESET ............................................................................................................... 92*

*8.13.7* *ADD TO DB ......................................................................................................................... 93*

*8.13.8* *START IMPORT ................................................................................................................... 93*

*8.13.9* *MODIFY IMPORT ............................................................................................................... 93*

*8.13.10* *FINISH IMPORT.............................................................................................................. 94*

8.14  EXPIRY DATABASE........................................................................................................................ 94

*8.14.1* *EXPIRY DB RESTORE......................................................................................................... 94*

*8.14.2* *AUTO DELETE MODE........................................................................................................ 94*

*8.14.3* *AUTO DELETE TIME.......................................................................................................... 95*

*8.14.4* *EXPIRATION DATE............................................................................................................. 95*

*8.14.5* *DELETE EXPIRED FILES................................................................................................... 96*

8.15  MPEG I/O BOARDS....................................................................................................................... 97

*8.15.1* *ENCODER NAME................................................................................................................ 97*

*8.15.2* *TRANSPORT STREAM NUMBER OF FILES..................................................................... 97*

*8.15.3* *PROGRAM ASSOCIATION TABLE..................................................................................... 97*

*8.15.4* *PROGRAM INFORMATION QUERY .................................................................................. 97*

*8.15.5* *PROGRAM STREAM INFORMATION QUERY .................................................................. 98*

*8.15.6* *MPEG OUT RATE................................................................................................................ 98*

*8.15.7* *MPEG OUT PACKET IDENTIFIER.................................................................................... 98*

*8.15.8* *MPEG OUT LANGUAGE CODE......................................................................................... 99*

**CUEING WHILE TRANSFERRING OR RECORDING............................................................ 100**

9.1  REQUIREMENTS AND CONSTRAINTS ............................................................................................ 100

*9.1.1* *Cueing to SOC while Transferring ..................................................................................... 101*

*9.1.2* *Cueing to Arbitrary Mark-In while Transferring ............................................................... 102*

*9.1.3* *Cueing to SOC while Recording......................................................................................... 103*

*9.1.4* *Cueing to an Arbitrary Mark-In while Recording .............................................................. 104*

9.2  PSEUDO CODE FOR CUEING WHILE TRANSFERRING OR RECORDING............................................ 105

**10**

**TRIMMING WHILE RECORDING .........................................................................................** **108**

10.1 AMOUNT OF VIDEO AVAILABLE FOR TRIMMING ......................................................................... 108

10.2 BASIC TRIMMING COMMANDS .................................................................................................... 108

*10.2.1* *Pause .................................................................................................................................. 109*

*10.2.2* *Jog ...................................................................................................................................... 109*

*10.2.3* *Shuttle ................................................................................................................................. 109*

10.3 EXPANDING THE TRIMMABLE AREA............................................................................................. 110

10.4 DETERMINING THE END OF THE FRAME TABLE ............................................................................ 110

10.5 DECODER CONFIGURATION ......................................................................................................... 113

**11**

11.1

11.2

11.3

**12**

**PLAYING A SHORT SPOT AT THE BEGINNING OF A POD............................................ 114**

CASE WITH NO ERRORS................................................................................................................  114

CASE WITH ERRORS ON THE FIRST CUE. .......................................................................................  117

CASE WITH ERRORS ON THE SECOND CUE. ...................................................................................  117

**INDEX OF COMMANDS ...........................................................................................................** **120**

Command States

page 5

**1** **LIST OF TABLES**

TABLE 1.

TABLE 2.

TABLE 3.

TABLE 4.

TABLE 5.

TABLE 6.

TABLE 7.

TABLE 8.

TABLE 9.

TABLE 10.

TABLE 11.

TABLE 12.

TABLE 13.

TABLE 14.

TABLE 15.

TABLE 16.

TABLE 17.

TABLE 18.

TABLE 19.

TABLE 20.

TABLE 21.

TABLE 22.

TABLE 23.

TABLE 24.

TABLE 25.

TABLE 26.

TABLE 27.

TABLE 28.

TABLE 29.

TABLE 30.

TABLE 31.

TABLE 32.

TABLE 33.

TABLE 34.

TABLE 35.

TABLE 36.

TABLE 37.

TABLE 38.

TABLE 39.

TABLE 40.

TABLE 41.

TABLE 42.

TABLE 43.

TABLE 44.

TABLE 45.

TABLE 46.

TABLE 47.

TABLE 48.

TABLE 49.

TABLE 50.

TABLE 51.

TABLE 52.

page 6

ERS DOCUMENT REVISION HISTORY ..........................................................................................  9

SINGLE SPOT TRIMMING............................................................................................................  14

EXTRACTING MULTIPLE CONTENTS ..........................................................................................  14

CONTENT SEGMENTATION.........................................................................................................  15

RS422 PIN ASSIGNMENTS ........................................................................................................  17

BINARY PROTOCOL FORMAT.....................................................................................................  17

COMMAND TYPES......................................................................................................................  18

NAK ERROR BYTE ....................................................................................................................  18

ASCII PROTOCOL RESPONSE EXAMPLES ..................................................................................  19

DECODER STATE DIAGRAM TABLE .......................................................................................  23

MSS PROTOCOL SUMMARY ..................................................................................................  24

SYSTEM OUTPUT DURING BACK-TO-BACK PLAY ....................................................................  29

RELATIONSHIP BETWEEN SHUTTLE SPEEDS AND I-FRAMES ..................................................  33

SHUTTLE VELOCITY ..............................................................................................................  33

SHUTTLE VELOCITY ROUNDING EXAMPLES ..........................................................................  34

PORT STATUS QUERY (STAT?) BITMAP ...............................................................................  36

PORT STATUS ITEM 1: STATE STATUS ..................................................................................  36

PORT STATUS ITEM 2: LIST STATUS......................................................................................  36

EXTENDED PORT STATUS ITEM 2: LIST STATUS ...................................................................  36

PORT STATUS ITEM 3: ERROR STATUS..................................................................................  37

PORT STATUS ITEM 4: SETTINGS ..........................................................................................  37

PORT STATUS ITEM 5: SUPPORTED MEDIA FORMATS ...........................................................  37

IDRQ? QUERY RETURN BYTE...............................................................................................  57

INFO? QUERY BITMAP .........................................................................................................  59

ID INFO 1: COMMAND VIDEO ATTRIBUTES ..........................................................................  59

ID INFO 2: ENCODE RATES ...................................................................................................  59

ID INFO 3: TIMECODE / DURATION .......................................................................................  60

ID INFO 4: AUDIO INFORMATION..........................................................................................  60

INPUT PORT PRESET VALUES ................................................................................................  65

VSTD PARAMETERS .............................................................................................................  66

MPRM PARAMETERS............................................................................................................  69

AUDF PARAMETERS .............................................................................................................  70

CSET? QUERY BITMAP .........................................................................................................  71

COMPRESSION SETTING 1: CURRENTLY UNUSED .................................................................  71

COMPRESSION SETTING 2: AUDIO SAMPLE RATE AND MODE...............................................  71

COMPRESSION SETTING 3: SYSTEM BIT RATE .......................................................................  71

COMPRESSION SETTING 6: VIDEO STANDARD AND RESOLUTION ..........................................  71

COMPRESSION SETTING 7: CLOSED CAPTION ........................................................................  72

ENCODER CONFIGURATION (ECFG) BITMAP ........................................................................  73

ENCODER TYPES SUPPORTING CONFIGURATION (ECFG) FUNCTIONS...................................  73

HSYN PARAMETERS .............................................................................................................  75

DCFG COMMAND BITMAP....................................................................................................  76

DECODER EOF / LAST FRAME ACTIONS................................................................................  77

AUDIO CUE MODES ...............................................................................................................  77

DECODER SETTINGS BITMAP ..........................................................................................  78

DECODER SETTINGS 1 – UNUSED .................................................................................  78

DECODER SETTINGS 6- VIDEO STANDARD ................................................................  78

UP-CONVERSION MODES ......................................................................................................  79

BACKGROUND COLORS .........................................................................................................  80

PLMT PARAMETERS .............................................................................................................  84

PSUP PARAMETERS ..............................................................................................................  84

SYSTEM STATUS QUERY (SYSS?) BITMAP ...........................................................................  85

Command States

TABLE 53.

TABLE 54.

TABLE 55.

TABLE 56.

TABLE 57.

TABLE 58.

TABLE 59.

TABLE 60.

TABLE 61.

TABLE 62.

TABLE 63.

TABLE 64.

SYSTEM STATUS 1 - STORAGE TIME REMAINING – SHORT OPTION .......................................  85

SYSTEM STATUS 1 - STORAGE TIME REMAINING – EXTENDED OPTION.................................  85

SYSTEM STATUS 2 - NUMBER OF IDS STORED – SHORT OPTION ............................................  86

SYSTEM STATUS 2 - NUMBER OF IDS STORED – EXTENDED OPTION......................................  86

SYSTEM STATUS 3 - DISK STATUS .........................................................................................  86

SYSTEM STATUS 4 - SUBSYSTEM STATUS..............................................................................  86

SYSTEM STATUS 5- STANDARD TIME ....................................................................................  86

SYSTEM STATUS 6- MINIMUM FREE SPACE PERCENTAGE .....................................................  87

VIDEO FILE SYSTEM PLAY ONLY STATES .............................................................................  90

MIMP PARAMETERS .............................................................................................................  93

STREAM TYPES......................................................................................................................  98

STREAM INDECES FOR MPOP COMMAND .............................................................................  99

Command States

page 7

**2** **Preface**

Version 4.0 of the Avid Media Stream Server Platform Protocol ERS is the only document which

describes the binary and ASCII protocols implemented on the MSS. It is being made available to

all partners in order to allow them to design and implement code meeting the specifications

contained herein.

If you have any questions or comments regarding this document, please contact:

Keith Keilman

keith\_keilman@avid.com

+44 (0) 7810 837 284

Authors:

Editors:

Kyle Karnosh

Keith Keilman

Mike Manlove

Bill Reckwerdt

Sandeep Nirmale

page 8

Command States

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 1.0 | 03/27/96 | First Draft for Internal review |
| 1.1 | 05/27/96 | First document for general distribution |
| 1.5 | 09/03/97 | Trimming features added |
| 1.6 | 01/12/98 | Time Delay and C2D1 added |
| 1.7 | 06/12/98 | DCFG changes, remove remote archive, bits & pieces |
| 1.8 | 07/31/98 | Updates, 4:2:2 commands |
| 1.9 | 08/10/98 | Updates |
| 2.0 | 06/16/99 | WARP Release, DTV commands |
| 3.0 | 06/10/2000 | VARIABLE NAMES Added |
| 3.1 | 09/13/01 | Minor Corrections |
| 3.2 | 09/24/01 | More Corrections, CUEing info |
| 3.3 | 03/18/02 | Added MPIO, pre-roll. Corrections. Command index. |
| 3.4 | 10/02/02 | Added time delay commands. Improved drop frame  and SOC descriptions. |
| 3.5 | 07/09/04 | Added 10 bit VILI, reorganized sections and made  numerous minor corrections. |
| 3.7 | 01/04/05 | ASCII protocol corrections. ASCII list commands  have been wrong since 3.0. |
| 4.0 | 09/08/05 | PFS compliant. New command meta-language.  Removed unsupported archive commands. |

**3** **Introduction**

**3.1**

**Abstract**

The Avid Media Stream Server provides delivery of compressed video from a magnetic disk

based file system for broadcast applications. This ERS contains an overview of the system and a

detailed description of the interface protocol.

**3.2**

**Product Identification**

Project Name

Project Mnemonic

Target Releases

Product Name

Product Packaging

AVID Media Stream Server

Media Stream Server (MSS)

B.09.45 for Series 300 / 700 / 1600 / 900si

C.02.20 for Series 900/NS / 8000/NS / 8000si

Media Stream Server – Series 300 / 700 / 1600 / 900(NS/si) /

8000(NS/si)

Stand alone system

**3.3**

**Table 1.**

**ERS Document Revision History**

**Audience**

Command States

page 9

This document is intended for the designers and developers of MediaStream servers and the

applications controlling MediaStream Servers.

**3.4**

**Related Documentation**

The following list of references was used in the preparation of this external reference

specification. The reader is urged to consult them for more information.

•

VDCP Revision 4 August 2000

**3.5**

**Purpose of Document**

The External Reference Specification is intended to define the protocol of the Media Stream

Server, how it interfaces to automation control software and other partners who will control the

Media Stream Server. Users include product marketing engineers, product support engineers,

product assurance engineers, learning product engineers and other partners who will interface to

Media Stream Server. Enough of a high level description is presented to provide an

understanding of how each feature operates. After reading this External Reference Specification,

a software engineer should understand how to write code that uses and works with Media Stream

Server; a product marketing engineer should understand how Media Stream Server works and

how it meets marketing requirements; a product support engineer should understand how to use

Media Stream Server; a product assurance engineer should understand Media Stream Server

well enough to test the product and ensure its quality; products engineer should understand

Media Stream Server well enough to write the user documentation.

This is intended to be a ``living'' document; that is, as circumstances change, this document will

be altered to reflect those changes.

**3.6**

**Acknowledgments**

The authors would like to gratefully acknowledge the contributions of all of the people involved in

developing and supporting BVS, including fearless leader, Peter Fisher.

**3.7**

**Terms and Definitions**

**Archive**

**Any off-line storage device capable of holding Media Stream clips.**

**base-clip**

A base-clip is an ID for which at least one subclip is currently defined.

**BVS, MSS or server**

Media Stream Server. BVS 1.5 is the release called Maverick and the first release to use the

product name Avid Media StreamServer. The new platform is called Warp and the first release of

this software is B.08.20.

**cli**

page 10

Command States

Command Line Interface. This executable provides an ASCII command interface to the Media

Stream Server using the commands defined in this ERS.

**clip or ID**

A piece of content.

**content**

Content is a VFS term. It is a clip wrapper for one or more tracks with metadata.

**Duration**

The minimum number of frames required to span an interval of clock time expressed in TC

format.

**EOF**

End of File. The timecode returned by the SIZE? command.

**EOM**

End of Material. EOM is synonymous with mark-out.

**fps**

Frames per second. NTSC video is shown at approximately 30 fps, while PAL is shown at 25 fps.

A frame is composed of a top field and a bottom field.

**Fps**

Fields per second. NTSC video is shown at approximately 60 Fps, and PAL is 50 Fps.

FSC

File System Controller. For systems where the VFS is maintained on a separate component from

the server node, the component controlling the file system is called an FSC. MSS 900 NS and

8000 are often refered to as FSC-based systems.

**JPEG**

Joint Photographic Experts Group. JPEG is a general-purpose compression standard designed to

meet the needs of continuous-tone, still-image applications. JPEG compression ratios can vary

quite a bit (no pun intended). High quality JPEG format may offer a 4:1 compression ratio and

bitrates at about 70 MegaBits/Second. JPEG is a format quite suitable for newsroom editing

markets.

**Mark-In**

Marks the beginning of play of the clip or spot, which may be different from the actual beginning

of the clip (SOC).

**Mark-Out**

Marks the end of play of the clip or spot. The actual end of the clip (EOF) may be different.

Media Vault

A third party SCSI archive available on older releases of MSS 300 / 700 / 1600. Support of the

Media Vault was discontinued starting at release B.09.00. For information on the protocol

interface to a Media Vault, see the Protocol ERS for releases which support the Media Vault.

**MPEG 2**

Moving Picture Experts Group standard for video transfer. MPEG-2 can span the marketplace to

cover video on demand at lower rates, 6MBit/Sec, to broadcast servers at 15MBit/Sec or greater

rates.

Command States

page 11

**MSi**

Media StreamServer Interface. An MS Windows based GUI interface to the Media Stream Server

functionality. Uses LAN access to the protocol layer of the BVS software. Also used as a console

for the CPU running the Media Stream Server software.

**NOP**

A command which does not change the state, status or configuration of the selected port is

considered a NOP command, or No Operation.

PFS

Palladium File System. The VFS on FSC-bases systems is called PFS. MSS 900 NS and 8000

are often referred to as PFS-based systems.

**RAID**

Redundant Array of Inexpensive Disks.

SAN

Storage Array Network. SAN refers to MSS servers which can have more than one node sharing

the same storage. All SAN systems are FSC-based. MSS 900 NS and 8000 NS are SAN

systems.

**SOC**

Start of clip. The timecode associated with the first frame of video in the file.

**SOM**

Start of material. SOM is synonymous with mark-in.

**spot**

Another term for clip.

**subclip**

A subclip is a method of assigning a unique name to a specific MARK-IN and DURATION for a

specific clip. Cue times for subclips follow the rules for the underlying PCWD command. In other

words, if a subclip has a non-zero MARK-IN to the base-clip, the cueing behavior of the subclip

will be that of a PCWD with a non-zero MARK-IN.

**timecode or TC format**

Timecode is used to correlate a video time reference to a specific frame of video. The format

used to express timecode on Media Stream can be found in the section titled “Timecode (TC

format)”.

**track**

A VFS term. Tracks only exist within a content wrapper. Tracks represent a subset of a clip,

whether it is the video portion of the clip or clip related information used for trimming.

**VFS**

The Media Stream Server's file system, the Video File System.

**VME**

Versa Module Eurocard. Introduced in 1981, a microprocessor independent bus that implements

a reliable, mechanical standard and allows independent vendors to build compatible products.

The wide variety of existing hardware and software is an attractive feature of VME.

**WARP**

The MSS 300, 700 and 1600 platforms are collectively known as Warp servers.

page 12

Command States

**4** **Product Overview**

The Media Stream Server provides video content delivery for broadcast applications. The basic

functions of the MSS are the ingesting and playing back of locally stored video content. The MSS

also has a limited set of trimming capabilities targeted at broadcast needs. The MSS can

exchange video content with other MSSs in a local cluster. In addition, the MSS supports the ftp

interface for exchanging video content with non-MSS systems and archives.

**4.1**

**Product Architecture**

**4.1.1 Control Ports**

There are two types of ports on the MSS. From a logical point of view, each port of the Media

Stream Server functions as a separate video tape machine. Input ports function as record only

decks and output ports function as play only decks. Each port is independently controllable

through the binary and ASCII control protocols. All ports may be active and process commands

at the same time.

**4.1.2 Video File System**

The video content is stored locally on the MSS’s Video File System. The VFS is organized into

files residing in a single directory. Each file must have a unique name. File names come in two

varieties.

Fixed length file names are exactly 8-characters long and must be blank-padded to the right.

Variable length file names were introduced in release B.08.60 and consist of a length byte

followed by at least 1 and no more than 32 characters. Legal characters are alpha-numeric

characters, dash and underscore. Blanks are not allowed in variable length names.

In the binary protocol, the upper bit of the first command byte determines the name format used

for the command. Although only the commands having ID names as parameters or responses

are listed in this document, variable length versions exist for all 0X, 2X, 3X and 5X commands.

The ASCII protocol uses variable length file names, unless otherwise indicated. Fixed length file

names may be represented in the ASCII protocol by blank-padding the name to exactly 8

characters.

The number of files supported depends on the storage option chosen, the compression rate, and

the length of the files. The absolute maximum number of files supported is 12,000 on the

standard product or, 25,000 with a high file count upgrade.

The minimum file length supported is 3 seconds, with the maximum being determined by the

system capacity and configuration.

**4.2**

**System Operation**

**4.2.1 Video Port Selection**

The system commands are organized with reference to the Video Port that they effect. The ports

consist of Video Input Ports and Video Output Ports. The Video Port consists of the audio and

Command States

page 13

|  |  |  |
| --- | --- | --- |
|  | **Spot** |  |

video channels configured for that port. Each port is referenced by a signed number which is

specified in the configuration file. Input ports are negative numbers while output ports are positive

numbers. Thus a typical system with one input and two outputs might have ports -1, 1 and 2

available for use. Port 0 is a system port and is not associated with a video port/hardware. It can

be used for system level queries and commands, such as list and delete. Any video port can be

controlled from any RS422 port with the following Port assignment commands and querieC:

OPEN, CLOSE and SELECT. Only one communications port can have a given video port open

at a given time. Only the use of RS422 ports physically located in the same card cage as their

selected Video ports is supported.

**4.2.2 Video Input Ports**

A Video Input Port is the only Port that can record new content into the disk system. All recording

is done through one of these ports. There are three commands that are used to record new

material into the system:

•

•

•

RECORD CUE

RECORD

STOP

Recording a spot into the system requires the sequence RECORD CUE, RECORD, and

optionally STOP.

At any point in time an Input Port can be in one of three stateC: IDLE, CUE or RECORD. The

input mode can be selected, as well as several encoding rates and settings. Settings which affect

the record process cannot be changed while cueing or recording.

**4.2.3 Video Output Ports**

Playback can be done on any of the output ports. To play a file to air the file must be cued with a

PCUE or PCWD command, then played with a PLAY command on an open, selected play port.

**4.3**

**Trimming Features**

The applications that this product will support are:

•

•

•

trimming black screen from the beginning and end of a spot.

extracting multiple spots from one video feed.

segmenting a spot for advertisement insertion.

**Table 2.**

**Table 3.**

page 14

**Extracting Multiple Contents**

**Single Spot Trimming**

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Spot 1** | | **Spot 2** |  | **Spot 3** |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Segment1** |  | | **Segment2** |  | **Segment3** |  | | **Segment4** |  | |

**Table 4.**

The features below represent the video editing capabilities needed for on-air playback

**Content Segmentation**

applications.

•

•

Video Tools - allows an operator to set the start and stop points.

Arbitrary Play - allows the automation system to play content at any start and stop points.

**4.3.1 Video Tools**

Video tools allow the user to examine video content so that the mark-in and mark-out points can

be set via an external interface. External interfaces include ISV automation software and MSi.

Basic video tools include pause, get current timecode, jog, and shuttle. These tools aid

automation systems in implementing the previously mentioned trimming applications.

•

•

•

•

•

•

•

•

PAUS freezes the video on a single frame. When in pause mode, only one field of the frame

will be shown. Decoders based on 68k processors will pause at the next B-frame. Decoders

based on PPC processors will pause at the next frame.

GTTC? returns the current timecode of the requested ID. The ID can be paused, playing or

recording. A maximum request rate of 3 position request commands per second is

supported.

JOG jumps from one frame to another frame by specified increments in the forward or

reverse direction. The JOG command rate should be based on waiting for the previous JOG

command to complete. This can be detected by waiting for the JOG bit in Port Status to

clear. If it is desired to send the JOG command without checking for completion, a maximum

rate of 3 per second is supported.

SHTL skips frames at varying rates playing them in forward or backward time order. The

implemented velocities depend upon the GOP size of the video being trimmed. No audio is

played while shuttling.

When the decoder is paused after a shuttle command, the application must issue a JOG 0

command to get the displayed video and current timecode in synch.

When trimming commands are being executed, a second simultaneous operation cannot be

initiated on the same decoder. In other words, a play cue command cannot be issued when

trimming commands are being used.

Trimming commands cannot be sent to a port at a rate exceeding three trimming commands

per second.

If a play duration is in effect (from a play cue command with duration) when a trimming

command is issued, the duration will be reset to the EOF of the ID. The only exception to this

rule is the sequence PCWD/PCWF, PAUS, CONT which preserves the duration specified in

the play cue command.

**4.3.2 Arbitrary Play**

Command States

page 15

Arbitrary Play allows the automation system to specify content playout at an arbitrary starting

point (SOM) for an arbitrary duration. The SOM must be between the SOC and the EOF. The

duration cannot place the EOM beyond the EOF.

page 16

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STX | Byte  Count | CMD 1 | CMD 2 | Data 1 | Data 2 | Data N | Checksum |

|  |  |  |
| --- | --- | --- |
| PIN | Controlling Device | Controlled Device |
| 1 | Frame Ground | Frame Ground |
| 2 | Receive A | Transmit A |
| 3 | Transmit B | Receive B |
| 4 | Transmit Common | Receive Common |
| 5 | Spare | Spare |
| 6 | Receive Common | Transmit Common |
| 7 | Receive B | Transmit B |
| 8 | Transmit A | Receive A |
| 9 | Frame Ground | Frame Ground |

**5** **Control Protocol**

The system will interact with a controller using a master-slave methodology. The controller will

initiate all communications between the controller and the disk system. Any error conditions will

be logged in the appropriate status byte, which the controller may then query. Play and Record

commands will be executed on the REF interval following the command. Each reference interval

is approximately1/30th of a second and is supplied on the system REF input. Commands must

be received in the first field of the frame prior to the start of the next interval.

Two command protocols will be provided. The binary protocol used via the supplied RS422 ports

is the primary method of control for frame accurate operation. A compatible ASCII protocol is

also supported over LAN and via the command line interface on the system console. All

commands in the binary protocol have an ASCII equivalent. However, not all commands in the

ASCII protocol have a binary equivalent.

**5.1**

**Binary Protocol**

**5.1.1 RS-422 Physical Protocol**

The Communications signal will conform to EIA RS-422A, asynchronous bit serial, word serial

with a transfer rate of 38.4 Kb/s. The bit configuration will consist of 1 start bit (space), 8 data bits,

1 parity bit (odd) and 1 stop bit (mark).

The Connector will be a 9 pin subminiature D with the pinout shown below:

**5.1.2 Binary Protocol Format**

**Table 5.**

**RS422 PIN Assignments**

A legal binary command is a sequence of not less than two and not more than 256 bytes

adhering to the following format.

Where:

**Table 6.**

•

**Binary Protocol Format**

Start of Transmission (STX) is 02h

Command States

page 17

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Timeout | Framing  Error | Overrun  Error | Parity  Error |  | Checksum  Error | Syntax  Error | Undefined  Error |

|  |  |
| --- | --- |
| 0 | System Control |
| 1 | Immediate Control |
| 2 | Preset/Select Control |
| 3 | Sense Request |
| 4 | Data Request |

•

•

Byte Count indicates the number of bytes between and not including the byte count byte and

the checksum byte.

CMD 1 consists of two nibbles; a command type nibble and a unit address nibble, which

defines the address of a subsystem within a device. The unit address will not be used by the

system, but is provided for protocol compatibility. Command type is one of the following:

•

**Table 7.**

**Command Types**

•

CMD 2 is the actual command or query code and it determines the amount of data, if any,

that follows.

DATA - The number of data bytes is determined by the command. The first data may contain

a bitmap byte or bytes specifying the data to follow or the data to be returned for a query. For

all bit maps a bit set to 1 designates that the bit is set to true.

NOTE: Many NEW COMMANDS use a LENGTH byte for all string data. The reason for this is to

provide compatibility with the variable ID length protocol, which supports filenames that are no

longer fixed at 8 characters. For software versions prior to B.08.60 which have implemented

some of these commands, the LENGTH byte must be set to a value of 8 for all filename

parameters. Carefully CHECK THE BINARY DATA for the new commands to avoid problems in

implementation.

•

CHECKSUM - 2's complement of the least significant byte of the sum of all CMND and DATA

bytes (The sum is from the first command byte to immediately before the checksum byte).

Commands will be specified in this ERS by CMND1.CMND2: DATA, with the unit address being a

don't care (X). For example a DEVICE TYPE query will be listed as 3X.08: NO DATA, DATA 1-2

returned. The ASCII equivalent will be shown below the binary.

All commands (type 0, 1 and 2) will be responded to by the disk system with either an ACK (04h),

indicating that the command was recognized as a valid command, or NAK (05h), indicating that

the command or query was not recognized. With a few exceptions, which are noted in the

command documentation, this response will be returned within the frame in which the command

was issued. When a NAK is returned, it will be followed by a byte specifying the nature of the

error as shown below. Undefined commands will return an error byte of 1.

Query requests (command type 3 and 4) will return the data requested in the binary protocol

**Table 8.**

**NAK Error Byte**

format previously described, and will set the most significant bit of CMND 2 to a 1 when returning

page 18

Command States

|  |  |  |
| --- | --- | --- |
| Description | Command String | Response String |
| Command | “psel 1\n” | no response |
| Command with Response Completor | “psel 1;\*OPC?\n” | “1\n” |
| Query | “open? 1,1\n” | “+1\n” |
| Query with Response Completor | “open? 1,1\n;\*OPC?\n” | “+1;1\n” |
| Null Query | “;\*OPC?\n” | “1\n” |

data. As an example, a query request command of 29 would return the STX, Byte Count,

CMND1, an A9 in the CMND 2 byte, and the requested data followed by the checksum.

**5.2**

**ASCII Protocol**

The ASCII protocol is also refered to as the MSS/LCP (MediaStream Server Lan Control

Protocol). The MSS/LCP is controlled through a TCP socket connection. An application using

the ASCII protocol must create and establish a LAN connection to the server using a socket.

