MASSACHUSETTS INSTITUTE OF TECHNOLOGY

HAYSTACK OBSERVATORY

WESTFORD, MASSACHUSETTS 01886

Telephone: *978-692-4764*
Fax: *781-981-0590*

13 November 2006

TO: Distribution
FROM: A. R. Whitney and J. A. Ball
SUBJECT: Mark 5A command set (Revision 2.73)

1. Mark5A program

The commands detailed in this memo are implemented by a program named `Mark5A` running under Red Hat Linux on the Mark 5A system. The details concerning the operation of `Mark5A` are available in documents at http://web.haystack.mit.edu/mark5/Mark5.htm.

Note: The command-line options for Mark 5A revisions 2.5 and later are different than earlier versions. In particular, the startup command-line for *Mark5A* has been updated to be in accord with Linux conventions, as follows:

Mark5A –m [-1|0|1|2|3] *–f* [0|1] *–s* [1|2|3|4|5|6|7] *–d* [0|1] *-h* (defaults underlined)
where

m – message level (range –1 to 3, default 1)
- -1 A vast quantity of debug
- 0 Some debug
- 1 Normal operation; warnings and errors
- 2 Only errors and operational messages
- 3 Only fatal errors when the program dies

f – parsing mode (0 – 'informal' parsing; 1 – 'formal' parsing, i.e. VSI-S syntax; default 1)

s – maximum number of allowed socket connections (range 1 to 7; default 7)

d – operate in special 'disk-FIFO' mode (0 - off; 1 – on); default 0. See Section 7

h – help on startup parameters; other options are ignored if –h is present

2. Notes on command set

The following should be noted with respect to the command set:

1. All commands/queries are implemented using the VSI-S communications protocol and command/response syntax.
2. Commands/queries are case *insensitive*.
3. Versions of program 'Mark5A' with a revision date earlier than the date on this memo may not implement all commands indicated in this memo or, in some cases, may implement them in a different way (use 'DTS_id' query to get revision date of current system software – see 'System Queries and Responses').

3. VSI-S Command, Query and Response Syntax

The following explanation of the VSI-S syntax may be useful in understanding the structure of commands, queries and their respective responses. This explanation has been lifted directly from the VSI-S specification.

3.1 *Command Syntax*

Commands cause the system to take some action and are of the form

<keyword> = <field 1> : <field 2> : …. ;

where <keyword> is a VSI-S command keyword. The number of fields may either be fixed or indefinite; fields are separated by colons and terminated with a semi-colon. A field may be of type decimal integer, decimal real, integer hex, character, literal ASCII or a VSI-format time code. White space between tokens in the command line is ignored, however most character fields disallow embedded white space.

3.2 *Command-Response Syntax*

Each command elicits a response of the form

!<keyword> = < return code > [:<DTS-specific return> :….] ;

where

<keyword> is the command keyword

<return code> is an ASCII integer as follows:

0 - action successfully completed
1 - action initiated or enabled, but not completed
2 - command not implemented or not relevant to this DTS
3 - syntax error
4 - error encountered during attempt to execute
5 - currently too busy to service request; try again later
6 - inconsistent or conflicting request[1]
7 - no such keyword
8 - parameter error

<DTS-specific return> - one or more optional fields specific to the particular DTS, following the standard fields defined by VSI-S; fields may be of any type, but should be informative about the details of the action or error.

3.3 *Query and Query-Response Syntax*

Queries return information about the system and are of the form

<keyword> ? <field 1> : <field 2> : …. ;

with a response of the form

!<keyword> ? <field 1(return code)> : <field 2> : <field 3> : …: [<DTS-specific return>];

where

<return code> is an ASCII integer as follows:

0 - query successfully completed
1 - action initiated or enabled, but not completed
2 - query not implemented or not relevant to this DTS
3 - syntax error

[1] For example, it is illegal to attempt to record during playback or position unloaded media.

4 - error encountered during attempt to execute query
5 - currently too busy to service request; try again later
6 - inconsistent or conflicting request
7 - no such keyword
8 - parameter error
9 - indeterminate state

Note: *A 'blank' in a returned query field indicates the value of the parameter is unknown.*
A '?' in a returned query field indicates that not only is the parameter unknown, but that some sort of error condition likely exists.