Once a connection is established, commands are sent to the socket in the form of ASCII

character strings. The server reads the data, processes the command, and then responds

where appropriate.

The socket must use the TCP protocol and the connection must be made to port number 6000.

Multiple socket connections can be created with the server. However, the server has a limited

number of connections available and care must be taken to manage the number of connections.

The pool of connections is shared with other applications communicating with the server via LAN.

These applications include the MSi and applications using the hpvslapi DLL. The connection limit

is subject to change, but at this time, the total number of connections available on the

MSS700/1600 is 42. When the server reaches the connection limit, a socket connection may

appear to be successfully created. Because of this, the application should validate each

connection by sending a null query. See the table titled “ASCII Protocol Response Examples” for

an example of a null query. Because there are limitations on the number of LAN connections the

server can maintain, it is important to close socket connections when they are no longer needed.

**5.2.1 ASCII Protocol Responses**

Unlike the binary protocol, MSS/LCP does not always return a response. If a response is

returned, it will be terminated with a newline, i.e., the string “\n”. In other words, an application

can determine that it has received the complete response when it receives the newline.

The MSS/LCP has a mechanism that tells the server to respond with an acknowledgement to a

command even when there is no data to be returned. This provides a way for the controlling

application to know that the server has received the command. It is recommended that all

commands and queries teminate with this response completor. The response completor string is

“;\*OPC?\n” and is appended onto the end of the command. The string can be added to any

command or query and will result in a “1” being returned for those commands that do not return

data. Queries will have the string “;1” appended to the response when the response completor is

sent. The following table demonstrate how this works.

**Table 9.**

**ASCII Protocol Response Examples**

A given query command returns the same number of parameters regardless of the amount of

return data requested. Queries which have BITMAPs to determine the amount of data to return

will pad the return parms with zeros to maintain the maximum number of return parms for that

query. For example, the STAT? query always returns 15 integers regardless of the BITMAP

specified. Note that a given parm is not always positional for these queries. If, for example, item

Command States

page 19

2 is the only item requested on a STAT? query, the values for item 2 will immediately follow the

returned BITMAP. The examples below illustrate this point.

The command “stat? 1;\*OPC?\n” will return the following string.

“+3,+1,+1,+1,+0,+0,+0,+0,+0,+0,+0,+0,+0,+0,+0;1\n”

The command “stat? 2;\*OPC?\n” will return the following string.

“+2,+2,+4,+0,+0,+0,+0,+0,+0,+0,+0,+0,+0,+0,+0;1\n”

The command “stat? 3;\*OPC?\n” will return the following string.

+4,+3,+1,+1,+4,+0,+0,+0,+0,+0,+0,+0,+0,+0,+0

**5.2.2 ASCII Command Format**

The ASCII protocol consists of a command name followed by a fixed list of comma-separated

string and numeric parameters. Strings must be enclosed in double-quotes. All numbers are

decimal unless otherwise indicated.

Commands in the ASCII protocol do not return a response. Queries return a fixed list of comma-

separated string and numeric parameters.

Lengths, checksums and response opcodes are not used in the ASCII protocol.

**5.3**

**Comparison of Binary and ASCII protocols**

The following ASCII command uses fixed length file names.

‘PCWD "5sec

",12,0,0,0,13,4,0,0;\*OPC?\n’

It's hex equivalent in the binary protocol is as follows.

02 12 20 25 35 73 65 63 20 20 20 20 12 00 00 00 13 04 00 00 A2

The following ASCII command uses variable length file names.

‘PCWD "5sec",12,0,0,0,13,4,0,0;\*OPC?\n’

It's hex equivalent in the binary protocol is as follows.

02 0F A0 25 04 35 73 65 63 12 00 00 00 13 04 00 00 9E

**5.4**

**Command Response Times**

A command received in its entirety by the end of the first field of a frame will be accepted for

processing in that frame. If the full command is not received by the end of the first field, it will be

accepted for processing in the next frame. Only one command can be accepted in any given

frame.

All immediate commands will be executed with frame accuracy. They will be executed at the

beginning of the frame after the one in which they are accepted. In other words, if a PLAY

command is fully received in the first field of frame 10:11:12.00, the decoder will start playing at

page 20

Command States

the beginning of frame 10:11:12.01. If a PAUS command is not fully received until the second

field of frame 10:11:12.00, the decoder will pause at the beginning of frame 10:11:12.02.

With a few exceptions, which are noted in the Command Descriptions section, queries will return

the requested data within the video frame in which the request was accepted. Full device status

will be provided in less than half a frame. A status request cannot be made in the same frame as

another command or query.

Any response from the server system will always return within 500mS of receipt of the last byte of

the command. If no message is returned within 500mS, the user should assume a system

controller fault has occurred and corrective action should be taken.

Some commands and operations require multiple frames to complete. A few examples include

the IDEL and STOP commands, switching from one play stream to another and any failure

recovery mode. During these commands and operations, the PORT BUSY bit in the Port Status

State Byte will be set prior to the end of the frame in which the command is accepted. When the

BUSY bit is set, only immediate, Port Status and System Status commands will be executed. All

other commands and queries will generate a Command Received While Busy Error in Port Status

Error Byte 1 and will be ignored. As a general practice the PORT BUSY bit should always be

checked prior to issuing a command. This will maximize the throughput of commands and

minimize the amount of error recovery.

**5.5**

**Command Description Format**

The syntax of each command is specified in the section titled “Command Descriptions”. The

format used to describe the syntax is as follows.

The characters “<”, “>”, “[“, “]”, “+”, “\*” “’” and “\” are special characters. The backslash (“\”) is

used to override a special character’s meaning and render it as a literal. The single-quote (“’”)

can be used to begin and end a literal for clarity. Any characters enclosed in single-quotes are

literals.

There are three types of tokens.

Literals are sequences of non-special ASCII characters.

Symbolic tokens have the format <ID T:C:L>. ID is the symbolic token name. Symbolic token

names begin with an alphabetic character and contain no blanks. The section after the symbolic

token name, after the first blank, is a format specification.

Modified tokens have the form <token :C:L>. If the first character after the “<” is not an

alphabetic, the token is a modified token. The tokens <,> and <’,’> are both modified literals

containing a single comma. The token <<ID> :N:,> is a modified symbolic token.

The format specification is defined as follows.

T is the data type for the symbolic token. The default data type is I for integer.

S is the count or size of the token. In binary, I:2 indicates a 16-bit signed integer, U:4 indicates a

32-bit unsigned integer and C:8 indicates an eight character string. There are two modifiers for

the size specification. A plus-sign (“+”) indicates a variable size between 1 and S. An asterisk

(“\*”) indicates a variable size between 0 and S. Thus, I:4\* indicates an optional signed integer of

8, 16, 24 or 32 bits. A specification of C:32+ indicates a character string of any length between 1

and 32. The default size is 1.

Command States

page 21

L indicates the list character separating the units being specified. The specification C:4:,

indicates four characters separated by commas. The default list character is no separation

between items.

The following data types are used.

I indicates a signed integer. In binary, this is a one byte binary number and the data size

indicates the number of bytes. In ASCII, it is a single decimal number .

U indicates an unsigned integer. In binary, this is a one byte binary number. In ASCII, it is a

single decimal number.

C indicates a single ASCII character.

Fixed length file names are indicated with the specification C:8. Variable length file names are

indicated with the specification C:32+. The “+” indicating that the name is between 1 and 32

characters in length.

TC indicates Timecode (TC format). Timecode is expressed as frames, seconds, minutes, hours.

In the binary protocol, these values are in BCD format. In the ASCII protocol, these values are an

ASCII representation of a decimal format. Thus, the timecode 1 hour, 5 minutes, 10 seconds,

and 15 frames is represented as follows.

In binary protocol, the timecode 1:5:10:15 is represented as the hexidecimal number

0x15100501.

In ASCII protocol, the timecode 1:5:10:15 is represented as the string “+15,+10,+5,+1”.

B indicates a byte of binary-coded-decimal.

Opcodes are represented as modified literal tokens in the binary format since the second nibble

of CMD1 is not a fixed literal. The modified token <AX.10> indicates the opcodes A010, A110,

A210 through AF10.

page 22

Command States

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Port Status  Mode | CMD/  STATE | PLAY | PAUS | CONT | STEP | JOG | SHTL | PCUE | PCU2 | STOP |
| 0x01 | IDLE | ILLEGAL | ILLEGAL | ILLEGAL | ILLEGAL | ILLEGAL | ILLEGAL | CUE | ILLEGAL | NOP |
| 0x02 | CUE | PLAY | PAUS | ILLEGAL | ILLEGAL | ILLEGAL | NOP | CUE | CUE-CUE2 | IDLE |
| 0x02 | CUE-  CUE2 | PLAY-  CUE | PAUS-  CUE | ILLEGAL | ILLEGAL | ILLEGAL | ILLEGAL | CUE-  CUE2 | CUE-CUE2 | IDLE |
| 0x04 | PLAY | NOP | PAUS | NOP | TRIM | TRIM | TRIM | PLAY-  CUE | ILLEGAL | IDLE |
| 0x06 | PLAY-  CUE | PLAY  switch | PAUS-  CUE | NOP | ILLEGAL | ILLEGAL | ILLEGAL | PLAY-  CUE | ILLEGAL | IDLE |
| 0x24 0x44 | TRIM | PLAY | PAUS | PLAY | TRIM | TRIM | TRIM | ILLEGAL | ILLEGAL | IDLE |
| 0x0C | PAUS | PLAY | NOP | PLAY | TRIM | TRIM | TRIM | PAUS-  CUE | ILLEGAL | IDLE |
| 0x0E | PAUS-  CUE | ILLEGAL | PAUS  Switch | PLAY-  CUE | ILLEGAL | ILLEGAL | ILLEGAL | PAUS-  CUE | ILLEGAL | IDLE |

**6** **Decoder Command States**

The decoders on the MSS can be in a variety of states. There are 2 distinct types of cueing

(PCUE and PCU2) as well as many trimming commands (PAUS, CONT, STEP, JOG, and SHTL).

The operations of these commands (and PLAY and STOP) determine the state of the decoder.

The following table shows a simplified state diagram depicting how the decoders react to

incoming commands.

DefinitionC:

1.

2.

3.

4.

5.

6.

7.

The columns indicate the command being issued.

The rows indicate the current state of the decoder.

PCUE and PCWD can be interchangeably used in this chart.

PCU2 and PCD2 can be interchangeably used in this chart.

Trim indicates that the decoder is either jogging, stepping, or shuttling.

Switch indicates a transition from the clip being played to the clip that is cued.

NOP indicates that the system does nothing with the possible exception of producing

an error.

8. ILLEGAL indicates that the behavior is unpredictable and an attempt will be made to

issue an error.

9. Cueing a channel that is cued unloads the cued clip and reissues another cue

command.

**Table 10. Decoder State Diagram Table**

Command States

page 23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| COMMAND | | | RESPONSE | | |
| HEX-  CODE | TRACE FLAG | NAME | HEX-CODE | TRACE FLAG | DESCRIPTION |
| System control | | | | | |
| **0X.01** | **exe +tracev** | **PTCL** | **04** |  | **ACK** |
| **0X.02** | **exe +tracev** | **PTCL** | **04** |  | **ACK** |
| **0X.11** | **exe +tracev** | **DTYP?** | **0X.91** | **exe +resultv** | **DEVICE TYPE** |
| 0X.15  8X.15 | exe +tracev | IDPR | 04 |  | ACK |
| 0X.16  8X.16 | exe +tracev | IDUP | 04 |  | ACK |
| Immediate Commands | | | | | |
| 1X.00 | exe +trace | STOP | 04 |  | ACK |
| 1X.01 | exe +trace | PLAY | 04 |  | ACK |
| 1X.02 | exe +trace | RECD | 04 |  | ACK |
| 1X.04 | exe +trace | PAUS | 04 |  | ACK |
| 1X.05 | exe +trace | STEP | 04 |  | ACK |
| 1X.06 | exe +trace | CONT | 04 |  | ACK |
| 1X.07 | exe +trace | JOG | 04 |  | ACK |
| 1X.08 | exe +trace | SHTL | 04 |  | ACK |
| Preset/Select Commands | | | | | |
| 2X.1D  AX.1D | exe +tracem | IDRN | 04 |  | ACK |
| 2X.1F  AX.1F | exe +tracem | NCPY | 04 |  | ACK |
| 2X.21 | exe +trace | CLOS | 04 |  | ACK |
| 2X.22 | exe +trace | PSEL | 04 |  | ACK |
| 2X.23  AX.23 | exe +trace | RCUE | 04 |  | ACK |
| 2X.24  AX.24 | exe +trace | PCUE | 04 |  | ACK |
| 2X.25  AX.25 | exe +trace | PCWD | 04 |  | ACK |
| 2X.26  AX.26 | exe +trace  or  exe +tracem | IDEL | 04 |  | ACK |
| 2X.27  AX.27 | exe +tracem | MARK | 04 |  | ACK |
| 2X.29 | exe +tracev | DCLR | 04 |  | ACK |
| 2X.2B | exe +tracev | MFRE | 04 |  | ACK |
| 2X.2C  AX.2C | exe +trace | RCWD | 04 |  | ACK |
| 2X.30 | exe +tracev | PRST | 04 |  | ACK |
| 2X.31 | exe +tracev | SBIT | 04 |  | ACK |
| **2X.3D** | exe +tracev | **VSTD** | **04** |  | **ACK** |
| **2X.3F** | exe +tracev | **VILI** | **04** |  | **ACK** |
| **2X.40** | exe +tracev | **SGPI** | **04** |  | **ACK** |
| 2X.41 | exe +tracev | SADJ | 04 |  | ACK |
| 2X.42 | exe +tracev | HSYN | 04 |  | ACK |
| 2X.50  AX.50 | exe +tracem | LCPY | 04 |  | ACK |
| 2X.52  AX.52 | exe +tracem | CTER | 04 |  | ACK |
| **2X.53** | **exe +tracem** | **COPY** |  |  |  |

**7** **VDCP Implementation on MSS**

The following table summarizes the VDCP protocol as implemented on MSS. The commands in

bold have been added to VDCP for the MSS platform. The commands are a conjunction of the

commands for all platforms. See the specific commands for supported platforms. The trace flags

are based upon the PFS code set except for commands which are Warp only.

**Table 11. MSS Protocol Summary**

page 24

Command States

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AX.53** |  |  |  |  |  |
| **2X.60**  **AX.60** | **exe +tracem** | **UOPN** | **04** |  | **ACK** |
| **2X.61**  **AX.61** | **exe +tracem** | **UCLS** | **04** |  | **ACK** |
| **2X.62** | **exe +tracem** | **UADD** | **04** |  | **ACK** |
| **2X.63** | **exe +tracem** | **UMOD** | **04** |  | **ACK** |
| **2X.64** | **exe +tracem** | **UDEL** | **04** |  | **ACK** |
| **2X.65** | **exe +tracev** | **EAUD** | **04** |  | **ACK** |
| **2X.66**  **AX.66** | **exe +trace** | **PCWF** |  |  |  |
| **2X.67** | **exe +trace** | **PGID** |  |  |  |
| **2X.70** | **exe +tracev** | **GLOC** | **04** |  | **ACK** |
| **2X.71** | **exe +tracev** | **DCFG** | **04** |  | **ACK** |
| **2X.72** | **exe +tracev** | **DRFR** | **04** |  | **ACK** |
| **2X.73** | **exe +tracev** | **ENAS** | **04** |  | **ACK** |
| **2X.74**  **AX.74** | **exe +trace** | **TCUE** | **04** |  | **ACK** |
| **2X.75**  **AX.75** | **exe +trace** | **TCWD** | **04** |  | **ACK** |
| **2X.78** | **exe +trace** | **ACUE** | **04** |  | **ACK** |
| **2X.79** | **exe +tracev** | **MPRM** | **04** |  | **ACK** |
| **2X.7A** | **exe +tracev** | **AUDF** | **04** |  | **ACK** |
| **2X.7F**  **AX.7F** | **exe +tracev** | **SSOC** | **04** |  | **ACK** |
|  | **exe +tracev** | **ENVT** |  |  | **ACK** |
|  | **exe +trace** | **PRER** |  |  | **ACK** |
|  | **exe +tracev** | **ECFG** |  |  | **ACK** |
|  | **exe +tracev** | **CREF** |  |  | **ACK** |
|  | **exe +tracev** | **DAGN** |  |  | **ACK** |
|  | **exe +tracev** | **DCMD** |  |  | **ACK** |
|  | **exe +tracev** | **UCMD** |  |  | **ACK** |
|  | **exe +tracev** | **BCLR** |  |  | **ACK** |
|  | **exe +tracev** | **CCLR** |  |  | **ACK** |
|  |  | **C2DC** |  |  | **ACK** |
|  | **exe +tracev** | **AUDG** |  |  | **ACK** |
|  | **exe +tracev** | **CVFS** |  |  | **ACK** |
|  | **exe +tracev** | **LVFS** |  |  | **ACK** |
|  | **exe +tracev** | **VFIX** |  |  | **ACK** |
|  | **exe +tracev** | **FCVF** |  |  | **ACK** |
|  | **exe +tracev** | **ACVF** |  |  | **ACK** |
|  | **exe +tracev** | **OMOD** |  |  | **ACK** |
|  | **exe +tracev** | **PXFR** |  |  | **ACK** |
|  | **exe +tracev** | **RDEL** |  |  | **ACK** |
|  | **exe +trace** | **DRST** |  |  | **ACK** |
|  | **exe +tracem** | **ADDB** |  |  | **ACK** |
|  | **exe +tracev** | **ERST** |  |  | **ACK** |
|  | **exe +trace** | **ENCN** |  |  | **ACK** |
|  | **exe +trace** | **MPOR** |  |  | **ACK** |
|  | **exe +trace** | **MPOP** |  |  | **ACK** |
|  | **exe +trace** | **LCOD** |  |  | **ACK** |
| Sense Commands | | | | | |
| 3X.01 | exe +trace | OPEN? | 3X.81 | exe +result | GRANTED/DENIED |
| 3X.02  BX.02 | exe +tracem | NEXT?  VNXT?  LNXT? | 3X.82 | exe +resultm  +  exe +list | LIST OF ID’S |
| 3X.03 | exe +tracem | LAST? | 3X.83 | exe +resultm | LAST STATUS DATA |
| 3X.05 | exe +status  (use with caution) | STAT? | 3X.85 | exe +status | PORT RETURN |
| 3X.06 | exe +tracev | GTTC? | 3X.86 | exe +resultv | TIME REMAINING/  CURRENT POSITION |
| 3X.07  BX.07 | exe +tracev | IDAC?  VIDA?  LIDA? | 3X.87 | exe +resultv | ACTIVE ID |
| 3X.08 | exe +tracev | DTYP? | 3X.88 | exe +resultv | DEVICE TYPE |
| 3X.10 | exe +status  (use with caution) | SYSS? | 3X.90 | exe +status | SYSTEM RETURN |

Command States

page 25

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3X.11  BX.11 | exe +tracem | LIST?  VLST?  LLST? | 3X.91 | exe +list | LIST OF IDS |
| 3X.14  BX.14 | exe +tracev | SIZE? | 3X.94 | exe +resultv | DURATION |
| 3X.15  BX.15 | exe +tracem | IATM?  VATM?  LATM? | 3X.95 | exe +tracem +  exe +list | LIST OF IDS |
| 3X.16  BX.16 | exe +tracem | IDRQ? | 3X.96 | exe +resultm | ID STATUS |
| 3X.17 | exe +tracev | CSET? | 3X.97 | exe +resultv | COMPRESSION  SETTINGS |
| 3X.18  BX.18 | exe +tracem | IADD?  VADD?  LADD? | 3X.98 | exe +tracem +  exe +list | LIST OF IDS |
| 3X.19  BX.19 | exe +tracem | IDLT?  VDLT?  LDLT? | 3X.99 | exe +tracem +  exe +list | LIST OF IDS |
| **3X.23** | **exe +tracev** | **DSET?** | **3X.A3** | **exe +resultv** | **DECODER SETTINGS** |
| **3X.60** | **exe +tracev** | **SREV?** | **3X.E0** | **exe +resultv** | **SOFTWARE REVISION**  **NUMBER** |
| **3X.61** | **exe +tracev** | **DCFG?** | **3X.E1** | **exe +resultv** | **DECODER**  **CONFIGURATION** |
| **3X.62** | **exe +tracev** | **AUDF?** | **3X.E2** | **exe +resultv** | **AUDIO FORMAT** |
| **3X.63** | **exe +tracem** | **UQRY?** | **3X.E3** | **exe +resultm** | **USER INFO** |
| **3X.64** | **exe +tracem** | **ULST?** | **3X.E4** | **exe +resultm** | **USER INFO LIST** |
| **3X.65** | **exe +tracev** | **EAUD?** | **3X.E5** | **exe +resultv** | **EMBEDDED AUDIO** |
| **3X.66** | **exe +tracev** | **GTFR?** |  | **exe +resultv** | TIME REMAINING/  **CURRENT POSITION** |
| **3X.67** | **exe +trace** | **PGID?** |  | **exe +result** | **PROGRAM ID** |
| **3X.6F**  **BX.6F** | **exe +tracev** | **IDQY?** | **3X.EF** | **exe +resultv** | **ID INFO** |
| **3X.70**  **BX.70** | **exe +tracev** | **INFO?** | **3X.F0** | **exe +resultv** | **USER INFO** |
| **3X.71**  **BX.71** | **exe +tracev** | **IDBY?** | **3X.F1** | **exe +resultv** | **CURRENT/FINAL SIZE** |
| **3X.72** | **exe +tracev** | **MPRM?** | **3X.F2** | **exe +resultv** | **MEDIA PARAMETERS** |
| **3X.73** | **exe +tracev** | **PLMT?** | **3X.F3** | **exe +resultv** | **PARAMETER LIMITS** |
| **3X.74** | **exe +tracev** | **PSUP?** | **3X.F4** | **exe +resultv** | **PARAMETERS**  **SUPPORTED** |
| **3X.75** | **exe +tracev** | **ENAS?** | **3X.F5** | **exe +resultv** | **ENCODER AUTO STOP** |
| **3X.76** | **exe +tracev** | **DRFR?** | **3X.76** | **exe +resultv** | **DROP FRAME** |
| **3X.77** | **exe +trace** | **ACUE?** | **3X.F7** | **exe +result** | **AUDIO QUEUE** |
| **3X.78** | **exe +tracev** | **VSTD?** | **3X.F8** | **exe +resultv** | **VIDEO STANDARD** |
| **3X.79** | **exe +tracev** | **GLOC?** | **3X.F9** | **exe +resultv** | **GENLOCK MODE** |
| **3X.7A** | **exe +tracev** | **SADJ?** | **3X.EA** | **exe +resultv** | **SUBCARRIER ADJUST** |
| **3X.7B** | **exe +tracev** | **HSYN?** | **3X.8B** | **exe +resultv** | **HSYNC TIMING** |
| **3X.7C** | **exe +tracev** | **VILI?** | **3X.FC** | **exe +resultv** | **VERTICAL LINE INFO** |
| **3X.7D** | **exe +tracev** | **DSIZ?** | **3X.FD** | **exe +resultv** | **DISK STORAGE SIZE** |
| **3X.7E** | **exe +tracev** | **PTCL?** | **3X.FE** | **exe +resultv** | **PROTOCOL** |
| **3X.7F** | **exe +tracev** | **ALVL?** | **3X.FF** | **exe +resultv** | **AUDIO LEVEL** |
|  | **exe +tracev** | **SBIT?** |  | **exe +resultv** | **BITRATE** |
|  | **exe +tracev** | **ENVT?** |  | **exe +resultv** | **LINE** |
|  | **exe +trace** | **PRER?** |  | **exe +result** | **FRAMES** |
|  |  | **ECFG?** |  |  | **SETTINGS** |
|  | **exe +tracev** | **CREF?** |  | **exe +resultv** | **VALUE** |
|  | **exe +tracev** | **DAGN?** |  | **exe +resultv** | **GAIN** |
|  | **exe +tracev** | **UCMD?** |  | **exe +resultv** | **MODE** |
|  | **exe +tracev** | **DCMD?** |  | **exe +resultv** | **MODE** |
|  | **exe +tracev** | **BCLR?** |  | **exe +resultv** | **COLOR** |
|  | **exe +tracev** | **CCLR?** |  | **exe +resultv** | **COLOR** |
|  | **exe +tracev** | **MFRE?** |  | **exe +resultv** | **PERCENT FREE** |
|  |  | **C2DC?** |  |  |  |
|  | **exe +tracev** | **AUDG?** |  | **exe +resultv** | **AUDIO GAIN** |
|  | **exe +tracev** | **CVFS?** |  | **exe +resultv** |  |
|  | **exe +tracev** | **LVFS?** |  | **exe +resultv** |  |
|  | **exe +tracev** | **CVIP?** |  | **exe +resultv** |  |

page 26

Command States

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **exe +tracev** | **NDVF?** |  | **exe +resultv** |  |
|  | **exe +tracev** | **VFPO?** |  | **exe +resultv** | **PLAY ONLY** |
|  | **exe +tracev** | **OMOD?** |  | **exe +resultv** | **OPERATION MODE** |
|  | **exe +tracev** | **PXFR?** |  | **exe +resultv** | **PLAY WHILE XFER** |
|  | **exe +tracev** | **RDEL?** |  | **exe +resultv** | **RIPPLE DELETE** |
|  | **exe +tracev** | **NLST?** |  | **exe +tracev**  **+**  **exe +list** | **NODE LIST** |
|  | **exe +tracev** | **LSIZ?** |  | **exe +resultv** | **SYSTEM DISK SPACE** |
|  | **exe +tracem** | **SIMP?** |  | **exe +resultm** | **OK** |
|  | **exe +tracem** | **MIMP?** |  | **exe +resultm** | **OK** |
|  | **exe +tracem** | **FIMP?** |  | **exe +resultm** | **OK** |
|  | **exe +trace** | **ENCN?** |  | **exe +result** | **NAME** |
|  | **exe +trace** | **TSNF?** |  | **exe +result** | **PROGRAM COUNT** |
|  | **exe +trace** | **PAT?** |  | **exe +result** | **PROGRAM INFO** |
|  | **exe +trace** | **PDNF?** |  | **exe +result** | **PROGRAM INFO** |
|  | **exe +trace** | **PMT?** |  | **exe +result** | **STREAM INFO** |
|  | **exe +trace** | **MPOR?** |  | **exe +result** | **RATE** |
|  | **exe +trace** | **MPOP?** |  | **exe +result** | **SETTINGS** |
|  | **exe +trace** | **LCOD?** |  | **exe +result** | **LANGUAGE** |

Command States

page 27

**8** **Command Descriptions**

Each supported command is given in two forms: a binary format and an ASCII format. The binary

format is used over RS-422, while the ASCII format is used over LAN. The binary format is listed

as followC: CMD1X.CMD2: data specification. ASCII commands are listed with the command

name and command parameters.

All commands are implemented on all MSS platforms listed in the preface, unless otherwise

indicated.

**8.1**

**Immediate Commands**

The playable video is defined as the area of the currently playing file (or cued file if no file is

playing) between the SOM and the EOM. No commands may be issued which would cause

video to be accessed outside the playable video. This includes, but is not limited to, a PLAY or

CONT command while paused on the last frame of the playable video or a JOG command which

references a frame outside the playable video.

**8.1.1 STOP**

**BINARY: <1X.00>**

**ASCII: STOP**

The STOP command returns the selected port to the IDLE state. The STOP command can be

issued to a port in any state..

When the STOP command is issued to a play port, the decoder will display black at the next REF

interval. Any CUE operations will be aborted and any cued files will be unloaded. If the STOP

command is received when the port is IDLE, then no action results.

If an input port is in the RECORD state when a STOP is issued, the system will stop recording

within ½ second of the next REF interval. If Auto Stop is enabled, the recording will be

terminated and the length of the file will be updated to reflect this reduced length. The file is kept

stored on the disk and is available for play.

NOTE: The STOP command will be executed immediately, but will require some cleanup time to

close streams, which will result in the busy bit being set. If the busy bit was already

set when the STOP command was received, it will remain set until both the original command

and the STOP cleanup is finished.

An error condition will result in the appropriate bit being set in the port status error bytes.

**8.1.2 RECORD**

**BINARY: <1X.02>**

**ASCII: RECD**

The RECORD command causes the selected input port to begin recording. This command is

only legal when received on an input port. The RECORD command should only be issued when

the port is in the CUE state and the CUE\_DONE flag is set in the port status. If a RECORD CUE

page 28

Command States

|  |  |  |
| --- | --- | --- |
| Time | Command | Output |
| 0.0 | pcue <IDI> | black |
| 0 |  | black |
| 2.9 |  | black |
| 3.0 | play | ID1 |
|  |  | ID1 |
| 30.0 | pcue <ID2> | ID1 |
|  |  | ID1 |
| 32.9 |  | ID1 |
| 33.0 | play | ID2 |

command had not been issued prior to the RECORD command, an error will result and the port

will remain in its former state. If the RECORD command is received in the first field of the frame,

the recording will start at the next REF interval. Otherwise, the recording will start at the following

REF interval (i.e. 1 frame shift).

If the encoder is in Auto Stop mode, the port will complete the record and return to the IDLE state

when the duration specified in the RCUE or RCWD command is reached. If Auto Stop is not

enabled, the port will continue to record until a STOP command is received. In either case, the

encode may record up to ½ second past the expected termination of the record. In Auto Stop

mode, the duration of the file, as reported by the SIZE? query, will be the duration specified in the

RCUE or RCWD command. If Auto Stop is not enabled, the duration of the file, as reported by

the SIZE? query, will be the actual recorded duration.

If the Play While Transfer (PXFR) setting is enabled, the new ID will be available for playout as

soon as sufficient data has been recorded. The Ids Added bit will be set in the port status at that

time, and an IDRQ? or IDQY? will have the ID\_IN\_DISK and ID\_IN\_XFER bits set. If Play While

transfer is not enabled, the ID will not appear in the IDS-ADDED list and the ID\_IN\_DISK bit will

remain cleared, until the record has finished.

An error condition will result in the appropriate bit being set in the port status error bytes. If the

LENGTH is zero, then the recording will stop when a STOP command is issued or the server runs

out of disk space.

**8.1.3 PLAY**

**BINARY: <1X.01>**

**ASCII: PLAY**

The PLAY command causes the currently cued file to be played out of the selected port. The port

must be in CUE state with the CUE\_DONE flag set or in one of the trimming states (SHTL, STILL,

JOG). Otherwise, an error will be logged. If the PLAY command is received in the first field of

the frame prior to a REF interval, the file will begin playing at that REF interval. An error condition

will result in the appropriate bit being set in the port status error bytes. It is illegal to issue a PLAY

command when the decoder is paused on the final frame of the duration specified in the CUE

command. The behavior in this case is undefined.

Example:

Assume ID1 is 30 seconds, ID2 is 60 seconds, ID3 is 15 seconds and ID 4 is 10 seconds in

recorded length.

**Table 12. System output during back-to-back play**

Command States

page 29

|  |  |  |
| --- | --- | --- |
|  |  | ID2 |
| 40.0 | pcue <ID3> | ID2 |
|  |  | ID2 |
| 42.9 |  | ID2 |
| 43.0 | play | ID3 |
|  |  | ID3 |
| 57.9 |  | ID3 |
| 58.0 | pcue <ID4> | black |
|  |  | black |
| 60.9 |  | black |
| 61.0 | play | ID4 |
|  |  | ID4 |
| 70.9 |  | ID4 |
| 71 |  | black |

In this example ID1 is played for its entire length of 30 seconds. It is followed by the first 10

seconds of ID2, which is cut off by ID3. ID3 plays for its entire length of 15 seconds. ID3 is

followed by 1 second of BLACK output since no play command was received. ID4 is then played

for its entire length of 10 seconds. The system then outputs BLACK unless freeze on last frame

is set.

**8.1.4 PAUSE**

**BINARY: <1X.04>**

**ASCII: PAUS**

The PAUS command has the following behavior depending upon the state of the decoder.

If the decoder is in PLAY or PLAY-CUE state, the PAUS command causes the currently playing

ID to pause and display a still frame of video.

If the decoder is in CUE or CUE-CUE2 state, the PAUS command causes the first frame of the

cued file to be displayed.

If the decoder is in PAUS-CUE state, the PAUS command causes a pause switch. The display

changes from the current frame of the paused file to the first frame of the cued file.

If the decoder is in TRIM state, the PAUS command causes the decoder to enter PAUS state and

display a still frame of video.

If the decoder is IDLE the NOT\_IN\_CUE\_STATE error condition will be set in the port status

bytes.

If the decoder is in the PAUS state the PAUS command is a NOP.

When the PAUS command is completed, the port will transition to the PAUS or PAUS-CUE state

and the PLAY and STILL status bits will be set in the port status. The PAUS command is not

complete until the STILL bit has been set in the port status. Any commands effecting the decoder

state should not be sent until this transition has occurred, or results could be unpredictable. An

error condition will result in the appropriate bit being set in the port status error bytes.

Different decoder types have different pausing characteristics. When transitioning from PLAY or

TRIM to PAUS, some will display the next frame of video and some will skip a few frames. Exact

positioning of the displayed video should be achieved with the GTTC? and JOG commands.

page 30

Command States

The PAUS command may make the port busy during execution, i.e., prior to either failure or

completion. During this time the port busy condition may be raised.

The PAUS command is ignored when the port BUSY condition is raised.

**8.1.5 STEP**

**BINARY: <1X.05>**

**ASCII: STEP**

The STEP command causes the currently paused file to advance to the next frame and pause

again. The ID must have a frame table track in order for this command to function. The output

port must be in PLAY and STILL state or an error will be logged. When the STEP command is

received in the first field of the frame prior to a REF interval, the spot will advance one frame at

that REF interval. An error condition will result in the appropriate bit being set in the port status

error bytes.

**8.1.6 CONTINUE**

**BINARY: <1X.06>**

**ASCII: CONT**

The CONT command causes the currently paused or trimming ID to exit PAUS or TRIM state and

transition to the PLAY state. If the port is in PAUS-CUE state, the cued file will remain cued. The

command is a NOP in PLAY or PLAY-CUE state. The command is illegal in all other states and

the behavior is undefined.

The rules for cueing while transferring or recording apply to the use of the CONT command. If

there is not enough frame table or video on disk to perform a cue to the currently paused frame,

the CONT command will fail and the OPERATION\_FAILED condition will be set in the port status

error bytes. It is illegal to issue a CONT command when the decoder is paused on the final frame

of the duration specified in the CUE command.

When the decoder is in the PAUS state, the CONT command causes a PLAY transition in the

next frame. It is illegal to issue a CONT command when the decoder is paused on the final frame

of the duration specified in the CUE command.

When the decoder is in the TRIM state, the CONT command causes a PLAY transition which

may take several frames to complete.

If the decoder is in CUED state, the CONT command is a NOP.

An error condition will result in the appropriate bit being set in the port status error bytes.

The CONT command may make the port busy during execution, i.e., prior to either failure or

completion. During this time the port busy condition may be raised.

The CONT command is ignored when the port BUSY condition is raised.

Command States

page 31

**8.1.7 JOG**

**BINARY: <1X.07><FRAMES I:4+>**

**ASCII: JOG <FRAMES>**

FRAMES represent the number of frames to jog.

The JOG command causes the decoder to move the specified number of frames in the forward or

reverse direction relative to the current position. The ID must have a frame table track in order for

this command to function. The PLAY and JOG bits are set while the jog operation is in progress.

An error will occur if the output port is not in the PLAY, CUED, or PAUS state. The JOG

command rate should be throttled by waiting for the previous JOG command to complete. This

can be detected by waiting for the JOG bit in Port Status to clear. If it is desired to send the JOG

command without checking for completion, a maximum request rate of 3 commands per sec is

recommended.

If the decoder is in CUED state, the JOG command is a NOP.

In the binary protocol, the byte count determines the number of DATA bytes. The number of

frames is a signed integer. For example, if there are four DATA bytes the range is from negative

2,147,483,648 to 2,147,483,647.

The JOG command is ignored when the port BUSY condition is raised.

**8.1.8 SHUTTLE**

**BINARY: <1X.08><VELOCITY I:3>**

**ASCII: SHTL <VELOCITY>**

VELOCITY represents the direction and speed at which the output port should be played.

The SHTL command instructs the decoder to play the video at the specified velocity. The SHTL

command may be issued to a port in CUED, PLAY or TRIM states. The command is illegal in all

other states and the behavior is undefined. If the decoder is in CUED state, the SHTL command

is a NOP.

The SHTL command should be sent no faster than a rate of 3 commands per second.

If VELOCITY is zero, the command is equivalent to a PAUS command and the decoder will

transition to a PAUS state. If VELOCITY is 0x001000, the command is equivalent to a CONT

command and the decoder will transition to the PLAY state. For all other values of VELOCITY,

the behavior is dependent upon the implemented velocities as described below. If the calculated

velocity does not round to a resulting velocity of 0 or 1, the decoder will begin a shuttle operation.

The PLAY and SHTL bits are set and the port is in TRIM state during shuttle operation. Shuttle

operation will continue until the state is altered or EOF/SOC is reached.

The implemented velocities for shuttling are dependent on the GOP size of the video being

trimmed. To calculate the velocity, multiply the desired speed and direction by 65536 or

0x10000. The calculated velocity will be rounded to one of the implemented velocities supported

by the GOP size. The implemented velocities are listed in the table titled “Shuttle Velocity”.

The calculated velocity is rounded to the nearest implemented velocity. If the calculated velocity

falls midway between two implemented velocities, the rounding is not guaranteed to follow a

page 32

Command States

|  |  |  |  |
| --- | --- | --- | --- |
| Decimal  Value | Hex Value | Resulting velocity | GOP size supported |
| 14155776 | 0x D80000 | 40 times play speed reverse | 15-frame GOP |
| 15466496 | 0x EC0000 | 20 times play speed reverse | 15-frame GOP |
| 16121856 | 0x F60000 | 10 times play speed reverse | 15-frame GOP |
| 16449536 | 0x FB0000 | 5 times play speed reverse | 15-frame GOP |
| 16613376 | 0x FD8000 | 2.500 times play speed reverse | 15-frame GOP |
| 16621568 | 0x FDA000 | 2.375 times play speed reverse | 15-frame GOP |
| 16629760 | 0x FDC000 | 2.250 times play speed reverse | 15-frame GOP |
| 16637952 | 0x FDE000 | 2.125 times play speed reverse | 15-frame GOP |
| 16646144 | 0x FE0000 | 2.000 times play speed reverse | 15-frame GOP |
| 16654336 | 0x FE2000 | 1.875 times play speed reverse | 15-frame GOP |
| 16662528 | 0x FE4000 | 1.750 times play speed reverse | 15-frame GOP |
| 16670720 | 0x FE6000 | 1.625 times play speed reverse | 15-frame GOP |

specific rounding algorithm. Some rounding examples can be found in the table titled “Shuttle



Velocity Rounding Examples”.

When shuttling in slow motion, only one field of the frame will be shown. This will prevent jitter

from both fields being shown simultaneously in the frame. The implemented velocities for slow

motion are listed in the table titled “Shuttle Velocity”. Currently, audio is not played.

During fast motion shuttling, not every frame is played. Only I-frames are played. At 2.5 times

faster, every I-frame is played. At 5.0 times faster, every other I-frame is played. The faster the

speed, the more I-frames are skipped. A smaller GOP size means slower speed. The

implemented velocities for fast motion are listed in the table titled “Shuttle Velocity”. Currently,

audio is not played.

Fast forward and reverse on the MSS are not intended to fully emulate VTR shuttling. The

reason is that MSS is digital and a VTR is analog. MSS will show a frame at a time, while a VTR

shows parts of different frames on the screen simultaneously giving the view a smoother look.

Since only I-frames are shown in fast motion shuttling, the MSS will always show the same I-

frame at the same speed. The table below shows how the speed of fast motion is related to

skipping I-frames. The faster the speed, the more I-frames are skipped.

**Table 13. Relationship Between Shuttle Speeds and I-frames**

**MPEG Stream**

**I B B P**

**I B B P**

**I B B P**

**I B B P**

**I**

**B B P**

**2.5**

**5.0**

**2.5**

**2.5**

**5.0**

**2.5**

**2.5**

**5.0**

**10.0**

**Speed relative to normal play**

**10.0**

**Table 14.  Shuttle Velocity**

Command States

page 33

|  |  |  |  |
| --- | --- | --- | --- |
| Desired  Velocity | Calculated  Veocity | Implemented Velocity | Resulting Velocity |
| 50 x R | 0xCE0000 | 0xD80000 | 40 times play speed reverse |
| 30 x R | 0xE20000 | 0xD80000 or 0xEC0000 | 40 or 20 times play speed reverse |
| 25 x R | 0xE70000 | 0XEC0000 | 20 times play speed reverse |
| 2½ x R | 0xFD8000 | 0xFD8000 | 2.500 times play speed reverse |
| 0.7 x R | 0xFF4CCD | 0xFF4000 | 0.75 times play speek reverse |
| 0.937 x F | 0x00F001 | 0x010000 | standard play speed forward |
| 1.062 x F | 0x010FFF | 0x010000 | standard play speed forward |
| 3xF | 0x030000 | 0x028000 | 2.500 times play speed forward |

|  |  |  |  |
| --- | --- | --- | --- |
| 16678912 | 0x FE8000 | 1.500 times play speed reverse | 15-frame GOP |
| 16687104 | 0x FEA000 | 1.375 times play speed reverse | 15-frame GOP |
| 16695296 | 0x FEC000 | 1.250 times play speed reverse | 15-frame GOP |
| 16703488 | 0x FEE000 | 1.125 times play speed reverse | 15-frame GOP |
| 16711680 | 0x FF0000 | 1.000 times play speed reverse | 15-frame GOP |
| 16719872 | 0x FF2000 | 0.875 times play speed reverse | 15-frame GOP |
| 16728064 | 0x FF4000 | 0.750 times play speed reverse | 15-frame GOP |
| 16736256 | 0x FF6000 | 0.625 times play speed reverse | 15-frame GOP |
| 16744448 | 0x FF8000 | 0.500 times play speed reverse | 15-frame GOP |
| 16752640 | 0x FFA000 | 0.375 times play speed reverse | 15-frame GOP |
| 16760832 | 0x FFC000 | 0.250 times play speed reverse | 15-frame GOP |
| 16769024 | 0x FFE000 | 0.125 times play speed reverse | 15-frame GOP |
| 16775032 | 0x FFF777 | 0.033 times play speed reverse | 15-frame GOP |
| 0 | 0x 000000 | 0        Still | 15-frame GOP |
| 2185 | 0x 000889 | 0.033 times play speed forward | 15-frame GOP |
| 8192 | 0x 002000 | 0.125 times play speed forward | 15-frame GOP |
| 16384 | 0x 004000 | 0.250 times play speed forward | 15-frame GOP |
| 24576 | 0x 006000 | 0.375 times play speed forward | 15-frame GOP |
| 32768 | 0x 008000 | 0.500 times play speed forward | 15-frame GOP |
| 40960 | 0x 00A000 | 0.625 times play speed forward | 15-frame GOP |
| 49152 | 0x 00C000 | 0.750 times play speed forward | 15-frame GOP |
| 57344 | 0x 00E000 | 0.875 times play speed forward | 15-frame GOP |
| 65536 | 0x 010000 | 1.000 Std play speed forward | 15-frame GOP |
| 73728 | 0x 012000 | 1.125 times play speed forward | 15-frame GOP |
| 81920 | 0x 014000 | 1.250 times play speed forward | 15-frame GOP |
| 90112 | 0x 016000 | 1.375 times play speed forward | 15-frame GOP |
| 98304 | 0x 018000 | 1.500 times play speed forward | 15-frame GOP |
| 106496 | 0x 01A000 | 1.625 times play speed forward | 15-frame GOP |
| 114688 | 0x 01C000 | 1.750 times play speed forward | 15-frame GOP |
| 122880 | 0x 01E000 | 1.875 times play speed forward | 15-frame GOP |
| 131072 | 0x 020000 | 2.000 times play speed forward | 15-frame GOP |
| 139264 | 0x 022000 | 2.125 times play speed forward | 15-frame GOP |
| 147456 | 0x 024000 | 2.250 times play speed forward | 15-frame GOP |
| 155648 | 0x 026000 | 2.375 times play speed forward | 15-frame GOP |
| 163840 | 0x 028000 | 2.500 times play speed forward | 15-frame GOP |
| 327680 | 0x 050000 | 5 times play speed forward | 15-frame GOP |
| 655360 | 0x 0A0000 | 10 times play speed forward | 15-frame GOP |
| 1310720 | 0x 140000 | 20 times play speed forward | 15-frame GOP |
| 2621440 | 0x 280000 | 40 times play speed forward | 15-frame GOP |

page 34

**Table 15.  Shuttle Velocity Rounding Examples**

Command States

The SHTL command is ignored when the port BUSY condition is raised.

**8.2**

**PORT CONTROL COMMANDS**

**8.2.1 OPEN QUERY**

**BINARY: <3X.01><PORT><LOCKMODE>**

**returns <STATUS>**

**ASCII: OPEN? <PORT>,<LOCKMODE>**

**returns <STATUS>**

PORT contains an 8 bit sign magnitude number specifying the video PORT.

LOCKMODE contains either a 1 for LOCKED mode or a 0 for UNLOCKED mode.

STATUS returns 1 for success, 0 for failure.

The OPEN command is used to establish a series of possible connections between a RS422

communications port and specified video ports. Only one communications port can have a given

video port open at a given time. There are two modes in which a port can be opened, LOCKED

and UNLOCKED. An OPEN request for an unopened video port will result in a 1 (port granted)

being returned in Data 1. An OPEN request for a currently open and LOCKED video port will be

refused automatically and a 0 (port denied) will be returned in rData 1. An OPEN request for a

video port that has been opened in the UNLOCKED mode will result in a 1 (port granted) being

returned to the requesting communications port and the PORT BUSY bit being set in the Port

Status until the video port has been reset to IDLE state. The communications port that previously

had the video port open will no longer have the port and if it was currently selected, it will have

the INVALID PORT bit set in the port error status. An error condition will result in the appropriate

bit being set in the port status error bytes.

**8.2.2 CLOSE**

**BINARY: <2X.21><PORT>**

**ASCII: CLOS <PORT>**

PORT contains an 8 bit sign magnitude number specifying the video port.

The CLOSE command is used to close a possible connection established by a proceeding OPEN

command. After the execution of a CLOSE command, the communications port is left connected

to the system port, PORT 0. An error condition will result in the appropriate bit being set in the

port status error bytes.

**8.2.3 PORT SELECT**

**BINARY: <2X.22><PORT>**

**ASCII: PSEL <PORT>**

PORT contains an 8 bit sign magnitude number specifying the video PORT.

The PORT SELECT command selects a port from the available open connections established by

a proceeding OPEN command and establish a link between the communications Port and the

selected Video Port. All subsequent commands arriving at the associated RS422 port will be

Command States

page 35

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Status 7 |  | Status 5 | Status 4 | Status 3 | Status 2 | Status 1 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE  DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/  RECORD | CUE | IDLE |
| SELECTED VIDEO PORT NUMBER | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | IDS ADDED  TO  MAIN/ARCH | IDS  DELETED | IDS  ADDED |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | IDS ADDED | IDS | IDS |  |

routed to the assigned Video Port until another PORT SELECT or CLOSE command is received.

PORT is a number representing the available video ports with input ports being negative numbers

and output ports being positive numbers. Thus the first input port would be port -1, while the

second output port would be port 2. An error will result if an invalid PORT is specified. Illegal

combinations of commands with the type of port selected will result in an error. If the PORT has

not already been OPENED by the requesting communications port, an INVALID PORT error will

be logged and the port control will remain unchanged. An error condition will result in the

appropriate bit being set in the port status error bytes.

**8.2.4 PORT STATUS QUERY**

**BINARY: <3X.05><BITMAP>**

**returns <rBITMAP>[<rSTAT1 I:2>][<rSTAT2 I:2+>][<rSTAT3 I:3>][<rSTAT4>][<rSTAT5 I:4\*>**

**ASCII: STAT? <BITMAP>**

**returns <rLEN>,<rBITMAP>[,<rSTAT1 I:2:,>][,<rSTAT2 I:2+:,>]**

**[,<rSTAT3 I:3:,>][,<rSTAT4>][,<rSTAT5 I:4\*:,>]<,+0 I:9-<rLEN>>**

BITMAP contains a bitmap specifying the Status items to be returned. The Port Status 7 bit

enables extended status for Item 2.

rLEN (ASCII ONLY) returns the byte count of the returned rSTAT items plus 1.

rBITMAP returns a bitmap specifying which status items have been returned.

rSTATX returns the status byte(s) specified below for Port Status Item X.

The STAT? query retuns current status for the selected port for the items specified in the

BITMAP.

The port status items available and their contents are shown in the tables below.

**Table 16.  Port Status Query (STAT?) Bitmap**

page 36

**Table 17.  Port Status Item 1:  State Status**

**Table 18.  Port Status Item 2:  List Status**

**Table 19.  Extended Port Status Item 2:  List Status**

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | TO  MAIN/ARCH | DELETED | ADDED |  |
|  |  |  |  |  |  |  | NO  TIMECODE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NOT  SUP-  PORTED | CMD  WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT  TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID  DELETE  PRO-  TECTED | XFER  COM-  PLETE | XFER  FAILED | ID CUED  OR  PLAY-  ING | ID STILL  RECORD  -ING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID  ID |
| SYSTEM  REBOOT |  | OPERA-  TION  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE  NOT  DONE | NOT IN  CUE  STATE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GPI  ENABLED |  |  | D1 | YUV | S VIDEO | COMPOSITE | OFF |

|  |
| --- |
| Number of Format bytes following |
| Byte representing format 1 |
| Byte representing format 2 |
| Byte representing format 3 |

The second byte of Item 2 is present only if extended status is enabled with the Port Status 7 bit.

The NO TIMECODE bit of Item 2 is set if VITC capture on encode is enabled but no valid

timecode is found on the specified line during CUE and RECORD. This bit is reset when the

record finishes or the cue is aborted without recording.

On PFS systems, the VFS information is centrally located on the FSC’s, but is distributed to each

node for performance reasons. This means that there may be a delay in the time it takes the

information to be updated on different nodes in the same cluster. For this reason, the results of

list and ID characteristic queries and the list status option of the port status query cannot be

shared across ports. For example, when an ID appears on the IDS\_ADDED list of one port, it

may not be IN\_DISK as far as other ports are concerned.

Available Media Formats for Item 5 are: JPEG = 1, MPEG420 = 2, MPEG422 = 3.

**Table 20.  Port Status Item 3:  Error Status**

**Table 21.  Port Status Item 4:  Settings**

**Table 22.  Port Status Item 5:  Supported Media Formats**

**8.2.5 LAST STATUS QUERY**

**BINARY: <3X.03>**

**returns <rBITMAP><STATUS I:13\*>**

**ASCII: LAST?**

Command States

page 37

**returns <LENGTH>,<rBITMAP>,<STATUS> I:13:,>**

The LAST STATUS query returns a copy of the last status data that was queried using any one of

the following commandC:

•

•

3X.05 PORT STATUS QUERY.

3X.10 SYSTEM STATUS QUERY.

Note that the status returned will not necessarily be the current status as the instrument state

may have changed since the original status query. If there has been no Status query since power

up, a 0 will be returned in DATA 1.

**8.2.6 ACTIVE ID QUERY**

**VARIABLE NAME:**

**BINARY: <BX.07>**

**returns <STATUS><LEN><ID C:32+>**

**ASCII: VIDA?**

**returns <STATUS>,”<ID C:32+>”**

**ASCII: LIDA?**

**returns <STATUS>,”<ID C:32+>”**

**FIXED NAME:**

**BINARY: <3X.07>**

**returns <STATUS><ID C:8>**

**ASCII: IDAC?**

**returns <STATUS>,”<ID C:8>”**

STATUS returns the active status.

LEN returns the length of the ID string.

ID returns the name of the active ID.

The IDAC? query returns information about whether the selected port is active and the name of

the active ID. An Active port is one that is either recording or playing. This query does not affect

the output of the system.

If the port is not active, a 0 is returned in STATUS and ID returns null. If a FIXED NAME ID is

active, STATUS returns 1 and all versions of the command return the name of the active ID. If a

variable name ID is active, STATUS returns 1 and the variable name commands return the name

of the active ID, but the FIXED NAME commands return a null string.

**8.2.7 POSTION REQUEST QUERY**

**BINARY: <3X.06>**

**returns <rTYPE><POS TC>**

**ASCII: GTTC? <TYPE>**

**returns <rTYPE>,<POS TC>**

TYPE specifies the type of position request. A 0 requests the time remaining and a 1 requests

current position.

page 38

Command States

rTYPE returns the type requested.

POS returns the requested position in TC format.

The GTTC? query returns timecode information on the file currently playing / recording on the

selected port.

Get timecode is valid when the selected port is in cue, play, pause, or record modes. The return

is valid every 15 frames.

If the active ID has a SOC, the timecode returned will be relative to the SOC.

**8.2.8 POSITION REQUEST BY FRAMES QUERY**

**BINARY: <3X.66><TYPE>**

**returns <rTYPE><POS U:4>**

**ASCII: GTFR? <TYPE>**

**returns <rTYPE>,<POS U>**

TYPE specifies the type of position request. A 0 requests the number of remaining frames, and a

1 requests the current frame number.

rTYPE returns the type requested.

POS returns the requested frame number as a 32-bit unsigned integer.

The GTFR? query returns frame number information on the file currently playing / recording on

the selected port.

This query is valid when the selected port is in cue, play, pause, or record modes. The return is

valid every 15 frames.

Any SOC set on the active ID will be ignored.

Command States

page 39

**8.3**

**CUEING COMMANDS**

**8.3.1 RECORD CUE**

**VARIABLE NAME BINARY: <AX.23><LEN><ID C:32+><DURATION TC>**

**FIXED NAME BINARY: <2X.23><ID C:8><DURATION TC>**

**ASCII: RCUE "<ID C:32+>”,<DURATION TC>**

LEN contains the length in bytes of the ID name.

ID contains the ID name.

DURATION contains the desired length of time for the recording in TC format.

Issuing the RECORD CUE command with a Video Input Port selected causes the server to

prepare for recording a spot. If an output port has been selected, an error is generated. The

RECORD CUE command consists of the command itself followed by an ID, followed by a

DURATION. The DURATION is the duration of record in TC format. An error will be logged if

DURATION is outside the specified range. The RECORD CUE command should only be issued

to the port in the IDLE state. If the port is not in the IDLE state when a RECORD CUE command

is received, an error is logged and the port remains in its prior state. If in the IDLE state, the port

will check if the ID is unique. If the ID is not unique, then an error is logged and the port is left in

the IDLE state. If the ID is unique then the server checks for enough disk space remaining to

record the DURATION specified. If there is not enough disk space, then an error is generated

and the port is left in the IDLE state. If there is enough space, then the port is put into the CUE

state and is ready to receive a RECORD command when the CUE is done. When the CUE has

been completed, the CUE\_DONE flag is set in the Port Status 1 byte.

An error condition will result in the appropriate bit being set in the port status error bytes.

**8.3.2 RECORD CUE WITH DATA**

**VARIABLE NAME BINARY: <AX.2C><LEN><ID C:32+><START TC><DURATION TC>**

**FIXED NAME BINARY: <2X.2C><ID C:8><START TC><DURATION TC>**

**ASCII: RCUE "<ID C:32+>”,<START TC>,<DURATION TC>**

LEN contains the length in bytes of the ID name.

ID contains the ID name to be recorded.

START contains the SOC for the ID to be recorded.

DURATION contains the desired DURATION timecode.

Please see the section titled “Timecode (TC format)” for details on the TC format.

The RECORD CUE WITH DATA command is almost identical to the RECORD CUE command

except that it specifies the START timecode for the id to be recorded. If NTSC drop frame is

enabled and the SOC is not a valid dropframe timecode, the SOC will not be set and the

OPERATION\_FAILED status flag will be raised. An error condition will result in the appropriate

bit being set in the port status error bytes.

**8.3.3 TIME DELAY RECORD CUE**

**VARIABLE NAME BINARY: <AX.74><LEN><ID C:32+><DUR TC><MAX\_TIME TC>**

**FIXED NAME BINARY: <2X.74><ID C:8><DUR TC><MAX\_TIME TC>**

**ASCII: TCUE "<ID C:32+>”,<DUR TC>,<MAX\_TIME TC>**

page 40

Command States

LEN contains the length in bytes of the ID Name.

ID contains the specified ID Name.

DURATION contains the Duration timecode of the ID.

MAX\_TIME contains the timecode representing the maximum delay that will ever be required for

this ID.