4. Simplified Diagrams of Various Mark 5 Data Transfer Modes

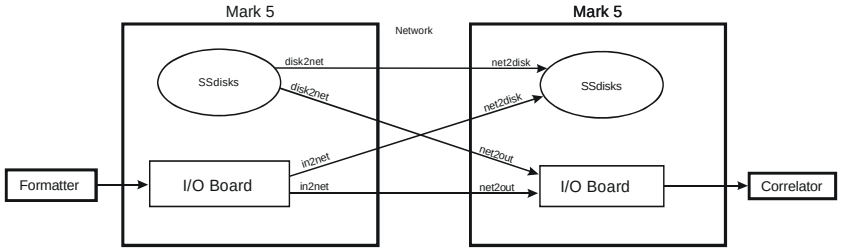

Figure 1: Mark 5 to Mark 5 transfer through network

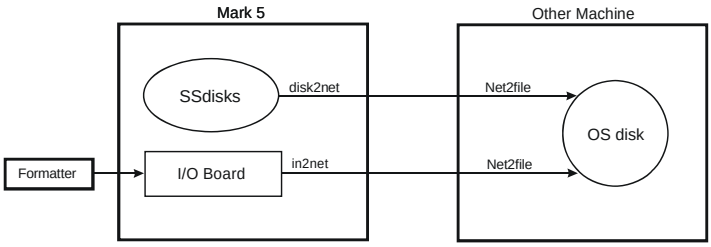

Figure 2: Mark 5 to file transfer through network

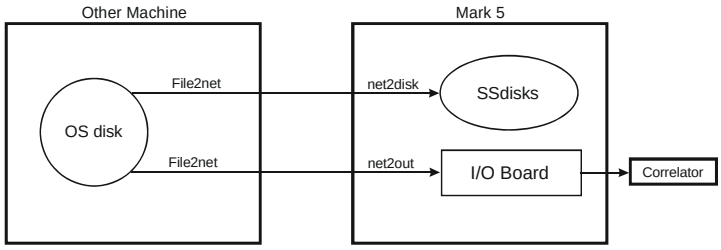

Figure 3: File to Mark 5 transfer through network

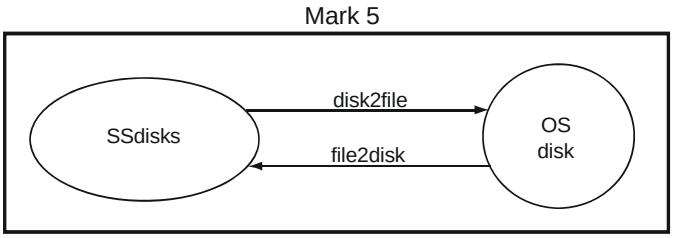

Figure 4: Internal Mark 5 to file transfer

5. Comments on 'Play Pointer', 'Record Pointer' and 'Scan Pointer'

Three different pointers are maintained by the Mark 5A system and it is important to understand what they are, what they mean, and how they are managed. The *play pointer* and *record pointer* are byte numbers (number of bytes), *not* pointers in the sense of C programs; the *scan pointer* points to a particular recorded scan.

5.1 Record Pointer

The Mark 5 system records data to a disk set much as if it were a tape. That is, recording starts from the beginning and gradually fills the disk set as scans are recorded one after another. The 'record pointer' indicates the current recording position (in bytes, always a multiple of 8) which, at any instant, is just the current total number of recorded bytes. Arbitrary recorded scans cannot be erased; however, individual scans may be erased in order from last to first. The entire disk set is erased by setting the record pointer back to zero using the 'reset=erase' command.

The record pointer can be modified in the following ways:

1. A 'reset=erase' command forces the record pointer to zero.
2. A 'reset=erase_last_scan' sets the record pointer to the beginning of the space occupied by the erased scan.
3. A 'record=on' command causes recording to start at the current record pointer position and increment at the total recording data rate.
4. The 'net2disk' and 'file2disk' commands act similarly to the 'record=on' command, except that the data originates from either a network connection or a Linux file, respectively.

The current value of the record pointer can be queried with the 'position' query.

5.2 Play Pointer

The 'play pointer' indicates the current playback position (in bytes, always a multiple of 8) from the beginning of the disk set. The play pointer may never be larger than the 'record pointer.' The play pointer can be modified in the following ways:

1. A 'reset=erase' commands forces the play pointer to zero.
2. On 'play=on:<start byte#>', the play pointer is set to <start byte#> before play starts. If <start byte#> is not specified, playback start at the current play-pointer position.
3. On 'play=off' or when playback reaches the end of recording, the play pointer is updated to the point at which playback stopped. If not at the end of the recording, a subsequent 'play=on' command will continue play from this point. On 'play=off:<byte#>', playback is stopped and the playback pointer is set to the specified byte number.
4. On 'record=off' or end-of-media (following a 'record=on'), the play pointer is set to the beginning of the just-recorded scan.
5. A 'scan_set=<scan label|scan#>:….' command sets the play pointer to the specified point within the specified scan; it also sets the *scan pointer* to the specified scan (see below).
6. A 'scan_play' command sets the play pointer to the *beginning* of the scan corresponding to the current *scan pointer* (see below) and commences play. Play stops at the end of the scan and the playback pointer updated to the stop position.

The current value of the play pointer can be queried with the 'position' query.

5.3 Scan Pointer

For the convenience of the user, the notion of a 'scan pointer' exists with respect to several commands; in actuality, the 'scan pointer' refers to a particular recorded scan. Actions affecting the scan pointer also often affect the play pointer. The 'scan pointer' can be modified in only two ways:

1. On 'record=off' or end-of-media (following a 'record=on'), the scan pointer is set to the just-recorded scan; the play pointer is set to the beginning of the just-recorded scan.