The TCUE command functions very similarly to the record cue (RCUE) command except that it

configures the file for time delay use. Every time the recording reaches a multiple of MAX\_TIME,

the file will be automatically overwritten from the beginning as it continues recording. The file size

never exceeds MAX\_TIME. If MAX\_TIME is 0:0:0:0, then the auto erase functionality is turned

off. MAX\_TIME must be zero, or greater than15 minutes and less than the largest available chunk

of disk space.

The following restrictions apply to cueing and playing from a circular file.

•

•

•

•

Do not attempt to use a mark in (PCWD and PCWF) which is less than 2 minutes from

the current end of file. Thus, the minimum time delay possible using these commands is

two minutes.

When using a circular file (TCUE) for recording, do not mark in (PCWD and PCWF) to a

point that has been removed from the buffer. For example, after 4 hours of recording a

circular file with a 1 hour MAX\_TIME, the legal values for mark in are between 3:0:0.0

and 3:58:0.0.

Playing from a circular file (TCUE) must also conform to the above restriction. The PLAY

command must be issued before the point which is currently cued is overwritten in the

circular file.

A PCUE command can only be used to cue a curcular file (TCUE) between the time the

file becomes IN\_DISK and the first time the file is overwritten (MAX\_TIME). In this

manner, a delay can be cued as soon as the file becomes IN\_DISK, and played as soon

as the cue reaches the CUE\_DONE state.

Violation of these rules will result in unpredictable behavior.

**8.3.4 TIME DELAY RECORD CUE WITH DATA**

**VARIABLE NAME BINARY:**

**<AX.75><LEN>< ID C:32+> <START TC><DURATION TC><MAX\_TIME TC>**

**FIXED NAME BINARY:**

**<2X.75><ID C:8> <START TC><DURATION TC><MAX\_TIME TC>**

**ASCII: TCWD "<ID C:32+>",<START TC>,<DURATION TC>,<MAX\_TIME TC>**

LEN contains the length in bytes of the ID name.

ID contains the specified ID Name.

START specifies the starting timecode of the ID to be recorded.

DURATION is the Duration timecode of the ID in TC format.

MAX\_TIME is the timecode representing the maximum delay that will ever be required for this ID.

The TCWD command functions very similarly to the record cue (RCUE) command except that it

configures the file for time delay use. Every time the recording reaches a multiple of MAX\_TIME,

the file will be automatically overwritten from the beginning as it continues recording. The file size

never exceeds MAX\_TIME. If MAX\_TIME is 0:0:0:0, then the auto erase functionality is turned

off. MAX\_TIME must be zero, or greater than15 minutes and less than the largest available chunk

of disk space. START allows the specification of a starting time code (SOC) other than 0:0:0.0. If

Command States

page 41

NTSC drop frame is enabled and the SOC is not a valid dropframe timecode, the SOC will not be

set and the OPERATION\_FAILED status flag will be raised.

The following restrictions apply to cueing and playing from a circular file.

•

•

•

•

Do not attempt to use a mark in (PCWD and PCWF) which is less than 2 minutes from

the current end of file. Thus, the minimum time delay possible using these commands is

two minutes.

When using a circular file (TCUE) for recording, do not mark in (PCWD and PCWF) to a

point that has been removed from the buffer. For example, after 4 hours of recording a

circular file with a 1 hour MAX\_TIME, the legal values for mark in are between 3:0:0.0

and 3:58:0.0.

Playing from a circular file (TCUE) must also conform to the above restriction. The PLAY

command must be issued before the point which is currently cued is overwritten in the

circular file.

A PCUE command can only be used to cue a curcular file (TCUE) between the time the

file becomes IN\_DISK and the first time the file is overwritten (MAX\_TIME). In this

mannor, a delay can be cued as soon as the file becomes IN\_DISK, and played as soon

as the cue reaches the CUE\_DONE state.

Violation of these rules will result in unpredictable behavior.

**8.3.5 PLAY CUE**

**VARIABLE NAME BINARY: <AX.24><LEN><ID C:32+>[<OPT\_LEN><OPT\_FUNC C:M\*>]**

**FIXED NAME BINARY: <2X.24><ID C:8>[<OPT\_LEN><OPT\_FUNC C:M\*>]**

**ASCII: PCUE "<ID C:32+>"**

**ASCII: PCU2 "<ID C:32+>"**

LEN contains the length in bytes of the ID string.

ID contains the ID name to be cued for play.

OPT\_LEN contains the number of optional bytes if an optional function is provided.

OPT\_FUNC contains the optional function.

The PLAY CUE command causes the selected port to prepare to play the specified ID. If the ID

is not found, an error occurs and the state returns to either IDLE or PLAY if the port is currently

playing another file. When the CUE has been completed, the CUE\_DONE flag is set in the Port

Status 1 byte. If the PLAY command is received prior to the CUE\_DONE bit being set in the port

status, an error will be logged in the appropriate port status error byte.

If PLAY-WHILE-TRANSFER is disabled when a PCUE command is received for a MARKed ID

that has not completed transferring, an error will be logged. If PLAY-WHILE-TRANFER is

enabled, the rules in the section titled**Cueing While Transferring or Recording** apply.

If another CUE command is received prior to receiving the PLAY command, then the previous

CUE will be aborted and the new item cued. When the PLAY command is received, the spot will

begin playing on the next REF interval.

An error condition will result in the appropriate bit being set in the port status error bytes. When a

CUE command is issued (PCUE, PCWD or PCWF), no more CUE commands should be sent to

the port until either the CUE\_DONE bit is set or an error bit is set.

The following restrictions apply to cueing and playing files using the PCUE and PCU2 commands.

page 42

Command States

•

•

A PCUE command can only be used to cue a circular file (TCUE) between the time the

file becomes IN\_DISK and the first time the file is overwritten (MAX\_TIME). In this

manner, a delay can be cued as soon as the file becomes IN\_DISK, and played as soon

as the cue reaches the CUE\_DONE state.

Playing from a circular file (TCUE) must also conform to the above restriction. The PLAY

command must be issued before the point which is currently cued is overwritten in the

circular file.

Violation of these rules will result in unpredictable behavior.

In the binary protocol, if optional data is sent, then OPT\_LEN will indicate the number of bytes

optional data to follow.

If OPT\_FUNC is 1, OPT\_LEN must be 1 and a normal play cue command will be executed.

If OPT\_FUNC is 2, OPT\_LEN must be 1 and the PCU2 operation will be performed. This function

is useful for playing spots at the beginning of a POD which are too short to allow time for the

subsequent cue operation. One example is playing 1 second station IDs at the beginning of a

commercial segment. Once the short spot is cued with the PCUE/PCWD command and the

subsequent spot is cued with the PCU2 command, two successive PLAY commands will play the

two IDs. See the section entitled**Playing a Short Spot at the Beginning of a POD** for more

details.

If another PCU2 is received prior to receiving the play command, then the second cue will be

aborted and the new item will be cued in its place as the second cue.

The video port selected must be in the CUE state or the PCU2 command will fail and the

appropriate bit will be set in the port status error byte.

**8.3.6 PLAY CUE WITH DATA**

**VARIABLE NAME BINARY:**

**<AX.25><LEN><ID C:32+>< POSITION TC>,<DURATION TC>[<OPT\_LEN><OPT\_FUNC>]**

**FIXED NAME BINARY:**

**<2X.25><LEN><ID C:8>< POSITION TC>,<DURATION TC>[<OPT\_LEN><OPT\_FUNC>]**

**ASCII: PCWD "<ID C:32+>", <POSITION TC>,<DURATION TC>**

**ASCII: PCD2 "<ID C:32+>", <POSTION TC>,<DURATION TC>**

LENcontains the length in bytes of the ID name.

ID contain the ID name to be cued for play.

POSITION contains the position (in TC format) within the ID to which the decoder will be cued.

DURATION contains the duration of time (in TC format) the ID should be played.

OPT\_DATA contains an optional length if optional data is provided.

OPT\_FUNC contains optional data.

The PLAY CUE WITH DATA command causes the selected port to prepare to play the specified

ID starting at the specified position and for the specified duration. The command is identical to

the PLAY CUE command with the added functionality of specifying a POSITION and DURATION.

The POSITION specifies the offset into the recorded ID at which playing should begin in TC

format.

The DURATION is the duration of time that the ID should be played in TC format.

Please see the section titled “Timecode (TC format)” for details on the TC format.

Command States

page 43

A POSITION value of 0:0:0.0 will specify that there is no starting offset and a DURATION value of

0:0:0.0 will specify that the recorded duration of the ID be used. If the POSITION is not zero, then

a frame table must be present. If the POSITION is not zero and a frame table is not present, then

the CUE will fail and the OPERATION\_FAILED bit will be set in the port status error bytes. When

a CUE command is issued (PCUE, PCWD or PCWF), no more CUE commands should be sent to

the port until either the CUE\_DONE bit is set or an error bit is set. If the POSITION is a timecode

before the SOC (start-of-clip), the SOC will be used. If the POSITION is a timecode after the end

of the clip, the last frame of the clip will be used. If NTSC drop frame is enabled and the

POSITION is not a valid dropframe timecode, the POSITION will be adjusted forward to the

nearest valid dropframe timecode.

Once a file is paused on the last frame of the specified duration, a PLAY command cannot be

issued without first re-cueing the decoder. A JOG or SHTL command resets the DURATION to 0

and overrides this rule. Also, once a JOG or SHTL command has been issued, playout will not

stop at SOC+POSITION+DURATION, but at the end of the file.

The following restrictions apply to cueing and playing files using the PCWD and PCD2

commands.

•

•

•

Do not attempt to use a mark in (PCWD and PCWF) which is less than 2 minutes from

the current end of file. Thus, the minimum time delay possible using these commands is

two minutes.

When using a circular file (TCUE) for recording, do not mark in (PCWD and PCWF) to a

point that has been removed from the buffer. For example, after 4 hours of recording a

circular file with a 1 hour MAX\_TIME, the legal values for mark in are between 3:0:0.0

and 3:58:0.0.

Playing from a circular file (TCUE) must also conform to the above restriction. The PLAY

command must be issued before the point which is currently cued is overwritten in the

circular file.

Violation of these rules will result in unpredictable behavior.

In the binary protocol, if optional data is sent, then OPT\_LEN will indicate the number of bytes of

optional data to follow. Currently only one optional byte is supported in OPT\_FUNC. If the

number of bytes is 1 and the optional Data is 2, the PCD2 operation will be performed. If the

number of bytes and optional data are any other value, a normal play cue with data command will

be executed.

PCD2 requires that the video port selected be in CUE state and results in the specified ID being

cued with data as well. This results in two cued items and is useful for playing very short spots

that are too short to allow time for a second cue operation, such as some station Ids. If another

PCD2 is received prior to receiving the play command, then the second cue will be aborted and

the new item will be cued in its place as the second cue.

**8.3.7 PLAY CUE WITH FRAMES**

**VARIABLE NAME BINARY: <AX.66><LEN><ID C:32+><POSITION I:4><DURATION I:4>**

**FIXED NAME BINARY: <2X.66><ID C:8><POSITION I:4><DURATION I:4>**

**ASCII: PCWF "<ID C:32+>", <POSITION>,<DURATION>**

LENcontains the length in bytes of the ID name.

ID contain the ID name to be cued for play.

POSITION contains the number of frames within the ID to which the decoder will be cued.

page 44

Command States

DURATION contains the number of frames which should be played.

The PLAY CUE WITH FRAMES command causes the selected port to prepare to play the

specified ID starting at the specified position and for the specified duration. The command is

almost identical (exceptions noted below) to the PLAY CUE command with the added

functionality of specifying a POSITION and DURATION. The POSITION specifies the starting

frame of the recorded ID at which playing should begin. The DURATION is the number of frames

that the ID should be played. An error will be logged if START or DURATION is outside the range

of the ID. A START frame of zero will specify that there is no starting offset and a DURATION of

zero frames will specify that the recorded duration of the ID be used. If the Start POSITION is not

zero, then a frame table must be present. If the start position is not zero and a frame table is not

present, then the CUE will fail and the OPERATION\_FAILED bit will be set in the port status error

bytes. When a CUE command is issued (PCUE, PCWD or PCWF), no more CUE commands

should be sent to the port until either the CUE\_DONE bit is set or an error bit is set.

Unlike other CUE commands, the PLAY CUE WITH FRAMES command does not allow another

CUE command before the PLAY command. If it is desired to cue again, a STOP command must

be sent to the port first.

PCWF is designed for time delay applications. The following should be kept in mind when using

PCWF.

•

•

•

Do not attempt to use a mark in (PCWD and PCWF) which is less than 2 minutes from

the current end of file. Thus, the minimum time delay possible using these commands is

two minutes.

When using a circular file (TCUE) for recording, do not mark in (PCWD and PCWF) to a

point that has been overwritten in the file. For example, after 4 hours of recording a

circular file with a 1 hour MAX\_TIME, the legal values for mark in are between 3:0:0.0

and 3:58:0.0.

Playing from a circular file (TCUE) must also conform to the above restriction. The PLAY

command must be issued before the point which is currently cued is overwritten in the

circular file.

Violation of these rules will result in unpredictable behavior.

Command States

page 45

**8.4**

**Material Management Commands**

**8.4.1 MARK**

**VARIABLE NAME BINARY: <AX.27><LEN><ID C:32+>**

**FIXED NAME BINARY: <2X.27><ID C:8>**

**ASCII: MARK "<ID C:32+>"**

**IMPLEMENTED: 300 / 700 / 1600**

LEN contains the length in bytes of the ID name.

ID contains the ID name to be marked for transfer.

The MARK command is used on systems in operational mode BUFFER (see OMOD), to transfer

content from another BVS to the local disk system, usually when a system of more than one BVS

is setup in a main multiple buffer configuration. For systems operating in PEER mode, this

opcode is a NOP. In PEER mode the COPY command is used to transfer content between

systems. The MARK command can be issued with the system in any state and selecting a video

port is not necessary. Issuing the MARK command causes the system to mark the specified ID

for transfer to the local disk system. Sequential MARK commands are added to a list.

In BUFFER mode, when the MARK command is received, the system checks if the ID is present

in the remote system. If the item is found on the corresponding MAIN system, the ID is added to

the mark list.

If the ID specified in the MARK command already exists the operation will fail and the

ID\_ALREADY\_EXISTS error bit will be set in the port error status. If an error occurs while

attempting to transfer/restore the ID, the XFER\_FAILED error bit is set, the ID is added to the ids

Deleted list and the IDS\_DELETED bit is set in the port status. Due to the fact that the ids

Deleted list is used as an error reporting feature, the presence of an ID on the list can occur

multiple times and for various reasons. Using the port error status and IDRQ/IDQY query to verify

the status of the ID is recommended.

If the local disk system does not have enough space to hold the video file being retrieved, the

MARK will fail with the INSUFFICIENT\_DISKSPACE error bit set in the port status. If the

specified ID does not exist in the peer system, the MARK command sets the ID NOT FOUND and

XFER\_FAILED error bits in port error status.

If Play While Transfer has not been enabled a video file cannot be played until the entire file has

finished transferring. When playing while transferring, care must be taken that the transfer

bandwidth is al least equal to the playout bandwidth. Speak with an Avid representative for more

details.

If the Play While Transfer (PXFR) setting is enabled, the new ID will be available for playout as

soon as sufficient data has recorded. The Ids Added bit will be set in port status at that time and

an IDRQ? or IDQY? will show the ID status as ID\_IN\_DISK with ID\_IN\_XFER bit set. If Play

While transfer is not enabled, the ID will be added to the system database when recording is

finished and can not be accessed until that time. When the transfer of the ID to the video disk has

completed, the ID is added to the "ID's Added List" for every communications port in the local disk

system. The "ID's Added" bit in the PORT STATUS for every communications port is also set.

Deleting (IDEL) an item that is on the MARK list for transfer from another server results in it being

removed from the list and from the local disk system. MARKing the same item twice has no

effect. There is a limit of 500 to the number of mark commands that can be received in very rapid

succession. Therefore it is suggested that for optimal performance either the mark commands

page 46

Command States

not be spaced closely or rapid bursts of mark commands be limited to a more moderate number.

The total number of ids that can be marked is limited to the number of Ids a system can store. An

error condition will result in the appropriate bit being set in the port status error bytes.

**8.4.2 COPY**

**VARIABLE NAME BINARY:**

**<AX.53><SRC\_N><LEN><SRC\_ID C:32+><MASK><DEST\_N><LEN2><DEST\_ID C:32+>**

**SHORT NAME BINARY:**

**<2X.53><SRC\_N><SRC\_ID C:8><MASK><DEST\_N><DEST\_ID C:8>**

**ASCII: COPY <SRC\_N>,"<SRC\_ID C:32+>",<MASK>, <DEST\_N>,"<DEST\_ID C:32+>”**

**IMPLEMENTED: 300 / 700 / 1600**

SRC\_N contains the node ID of the source disk system.

LEN contains the length in bytes of the SRC\_ID.

SRC\_ID contains the ID name of the source item.

MASK is a bit mask specifying the characteristics of the copy.

DEST\_N contains the node ID of the destination server.

LEN2 contains the length in bytes of the DEST\_ID.

DEST\_ID contains the ID name of the destination item.

The COPY command is used to transfer content between servers in a Peer to Peer cluster. The

BVS receiving the command must be in OMOD PEER. The command must be sent to the

destination node. The node numbers are obtained from the NLST? Query on the destination

server.

To copy an ID to the destination without renaming it, bit zero of MASK is set to 0. In this case,

the DEST\_ID is not sent.

To rename an ID on the destination node, bit zero of MASK is set to 1. In this case, the DEST\_ID

must be specified.

**8.4.3 LOUTH COPY**

**VARIABLE NAME BINARY: <AX.50><LEN><ID NAME C:32+><SRC NODE><DEST NODE>**

**FIXED NAME BINARY: <2X.50><ID NAME C:8><SRC NODE><DEST NODE>**

**ASCII: LCPY "<ID NAME C:32+>",<SRC NODE>,<DEST NODE>**

**IMPLEMENTED: 300 / 700 / 1600**

LEN contains the length in bytes of the ID name.

ID\_NAME contains the ID name of the item to be copied.

SRC\_NODE contains the node number of the source server.

DEST\_NODE contains the node number of the destination server.

NOTE: In order for this command to function correctly with Louth automation control, the bvsrc

and the /etc/hosts file must be edited. The "aardvark" CPU item must be uncommented in the

bvsrc file and added to the /etc/hosts file. This item will then become server number 0, which

Louth automation control does not recognize.

Command States

page 47

The LCPY command is used to transfer content between peer systems in a Peer to Peer cluster

without renaming the content. In all other respects it is the same as the COPY command.

**8.4.4 COPY TERMINATE**

**VARIABLE NAME BINARY: <AX.52><DEST\_NODE><LEN><ID C:32+>**

**FIXED NAME BINARY: <2X.52><DEST\_NODE><ID C:8>**

**ASCII: CTER <DEST\_NODE>, "<ID C:32+>"**

**IMPLEMENTED: 300 / 700 / 1600**

DEST\_NODE contains the node number of the destination server.

LEN contains the length in bytes of the ID name.

ID contains the ID name of the destination item to be terminated.

The CTER command terminates an active copy or, if the copy has not yet started, removes a

copy request from the copy que of the destination server. If the specified ID is neither an active

copy nor found in the copy que, the ID NOT FOUND bit will be set in the Port Status. If the copy

for the specified ID has completed, then XFER COMPLETE bit will be set in the Port Status.

The CTER command is only valid for BVS systems in PEER mode. The CTER command must

be issued to the destination server.

**8.4.5 NEW COPY**

**VARIABLE NAME BINARY:**

**AX.1F><LEN><ID1 C:32+><LEN2><ID2 C:32+><POS TC><DUR TC>**

**FIXED NAME BINARY: <2X.1F><ID1 C:8><ID2 C:8><POS TC><DUR TC>**

**ASCII: NCPY "<ID1 C:32+>","<ID2 C:32+>",<POS TC>,<DUR TC>**

**IMPLEMENTED: 300 / 700 / 1600**

LEN contains the length of string ID1.

ID1 contains the name of the source ID.

LEN2 contains the length of string ID2.

ID2 contains the name of the destination ID.

POS contains the timecode within ID1 which will be the first frame of ID2.

DUR contains the DURATION of ID2.

Creates a new file named ID2 which contains the video from ID1 starting at POS and ending at

POS+DUR. The busy bit is set during the copy operation. The new ID2 will consume disk space.

If the POS is before the SOC, the SOC will be used. If NTSC drop frame is enabled and the POS

is not a valid dropframe timecode, the POS will be adjusted forward to the nearest valid

dropframe timecode.

The busy bit is set while the operation is in progress. The output file (ID2) will not be accessible

until the NCPY operation has completed. The source file cannot be open by any other activity

during the NCPY operation.

**8.4.6 ID RENAME**

page 48

Command States

**VARIABLE NAME BINARY: <AX.1D><LEN><ID1 C:32+><LEN2><ID2 C:32+>**

**FIXED NAME BINARY: <2X.1D><ID1 C:8><ID2 C:8>**

**ASCII: IDRN "<ID1 C:32+>","<ID2 C:32+>"**

LEN contains the length of the original ID name.

ID1 contains the original ID name.

LEN2 contains the length of the new ID name.

ID2 contains the new ID name.

IDRN renames ID1 to ID2. The busy bit will be set during the rename operation. The file should

not be accessed while the rename operation is in progress.

**8.4.7 SET START OF CLIP**

**VARIABLE NAME BINARY: <AX.7F><LEN><ID C:32+><SOC TC>**

**FIXED NAME BINARY: <2X.7F><ID C:8><SOC TC>**

**ASCII: SSOC "<ID C:32+>", <SOC TC>**

LEN contains the length of the ID name.

ID contains the ID name.

SOC contains the timecode to be set as the start of ID.

The SET START OF CLIP command sets the start time code for the specified ID. The ID must

have been successfully cued for record prior to issuing the SSOC command. This value will be

used to offset the return of the position request command and in the computation of the

POSITION in the PCWD command.

The SOC value can be queried with the INFO? query (binary 3X.70).

A SOC value cannot be set for a subclip.

The SOC value cannot be changed for the base-clip after a subclip has been created using the

base-clip.

An error condition will result in the appropriate bit being set in the port status error bytes. If

NTSC drop frame is enabled and the SOC is not a valid dropframe timecode, the SOC will not be

set and the OPERATION\_FAILED status flag will be raised.

**8.4.8 ID DELETE**

**VARIABLE NAME BINARY: <AX.26><LEN><ID C:32+>**

**FIXED NAME BINARY: <2X.26><ID C:8>**

**ASCII: IDEL "<ID C:32+>"**

LEN contains the length of ID.

ID contains the name of the ID to be deleted.

The ID DELETE command is used to remove material from the server. Upon receipt of a ID

DELETE command, the system will check if the ID is present. If it is not, then an error will be

logged. If it is present, the system will check if the ID is currently being cued or played. If it is,

an error will be logged. Otherwise, the ID will be deleted. The busy bit will be set until the delete

has completed.

Command States

page 49

If the server is in OMOD MAIN, the ID will also be deleted from any servers configured as buffers

(OMOD 2) to this server of they have ripple through delete enabled (RDEL utility).

**8.4.9 DELETE PROTECT**

**VARIABLE NAME BINARY: <8X.15><LEN><ID C:32+>**

**FIXED NAME BINARY: <0X.15><ID C:8>**

**ASCII: IDPR "<ID C:32+>"**

LEN contains the length of ID.

ID contains the name of the ID to be delete protected.

The ID DELETE PROTECT command is used to prevent an ID from being removed from the

server. Upon receipt of an ID DELETE PROTECT command, the system will check if the ID is

present in the server. If it is not, then an error will be logged. If it is present, the system will

check if the ID is currently being cued or played. If it is, an error will be logged. Otherwise, the ID

will be delete protected and cannot be deleted until an ID DELETE UNPROTECT command has

been received.

**8.4.10 DELETE UNPROTECT**

**VARIABLE NAME BINARY: <8X.16><LEN><ID C:32+>**

**FIXED NAME BINARY: <0X.16><ID C:8>**

**ASCII: IDUP "<ID C:32+>"**

LEN contains the length of ID.

ID contains the ID name to be delete unprotected.

The ID DELETE UNPROTECT command is used to allow an ID that has been delete protected to

be removed from the server. Upon receipt of an ID DELETE UNPROTECT command, the

system will check if the ID is present in the server. If it is not, then an error will be logged. If it is

present, the system will check if the ID is currently being cued or played. If it is, an error will be

logged. Otherwise, the ID will be delete unprotected and can be deleted from the server.

**8.4.11 DISK CLEAR**

**BINARY: <2X.29>**

**ASCII: DCLR**

The purpose of the DISK CLEAR command is to remove all files that are currently on the server

and reset the mark list. This is a very dangerous command, meant for use mainly on systems in

OMOD BUFFER in a main multiple buffer system configuration. Should there be a loss of

apparent disk space due to unequal numbers of MARK and DELETE commands, this command

places the buffer system in a known state (empty) and enables a new mark list to be started from

scratch.

The DISK CLEAR command can only be issued with the selected port in the IDLE state.

An error condition will result in the appropriate bit being set in the port status error bytes.

page 50

Command States

**8.5**

**LIST QUERIES**

On PFS systems, the VFS information is centrally located on the FSC’s, but is distributed to each

node for performance reasons. This means that there may be a delay in the time it takes the

information to be updated on different nodes in the same cluster. For this reason, the results of

list and ID characteristic queries and the list status option of the port status query cannot be

shared across ports. For example, when an ID appears on the IDS\_ADDED list of one port, it

may not be IN\_DISK as far as other ports are concerned.

**3.6.1 LIST QUERY**

**VARIABLE NAME:**

**BINARY: <BX.11><TYPE>**

**returns <TYPE><IDS\_REMAINING I:2><<LEN><ID C:32+> :X>**

**ASCII: VLST? <TYPE>**

**returns <TYPE><IDS\_REMAINING I:2:,><,”<ID C:32+>” :X><,”” :20-X>**

TYPE contains the sort order for the list. TYPE 0 indicates alphanumeric order. Type 1 indicates

FIFO.

IDS\_REMAINING returns the number of ids remaining.

LEN returns the length of the name of the subsequent ID.

ID returns the name of the ID.

LEN / ID is repeated X-1 times for a total of X LEN / ID pairs returned by command.

The number of items returned is limited to the first X for which the sum of their lengths is less than

or equal to 80. The return data length, in this case, is 82+X. In the ASCII protocol, 20 strings

are always returned and 20-X strings will be null.

The VLST? query returns the first X entries of the list of all IDs currently stored on the server.

The number of IDs remaining in the list is returned in IDS\_REMAINING followed by the X length /

name pairs. The VNXT? query is used to transfer any remaining IDS and is called repeatedly

until IDS\_REMAINING reaches 0. All file names (variable or fixed) are returned in variable

format.

**ASCII: LLST? <TYPE>**

**returns <TYPE><IDS\_REMAINING I:2:,><,”<ID C:32\*>” :10>**

TYPE contains the sort order for the list. TYPE 0 indicates alphanumeric order. Type 1 indicates

FIFO.

**IDS\_REMAINING** returns the number of ids remaining.

ID returns the name of the ID. Nulls will be returned if there are less than 10 ID names to return.

The LLST? query returns the first 10 entries of the list of all IDs currently stored on the server.

The number of IDs remaining in the list is returned in IDS\_REMAINING followed by the 10

names. The LNXT? query is used to transfer any remaining IDS and is called repeatedly until

IDS\_REMAINING reaches 0. All file names (variable or fixed) are returned in variable format. If

there are not 10 ID names to return, the response will be padded with null strings.

**FIXED NAME:**

**BINARY: <3X.11><TYPE>**

Command States

page 51

**returns <TYPE><IDS\_REMAINING I:2><<ID C:8> :10>**

**ASCII: LIST? <TYPE>**

**returns <TYPE><IDS\_REMAINING I:2:,><,”<ID C:8>” :10>**

TYPE contains the sort order for the list. TYPE 0 indicates alphanumeric order. Type 1 indicates

FIFO.

IDS\_REMAINING returns the number of ids remaining.

ID returns the name of the ID.

Nulls will be returned if there are less than 10 ID names to return.

The return data length is 82.

The LIST? query returns the first 10 entries of the list of all fixed-length named IDs currently

stored on the server. The number of IDs remaining in the list is returned in IDS\_REMAINING

followed by the 10 names. The NEXT? query is used to transfer any remaining IDS and is called

repeatedly until IDS\_REMAINING reaches 0. The names are returned in fixed format.

**3.3.2 NEXT QUERY**

**VARIABLE NAME:**

**BINARY: <BX.02>**

**returns <IDS\_REMAINING I:2><<LEN><ID C:32+> :X>**

**ASCII: VNXT?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:32+>” :X><,”” :20-X>**

IDS\_REMAINING returns the number of ids remaining.

LEN returns the length of the name of the subsequent ID.

ID returns the name of the ID.

LEN / ID is repeated X-1 times for a total of X LEN / ID pairs returned by command.

The number of items returned is limited to the first X for which the sum of their lengths is less than

or equal to 80. The return data length, in this case, is 82+X. In the ASCII protocol, 20 strings

are always returned and 20-X strings will be null.

The VNXT? query is used to transfer the next X items following a VLST?, VADD?, VATM? or

VDLT? query which returned a non-zero IDS\_REMAINING.

**ASCII: LNXT?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:32\*>” :10>**

**IDS\_REMAINING** contains the number of ids remaining.

ID returns the name of the ID. Null strings will be returned if there are less than 10 ID names to

return.

The LNXT? Query transfers the next 10 items following an LLST? LADD?, LATM? Or LDLT query

which returned a non-zero IDS\_REMAINING.

**FIXED NAME:**

**BINARY: <3X.02>**

**returns <IDS\_REMAINING I:2><<LEN><ID C:8> :10>**

**ASCII: NEXT?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:8>” :10>**

page 52

Command States

IDS\_REMAINING returns the number of ids remaining.

ID returns the name of the ID.

Nulls will be returned if there are less than 10 ID names to return..

The return data length is 82.

The NEXT? query is used to transfer the next 10 items following a LIST?, IADD?, IATM? or

IDLT? query which returned a non-zero IDS\_REM (MSB,LSB).

**8.5.1 IDS ADDED TO MAIN QUERY**

**VARIABLE NAME:**

**BINARY: <BX.15><TYPE>**

**returns <IDS\_REMAINING I:2><<LEN><ID C:32+> :X>**

**ASCII: VATM?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:32+>” :X><,”” :20-X>**

IDS\_REMAINING returns the number of ids remaining.

LEN returns the length of the name of the subsequent ID.

ID returns the name of the ID.

LEN / ID is repeated X-1 times for a total of X LEN / ID pairs returned by command.

The number of items returned is limited to the first X for which the sum of their lengths is less than

or equal to 80. The return data length, in this case, is 82+X. In the ASCII protocol, 20 strings

are always returned and 20-X strings will be null.

The VATM? query returns the first X entries of the list of all IDs added to main. The number of

IDs remaining in the list is returned in IDS\_REMAINING followed by the X length / name pairs.

The VNXT? query is used to transfer any remaining IDS and is called repeatedly until

IDS\_REMAINING reaches 0. All file names (variable or fixed) are returned in variable format.

On PFS-based systems, this command always returns zero IDS\_REMAINING.

**ASCII: LATM?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:32\*>” :10>**

IDS\_REMAINING contains the number of ids remaining.

ID returns the name of the ID. Null strings will be returned if there are less than 10 ID names to

return..

The LATM? query returns the first 10 entries of the list of all IDs added to main. The number of

IDs remaining in the list is returned in IDS\_REMAINING followed by the 10 length / name pairs.

The LNXT? query is used to transfer any remaining IDS and is called repeatedly until

IDS\_REMAINING reaches 0. All file names (variable or fixed) are returned in variable format.

On PFS-based systems, this command always returns zero IDS\_REMAINING.

**FIXED NAME:**

**BINARY: <3X.15>**

**returns <IDS\_REMAINING I:2><<ID C:8> :10>**

**ASCII: IATM?**

Command States

page 53

**returns <IDS\_REMAINING I:2:,><,”<ID C:8>” :10>**

IDS\_REMAINING returns the number of ids remaining.

ID returns the name of the ID.

Nulls will be returned if there are less than 10 ID names to return.

The return data length is 82.

The IATM? query returns the first 10 entries of the list of all fixed-length named IDs added to

main. The number of IDs remaining in the list is returned in IDS\_REMAINING followed by the list

of 10 names. The NEXT? query is used to transfer any remaining IDS and is called repeatedly

until IDS\_REMAINING reaches 0. The names are returned in fixed format.

For systems in OMOD BUFFER, these three queries return the begining of the list of IDs that

have been added to the main since the last instance of the query or the most recent power up.

The appropriate NEXT query is used to transfer any remaining IDs. NEXT should be called

repeatedly until IDS\_REMAINING reaches 0. After all data has been transmitted, the

ID\_ADDED\_TO\_MAIN bit in the vport status of the selected video port will be reset. When a new

ID is recorded or transferred onto a main server, the ID\_ADDED\_TO\_MAIN bit will be set.

On PFS-based systems, this command always returns zero IDS\_REMAINING.

**8.5.2 IDS ADDED QUERY**

**VARIABLE NAME:**

**BINARY: <BX.18>**

**returns <IDS\_REMAINING I:2><<LEN><ID C:32+> :X>**

**ASCII: VADD?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:32+>” :X><,”” :20-X>**

IDS\_REMAINING returns the number of ids remaining.

LEN returns the length of the name of the subsequent ID.

ID returns the name of the ID.

LEN / ID is repeated X-1 times for a total of X LEN / ID pairs returned by command.

The number of items returned is limited to the first X for which the sum of their lengths is less than

or equal to 80. The return data length, in this case, is 82+X. In the ASCII protocol, 20 strings

are always returned and 20-X strings will be null.

The VADD? query returns the first X entries of the list of all IDs added to the server. The number

of IDs remaining in the list is returned in IDS\_REMAINING followed by the X length / name pairs.

The VNXT? query is used to transfer any remaining IDS and is called repeatedly until

IDS\_REMAINING reaches 0. All file names (variable or fixed) are returned in variable format.

**ASCII: LADD?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:32\*>” :10>**

IDS\_REMAINING returns the number of ids remaining.

ID returns the name of the ID. Nulls will be returned if there are less than 10 ID names to return.

The LADD? query returns the first 10 entries of the list of all IDs added to the server. The number

of IDs remaining in the list is returned in IDS\_REMAINING followed by the 10 length / name pairs.

The LNXT? query is used to transfer any remaining IDS and is called repeatedly until

IDS\_REMAINING reaches 0. All file names (variable or fixed) are returned in variable format.

page 54

Command States

This query is only available on the CLI /LAN and is similar to VADD? As described above with

the exception that 10 ids are always returned, regardless of the ID length.

**FIXED NAME:**

**BINARY: <3X.18>**

**returns <IDS\_REMAINING I:2><<ID C:8> :10>**

**ASCII: IADD?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:8>” :10>**

IDS\_REMAINING returns the number of ids remaining.

ID returns the name of the ID.

Nulls will be returned if there are less than 10 ID names to return..

The return data length is 82.

The IADD? query returns the first 10 entries of the list of all fixed-length named IDs added to the

server. The number of IDs remaining in the list is returned in IDS\_REMAINING followed by the

list of 10 names. The NEXT? query is used to transfer any remaining IDS and is called

repeatedly until IDS\_REMAINING reaches 0. The names are returned in fixed format.

These three queries return a list of the IDs that have been added to the server since the last list

query or power up. The format is identical to the list command. The number of IDs remaining to

be transmitted in subsequent queries will be returned in IDS\_REMAINING, followed by a list of

IDs. The appropriate NEXT? (VNXT? or LNXT?) query is used to transfer any remaining IDs.

The NEXT? query is called repeatedly until all IDs have been transferred. After all data has been

transmitted, the ID\_ADDED bit in the port status of the selected video port will be reset. When a

new ID is recorded or transferred, the ID\_ADDED bit will be set.

If an item is deleted immediately after being recorded/copied into the system, it is possible to

have the delete occur between the time the IDS ADDED bit is set and when the user queries the

ISADDED LIST. This may result in an empty IDS ADDED list.

**8.5.3 IDS DELETED QUERY**

**VARIABLE NAME:**

**BINARY: <BX.19>**

**returns <IDS\_REMAINING I:2><<LEN><ID C:32+> :X>**

**ASCII: VDLT?**

**returns <<IDS\_REMAINING> I:2:,><,”<ID C:32+>” :X><,”” :20-X>**

IDS\_REMAINING returns the number of ids remaining.

LEN returns the length of the name of the subsequent ID.

ID returns the name of the ID.

LEN / ID is repeated X-1 times for a total of X LEN / ID pairs returned by command.

The number of items returned is limited to the first X for which the sum of their lengths is less than

or equal to 80. The return data length, in this case, is 82+X. In the ASCII protocol, 20 strings

are always returned and 20-X strings will be null.

The VDLT? query returns the first X entries of the list of all IDs deleted from the server. The

number of IDs remaining in the list is returned in IDS\_REMAINING followed by the X length /

name pairs. The VNXT? query is used to transfer any remaining IDS and is called repeatedly

until IDS\_REMAINING reaches 0. All file names (variable or fixed) are returned in variable

format.

Command States

page 55

**ASCII: LDLT?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:32\*>” :10>**

IDS\_REMAINING returns the number of ids remaining.

ID returns the name of the ID. Nulls will be returned if there are less than 10 ID names to return.

The LDLT? query returns the first 10 entries of the list of all IDs deleted from the server. The

number of IDs remaining in the list is returned in IDS\_REMAINING followed by the 10 length /

name pairs. The LNXT? query is used to transfer any remaining IDS and is called repeatedly

until IDS\_REMAINING reaches 0. All file names (variable or fixed) are returned in variable

format.

**FIXED NAME:**

**BINARY: <3X.19>**

**returns <IDS\_REMAINING I:2><<ID C:8> :1+>**

**ASCII: LIST?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:8>” :10>**

IDS\_REMAINING returns the number of ids remaining.

ID returns the name of the ID.

Nulls will be returned if there are less than 10 ID names to return..

The return data length is 82.

The IDLT? query returns the first 10 entries of the list of all fixed-length named IDs deleted from

the server. The number of IDs remaining in the list is returned in IDS\_REMAINING followed by

the list of 10 names. The NEXT? query is used to transfer any remaining IDS and is called

repeatedly until IDS\_REMAINING reaches 0. The names are returned in fixed format.

These 3 queries return a list of the IDs that have been deleted from the server since the last list

query, IDLT query or power up. The format is identical to the list command. The number of IDs

remaining to be transmitted in subsequent queries will be returned in IDS\_REMAINING followed

by a list of IDs. The appropriate NEXT query is used to transfer any remaining IDs. NEXT is

called repeatedly until all IDs have been transferred. After all data has been transmitted, the

ID\_DELETED bit in the port status of the selected video port will be reset. When an existing ID is

deleted, the ID\_DELETED bit will be set. IDs are also placed on this list when an attempt to

transfer them from another server has failed. Due to the mechanics of this it is possible with

repeated copy failures to end up with the same id listed in the IDS DELETED list more than once.

page 56

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| OPERA-  TION  PENDING | ID IN XFER |  |  | ID DELETE  PROTECTED | ID IN  REMOTE  SYSTEM | ID MARKED  FOR XFER | ID IN DISK |

**8.6**

**ID CHARACTERISTIC QUERIES**

On PFS systems, the VFS information is centrally located on the FSC’s, but is distributed to each

node for performance reasons. This means that there may be a delay in the time it takes the

information to be updated on different nodes in the same cluster. For this reason, the results of

list and ID characteristic queries and the list status option of the port status query cannot be

shared across ports. For example, when an ID appears on the IDS\_ADDED list of one port, it

may not be IN\_DISK as far as other ports are concerned.

**8.6.1 SIZE QUERY**

**VARIABLE NAME BINARY: <BX.14><LEN><ID C:32+>**

**returns <DURATION I:4>**

**FIXED NAME BINARY: <3X.14><ID C:8>**

**returns <DURATION I:4>**

**ASCII: SIZE? "<ID C:32+>"**

**returns <DURATION>**

LEN contains the length of the ID name.

ID contains the ID name.

DURATION returns the duration of the ID.

The SIZE? query returns the duration of the specified ID.

**3.3.13 ID REQUEST QUERY**

**VARIABLE NAME BINARY: <BX.16><ID C:N>**

**returns <ID\_STATUS>**

**FIXED NAME BINARY: <3X.16><ID C:8>**

**Returns <ID\_STATUS>**

**ASCII: IDRQ? "<ID C:32+>"**

**returns <ID\_STATUS>**

LEN contains the length of the ID name.

ID contains the ID name.

ID\_STATUS returns the data described in the table below.

The IDRQ? query returns a status byte indicating current state the specified ID.

Command States

**Table 23.  IDRQ? Query Return Byte**

page 57

ID IN DISK indicates that the ID was found on the local video disk. If the ID is being recorded or

PXFR has been enabled and the ID is being transferred, then ID IN DISK will be set as soon as

there is sufficient data to enable the file to be played. ID\_IN\_XFER will be set until the ID is

completely on the video disk.

ID MARKED FOP XFR indicates that the ID is on the list to be transferred to the local video disk.

If the server is in BUFFER OMOD, ID IN REMOTE SYSTEM indicates the ID is on a peer server.

ID DELETE PROTECTED indicates that the ID on the local video disk is delete protected.

ID\_IN\_XFER is only set when the ID\_IN\_DISK bit is also set. It indicates that while the ID may

be able to be played, it is still in the process of being recorded or transferred and is not yet

complete on the video disk.

OPERATION PENDING indicates that the requested information is being retrieved but is not yet

available.

When the OPERATION PENDING bit is clear (0), all of the above bits are valid. When the

OPERATION PENDING bit is set (1), none of the above bits are valid. In this case, the IDRQ?

query must be re-issued to obtain valid information in the other bits. It may take several attempts

before the query is completed and the OPERATION PENDING bit is cleared (0). Due to the time

that this operation may take on remote systems, it is recommended that one of the following

algorithms be employed when querying long lists if IDs.

1.

2.

Upon receiving an OPERATION PENDING return from IDRQ?, continue to poll IDRQ? on

the same ID every 5 frames until a non OPERATION PENDING response is received. At

this point an IDRQ? for a new ID may be issued.

Limit the total number of outstanding IDRQ? queries from all ports on the server to less

than 400. Introduce new IDs to the list according to one of the algorithms below.

a.

b.

Poll IDRQ? until a non OPERATION PENDING response is received on all

requests and then issue 400 more requests.

Poll IDRQ? for the first ID that was queried until a non OPERATION PENDING

response is received and then issue an IDRQ? query for a new ID.

**8.6.2 ID REQUEST DISK QUERY**

**VARIABLE NAME BINARY: <BX.6F><LEN<ID C:32+>**

**returns <ID\_STATUS>**

**FIXED NAME BINARY: <3X.6F><ID C:8>**

**returns <ID\_STATUS>**

**ASCII: IDQY? "<ID C:32+>"**

**returns <ID\_STATUS>**

LEN contains the length of the ID name.

ID contains the ID name.

ID\_STATUS returns the data described in the table below.

page 58

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | ID INFO 4 | ID INFO 3 | ID INFO 2 | ID INFO 1 |

|  |
| --- |
| Media Standard |
| Media Format |
| Media Subformat |
| Reserved |

The IDQY? query is identical to the IDRQ? query. It is included for compatibility with releases

which supported the Media Vault. Please see the IDRQ? query for details of the operation of this

query.

**8.6.3 ID INFO RETRIEVAL QUERY**

**VARIABLE NAME BINARY: <BX.70><BITMAP><LEN><ID C:32+>**

**returns**

**<rLEN><rBITMAP>[<rINFO1 I:4>][<rINFO2 I:4>][<rINFO3 TC:2>][<LEN><rINFO4 I:8\*>]**

**FIXED NAME BINARY: <3X.70><BITMAP><ID C:8>**

**returns**

**<rLEN><rBITMAP>[<rINFO1 I:4>][<rINFO2 I:4>][<rINFO3 TC:2>][<LEN><rINFO4 I:8\*>]**

**ASCII: INFO? <BITMAP>, "<ID>"**

**returns <rLEN>,<rBITMAP>[,<rINFO1 I:4:,>][,<rINFO2 I:4:,>]**

**[,<rINFO3 TC:2:,>][,<LEN>,<rINFO4 I:8\*:,>]<,+0 I:25-<rLEN>>**

BITMAP contains a bitmap specifying which information should be returned.

LEN contains the length of the ID name.

ID contains the name of the ID.

rLEN returns the byte count of M+1.

rBITMAP returns the mask of what information was returned.

rINFOx returns the values specified for ID INFO x below.

The INFO? query returns information about the specified ID. BITMAP determines which

information will be returned. If the bit in the ID INFO Bitmap diagram is set, the associated data

is included in the reply.

rINFO 1 contains the Video Media Standard. Valid values can be found in the table titled “VSTD

**Table 24.  INFO? Query Bitmap**

**Table 25.  ID Info 1:  Command Video Attributes**

Parameters”.

rINFO 2 contains the Media Format. Valid values are 2 for MPEG 4:2:0 and 3 for MPEG 4:2:2.

rINFO 3 contains the Media SubFormat. Valid values are 0 for GOP IBP, 1 for GOP IB and 2 for

GOP I only.

rINFO 4 is reserved for future use.

**Table 26. ID Info 2: Encode Rates**

Command States

page 59

|  |
| --- |
| BitRate MSB |
| BitRate (middle byte) |
| BitRate LSB |
| Reserved |

|  |
| --- |
| SOC Frames |
| SOC Seconds |
| SOC Minutes |
| SOC Hours |

|  |
| --- |
| DURATION Frames |
| DURATION Seconds |
| DURATION Minutes |
| DURATION Hours |

|  |
| --- |
| number of remaining bytes |
| Audio Pair 1 Format |
| Audio Pair 1 Value |
| Audio Pair 2 Format |
| Audio Pair 2 Value |
| Audio Pair 3 Format |
| Audio Pair 3 Value |
| Audio Pair 4 Format |
| Audio Pair 4 Value |

rINFO 1-3 contains the bitrate at which the file was encoded.

rINFO 4 is reserved for future use.

Note: This info request returns binary values in both binary and ASCII protocol, not BCD.

**Table 27. ID Info 3: Timecode / Duration**

**Start of Clip Timecode**

rINFO 1 contains the frames value for the Start-of-Clip timecode.

rINFO 2 contains the seconds value for the Start-of-Clip timecode.

rINFO 3 contains the minutes value for the Start-of-Clip timecode.

rINFO 4 contains the hours value for the Start-of-Clip timecode.

**Clip Duration**

rINFO 1 contains the frames value for the ID DURATION.

rINFO 2 contains the seconds value for the ID DURATION.

rINFO 3 contains the minutes value for the ID DURATION.

rINFO 4 contains the hours value for the ID DURATION.

rINFO 1 contains the number of remaining bytes of audio data.

**Table 28.  ID Info 4:  Audio Information**

rINFO 2 contains the Audio Format for the first audio pair.

rINFO 3 contains the Audio Value for the first audio pair.

page 60

Command States

rINFO 4-M contain the Audio Format and Value for the subsequent audio pairs. Currently, up to 4

audio pairs are supported.

See the AUDF command for valid values and their meaning.

**8.6.4 ID BYTE COUNT QUERY**

**VARIABLE NAME BINARY: <BX.71><LEN><ID C:32+>**

**returns <CUR\_SIZE I:4><FINAL\_SIZE I:4>**

**FIXED NAME BINARY: <3X.71><ID C:8>**

**returns rDATA 1-8**

**ASCII: IDBY? "<ID C:32+>"**

**returns <CUR\_SIZE>,<FINAL\_SIZE>**

LEN contains the length of the ID name.

ID contains the name of the ID..

CUR\_SIZE returns the video file system’s current size for the ID.

FINAL\_SIZE returns the video file system’s final size for the ID.

The IDBY? query returns the current size in DATA 1-4 and the final size in DATA 5-8 of the

specified id in Kbytes.

If the file is not currently being transferred, restored or recorded then both numbers returned will

be the final file size. If the file is being transferred, restored or recorded, the final size is the

estimated final size of the standard track.

This information can be used to calculate statistics on recording and transferring IDs. Examples

of these statistics include percent of transfer complete and average transfer speed.

Command States

page 61

**8.7**

**USER INFO COMMANDS**

The MediaStream Server provides for a "User Info" track for each file in the system. The User

Info track is provided specifically for automation systems to store and retrieve information about

the material that is not included in the information already stored by the server. This includes

database information tracked by the automation system or other user information data associated

with the material. The MSS server does not use the information in the user info track in any way.

The server simply carries the information along with the material for retrieval by the automation

system.

The User Info track is organized as a collection of KEYWORD / VALUE pairs. The KEYWORD is

a unique string identifier not to exceed 32 alphanumeric, underscore and hyphen characters. The

VALUE is the ASCII representation of the information to be stored for that KEYWORD and

consists of a string of not more than 128 non-null 7-bit ASCII characters.

It should be noted that many of the commands to access the User Info track are not real time

commands and therefore do not meet the 1 frame response specification. As a consequence,

care should be taken not to issue these commands when they might delay the execution of frame

accurate immediate commands. See documentation of the individual commands for details.

**8.7.1 USER INFO OPEN**

**VARIABLE NAME BINARY: <AX.60><MODE><LEN><ID C:32+>**

**FIXED NAME BINARY: <2X.60><MODE><ID C:8>**

**ASCII: UOPN <MODE>,"<ID C:32+>"**

MODE contains the access Mode for the open operation.

LEN contains the length of the ID name.

ID contains the ID name.

NOTE: Do not open, close or select video ports while a user info track is open. If it is necessary

to change ports, close the currently open user info track before proceeding.

The UOPN command opens the User Info track of the specified ID for reading or writing. To open

the ID for READ access set MODE to 1. To open the ID for WRITE ONLY access set MODE to

2. To open the ID for READ AND WRITE access set MODE to 3. The busy bit will be set while

the user info track is in the process of being opened. If a User Info track is already open, it will be

closed and written out to disk prior to opening the new User Info track.

**8.7.2 USER INFO CLOSE**

**BINARY: <AX.61>**

**BINARY: <2X.61>**

**ASCII: UCLS**

The UCLS command closes the currently open User Info track and writes the data out to disk.

The busy bit will be set while the track is being closed.

page 62

Command States

**8.7.3 USER INFO ADD**

**BINARY: <2X.62><LEN><KEYWORD C:32+><LEN2><VALUE C:M+>**

**ASCII: UADD "<KEYWORD C:32+>","<VALUE C:M+>"**

LEN contains the length of the KEYWORD string.

KEYWORD contains the name of the user info field.

LEN2 contains the length of the VALUE string.

VALUE contains the value string.

The UADD command associates VALUE with KEYWORD in the currently open User Info track.

KEYWORD is a unique string identifier not to exceed 32 alphanumeric, underscore and hyphen

characters. The VALUE is an ASCII representation of the information to be stored for that

KEYWORD and consists of a string of not more than 128 non-null 7-bit ASCII characters.

For example, the binary sequence for the KEYWORD of "Title" and the VALUE of "test1" would

be 0x5, ‘T’, ‘I’, ‘T’, ‘L’, ‘E’, 0x5, ‘t’, ‘e’, ‘s’, ‘t’, ‘1’.

The busy bit will be set while this operation is performed. This command may not return within a

frame.

**8.7.4 USER INFO MODIFY**

**BINARY: <2X.63><LEN><KEYWORD C:32+><LEN2><VALUE C:M+>**

**ASCII: UMOD "<KEYWORD C:32+>","<VALUE C:M+>"**

LEN contains the length of the KEYWORD string.

KEYWORD contains the name of the User Info field.

LEN2 contains the length of the VALUE string.

VALUE contains the VALUE string.

The UMOD command modifies the VALUE in the user info data associated with KEYWORD. The

KEYWORD is a unique string identifier not to exceed 32 alphanumeric, underscore and hyphen

characters. The VALUE is the ASCII representation of the information to be stored for that

KEYWORD and consists of a string of not more than 128 non-null 7-bit ASCII characters. For

example, the binary sequence for the KEYWORD of "Title" and the VALUE of "test1" would be 5

T I T L E 5 t e s t 1.

The busy bit will be set while this operation is performed. This command may not return within a

frame.

**8.7.5 USER INFO DELETE**

**BINARY: <2X.64><LEN><KEYWORD C:32+>**

**ASCII: UDEL "<KEYWORD C:32+>"**

LEN contains the length of the KEYWORD string.

KEYWORD contains the name of the User Info field.

The UDEL command deletes the KEYWORD and associated VALUE from the user info data.

KEYWORD is a unique string identifier not to exceed 32 characters consisting of alphanumeric

underscore and hyphen characters. The busy bit will be set while this operation is performed.

This command may not return within a frame.

Command States

page 63

**8.7.6 USER INFO QUERY**

**BINARY: <3X.63><LEN><KEYWORD C:32+>**

**returns <rLEN><VALUE C:M+>**

**ASCII: UQRY? "<KEYWORD C:32+>"**

**returns <VALUE C:M+>**

LEN contains the length of the KEYWORD string.

KEYWORD contains the name of the User Info field.

rLEN returns the length of the value string.

VALUE returns the value of the User Info field.

The UQRY? query returns the VALUE associated with the specified KEYWORD for the currently

open user info track. Another UQRY? cannot be sent until the first has completed. This command

will take a long time to complete, depending on the size of the file. Expected completion time is

less than or equal to 1 sec.

**8.7.7 USER INFO LIST**

**BINARY: <3X.64>**

**returns <REMAINING I:2><LEN><KEYWORD><LEN2><VALUE>**

**ASCII: ULST?**

**returns <REMAINING I:2>, “<KEYWORD C:32+>”, “<VALUE C:M+>”**

REMAINING contains the number of KEYWORDs remaining.

LEN contains the length of the KEYWORD string.

KEYWORD contains the name of the User Info field.

LEN2 contains the length of the VALUE string.

VALUE contains the value of the User Info field.

The ULST? query returns a list of KEYWORDS and VALUES defined for the currently open User

Info track. One KEYWORD and VALUE at a time are returned until all KEYWORDS have been

listed. Returns 2 bytes indicating the remaining list items followed by the list of KEYWORDS.

ULST? is called repeatedly until list is finished. If the User Info file does not follow standard

format, no KEYWORDS will be returned. This command will take a long time to complete,

depending on the size of the file. Expected completion time is less than or equal to 1 sec.

page 64

Command States

|  |  |
| --- | --- |
| System Bit Rate | 10Mb/s |
| Vertical Line Info | All Items OFF |

**8.8**

**ENCODER CONFIGURATION COMMANDS**

The encoder port must be in the IDLE state when configuration commands are executed.

**8.8.1 PRESET**

**BINARY: <2X.30>**

**ASCII: PRST**

The PRST command applies the default settings described below to the selected port. The PRST

command can be issued to a port in any state.

For input ports, the encoder is set to the following values.

**8.8.2 SYSTEM BIT RATE**

**Table 29.  Input Port Preset Values**

**BINARY: <2X.31><BIT\_RATE U:4>**

**ASCII: SBIT <BIT\_RATE>**

BIT\_RATE contains a 32 bit unsigned number representing bit rate in bits/sec.

The SBIT command sets the aggregate system bit rate for the compressed video/audio stream

when recording on the selected input port. The resolution is 0.1 Megabits/sec. Input video will be

sampled at CCIR 601 rates. An error condition will result in the appropriate bit being set in the

port status error bytes.

**8.8.3 SYSTEM BIT RATE QUERY**

**ASCII: SBIT?**

**returns <BIT\_RATE>**

BIT\_RATE returns a 32 bit unsigned number representing the bit rate for the selected input port in

bits/sec.

The SBIT? query returns the system bit rate setting for the selected input port. See the SBIT

command for more details.

**8.8.4 VIDEO STANDARD**

**BINARY: <2X.3D><STD>**

**ASCII: VSTD <STD>**

STD contains the value for the desired video standard.

The VSTD command sets the video standard for the selected input port.

The following table describes the supported video standards.

Command States

page 65

|  |  |
| --- | --- |
| VALUE | DESCRIPTION |
| 0 | NTSC |
| 1 | PAL |
| 32 | HD 720P 59.94 frames/second |
| 40 | HD 1080I 29.97 frames/second |
| 41 | HD 1080I 25 frames/second |

An error condition will result in the appropriate bit being set in the port status error bytes.

**Table 30.  VSTD Parameters**

**8.8.5 VIDEO STANDARD QUERY**

**BINARY: <3X.78>**

**Returns <STD>**

**ASCII: VSTD?**

**returns <STD>**

rDATA 1 (MODE) returns the value for the current video standard.

The VSTD? query returns the video standard setting for the selected input port. See the VSTD

command for parameter definitions.

**8.8.6 VERTICAL LINE INFO**

**BINARY: <2X.3F><ITEM><LINE><PARM>**

**ASCII: VILI <ITEM>, <LINE>, <PARM>**

ITEM contains the item number.

LINE contains the line number to be encoded. In NTSC, supported values are 0 and 10 through

21 inclusive. In PAL, supported values are 0, 7 through 22 inclusive and 320 through 335

inclusive.

PARM contains additional data.

The VILI command enables and disables encoding of vertical interval lines. The encoder can

capture 8-bit or 10-bit VBI data depending upon the setting of the VBI Bits field in the ECFG

command. Changing the VBI configuration with the ECFG command will clear all previous VBI

settings.

If the encoder is set to 8 VBI Bits of data, 10 items can be captured. In NTSC, PARM specifies

which field of the line is captured. Valid values are 1 and 2. In PAL, PARM is ignored.

If the encoder is set to 10 VBI Bits of data, 4 items can be captured.

Sending the VILI command for a given item with a line number of 0 will disable capturing of that

item.

An error condition will result in the appropriate bit being set in the port status error bytes.

The following restrictions apply to 68k-based and REV1 HD decoders.

page 66

Command States

•

•

•

10-bit VBI data is not supported.

A maximum of 6 lines can be displayed.

ARIB functionality is not supported. (See ECFG command for details.)

**8.8.7 VERTICAL LINE INFO QUERY**

**BINARY: <3X.7C><ITEM>**

**returns <LINE><PARM>**

**ASCII: VILI? <ITEM>**

**returns <LINE>,<FIELD>**

ITEM contains the item number. Valid values are 1 through 16.

LINE returns the LINE value currently set for the specified ITEM.

PARM returns the PARM value currently set for the specified ITEM.

The VILI? query returns the vertical interval line info for the item number specified. Supported

item numbers are 1 through 16. See the VILI command for more details.

**8.8.8 OUTPUT PROGRAM ID**

**BINARY: <2X.67><ID I:2>**

**ASCII: PGID <ID>**

ID contains the program ID.

The PGID command sets the output program ID for the selected input port. The default value is

to 0, which means to ingest the first program in the transport stream Program Association Table

(PAT). Valid program IDs are 1 through 65535.

**8.8.9**

**OUTPUT PROGRAM ID QUERY**

**BINARY: <3X.67>**

**returns <ID>**

**ASCII: PGID?**

**returns <ID>**

ID returns the currently set program ID for the selected encoder port.

The PGID? query returns the Program ID set by a previous PGID command.

**8.8.10 DROP FRAME**

**BINARY: <2X.72><MODE>**

**ASCII: DRFR <MODE>**

MODE contains the value to which the dropframe state will be set.

Command States

page 67

The DRFR command enables (1), or disables (0) drop frame mode on the selected port. Once

the drop frame state has been set, it is remembered after power cycle. The drop frame mode

only affects recording or playing of NTSC video.

Drop frame timecode drops 108 frames every hour to make up for the slower than 30 frames per

second rate of NTSC video. Timecodes in dropframe do not use frames 0 or 1 at the start of

each minute EXCEPT for minutes 0, 10, 20, 30, 40 , and 50.

If drop frame is enabled, the displayed time code (GTTC), start of clip (SSOC, RCWD), and the

mark-in position (PCWD, PCD2, NCPY) will follow the standard drop frame calculations. Duration

is not represented as dropframe timecode.

The duration is converted to a frame number by multiplying by 29.97 and rounding up. If drop

frame is disabled, the duration is converted to a frame number by multiplying by 30.

**8.8.11 DROP FRAME QUERY**

**BINARY: <3X.76>**

**returns <MODE>**

**ASCII: DRFR?**

**returns <MODE>**

MODE contains the drop frame setting.

The DRFR? query returns the drop frame mode of the selected port. DRFR? returns a 1 if drop

frame is enabled and a 0 if drop frame is disabled. The drop frame only affects recording or

playing NTSC video.

**8.8.12 ENCODER AUTO STOP**

**BINARY: <2X.73><MODE>**

**ASCII: ENAS <MODE>**

MODE contains the value to set.

The ENAS command sets the auto-stop mode of the selected input port. When auto stop is

enabled (1), the encoder will stop recording within ½ second of the duration set by the RCUE or

RCWD command and set the clip’s duration to this value. If auto stop is disabled (0), the encoder

will record until a STOP command is issued. In this case, the recording will terminate within ½

second of the frame following the STOP command. See the RECD command for more

information on this functionality.

**8.8.13 ENCODER AUTO STOP QUERY**

**BINARY: <3X.75>**

**returns <MODE>**

**ASCII: ENAS?**

**returns <MODE>**

MODE contains the value of the auto stop mode.

page 68

Command States

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CMD | FUNCTION | PARAMETER | OPT  PARM | DESCRIPTION |
| 0 | CLEAR | 0 |  | Clear all intermediate settings since the last set (MPRM  255,0) command. |
| 1 | SET FORMAT | Media Format |  | Media FormatC: 2 = Mpeg420  3 = Mpeg 422  4 = SMPTE-356M |
| 2 | SET  SUBFORMAT | Media  Subformat |  | For Mpeg formatC: Subformat 0 = IBP  1 = IB  2 = I only |
| 255 |  | 0 |  | Check and set the intermediate parameters entered. |

The ENAS? query returns the encoder auto stop mode of the selected input port. See the ENAS

and RECD commands for details of this functionality..

**8.8.14 MEDIA PARAMETERS**

**BINARY: <2X.79><CMD><PARM I:2+>**

**ASCII: MPRM <CMD>, <PARM>**

CMD contains the command code for the media parameter.

PARM contains the value for that parameter.

The MPRM command sets media format parameters on the selected input port. The MPRM

command has a CMD parameter which specifies the action of the command. The decoder

characteristics are only changed when the set CMD (MPRM 255,0) is received.

**8.8.15 MEDIA PARAMETERS QUERY**

**Table 31.  MPRM Parameters**

**BINARY: <3X.72><CMD>**

**returns <PARM I:2+>**

**ASCII: MPRM? <CMD>**

**returns PARM**

CMD contains the command code for the media parameter.

PARM returns the value for the parameter specified in CMD.

The MPRM? query returns the current setting for the parameter specified by CMD for the

currently selected input port. MPRM? 255 returns 0. For details see the table in the description of

the MPRM command.

**8.8.16 AUDIO FORMAT**

**BINARY: <2X.7A><PAIR\_ID><FORMAT><VALUE>**

**ASCII: AUDF <PAIR\_ID>, <FORMAT>, <VALUE>**

PAIR\_ID contains the number for the audio pair to be set. Up to 4 audio pairs are supported,

depending upon the hardware installed.

FORMAT contains the format for this audio pair.

Command States

page 69

|  |  |  |
| --- | --- | --- |
| FORMAT | VALUE | DESCRIPTION |
| 0 | 0 | Audio OFF. |
| 1 | 1 | MPEG L2(Musicam): 256,000 bps (default) |
| 1 | 2 | MPEG L2(Musicam): 384,000bps |
| 2 | 0 | Unsupported |
| 3 | 1 | Uncompressed 16 bit audio |
| 3 | 2 | Uncompressed 20 bit audio |
| 4 | 1 | Data 16 bit |
| 4 | 2 | Data 20 bit |

VALUE contains a qualifying characteristic for the format selected.

The AUDF command sets the audio format for the selected input port. For any of the audio pairs,

the format and value settings available are shown in the table below.

Data 16 can be used to store AC3 audio data.

**Table 32.  AUDF Parameters**

Data 20 can be used to store Dolby audio data.

**8.8.17 AUDIO FORMAT QUERY**

**BINARY: <3X.62><PAIR\_ID>**

**returns <FORMAT><VALUE>**

**ASCII: AUDF? <PAIR\_ID>**

**returns <FORMAT>,<VALUE>**

PAIR\_ID contains the index of the audio pair for which the settings are desired.

FORMAT returns the format for this audio pair.

VALUE returns a qualifying characteristic for the format selected.

The AUDF? query returns the FORMAT and VALUE information for the specified audio pair on

the selected input port. See AUDF for interpretation of returned values.

**8.8.18 AUDIO LEVEL QUERY**

**BINARY: <3X.7F><CHAN>**

**returns <VALUE>**

**ASCII: ALVL? <CHAN>**

**returns <VALUE>**

CHAN contains the index of the audio channel.

VALUE returns the audio level.

The ALVL? query returns the audio level (in dB) for the specified channel on the selected input

port. If CHAN is not a valid index for an audio channel, or there is no audio input, the operation

will fail and -90 dB will be returned. in VALUE.

NOTE: This query will not return in the normal specification of 1 frame and may take as long as 1

second to execute.

page 70

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Setting 8 | Setting 7 | Setting 6 | Setting 5 | Setting 4 | Setting 3 | Setting 2 | Setting 1 |

|  |
| --- |
| 0 |
| 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 48 kHz | 44.1 kHz | 32 kHz |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | MONO | JOINT | DUAL | STEREO |

|  |
| --- |
| 32 BIT VALUE REPRESENTING THE SYSTEM BIT RATE FOR THE |
| SELECTED VIDEO PORT IN BITS PER SECOND |
|  |
|  |

**8.8.19 COMPRESSION SETTINGS QUERY**

**BINARY: <3X.17><BITMAP>**

**returns <LEN><BITMAP><SETTINGS I:15+>**

**ASCII: CSET? <BITMAP>**

**returns <LEN>,<BITMAP><, SETTING I:25>**

BITMAP contains a bitmap specifying which Settings should be returned.

LEN returns the byte count of N-1 (BITMAP plus SETTINGs bytes).

BITMAP returns the BITMAP from the command sent.

SETTING returns the status byte(s) specified by the BITMAP as shown below.

The CSET? query returns information about the encoding parameters for the selected input port.

The compression settings available and their contents are shown in the tables below.

**Table 33.  CSET? Query Bitmap**

**Table 35. Compression Setting 2: Audio Sample Rate and Mode**

**Table 34.  Compression Setting 1:  Currently Unused**

**Table 37. Compression Setting 6: Video Standard and Resolution**

**Audio Sample Rate**

**Audio Mode**

**Table 36.  Compression Setting 3: System Bit Rate**

**Video Standard**

BYTE REPRESENTING VIDEO STANDARD. NTSC = 0, PAL = 1

**Horizontal Resolution**

Command States

page 71

|  |
| --- |
| 16 BIT VALUE REPRESENTING HORIZONTAL VIDEO RESOLUTION |
|  |

|  |
| --- |
| 16 BIT VALUE REPRESENTINGVERTICAL VIDEO RESOLUTION |
|  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | CLOSED  CAPTION |

**8.8.20 ENABLE VITC CAPTURE**

**Vertical Resolution**

**Table 38.  Compression Setting 7: Closed Caption**

**ASCII: ENVT < LINE>**

**ASCII: ENVT?**

**returns <LINE>**

LINE specifies the Line number in the vertical interval which contains the VITC value.

Supported values are 0 and 7 through 22 inclusive.

The ENVT command sets the vertical interval timecode capture mode on the selected input port.

A timecode is read from the specified line during the first frame of an encode. This value is then

stored as start of clip (SOC) for that ID. A valid line number will enable the vertical interval

timecode capture, a line number of 0 will disable capture. Before issuing this command, the VILI

command must be issued to enable encoding of the desired vertical interval line.

The ENVT? query returns the line number set on the selected input port.

**8.8.21 PREROLL**

**ASCII: PRER <FRAMES>**

**ASCII: PRER?**

**returns <FRAMES>**

FRAMES contains the number of preroll frames.

The PRER command sets the preroll for the selected input or output port to the specified number

of frames.

If FRAMES is 0, the selected port is set to normal operation, i.e., immediate commands are

executed in the frame after they are received (if received in the first field).

If FRAMES is a positive number, the execution of immediate commands are delayed the

specified number of frames from when they are received.

If FRAMES is negative, the operation fails with an ILLEGAL\_VALUE error condition.

page 72

Command States

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Value |  |  | SD Line  Count | MPEG  IN  Preview | ARIB  ancillary  Data | VBI Bits | Video  Packet  Type | MUX  Mode |
| 0 |  |  | 480/576 | OFF | OFF | 8 | PES | CBR |
| 1 |  |  | 512/608 | ON | ON | 10 | ES | VBR |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Encoder |  |  | SD Line  Count | MPEG  IN  Preview | ARIB  ancillary  Data | VBI Bits | Video  Packet  Type | MUX  Mode |
| 68k |  |  |  |  |  |  |  | YES |
| PPC  MPEG-IN |  |  |  | YES | YES |  | TBD |  |
| PPC SD  dual |  |  | YES |  |  | YES | YES | YES |

The PRER? querry command returns the current preroll setting for the selected port.

**8.8.22 ENCODER CONFIGURATION**

**ASCII: ECFG <BITMAP>**

**ASCII: ECFG?**

**returns <BITMAP>**

BITMAP contains the configuration settings. See the table below for details.

The ECFG command sets the configuration values for the selected input port.

The default value for all bits is zero.

**Table 39.  Encoder Configuration (ECFG) Bitmap**

The following table summarizes the bits supported by encoder type. An empty cell indicates that

the bit is not supported for that encoder.

Changing the VBI configuration will clear all previous VBI settings.

**Table 40.  Encoder Types Supporting Configuration (ECFG) Functions**

Command States

page 73

**8.9**

**DECODER CONFIGURATION COMMANDS**

The decoder port must be in the IDLE state when configuration commands are executed.

The DRFR and PRER commands and queries also affect decoders. See DRFR in the Encoder

Configuration Commands section for details.

**8.9.1 SET GPI**

**BINARY: <2X.40><MODE>**

**ASCII: SGPI <MODE>**

NOTE: This command is currently NOT IMPLEMENTED

MODE is set to 0 to disable the GPI input and 1 to enable it.

**8.9.2 SUBCARRIER ADJUST**

**BINARY: <2X.41><BURST\_PHASE I:2><HUE\_PHASE I:2>**

**ASCII: SADJ <BURST\_PHASE>, <HUE\_PHASE>**

BURST\_PHASE contains a value from -1800 to +1800 representing 0.1 degree steps.

HUE\_PHASE contains a value from -1800 to +1800 representing 0.1 degree steps.

The SADJ command sets the burst and hue phase for the selected output port. BURST\_PHASE

specifies the phase of the color burst. HUE\_PHASE specifies the phase of the active video

relative to color burst.

**8.9.3 SUBCARRIER ADJUST QUERY**

**BINARY: <3X.7A>**

**returns <BURST\_PHASE I:2><HUE\_PHASE I:2>**

**ASCII: SADJ?**

**returns <BURST\_PHASE>, <HUE\_PHASE>**

The SADJ? query returns the values set by the SADJ command. See the SADJ command for

parameter definitions.

**8.9.4 HORIZONTAL SYNC TIMING**

**BINARY: <2X.42><VALUE I:2>**

**ASCII: HSYN <VALUE>**

VALUE contains a number representing the horizontal sync offset. See the HSYN Parameters

Table for details.

The HSYN command sets the horizontal synch offset for the selected port.

page 74

Command States

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Decoder | Minimum  VALUE | offset | Maximum  VALUE | offset |
| 68K | 0 | - 8.192us | 8191 | 8.192us |
| Dual SD | 0 | - 8.192us | 8191 | 8.192us |
| MPEG OUT | 0 | - 8.192us | 8191 | 8.192us |
| HD | 0 | - 8.192us | 200,000 | 391.808us |

The default value is 3072 which corresponds to -2.048us. One unit of VALUE corresponds to

0.002us of horizontal synch offset. Values above 32,767 are only available in the ASCII protocol.

**8.9.5 HORIZONTAL SYNC TIMING QUERY**

**Table 41.  HSYN Parameters**

**BINARY: <3X:7B>**

**returns <VALUE>**

**ASCII: HSYN?**

**returns <VALUE>**

VALUE returns the horizontal sync offset previously set on the selected port with the HSYN

command.

The HSYN? query returns the horizontal synch offest for the selected port.

**8.9.6 EMBEDDED AUDIO**

**BINARY: <2X.65><MODE>**

**ASCII: EAUD <MODE>**

MODE contains a 0 (disable) or 1 (enable).

The EAUD command enables or disables the embedded audio functionality on the selected port.

When sent to an input port, the EAUD command determines whether embedded audio is

encoded from the input stream. If EAUD is enabled on an input port, there must be embedded

audio on the input stream.

When sent to an output port, the EAUD command determines whether embedded audio is

decoded from the stored file. If the EAUD command is enabled on an output port, there must be

embedded audio in the recorded file.

**8.9.7 EMBEDDED AUDIO QUERY**

**BINARY: <3X.65>**

**returns <MODE>**

**ASCII: EAUD?**

**returns <MODE>**

MODE returns the embedded audio setting.

The EAUD? query returns the current embedded audio setting for the selected port.

Command States

page 75

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  | IRE  SETUP | EOF | LAST  FRAME |

**8.9.8 GENLOCK MODE**

**BINARY: <2X.70><MODE>**

**ASCII: GLOC <MODE>**

MODE contains the genlock mode. OFF is represented by 0 and ON by 1.

The GLOC command sets the genlock mode for the selected port.

**8.9.9 GENLOCK MODE QUERY**

**BINARY: <3X.79>**

**returns <MODE>**

**ASCII: GLOC?**

**returns <MODE>**

MODE returns the genlock mode.

The GLOC? query returns the genlock mode of the selected port.

**8.9.10 DECORDER CONFIGURATION**

**BINARY: <2X.71><I:4>**

**ASCII: DCFG <BITMAP>**

BITMAP contains the values to be set.

The DCFG command sets the parameters for the currently selected decoder. Currently, only the

EOF and last frame modes may be set.

**IRE Setup**

**Table 42.  DCFG Command Bitmap**

The decoder can be configured with a 7.5 IRE Setup enabled (0) or disabled (1).

**EOF / Last Frame**

End of File (EOF) is defined to be the final frame displayed when the duration equals the size for

the ID.

Last Frame (EOM) is defined to be the frame played when the specified duration is reached.

The default behavior with both bits cleared is to switch to the background color (set by the BCLR

command) when playout completes.

If the specified duration is greater than zero and is less than or equal to the clip length, the

behavior when playout reaches this duration is defined by the Last Frame bit. If this bit is clear,

page 76

Command States

|  |  |
| --- | --- |
| MODE Value | Operaion |
| 0 | No audio cueing. |
| 1 | Audio cueing enable. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Last Frame = 1** | **Last Frame = 0** | **EOF = 1** | **EOF = 0** |
| 0<EOM<=EOF | STILL / last frame | IDLE / BCLR | N/A | N/A |
| EOM=0 or  EOM>EOF | N/A | N/A | STILL / last frame | IDLE / BCLR |

the decoder will switch to the BCLR. If the Last Frame bit is set, the decoder will freeze on the

Last Frame.

If the specified duration is either 0 (no duration) or is greater than the clip length, the behavior

when playout reaches the clip end is governed by the EOF bit. If the EOF bit is clear, the

decoder switches to BCLR. If the bit is set, the decoder will freeze on the EOF frame.

The STOP command will always set the decoder state to IDLE and display BCLR.

Note: If the last frame is black in freeze on last frame mode, then the decoder state will be STILL

**Table 43.  Decoder EOF / Last Frame Actions**

but the screen will be displaying black.

**8.9.11 DECORDER CONFIGURATION QUERY**

**BINARY: <3X.61>**

**returns <MODE I:4>**

**ASCII: DCFG?**

**returns <MODE>**

DATA returns the decoder configuration.

The DCFG? query returns the decoder configuration bitmap as described in the DCFG Command

Bitmap Table.

**8.9.12 AUDIO CUE**

**BINARY: <2X.78><MODE><PARM>**

**ASCII: ACUE <MODE>,<PARM>**

MODE contains the mode value to set.

PARM is currently ignored.

The ACUE command sets the audio cue mode for the selected output port.

Legal values for MODE are described in the table below.

**8.9.13 AUDIO CUE QUERY**

**Table 44.  Audio Cue Modes**

Command States

page 77

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Decoder  Settings 8 | Decoder  Settings 7 | Decoder  Settings 6 | Decoder  Settings 5 | Decoder  Settings 4 | Decoder  Settings 3 | Decoder  Settings 2 | Decoder  Settings 1 |

|  |
| --- |
| 0 |
| 0 |

**BINARY: <3X.77>**

**returns <MODE><LOOP>**

**ASCII: ACUE?**

**returns <MODE>,<LOOP>**

MODE contains the audio cue mode setting.

LOOP contains the loop setting.

The ACUE? query returns the Audio Cue Settings for the selected output port.

**8.9.14 DECODER SETTINGS QUERY**

**BINARY: <3X.23><BITMAP>**

**returns <LEN><BITMAP><SETTING I:3+>**

**ASCII: DSET? <BITMAP>**

**Returns <LEN>,<BITMAP>,<SETTING I:23+:,>**

BITMAP contains a bitmap specifying which information should be returned.

LEN contains the number of valid items returned.

The DSET? query returns to the user information about the decoder settings for the selected

video port.

**Table 45. DECODER SETTINGS Bitmap**

**Table 47. DECODER SETTINGS 6- VIDEO STANDARD**

**Table 46.  DECODER SETTINGS 1 – UNUSED**

BYTE REPRESENTING VIDEO STANDARD. NTSC = 0, PAL = 1

**8.9.15 CHROMA GAIN REFERENCE**

**ASCII: CREF < VALUE>**

**ASCII: CREF?**

**returns <VALUE>**

VALUE contains an integer value between -61 and 194.

The CREF command sets the reference for the automatic chroma gain control. This setting

affects the color saturation seen on the output.

The CREF? query returns the reference value for the automatic chroma gain control.

page 78

Command States

|  |  |
| --- | --- |
| MODE | Operaion |
| 0 | Use the system "default" upconversion. This is currently anamorphic. |
| 1 | Add sidebars. |
| 2 | Tilt and scan. |
| 3 | Anamorphic upconversion. |
| 4 | Add partial sidebars. This mode trims a bit off the top and bottom but  has narrower sidebars. |
| 5 | Non-uniform scale. This scales the center linearly, but does an  anamorphic conversion on the sides. |
| 6 | "Glass prism". This is a sidebar conversion, but the left and right pixel  of each line get extended to the side of the screen. |
| 7 | Pre-letterboxed. This assumes the original SD content was already  letterboxed, so the top is trimmed and bottom bands and magnify the  middle. |
| 8 | Partial pre-letterbox. |

**8.9.16 DECODER AUDIO GAIN**

**ASCII: DAGN <CHAN>, <GAIN>**

**ASCII: DAGN? <CHAN>**

**returns <GAIN>**

CHAN contains the audio channel number.

GAIN contains the audio gain in units of 1/10 dB.

The DAGN command sets the audio gain for the specified audio channel on the selected output

port.

The GAIN specified must be in increments of 0.5 dB. Valid values range from –1025 to +250 (–

102.5 to 25.0 dB).

The DAGN? query returns the audio gain for the specified audio channel on the selected output

port.

**8.9.17 UPCONVERSION MODE**

**ASCII: UCMD <MODE>**

**ASCII: UCMD?**

**returns <MODE>**

MODE contains the mode of upconversion.

The UCMD command sets the method to be used when upconverting SD files for playout on an

HD decoder. Valid modes are listed in the Up-Conversion Modes Table.

The UCMD? query returns the upconversion mode setting for the selected output port.

**Table 48.  Up-Conversion Modes**

Command States

page 79

|  |  |  |  |
| --- | --- | --- | --- |
| Color | Y | Cb | Cr |
| Blue | 35 | 212 | 114 |
| Black | 16 | 128 | 128 |
| Red | 65 | 100 | 212 |

**8.9.18 DOWNCONVERSION MODE**

**ASCII: DCMD <MODE>**

**ASCII: DCMD?**

**returns <MODE>**

MODE contains the mode of upconversion.

The DCMD command sets the method to be used when downconverting HD files for playout on

an SD decoder. See the Up-Conversion Modes Table for valid modes.

The DCMD? query returns the downconversion mode setting for the selected output port.

**8.9.19 SET BACKGROUND COLOR**

**ASCII: BCLR <Y>,<Cb>,<Cr>**

**ASCII: BCLR?**

**returns <Y>,<Cb>,<Cr>**

Y contains the Y component of the background color.

Cb contains the Cb component of the background color.

Cr contains the Cr component of the background color.

The BCLR command sets the color displayed on the selected output port when it is IDLE. The

default background color is black. The Background Colors Table shows some common

combinations.

The BCLR? query returns the background color setting for the selected output port.

**Table 49.  Background Colors**

**8.9.20 SET CONVERSION COLOR**

**ASCII: CCLR <Y>,<Cb>,<Cr>**

**ASCII: CCLR?**

**returns <Y>,<Cb>,<Cr>**

Y contains the Y component of the conversion color.

Cb contains the Cb component of the conversion color.

Cr contains the Cr component of the conversion color.

The CCLR command sets the color of the non-video areas that can occur when in video

conversion mode. The top and bottom portions of the screen in letterbox mode are examples of

page 80

Command States

these non-video areas. The default conversion color is black. The Background Colors Table

shows some common combinations.

The CCLR? query returns the conversion color setting for the selected output port.

Command States

page 81

**8.10 SYSTEM CONTROL**

**8.10.1 DEVICE TYPE QUERY**

**BINARY: <3X.08>**

**BINARY: <0X.11>**

**returns <TYPE C:32+>**

**ASCII: DTYP?**

**returns <TYPE C:32+>**

TYPE returns a string specifying the model of the system.

The DTYP? query returns the system model number. For servers running A.07.XX or B.08.XX

releases, this string is "HP:E2524B".

**8.10.2 PROTOCOL**

**BINARY: <0X.01><MODE><PERM>**

**<0X.02><MODE><PERM>**

**ASCII: PTCL <MODE>, <PERM>**

MODE contains the desired protocol.

PERM contains a byte indicating whether to permanently save this as the default protocol (1) or

only temporarily modify the protocol(0).

The PTCL command sets the protocol for the selected port. Currently, the only supported

protocol is VDCP (1).

**8.10.3 PROTOCOL QUERY**

**BINARY: <3X.7E>**

**returns <MODE>**

**ASCII: PTCL?**

**returns <MODE>**

MODE returns the protocol setting.

The PTCL? query returns the protocol setting of the selected port. See the PTCL command for

parameter details. A return of 0 indicates that no video port has been selected.

**8.10.4 SOFTWARE REVISION QUERY**

**BINARY: <3X.60>**

**returns <REV C:X+>**

**ASCII: SREV?**

**retrurns <REV C:X+>**

REV returns the revision string. The Warp release B.09.21 returns the string “B.09.20: p1”. The

MSS\_NS release C.15.71 returns the string “C.15.70: p1”.

page 82

Command States

The SREV? query returns a string, which indicates the software revision number of the currently

running software, plus any patch number that may be installed.

**8.10.5 DISK STORAGE SIZE QUERY**

**BINARY: <3X.7D>**

**returns <TOTAL I:4><UNUSED I:4>**

**ASCII: DSIZ?**

**returns <TOTAL>,<UNUSED>**

TOTAL returns the total system disk capacity in KBytes.

UNUSED returns the number of KBytes of unused system disk space.

The DSIZ? query returns total and unused disk space on the system. The ratio of these values is

the same value used to determine the current free space for the Disk Near Full flag.

**8.10.6 MINIMUM FREE SPACE**

**BINARY: <2X.2B><PERCENT\_FREE\_SPACE>**

**ASCII: MFRE <PERCENT\_FREE\_SPACE>**

PERCENT\_FREE\_SPACE contains the percent of disk space to be kept free.

The MFRE command sets the minimum free space percentage for the system. When a record

cue or transfer to the system would cause the free space percentage to fall below this value, the

Disk Near Full flag will be set in the System Status. This setting does not effect whether the

record cue or transfer will be successful.

**8.10.7 MINIMUM FREE SPACE QUERY**

**ASCII: MFRE?**

**returns <PERCENT\_FREE\_SPACE>**

PERCENT\_FREE\_SPACE returns the percent of disk space to be kept free.

The MFRE? query returns the minimum free space percentage for the system.

**8.10.8 PARAMETER LIMITS QUERY**

**BINARY: <3X.73><CMD><PARM>**

**returns <MIN I:4><MAX I:4>**

**ASCII: PLMT? <CMD>, <PARM>**

**returns <MIN>, <MAX>**

CMD contains the command type to be queried.

PARM contains a parameter associated with the command.

MIN returns the minimum setting.

MAX returns the maximum setting.

Command States

page 83

|  |  |  |  |
| --- | --- | --- | --- |
| CMD | PARM | OPT PARAM | DESCRIPTION |
| 1 | Media Format |  | Returns the Min and Max supported bitrate values for the  specified Media Format: 2 = Mpeg 4:2:0 and 3 = Mpeg  4:2:2 |

|  |  |  |  |
| --- | --- | --- | --- |
| CMD | PARAMETER | OPT PARAM | DESCRIPTION |
| 1 | Media Format | Media subformat | Returns 1 if the format/sub format combination is  supported. Media FormatC: 2 = Mpeg420, 3 =  Mpeg 422, SubformatC: 0=IBP, 1=IB, 2=I only. |

The PLMT? query returns the minimum and maximum limits for the CMD and PARM values

specified below.

**8.10.9 PARAMETERS SUPPORTED QUERY**

**Table 50.  PLMT Parameters**

**BINARY: <3X.74><CMD><PARM1><PARM2>**

**returns <OK>**

**ASCII: PSUP? <CMD>, <PARM1>, <PARM2>**

**returns <OK>**

CMD contains the command type to be queried.

PARM1 contains the first parameter associated with the command.

PARM2 contains the second parameter associated with the command.

OK returns 1 if the combination is supported and 0 otherwise.

The PSUP? query returns indication whether the combination of the parameters specified is

supported by the system.

**Table 51. PSUP Parameters**

**8.10.10**

**SYSTEM STATUS QUERY**

**BINARY: <3X.10><BITMAP>**

**short option returns <rBITMAP>**

**[<rSTAT1 TC:2>][<rSTAT2 U:2>][<rSTAT3>][<rSTAT4 I:2>][<rSTAT5 TC>][<rSTAT6>]**

**extended option returns <rBITMAP>**

**[<rSTAT1 B:12>][<rSTAT2 U:4>][<rSTAT3>][<rSTAT4 I:2>][<rSTAT5 TC>][<rSTAT6>]**

**ASCII: SYSS? <BITMAP>**

**short option returns <rLEN>,<rBITMAP>**

**[,<rSTAT1 B:4:,>][,<rSTAT2 U:2:,>][,<rSTAT3>][,<rSTAT4 I:2:,>]**

**[,<rSTAT5 B:4:,>][,<rSTAT6>]<,+0 I:18-<rLEN>>**

**extended option returns <rLEN>,<rBITMAP>**

**[,<rSTAT1 B:6:,>][,<rSTAT2 U:4:,>][,<rSTAT3>][,<rSTAT4 I:2:,>]**

**[,<rSTAT5 B:4>][,<rSTAT6><,+0 I:24-<rLEN>>**

BITMAP contains a bitmap specifying which system status values should be returned.

rLEN (ASCII only) returns the byte count of the rBITMAP plus the rSTAT bytes).

rBITMAP returns the BITMAP for the items returned.

page 84

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Status 8 | EXT SS1+SS2 | Status 6 | Status 5 | Status 4 | Status 3 | Status 2 | Status 1 |

|  |  |
| --- | --- |
| Frames x 10 | Frames x 1 |
| Seconds x 10 | Seconds x 1 |
| Minutes x 10 | Minutes x 1 |
| Hours x 10 | Hours x 1 |

|  |  |
| --- | --- |
| Frames x 10 | Frames x 1 |
| Seconds x 10 | Seconds x 1 |
| Minutes x 10 | Minutes x 1 |
| Hours x 10 | Hours x 1 |

|  |  |
| --- | --- |
| Frames x 10 | Frames x 1 |
| Seconds x 10 | Seconds x 1 |
| Minutes x 10 | Minutes x 1 |
| Hours x 10 | Hours x 1 |
| Hours X1000 | Hours X100 |
| Hours X 100,000 | Hours X 10,000 |

rSTATX returns the status byte(s) specified below for status item X. Bit 7 of the BITMAP enables

extended status for System Status items 1 and 2.

The SYSS? query retuns current system status as specified in BITMAP.

The system status items available and their contents are shown in the tables below.

**Table 52.  System Status Query (SYSS?) Bitmap**

If the EXT SS1+SS2 bit is set true, then System Status 1 and System Status 2 will return the

extended version of their respective status bytes.

The Storage Time Remaining is an estimate of the time available for video storage. This does

not take into account disk fragmentation.

The Largest Available Chunk is an estimate of the largest file that could currently be recorded.

If the selected port is a video input port, this information is calculated using the current bitrate and

video standard settings for that port (i.e. more compression allows more storage). If the selected

port is an output port, the query will return the information based on the video standard setting for

that port and the maximum allowable system bitrate. For an unselected port, the information is

computed based upon NTSC and the maximum allowable system bitrate. Both values are

returned in binary coded decimal. For example, 53 seconds would be represented as a 5 in the

Seconds x10 nibble and a 3 in the Seconds x1 nibble, or 53 Hex.

**Table 53. System Status 1 - Storage Time Remaining – Short Option**

Total Time Remaining

Largest Available Chunk

**Table 54. System Status 1 - Storage Time Remaining – Extended Option**

Total Time Remaining

Command States

page 85

|  |  |
| --- | --- |
| Frames x 10 | Frames x 1 |
| Seconds x 10 | Seconds x 1 |
| Minutes x 10 | Minutes x 1 |
| Hours x 10 | Hours x 1 |
| Hours X1000 | Hours X100 |
| Hours X 100,000 | Hours X 10,000 |

|  |
| --- |
| Number of ids stored – most significant bye |
| Number of ids stored – least significant byte |

|  |
| --- |
| Number of ids stored – most significant bye |
| Number of ids stored – Byte 2 |
| Number of ids stored – Byte 3 |
| Number of ids stored – least significant byte |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | DISK  NEAR  FULL |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | PEER  AVAIL-  ABLE |
|  |  |  |  | SYSTEM  ARCHIVE  FULL | LOCAL  ARCHIVE  FULL | SYSTEM  ARCHIVE  AVAIL-  ABLE | LOCAL  ARCHIVE  AVAIL-  ABLE |

|  |  |
| --- | --- |
| Frames x 10 | Frames x 1 |
| Seconds x 10 | Seconds x 1 |
| Minutes x 10 | Minutes x 1 |
| Hours x 10 | Hours x 1 |

Largest Available Chunk

The Disk Near Full flag will be set when the percentage of free space on the disk reaches the

**Table 55.  System Status 2 - Number of Ids stored – Short Option**

**Table 56.  System Status 2 - Number of Ids stored – Extended Option**

value set by the Minimum Free Space command.

The standard time will always be 0:0:0.0.

**Table 57.  System Status 3 - Disk Status**

**Table 58.  System Status 4 - Subsystem Status**

page 86

**Table 59.  System Status 5- Standard Time**

Command States

**Table 60. System Status 6- Minimum Free Space Percentage**

BYTE REPRESENTING MINIMUM FREE SPACE PERCENTAGE

See the MFRE command for details of the Minimum Free Space Percentage function.

Command States

page 87

**8.11 COMPOSITE-TO-D1 UTILITIES**

**8.11.1 COMPOSITE-TO-D1 BOARD CONFIGURATION**

**ASCII: C2DC <BOARD>, <SECTION>, "<CONFIGNAME C:N>"**

**ASCII: C2DC? <BOARD>, <SECTION>**

**returns “<CONFIGNAME C:N>”**

BOARD contains the Composite-to-D1 board number. This board number is specified in the bvsrc

configuration file.

SECTION contains the section(0 or 1) of a given board.

CONFIGNAME contains the configuration string.

The C2DC command sets the configuration name for the specified board and section.

When the name is "COMP2D1\_NTSC" the section is configured to support NTSC. When the

name is "COMP2D1\_PAL" the section is configured to support PAL. When the name is

"COMP2D1\_NTSCJ" the section is configured to support NTSC Japan.

The C2DC? query returns the configuration name for the specified board and section.

**8.11.2 AUDIO GAIN**

**ASCII: AUDG <BOARD>, <SECTION>, <CHAN>, <L/R>, <INCREMENT>**

**ASCII: AUDG? <BOARD>,<SECTION>, <CHAN>,<L/R>**

**returns <INCREMENT>**

BOARD contains the Composite-to-D1 board number. This board number is specified in the

bvsrc configuration file.

SECTION contains the section (0 or 1) of the board.

CHAN contains the channel (0 or 1).

L/R contains the speaker (0 for left, 1 for right).

INCREMENT contains the desired modification. A positive integer indicates the number of 0.5 dB

steps to increase the specified audio level. A negative integer indicates the number of 0.5 dB

steps to decrease the specified audio level.

The AUDG command sets the audio level modification of the specified audio source.

The AUDG? query returns the audio level modification currently set for the specified audio

source.

page 88

Command States

**8.12 VIDEO FILE SYSTEM CHECK**

**8.12.1 CHECK VIDEO FILE SYSTEM**

**FIXED NAME:**

**ASCII: CVFS**

**ASCII: CVFS?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:8>” :10>**

**VARIABLE NAME:**

**ASCII: LVFS**

**ASCII: LVFS?**

**returns <IDS\_REMAINING I:2:,><,”<ID C:32+>” :10>**

**IMPLEMENTED: 300 / 700 / 1600**

**IDS\_REMAINING** contains the number of ids remaining.

**ID** returns the name of the ID.

The CVFS command starts the check video file system process. The busy bit is set until the

operation completes.

The CVF? query returns a list of the damaged video files found by the last check video file system

command.

**8.12.2 VIDEO FILE SYSTEM CHECK IN PROGRESS**

**ASCII: CVIP?**

**returns <STAT>**

**IMPLEMENTED: 300 / 700 / 1600**

STAT returns the current status of the check video files operaion.

The CVIP? query returns a 1 if a check video files is in progress and a 0 if no file system check is

in progress.

**8.12.3 NUMBER OF DAMAGED VIDEO FILES**

**ASCII: NDVF?**

**returns <COUNT>**

**IMPLEMENTED: 300 / 700 / 1600**

**COUNT returns the number of damaged files.**

The NDVF? query returns the number of damaged video files found by the last check video file

system command.

Command States

page 89

|  |  |
| --- | --- |
| STATE | DESCRIPTION |
| 0 | Clean. Normal mode. |
| 1 | PLAY ONLY. The check video files process must be run. |

**8.12.4 VIDEO FILE FIX**

**ASCII: VFIX "<ID C:32+>"**

**IMPLEMENTED: 300 / 700 / 1600**

**ID containes the name of the ID.**

The VFIX command attempts a fix of the specified damaged video file.

**8.12.5 FINISH CHECK VIDEO FILES**

**ASCII: FCVF**

**IMPLEMENTED: 300 / 700 / 1600**

The FCVF command removes the old results file from a check video files.

**8.12.6 VIDEO FILESYSTEM PLAY ONLY**

**ASCII: VFPO?**

**return <PLAY\_ONLY>**

**IMPLEMENTED: 300 / 700 / 1600**

PLAY\_ONLY returns the current play out status.

The VFPO? query returns the status of the video file system.

**8.12.7 ABORT CHECK VIDEO FILES**

**Table 61.  Video File System Play Only States**

**ASCII: ACVF**

The ACVF command aborts any currently running check video file system operation.

This command is not implemented.

page 90

Command States

**8.13 MISCELLANEOUS UTILITIES**

**8.13.1 OPERATION MODE**

**ASCII: OMOD <MODE>**

**ASCII: OMOD?**

**returns <MODE>**

NOTE: The system must be rebooted after the OMOD command is executed. If any other

command is received prior to reboot, the OMOD will be reset to the current value.

MODE contains the value for the operation mode.

The OMOD command sets the operation mode. Valid modes are PEER (0), MAIN (1) or BUFFER

(2). Currently the mode must be changed and then the system re-started in order for the change

to take effect. In MAIN OMOD, the unit does not initiate any communication with other units in

the system and cannot execute either the MARK or COPY commands. This mode is usually used

for the main content server in a main multiple buffer system configuration. In BUFFER OMOD,

the unit will periodically ping the system specified as the main in the configuration file and can

transfer content using the mark command without the user having to specify where the content is

located. If ripple through delete is enabled (see RDEL), an item deleted on the main will be

deleted on the buffer as well. If ripple through delete is not enabled, a delete on the main will

have no effect on the buffer contents. In PEER OMOD, used in a peer to peer cluster

configuration, the unit does not ping other units in the cluster, but can transfer items using the

COPY command. This is a more flexible protocol, but it requires the user to have more

knowledge of where in the system content resides.

The OMOD? query returns the current operation mode.

**8.13.2 PLAY WHILE TRANSFER MODE**

**ASCII: PXFR <MODE>**

**ASCII: PXFR?**

**returns <MODE>**

MODE specifies the value for the play while transfer mode.

The PXFR command sets the play-while-transfer mode of the server.

Play while transfer mode affects the ability to play an Id while it is being transferred to the playing

system. It does not affect the ability to play while recording an ID. Mode = 0 disables play while

transfer, Mode = 1 enables it.

The PXFR? query returns the play-while-transfer mode of the server.

**8.13.3 RIPPLE THROUGH DELETE MODE**

**ASCII: RDEL <MODE>**

Command States

page 91

**ASCII: RDEL?**

**returns <MODE>**

**IMPLEMENTED: 300 / 700 / 1600**

Mode specifies the ripple through delete operation mode.

The RDEL command sets the ripple delete mode for the server.

In OMOD BUFFER if an item is deleted from the system designated as the main, the delete can

be allowed to "ripple through" to the buffer, resulting in the same item being deleted from the

buffer system. Mode = 0 disables ripple through delete, Mode = 1 enables it.

The RDEL? query returns the ripple delete mode for the server.

**8.13.4 NODE LIST**

**ASCII: NLST?**

**returns <REM>,<N>,<”<ID C:8+>”,<IDX> :10,>**

REM returns the number of nodes still to be requested.

N returns the number of NODE/IDX pairs returned.

NODE returns the server name of the first node.

IDX returns the node index for NODE.

The NLST? query returns a list of the nodes in the local cluster sorted by IDX.

The server creates a node list by alphabetizing the BVS\_CPU entries in the bvsrc file and

assigning a node number (starting with zero) to each node. A maximum of 10 nodes will be

returned. If REM is non-zero, a subsequent NLST? query on the selected port will return more

NODE/IDX pairs.

**8.13.5 LONG DISK SIZE QUERY**

**ASCII: LSIZ?**

**returns <TOTAL I:2:,>,<UNUSED I:2:,>,<LARGEST I:2:,>**

TOTAL returns the total amount of disk space in KB.

UNUSED returns the amount of unused disk space in KB.

LARGEST returns the size of the largest chunk of unused space in KB.

The LSIZ? query returns the total system disk space, the total unused disk space and the largest

chunk of available disk space in Kbytes. Each value is a 64 bit integer and is returned as two

double word(32 bit) numbers. The first one is most significant double word and second one is

least significant double word.

**8.13.6 DECODER RESET**

**ASCII: DRST <PORT>, <FLAG>**

PORT contains the output port number.

FLAG contains the constant 7.

page 92

Command States

The DRST command resets the decoder specified by PORT to its start-up state. Used in the

case where an unrecoverable decoder error is encountered.

Due to the nature of the operations required to perform a decoder reset, other ports on the

system can be effected. The decoder reset should only be used as an extreme measure and

when the rest of the server is IDLE.

**8.13.7 ADD TO DB**

**ASCII: ADDB** “**<ID C:32+>”**

**IMPLEMENTED: 300 / 700 / 1600**

ID contains an ID name.

The ADDB command adds the specified ID to the internal database. This command is used by

Connect and other Avid utilities. It is not available for 3rd party use.

**8.13.8 START IMPORT**

**ASCII: SIMP?** “**<ID C:32+>”**

**returns <OK>**

ID contains an ID name.

OK returns the completion status.

The SIMP? query returns the status from starting an import on the specified ID. This command is

used by Connect and other Avid utilities. It is not available for 3rd party use.

Valid return calues are 0 on success, and a non-zero error code on failure. Currently, all failures

return the value 1.

**8.13.9 MODIFY IMPORT**

**ASCII: MIMP?** “**<ID C:32+>”,<action>,<value>**

**returns <OK>**

ID contains an ID name.

ACTION contains an integer specifying the action to be taken.

VALUE is a parameter associated with ACTION.

OK returns the completion status.

The MIMP? query returns the status from odifying the import state of the specified ID. This

command is used by Connect and other Avid utilities. It is not available for 3rd party use.

Supported values for ACTION are as listed in the table below.

**Table 62. MIMP Parameters**

Command States

page 93

|  |  |  |  |
| --- | --- | --- | --- |
| ACTION  VALUE | ACTION  MEANING | VALUE | DESCRIPTION |
| 1 | Update  Header | Expected size of  file in KB. | After the header track is transferred, this  command is executed to allow AMS to update its  DB. |
| 2 | Transfer  Done | N/A | Executed when all tracks have completed  tranferring. This command is a synonym for  FIMP? “<ID>”,0. |
| 3 | Play OK | N/A | Executed when file is to be marked PLAY-OK. |
| 4 | Update  Space | N/A | Executed to have AMS update its space tracking  with the actual size of the file. |

Retuns 0 on success, and a non-zero error code on failure. Currently, all failures return the value

1.

**8.13.10**

**FINISH IMPORT**

**ASCII: FIMP?** “**<ID C:32+>”,<STATUS>**

**returns OK**

ID containts an ID name.

STATUS contains the import status.

OK returns the completion status.

The FIMP? query returns the status from finishing the current inport on the specified ID. This

command is used by Connect and other Avid utilities. It is not available for 3rd party use.

If STATUS is 0, the ID is closed as a good file. If STATUS is non-zero, the ID is removed from

the AMS DB, but not from VFS.

Retuns 0 on success, and a non-zero error code on failure. Currently, all failures return the value

1.

**8.14 Expiry Database**

**8.14.1 EXPIRY DB RESTORE**

**ASCII: ERST**

**IMPLEMENTED: 300 / 700 / 1600**

The ERST command rebuilds the expiry database.

**8.14.2 AUTO DELETE MODE**

**BINARY: <2X.7C><MODE>**

**ASCII: AUTD <MODE>**

page 94

Command States

**IMPLEMENTED: 300 / 700 / 1600**

<MODE> contains the auto delete mode. A 0 sets the mode off and a 1 sets the mode on.

The AUTD command sets the autodelete mode.

**BINARY: <3X.6D>**

**returns <MODE>**

**ASCII: AUTD?**

**returns <MODE>**

**IMPLEMENTED: 300 / 700 / 1600**

<MODE> returns the auto delete mode.

The AUTD? query returns the autodelete mode.

**8.14.3 AUTO DELETE TIME**

**BINARY: <2X.7D><DAY\_COUNT><HOURS><MINUTES><SECONDS>**

**ASCII: ATIM <DAY\_COUNT>,<HOURS>,<MINUTES>,<SECONDS>**

**IMPLEMENTED: 300 / 700 / 1600**

<DAY\_COUNT> contains the interval (in days) bwtween auto deletion processes.

<HOURS> contains the hour of the auto deletion process.

<MINUTES> contains the hour of the auto deletion process.

<SECONDS> contains the second of the auto deletion process.

The ATIM command sets the autodelete time and interval.

**BINARY: <3X.6E>**

**returns <DAY\_COUNT><HOURS><MINUTES><SECONDS>**

**ASCII: ATIM?**

**returns <DAY\_COUNT>,<HOURS>,<MINUTES>,<SECONDS>**

**IMPLEMENTED: 300 / 700 / 1600**

The ATIM? query returns the autodelete time and interval.

**8.14.4 EXPIRATION DATE**

**VARIABLE NAME BINARY: <AX.7B><LEN><ID C:32+><DAY><MONTH><YEAR I:2>**

**FIXED NAME BINARY: <2X.7B><ID C:8><DAY><MONTH><YEAR I:2>**

**ASCII: EDAT “<ID C:32+>”,<DAY>,<MONTH>,<YEAR>**

**IMPLEMENTED: 300 / 700 / 1600**

The EDAT command sets the expiration date for the specified ID.

**VARIABLE NAME BINARY: <BX.6C><LEN><ID C:32+>**

**returns <DAY><MONTH><YEAR I:2>**

Command States

page 95

**FIXED NAME BINARY: <3X.6C><ID C:8>**

**returns <DAY><MONTH><YEAR I:2>**

**ASCII: EDAT? “<ID C:32+>”,<DAY>,<MONTH>,<YEAR>**

**returns <DAY>,<MONTH>,<YEAR>**

**IMPLEMENTED: 300 / 700 / 1600**

The EDAT? query returns the expiration date for the specified ID.

**8.14.5 DELETE EXPIRED FILES**

**BINARY: <2X.7E>**

**ASCII: EDEL**

**IMPLEMENTED: 300 / 700 / 1600**

The EDEL command deletes the expired files.

page 96

Command States

**8.15 MPEG I/O Boards**

**8.15.1 ENCODER NAME**

**ASCII: ENCN <NAME C:31+>**

**ASCII: ENCN?**

**returns <NAME C:31+>**

<NAME> is a string not exceeding 31 characters in length.

The ENCN command sets the name of the external encoder used by the MPEG IN board to

<NAME>, for the currently selected MPEG IN port.

The ENCN? query returns the external encoder name for the currently selected input port.

**8.15.2 TRANSPORT STREAM NUMBER OF FILES**

**ASCII: TSNF?**

**returns <NUM>**

The TSNF? query returns the number of programs in the incoming transport stream.

**8.15.3 PROGRAM ASSOCIATION TABLE**

**ASCII: PAT? <IDX>**

**returns <PROG>,<PID>**

IDX contains the index number in the PAT.

PROG returns the program ID for the selected index. If zero, there is no program in the selected

slot.

PID returns the PID of the program’s PMT (Program Map Table).

The PAT? query returns information about the program in slot IDX of the PAT (Program

Association Table).

**8.15.4 PROGRAM INFORMATION QUERY**

**ASCII: PGNF? <ID>**

**returns <PID>, <STREAMS>, <PCR>**

ID contains a program ID.

PID returns the real program ID for the specified program. This is useful when ID is zero.

STREAMS returns the number of elementary streams in the program.

PCR returns the PID of the PCR stream for the specified program.

The PGNF? query returns information about the program with program ID of <ID>. If <ID> is

zero, returns information about the program to be ingested if a recording was started now.

Command States

page 97

|  |  |
| --- | --- |
| **Value** | **Stream Type** |
| 1 | MPEG-1 video |
| 2 | MPEG-2 video |
| 3 | MPEG-1 (Musicam) audio |
| 4 | mpeg-2 audio |
| 5-6 | Private Data |
| 13 | Dolby E |
| 128 | User Private |
| 129 | AC-3 audio |
| 130-255 | User Private |

**8.15.5 PROGRAM STREAM INFORMATION QUERY**

**ASCII: PMT? <ID>, <IDX>**

**returns <PID>, <STYPE>, <BITRATE>, <LANG\_NUM>, <LANG>**

ID contains the program.

IDX contains the stream number.

PID returns the PID of the specified stream. If zero, the stream or program does not exist.

STYPE returns the stream type.

BITRATE returns the bitrate, if known.

LANG\_NUM returns the numeric language code, if it is known and the selected stream is an

audio stream.

LANG returns the language code string.

The PMT? query returns information about stream number IDX in the program selected by ID. If

ID is zero, the PMT? query returns information about the program to be ingested, if a recording

was to be started now. Currently, only ID zero is supported in this command.

The stream type is defined in ISO/IEC 13818-1, table 2-29. See the table “Stream Types” below

for some common values.

**8.15.6 MPEG OUT RATE**

**Table 63.  Stream Types**

**ASCII: MPOR <TYPE>,<VAL>**

**ASCII: MPOR? <TYPE>**

**returns <VAL>**

TYPE contains the value for rate. Valid values are 1 for PCR and 2 for PSI.

VALUE contains the rate in items per second (i.e. PCR samples per second).

The MPOR command sets the MPEG out rate for the selected input port.

The MPOR? querry returns the current setting for the specified type.

**8.15.7 MPEG OUT PACKET IDENTIFIER**

**ASCII: MPOP <IDX>,<PID>**

page 98

Command States

|  |  |
| --- | --- |
| **Value** | **Stream Type** |
| 0 | video |
| 1-4 | audio |
| 15 | data stream (not currently supported) |
| 16 | program map table |

**ASCII: MPOP? <IDX>**

**returns <PID>**

IDX contains the stream index.

PID contains an integer between 1 and 8191 (0x1fff).

The MPOP command sets the MPEG out packet identifier (PID) for the specified stream.

The MPOP? querry returns the current setting for the specified type.

The following table contains the valid stream indeces and their corresponding meaning.

**8.15.8 MPEG OUT LANGUAGE CODE**

**Table 64.  Stream Indeces for MPOP Command**

**ASCII: LCOD <STREAM>,<CODE>**

**ASCII: LCOD? <STREAM>**

**returns <CODE>**

STREAM contains one of the audio streams. Valid values are 1 through 4.

CODE contains a 3 character string representing the language. This string is converted to a 24-

bit integer and stored as the language code in the PMT.

The LCOD command sets the MPEG out language code for an audio stream.

The LCOD? query returns an integer containing the current language code for the specified audio

stream.

Command States

page 99

**9** **Cueing While Transferring or Recording**

The MediaStream server can be configured to play material that is currently being recorded or

transferred. Furthermore, the material can be played from the beginning, or from an offset into

the file. The following section outlines the requirements and constraints of the Media Stream

Server when implementing this functionality. The Play-While-Transfer functionality can be

configured through the MSI or with the PXFR command.

This section discusses both play while transferring, and play while recording. It does not cover

issuing trimming commands (JOG / SHTL) on a file that is transferring or recording. For

information on trimming while transferring or recording, refer to the section titled “Trimming while

Recording”.

**9.1**

**Requirements and Constraints**

In order to cue a file to a certain frame, the associated video data and frame table data must be

⎜ ⎜ (*BitRate* \*1.1) ⎟ ⎟

available for reading. The following expression guarantees that the server has the necessary

information to enable the material to be cued. If the expression is true, the file can be cued to the

desired MARK-IN. If the expression is false, the application must wait until the expression is true

before cueing to the desired MARK-IN. In most cases, the expression can be greatly simplified or

an alternate algorithm can be used. The expression is included here to explain the theory behind

the subsequent examples.

⎛ ⎛*CurrentSize* ⎞ ⎞

*FramesNeeded* < ⎜ ⎜ ⎟ −*Safety* ⎟ \**FrameRate*

⎜ ⎜ 8 \*1024 ⎟ ⎟

⎝ ⎝ ⎠ ⎠

Where FramesNeeded is the desired mark-in value in frames

CurrentSize is the current size of the file on the destination server in Kilobytes

BitRate is the bit rate of the file

FrameRate is the frame rate of the system, i.e. 25 (PAL) or 30 (NTSC)

Safety is 5 seconds for transferring, 120 seconds for recording

When FramesNeeded is 0, i.e., the file is cued with the MARK-IN equal to the SOC, the material

is ready to cue as soon as the ID\_IN\_DISK bit is set.

When material is transferred to the server, the frame table is transferred before the standard track

(video data). Therefore, frame table information will exist for the file before the associated video

data. When material is encoded (recorded) to the server, the standard track (video data) is

written in real time, but the frame table data is only updated approximately every 2 minutes.

Therefore, in the case of a recording file, the video data will exist before the associated frame

table data. A cue command requires the frame table information and video data to exist for the

desired mark-in.

When the mark-in value of a cue command is greater than the start-of-clip, the above algorithm

must be used to ensure sufficient information is available for the material to cue successfully.

The remainder of this section describes techniques, and provides examples, for implementing the

requirements and constraints for cuing while transferring or recording.

•

In the examples shown, it is assumed the port status command will be used to determine

port status information. Rather than confusing the examples by including every status

page 100

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | IDS ADDED TO  PEER / ARCH | IDS  DELETED | IDS ADDED |  |
|  |  |  |  |  |  |  | NO  TIMECODE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

transition, only the final status before the next command is generally displayed. Some



sequences may result in the PORT BUSY bit being set, for clarity this isn’t shown in the

examples.

The following BITMAP is used to show the state and flag status of the port status response:

In the examples shown, all commands are sent to the destination server (the server

•

where the file is to be played) unless otherwise stated.

**9.1.1 Cueing to SOC while Transferring**

When a piece of material is transferred from one server to another, the frame table is transferred

before the standard track (video data). Consequently, frame table information can exist for a file

before the actual video data has transferred.

The transfer happens ‘As Fast As Possible’ and is referred to as an AFAP transfer. The speed of

the transfer depends on the available bandwidth in the system, that is, the more bandwidth

available, the faster the transfer. Depending on system load, the transfer rate may vary

significantly. Consequently, one cannot simply wait a period of time before issuing the cue

command, this is in contrast to cue while recording.

Providing the material is cued with a mark-in equal to the start-of-clip, information in the IDRQ?

response indicates when the material is available to cue.

Suppose the file FILE0000 is to be copied from SERVER1 (node 0) to SERVER2 (node 1) and

renamed FILE0001.

**COPY 0,’FILE0000’,1,1,’FILE0001’**

Use the STAT? 2 command to check when new files appear on the server.

**STAT? 2**

Continue monitoring the IDS ADDED bit until it is set. When it is, issue an IADD? query and look

for the expected file, which in this case is FILE0001.

**IADD?**

If the expected file appears in the IADD list, the file can now be cued at the start-of-clip. If the

expected file does not appear, continue at the STAT? 2 command.

**PCUE ‘FILE0001’**

Observe the port status 1 response until the material is cued, i.e. the CUE DONE bit is set, or

until an error indicator appears in the port status 3 response.

Command States

page 101

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | IDS ADDED TO  PEER / ARCH | IDS  DELETED | IDS ADDED |  |
|  |  |  |  |  |  |  | NO  TIMECODE |

If the CUE DONE bit is set, the material may now be played.

⎜ ⎜ (*BitRate* \*1.1) ⎟ ⎟

**PLAY**

**9.1.2 Cueing to Arbitrary Mark-In while Transferring**

If the material is cued with a mark-in value greater than the start-of-clip, a more detailed algorithm

needs to be used. This is because the IN DISK bit only indicates there is sufficient material

available to play from the beginning of the file, i.e. the start-of-clip. To cue material with a mark-in

greater than the start-of-clip an application must ensure that sufficient data is available for the cue

to succeed. The IDBY? query is used to determine the current size of the destination file. Using

this value and the compression rate of the file, a calculation can be made to determine when

sufficient material is available for the cue to succeed.

The following expression must be used for determining when sufficient material is available.

⎛ ⎛*CurrentSize* ⎞ ⎞

*FramesNeeded* < ⎜ ⎜ ⎟ − 5⎟ \**FrameRate*

⎜ ⎜ 8 \*1024 ⎟ ⎟

⎝ ⎝ ⎠ ⎠

Where FramesNeeded is the desired mark-in value in frames

CurrentSize is the current size of the file on the destination server in Kilobytes

BitRate is the bit rate of the file

FrameRate is the frame rate of the system, i.e. 25 (PAL) or 30 (NTSC)

Once the expression is true the material can be cued to the desired mark-in value and played.

Suppose the file FILE0001 (length 5 minutes) is to be copied from SERVER1 to SERVER2 and

renamed FILE0002. Further suppose the bit rate is 8 Mb/s, the frame rate is 30 fps, and the

material is to be cued 1 minute into the file.

**COPY 0,’FILE0001’,1,1,’FILE0002’**

Use the STAT? 2 command to check when new files appear on the server.

**STAT? 2**

Continue monitoring the IDS ADDED bit until it is set. When it is, issue an IADD? query and look

for the expected file, which in this case is FILE0002.

**IADD?**

page 102

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

If the expected file appears in the IADD list, the file can now be cued at the start-of-clip. If the

⎜⎜ (8000000 \*1.1) ⎟ ⎟

expected file does not appear, continue at the STAT? 2 command.

Use the INFO? query to determine encoded bit rate of the file. Passing in a bitmap of 2 requests

the encoded bit rate of the file.

**INFO? 2, ‘FILE0002’**

Issue an IDBY? query to determine the current size of the destination file. Substitute the

CurrentSize, BiteRate, FrameRate and FramesNeeded values into the expression. If the

expression is true, the material can be cued. Otherwise, repeat this step as necessary.

**IDBY? ‘FILE0002’**

FramesNeeded = 60 \* 30 = 1800

BitRate = 8000000

FrameRate = 30

CurrentSize = 80000

⎛⎛ 80000 ⎞ ⎞

1800 < ⎜⎜ ⎟ − 5⎟ \*30

⎜⎜ 8 \*1024 ⎟ ⎟

⎝⎝ ⎠ ⎠

Once the equation is satisfied the material can be cued to the desired mark-in.

**PCWD ‘FILE0002’,0,0,1,0,0,0,0,0**

Observe the port status 1 response until the material is cued, i.e. the CUE DONE bit is set, or

until an error indicator appears in the port status 3 response.

If the CUE DONE bit is set, the material may now be played.

**PLAY**

**9.1.3 Cueing to SOC while Recording**

When a file is recording and the play-while-transfer feature in enabled, the file can be cued to the

start-of-clip when it appears in the IDS-ADDED list. Depending upon the bit rate used in the

recording, the file will typically appear in the list between 3 and 15 seconds into the recording. Be

sure to use the commands below to verify the availability of the file as these times may vary.

Use the STAT? 2 command to check when new files appear on the server.

**STAT? 2**

Command States

page 103

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | IDS ADDED TO  PEER / ARCH | IDS  DELETED | IDS ADDED |  |
|  |  |  |  |  |  |  | NO  TIMECODE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | IDS ADDED TO  PEER / ARCH | IDS  DELETED | IDS ADDED |  |
|  |  |  |  |  |  |  | NO  TIMECODE |

Continue monitoring the IDS ADDED bit until it is set. When it is, issue an IADD? query and look

for the expected file, which in this case is FILE0001.

**IADD?**

If the expected file appears in the IADD list, the file can now be cued at the start-of-clip. If the

expected file does not appear, continue at the STAT? 2 command.

**PCUE ‘FILE0001’**

Observe the port status 1 response until the material is cued, i.e. the CUE DONE bit is set

If the CUE DONE bit is set, the material may now be played.

**PLAY**

**9.1.4 Cueing to an Arbitrary Mark-In while Recording**

When material is encoded into the server the data-rate is constant, i.e., the recording process

happens in real time. As such, it is possible to time the issuance of a cue command to ensure

sufficient information is available. Frame table information, required when issuing a cue

command with a mark-in greater than the start-of-clip, is updated during the record process

approximately every 2 minutes. The following simple calculation can be used to determine when

the cue command can be issued.

Time currently encoded minus 2 minutes > desired mark-in

Suppose the file FILE0001 is currently being recorded, the following steps must be taken to

ensure it can be played.

Use the STAT? 2 command to check when new files appear on the server.

**STAT? 2**

Continue monitoring the IDS ADDED bit until it is set. When it is, issue an IADD? query and look

for the expected file, which in this case is FILE0001.

**IADD?**

page 104

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

If the expected file appears in the IADD list, the file can now be cued at the start-of-clip. If the

expected file does not appear, continue at the STAT? 2 command.

To determine if a particular mark-in can be cued, use the expression above. To obtain the time

currently encoded, send a GTTC? 1 command to the record port on which the file is recording.

Once the file has recorded 2 minutes past the desired mark-in, the file can be cued.

**PCUE ‘FILE0003’**

Observe the port status 1 response until the material is cued, i.e. the CUE DONE bit is set, or

until an error indicator appears in the port status 3 response.

If the CUE DONE bit is set, the material may now be played.

**PLAY**

**9.2**

**Pseudo Code for Cueing while Transferring or Recording**

The following pseudo-code further illustrates the process for determining when the material can

be cued. The algorithms used in this pseudo code can be used for cueing while transferring and

cueing while recording.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* setup for cue while record

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* cue while record needs 120 seconds for FT to catch up \*/

Safety = 120;

RCUE "file",duration

/\* check for CUE DONE or a failure. \*/

repeat {

usleep 200000;

IdState = IDRQ? “file”;

Errors = STAT? 4;

} until (( IdState & CUE\_DONE ) || Errors)

/\* if no errors, start recording \*/

if ( ! Errors )

RECD;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* end of setup for cue while record

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* setup for cue while transfer

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Command States

page 105

{COPY "file" or AGET "file"

/\* trim while transfer only needs 5 seconds \*/

SAFETY = 5;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* end of setup for cue while transfer

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* common code for determining whether to cue

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

InDisk = false;

/\* To determine when the file is available,

wait for the file to appear on the IDS-ADDED list \*/

Found = false;

Repeat {

NewIds = (stat? 2) & IDS\_ADDED;

if ( NewIds ) {

NewIdNames = IADD?;

Found = strstr( NewIdNames, “file” );

}

if ( ! Found )

usleep 200000; /\* this value limits the checks to 5 per second \*/

} until Found;

inDisk=true;

bitRate = INFO? 2,”file”;

idState = IDRQ? “file”;

inProgress = idState & XFER\_IN\_PROGRESS;

/\* if the record / transfer is NOT in progress, it has finished,

the file can be cued normally, so skip this section \*/

if ( inProgress ) {

/\* get the current size of the std track \*/

curSize, totalSize = IDBY? "file";

/\* use the expression to determine how many frames of video can

currently be cued \*/

framesOK = ( ((curSize / (bitRate \* 1.1)) / (8 \* 1024)) - Safety ) \* frameRate;

/\* negative value is meaningless \*/

framesOK = max ( 0, framesOK );

/\* convert the desired mark-in to frames \*/

frameNeeded = frames ( MarkIN );

/\* if there isn’t enough video to cue to the desired mark-in, recalculate

the available video 5 times per second. \*/

while ( framesNeeded > framesOK ) {

usleep 200000;

curSize, totalSize = IDBY? "file";

framesOK = ( (curSize / (bitRate \* 1.1) / (8 \* 1024)) – Safety )

\* frameRate;

framesOK = max ( 0, framesOK );

page 106

Command States

}

}

/\* the file can now be cue to the desired mark-in \*/

PCWD "file", MarkIN, Duration;

/\* check for CUE\_DONE or errors

repeat {

IdState = STAT? 1;

Errors = STAT? 4;

} until (( IdState & CUE\_DONE ) || Errors)

if ( ! Errors )

PLAY

Command States

page 107

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |
| --- | --- | --- | --- |
| Trimmable | content |  |  |

**10 Trimming while Recording**



The following section outlines the requirements and constraints of the Media Stream Server for

simultaneously trimming a video file that is still in the process of being recorded. It is assumed

that the developer is familiar with the basic trimming functions of the server, and the material

discussed in the section “Cueing while Transferring or Recording”.

Trimming is defined to be the procedure of navigating through a video file using the jog, shuttle

and pause commands.

**10.1 Amount of Video Available for Trimming**

The portion of video that can be trimmed while the record is in progress is defined aC:

**The duration of the record session at the time the ID is CUED minus 2 minutes**

For example, if the duration of a record is 20 minutes and the ID is cued 10 minutes into the

record, the first 8 minutes of the file can be trimmed.

minutes

0

8

minutes

20

10 minutes

cue issued

Once the file is cued, the amount of trimmable material will NOT increase as the record

continues. The file must be re-cued to have access to material recorded since the last cue. The

entire file can be trimmed only when the record has stopped and the file is re-cued.

The CONT command cannot be used to expand the trimmable content.

**10.2 Basic Trimming Commands**

The remainder of this section describes techniques, and provides examples, for implementing the

requirements and specifications for trim while recording.

•

•

The term ‘trimmable content’ refers to the section of currently recording file that may be

trimmed.

In the examples shown, it is assumed the port status command will be used to determine

port status information. Rather than confusing the examples by including every status

transition, only the final status before the next command is generally displayed. Some

sequences may result in the PORT BUSY bit being set, for clarity this isn’t shown in the

examples.

The following BITMAP is used to show the state and flag status of the port status response:

page 108

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

**10.2.1 Pause**



Both the jog and shuttle commands require a frame of video to be present on the decoder, i.e. the

decoder must be in a PLAY/RECORD, STILL, JOG, or SHUTTLE state. The PAUS command

can be used to bring up the first frame of active video once the decoder has been cued.

**10.2.2 Jog**

The jog command may be issued within the trimmable content. The frequency at which the jog

command is issued must be controlled by automation. Consecutive jog commands may only be

issued when the previous jog command has completed. Whilst the jog command is being

completed the JOG bit is set in the port status response, when the jog is complete the JOG bit is

cleared.

For example, assume a piece of material is cued on the decoder. The following commands will

result in the following status changes.

**PAUS**

**JOG**

(the JOG bit is set while the jog command is being completed)

.

If a JOG command is issued that will require a frame of video outside the trimmable content, the

.

first or last frame will be displayed (depending opon the direction of the JOG) and the STILL bit

will be set in the port status.

Any subsequent JOG commands requiring frames outside the trimmable content will be ignored

and will not be classified as errors.

**10.2.3 Shuttle**

The shuttle command may only be issued within the trimmable content. The shuttle command

may only be issued 3 times per second on a given port, i.e. the shuttle command may only be

issued every 333 milliseconds. While the decoder is shuttling the SHUTTLE bit will be set in the

PORT STATUS response.

For example, assume a piece of material is cued on the decoder.

**PAUS**

Command States

page 109

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |
| --- | --- | --- |
| Trimmable | content |  |

|  |  |
| --- | --- |
| Trimmable content |  |

**SHTL**



If a SHTL command is issued that will require a frame of video outside the trimmable content, the

first or last frame will be displayed (depending opon the direction of the JOG) and the STILL bit

will be set in the port status.

Any subsequent SHTL commands requiring frames outside the trimmable content will be ignored

and will not be classified as errors.

**10.3 Expanding the trimmable area**

To increase the duration of the file that can be trimmed while the file is being recorded the ID

should be STOPPED then re-cued.

Using the previous example, if the record has now been in progress for 15 minutes, the trimmable

portion may be updated by first issuing a STOP, waiting for the port status to return to IDLE, then

issuing another cue command.

minutes

0

8

minutes

20

15 minutes

0

Issue a STOP command,

Wait for the port to be IDLE,

Issue another cue command

minutes

13

minutes

20

Once the file has completed recording, the ID must be re-cued for the trimmable area to be

updated.

**10.4 Determining the end of the frame table**

page 110

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

The following section is only valid if the decoder is configured for freeze at EOM. If the decoder is



configured to go to black at EOM then the end of frame table cannot be determined, because the

decoder will stop decoding and go into an IDLE state.

If the end of the trimmable content is reached using the shuttle function the decoder will pause on

the last frame of the file that can be trimmed, the STILL bit in the port status will then be set.

Once this frame has been reached it is then only possible to jog/shuttle backwards into the

trimmable content.

An Application Developer can determine when the end of trimmable content has been reached

during shuttle because the STILL bit will be set in the decoder status. When this condition is

detected the application can issue a JOG 0 command to synch the decoder then a position

request command to locate the last frame of trimmable content. This is useful if a requirement of

the application is to provide continuity in shuttling. Once the necessary time requirement has

been met (record duration @ cue – 2 minutes), a STOP command can be sent, followed by port

status commands until the IDLE bit is set, followed by a PCWD command with a mark-in of the

previous last frame of trimmable content.

Using the previous example again.

1.

2.

3.

4.

Issue a RCUE command with a length of 20 minutes.

Using port status wait until the CUE DONE bit is set.

Issue the RECD command

Wait until 10 minutes has passed and issue a PCWD command, monitor the port status and

wait until the CUE DONE bit is set.

5. Issue a PAUSE command and monitor the port status until the STILL bit is set.

6. Issue a SHUTTLE command with a rate of 2x real time, monitor the port status until the

SHUTTLE bit is set.

7. Monitor the port status until the STILL bit is set, this will indicate the end of the frame table

has been reached.

8. Issue a JOG 0 command to synch the decoder.

9. Issue a GTTC command to find the current position within the file, store this position.

10. Issue a STOP command, observe the port status until the decoder returns to the IDLE state.

11. Issue a PCWD command with a mark-in value of the position noted previously.

12. Wait until the CUE DONE bit is set in the port status.

Command States

page 111

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT  BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |

Trimming can now continue within the new ‘trimmable’ content.

Determining the end of the trimmable content during the jog operation is slightly different. The

STILL bit, in this situation, does not indicate the end of the trimmable content since it is set every

time the jog command completes. By continuously monitoring the current position within the file

the Application Developer can determine the end of trimmable content. If the position within the

file does not change between two successive jog commands then the end of trimmable content

has been reached.

Using the previous example again.

1. Issue a RCUE command with a length of 20 minutes.

2. Using port status wait until the CUE DONE bit is set.

3. Issue the RECD command.

4. Wait until 10 minutes has passed and issue a PCWD command, monitor the port status and

wait until the CUE DONE bit is set.

5. Issue a PAUSE command and monitor the port status until the STILL bit is set.

6. Issue a JOG 1 command to advance forward 1 frame.

7. Wait until the JOG bit is cleared in the port status (the STILL bit will be set).

8. Issue a GTTC command to find the current position within the file, store this position.

9. Issue a JOG 1 command to advance forward 1 frame.

10. Wait until the JOG bit is cleared in the port status (the STILL bit will be set).

11. Issue a GTTC command to find the current position within the file, store this position.

12. Compare the position obtained in step 8 with the position obtained in step 11, if the values

are the same, they represent the position of the last trimmable frame.

page 112

Command States

**10.5 Decoder Configuration**

The decoder can be configured to either freeze on last frame or display black when either the End

of File (EOF) or End of Message (EOM) positions are reached. These terms have the following

meanings.

•

•

EOF is the end of file is the last frame of the file.

EOM is the end of message is the last frame of the duration specified in the PCWD

command.

The configuration parameters directly affect the behavior of the decoder when the last frame of

the trimmable portion of the file is reached. The following paragraphs describe the decoder’s

behavior in each of the possible configurations.

**Freeze on last frame at EOF**

After the last frame of the trimmable portion of the file is reached the decoder will pause, the last

frame will be shown, and the port status will have the STILL bit set. If a JOG, or SHUTTLE,

reverse command is issued the appropriate frame of video will be displayed.

**Black at EOF**

After the last frame of the trimmable portion of the file is reached the decoder will pause, black

will be shown, and the port status will have the STILL bit set. If a JOG, or SHUTTLE, reverse

command is issued the appropriate frame of video will be displayed.

**Freeze on last frame at EOM**

After the duration specified in the PCWD command has been reached the decoder will pause, the

last frame will be shown, and the port status will have the STILL bit set. If a JOG, or SHUTTLE,

reverse command is issued the appropriate frame of video will be displayed. Also, if more

trimmable file is available, a JOG or SHUTTLE forward will display the appropriate frame of video.

**Black at EOM**

After the duration specified in the PCWD command has been reached the decoder will have

finished decoding, the display will go to black, and the port status will be IDLE.

Although all configurations are supported, the trimming functions make more sense in certain

configurations than others. Due to the requirements of trimming while recording it is not sensible

to issue a PCWD with a non-zero duration, that is, issue a cue command with a finite duration.

Therefore, if one only issues an open ended cue, (PCWD with a zero duration), the settings for

EOM are irrelevant when trimming. Again, when trimming, it isn’t sensible to configure the

decoder for ‘Black at EOF’ because the decoder would display black when the EOF is reached.

Hence, for trimming operations, the recommended decoder configuration would be

**Freeze at EOM/EOF**

Command States

page 113

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

**11 Playing a Short Spot at the Beginning of a POD.**

Normally, the server cannot play files under 3 seconds in duration. However, a sub-3-second

spot can be played as the first of a group of files to play, i.e., at the beginning a POD. This is

accomplished by using the PCUE (or PCWD) command to cue up the short spot then, using the

PCU2 (or PCD2) command to cue up the following spot. Subsequent files are cued with the

PCUE (or PCWD) command and must have durations of 3 seconds or longer.

The following examples show the status transitions for playing a short spot at the beginning of a

POD. The basics of the algorithm for using this functionality are as follows.

1. Use the PCUE or PCWD command to cue the short spot.

2. Use the STAT? 5 command to get the error flags and state bits.

3. If there is an error, the short spot cannot be played, and the PCU2 / PCD2 cannot be issued.

4. Wait until the spot has cued, as indicated by the CUE\_DONE state bit.

5. Use the PCU2 or PCD2 command to cue the subsequent spot.

6. Use the STAT? 5 command to get the error flags and state bits.

7. If there is an error, the subsequent spot cannot be played.

8. Wait until the spot has cued, as indicated by the CUE\_DONE state bit.

9. The short spot can now be played with the PLAY command.

10. The play switch can now be triggered with the PLAY command.

**11.1 Case with no errors.**

/\* open and select the port \*/

**open? 1,1**

+1;1

**psel 1**

/\* cue up the short spot \*/

**pcue "shortspot"**

/\* the CUE bit is set while the file is cueing \*/

**stat? 15**

+8,+15,+2,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

**stat? 15**

+8,+15,+2,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* when the file is cued, the CUE\_DONE bit is set \*/

**stat? 15**

page 114

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

+8,+15,-126,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1



/\* cue up the spot after the short spot \*/

**pcd2 "longspot",0,1,0,0,0,10,0,0**

/\* when the pcu2 command is executed, the CUE\_DONE bit

is cleared and the CUE bit is set \*/

**stat? 15**

+8,+15,+2,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

**stat? 15**

+8,+15,+2,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

**stat? 15**

+8,+15,+2,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* when this spot is cued, the CUE\_DONE but is once again set. \*/

**stat? 15**

+8,+15,-126,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* play file shortspot \*/

**play**

/\* when the first file starts playing, the status bits are CUE\_DONE, PLAY,

and CUE. \*/

**stat? 15**

Command States

page 115

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

+8,+15,-122,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1



/\* play file longspot \*/

**play**

/\* when the second file starts playing, the status bits are PORT\_BUSY and

PLAY. \*/

**stat? 15**

+8,+15,+68,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* when the play switch completes, the only status bit set is PLAY. \*/

**stat? 15**

+8,+15,-4,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* close the port, when done. \*/

**clos 1**

page 116

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

**11.2 Case with errors on the first cue.**



/\* open and select the port \*/

**open? 1,1**

+1;1

**psel 1**

/\* cue up a spot that doesn’t exist. \*/

**pcue "junkspot"**

/\* the state byte stays IDLE and the error bit ID\_NOT\_FOUND is set. \*/

**stat? 15**

+8,+15,+1,+1,+0,+0,+2,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* At this point, since the port is not CUE\_DONE, the PCU2 command will

always fail. \*/

**pcu2 "longspot"**

**stat? 15**

+8,+15,+1,+1,+0,+0,+0,+1,+16,+0,+0,+0,+0,+0,+0;1

/\* close the port, when done. \*/

**clos 1**

**11.3 Case with errors on the second cue.**

This case demonstrates what happens when the PCU2 (or PCD2) command fails.

Command States

page 117

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

/\* open and select the port \*/



**open? 1,1**

+1;1

**psel 1**

/\* cue up the short spot \*/

**pcue "shortspot"**

/\* the CUE bit is set while the file is cueing \*/

**stat? 15**

+8,+15,+2,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* when the file is cued, the CUE\_DONE bit is set \*/

**stat? 15**

+8,+15,-126,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* cue up a non-existent spot \*/

**pcd2 "junkspot"**

/\* the error flag ID\_NOT\_FOUND is set and the state byte does not change. \*/

**stat? 15**

+8,+15,-126,+1,+0,+0,+2,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* play file shortspot \*/

**play**

page 118

Command States

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUE DONE | PORT BUSY | SHUTTLE | JOG | STILL | PLAY/REC | CUE | IDLE |
| NOT  SUPPORTED | CMD WHILE  BUSY | NOT  ENOUGH  DISK  SPACE | PORT  LOCKED | WRONG  PORT TYPE | INVALID  PORT | ILLEGAL  VALUE | SYSTEM  ERROR |
| ID DELETE  PROTECTED | XFER  COMPLETE | XFER  FAILED | ID CUED  OR  PLAYING | ID STILL  RECORDING | ID  ALREADY  EXISTS | ID NOT  FOUND | INVALID ID |
| SYSTEM  REBOOT |  | OPERATI  ON  FAILED | PORT  IDLE | PORT  ACTIVE | PORT NOT  IDLE | CUE NOT  DONE | NOT IN  CUE  STATE |

/\* when the first file starts playing, all state bits are cleared except PLAY. \*/



**stat? 15**

+8,+15,+4,+1,+0,+0,+0,+0,+16,+0,+0,+0,+0,+0,+0;1

/\* play file longspot \*/

**play**

/\* Since the second spot did not CUE, the play commands gets the error flag

ID\_NOT\_CUED. \*/

**stat? 15**

+8,+15,+4,+1,+0,+0,+0,+1,+16,+0,+0,+0,+0,+0,+0;1

/\* close the port, when done. \*/

**clos 1**

Command States

page 119

**12 Index of Commands**

ACUE ...................  77

ACUE?.................. 77

ACVF.................... 90

ADDB................... 93

AGET.................... 46

ALVL?.................. 70

ATIM....................  95

ATIM? ..................  95

AUDF ...................  69

AUDF?.................. 70

AUDG................... 88

AUDG?.................  88

AUTD ...................  94

AUTD? .................  94

BCLR.................... 80

C2DC ....................  88

C2DC? ..................  88

CLOS....................  35

CONT ...................  31

COPY.................... 47

CREF ....................  78

CSET?................... 71

CTER....................  48

CVF ......................  89

CVF?..................... 89

CVIP? ...................  89

DAGN................... 79

DAGN?.................  79

DCFG.................... 76

DCFG?.................. 77

DCLR.................... 50

DCMD ..................  80

DRFR.................... 67

DRFR? ..................  68

DRST ....................  92

DSET? ..................  78

DSIZ?.................... 83

DTYP? ..................  82

EAUD ...................  75

EAUD? .................  75

ECFG ....................  73

EDAT ....................95

EDAT? ..................95

EDEL.....................96

ENAS ....................68

ENAS?...................68

ENCN ....................97

ENCN? ..................97

ENVT ....................72

ENVT? ..................72

ERST .....................94

FCVF.....................90

FIMP?....................94

GLOC ....................76

GLOC? ..................76

GTFR?...................39

GTTC?...................38

HSYN ....................74

HSYN? ..................75

IADD? ...................54

IATM?...................53

IDAC? ...................38

IDBY? ...................61

IDEL......................49

IDLT? ....................55

IDPR......................50

IDQY? ...................58

IDRN .....................48

IDRQ? ...................57

IDUP......................50

INFO?....................59

JOG .......................32

LADD? ..................54

LAST? ...................37

LATM?..................53

LCOD ....................99

LCOD? ..................99

LCPY.....................47

LDLT?...................55

LIDA?....................38

LIST?.....................51

LLST?....................51

LNXT? ..................  52

LSIZ? ....................  92

MARK................... 46

MFRE.................... 83

MIMP? ..................  93

MPOP.................... 98

MPOP?..................  98

MPOR ...................  98

MPOR? .................  98

MPRM................... 69

MPRM?................. 69

NCPY.................... 48

NDVF?..................  89

NEXT? ..................  52

NLST?................... 92

OMOD ..................  91

OMOD?................. 91

OPEN? ..................  35

PAT? .....................  97

PAUS ....................  30

PCD2..................... 43

PCU2..................... 42

PCUE ....................  42

PCWD ...................  43

PCWF.................... 44

PGID .....................  67

PGID? ...................  67

PGNF?................... 97

PLAY ....................  29

PLMT? ..................  83

PMT?..................... 98

PRER..................... 72

PRER?................... 72

PRST .....................  65

PSEL .....................  35

PSUP? ...................  84

PTCL..................... 82

PTCL? ...................  82

PXFR..................... 91

PXFR?................... 91

RCUE.................... 40

RCWD ..................  40

RDEL.................... 91

RDEL? ..................  91

RECD.................... 28

SADJ..................... 74

SADJ?...................  74

SBIT...................... 65

SBIT?.................... 65

SGPI...................... 74

SHTL ....................  32

SIMP? ...................  93

SIZE? ....................  57

SREV? ..................  82

SSOC ....................  49

STAT? ..................  36

STEP..................... 31

STOP ....................  28

SYSS?...................  84

TCUE.................... 40

TCWD ..................  41

TSNF?................... 97

UADD................... 63

UCLS ....................  62

UCMD ..................  79

UDEL.................... 63

ULST? ..................  64

UMOD ..................  63

UOPN ...................  62

UQRY? .................  64

VADD? .................  54

VATM?................. 53

VDLT?.................. 55

VFIX..................... 90

VFPO? ..................  90

VIDA? ..................  38

VILI ......................  66

VILI? ....................  67

VLST? ..................  51

VNXT? .................  52

VSTD.................... 65

VSTD? ..................  66

page 120

Command States