2. A ‘scan_set=<scan label|scan#>:s|c|e’ command sets the ‘scan pointer’ to the specified scan, as well as setting the play pointer within the scan as specified.

The following actions are affected by the value of the scan pointer:

1. A ‘scan_play’ command sets the play pointer to the *beginning* of the scan corresponding to the current *scan pointer* and commences play; the scan pointer is not affected.
2. A ‘scan_set?’ query returns information about the scan pointed to by the scan pointer.
3. A ‘scan_check?’ query returns information regarding the scan pointed to by the scan pointer.

5.3 Directory management

The Mark 5 maintains a scan directory on each disk set. The following queries are used to retrieve directory information:

1. The ‘dir_info’ query reports the number of scans plus information about remaining disk space.
2. The ‘scan_set?’ query reports the directory information for the current *scan_set* scan. A ‘scan_set=inc’ command increments to the following scan (in order recorded on disks), which can then be queried with a ‘scan_set?’ query.

6. ‘Scan Name’, ‘Scan Label’ and ‘Filename’

Mark5 defines a ‘scan’ as a continuously recorded set of data. Each scan is identified by a scan name, which normally is derived from the scan name in the associated VEX file used in the scheduling of the experiment (see http://lupus.gsfc.nasa.gov/vex/vex.html). When Mark 5 records a scan, it assigns a scan label of the form:

<experiment name>_<station code>_<scan name> [Example: ‘grf103_ef_254-1056’]

where

<experiment name> is the name of the experiment (e.g. ‘grf103’); maximum 16 characters, but by convention corresponds to a standardized 6-character experiment name.

<station code> is the station code (e.g. ‘ef’); maximum 16 characters, but by convention corresponds to standardized 2-character codes.

<scan name> is the identifier for the scan (e.g. ‘254-1056’); max 16 characters.

No white space is allowed in any of these subfields, nor are any characters ‘/.\:;=_’, single-quote or double-quote. Trailing underscores (which may result from null subfields) are deleted from the scan label. The ‘+’ character is not allowed in <scan name>[2]. Lower-case characters in all subfields are preferred, but not enforced.

The scan label, with ‘.m5a’ suffix becomes the default destination filename for ‘disk2file’ operations, providing full self-identification of all scan files as they are moved between and among Mark 5 and Linux file systems (example: ‘abc789_xy_scan035.m5a’).

By agreed convention, the scan label may be augmented with optional additional information where necessary or useful, as follows:

<exp name>_<station code>_<scan name>[_<data start time>_<aux info1>_<aux info2>...]

where

<data start time> - start time of data in file; required if data start time is not unambiguously embedded in the data itself. Format may be either 1) VEX time format or 2) undelimited time of form ‘yyyydddhhmmss’ (13 digits), ‘dddhhmmss’ (9 digits), ‘yyyyddd’ (7 digits), or ‘hhmmss’ (6 digits). Fractional seconds should be specified as necessary. The <data start time field>

[2] The ‘+’ character is reserved for identifying scans which span more than one disk module.

is mandatory when a single scan is broken into a time series of files with the same <scan name>.

<aux info> - auxiliary information field(s) in format 'cc=ppp' where 'cc' is a standardized 2-char identifier for information and 'ppp' is the information value in some specified standardized format (example: 'bm=0x0000ffff' specifies the VSI 'bit mask' used in collecting the data)

Example filename: 'abc789_xy_scan035_154d12h43m10s_bm=0x000ffff.m5a'

Complete information on standardized VLBI filenaming conventions is available in the memo "e-VLBI File-Naming Conventions", available as memo 49 in the e-VLBI memo series at http://www.haystack.edu/tech/vlbi/evlbi/memo.html.

Mark 5 scans may be transferred to standard files using the 'disk2file' command. Though normal default usage of the 'disk2file' command is to transfer one scan to one file, it should be noted that the Mark 5 system allows multiple scans to copied to a single Linux file with the 'disk2file' program since 'disk2file' allows the user to specify arbitrary start and end bytes of the Mark 5 recording for transferring to a file. No standardized file-naming conventions have been adopted for this type of usage.

7. Disk-FIFO mode

A special 'disk-FIFO' mode augments 'in2net' to use a Mark 5 disk pack as the FIFO buffer in the case where network transfer during an experiment is desired, but network transfer rates are much slower than real-time. The disk-FIFO mode is usable up to a maximum data rate of 512 Mbps with an 8-disk module.

7.1 Usage

Use disk FIFO mode on a transmitting Mark 5, along with 'net2disk', 'net2out' or 'Net2file' on a receiving Mark 5, to transfer data to remote machines during an experiment, when network transfer speeds are slow compared to the real-time data rate from the telescope. Use 'ordinary' in2net with network connections fast enough to keep up with real-time data rates. In disk-FIFO mode, the disk module is used only to augment FIFO buffer storage and does not result in usable recorded data at the end of the experiment.

7.2 Setup

Disk-FIFO mode requires a scratch disk module in Bank A. Before starting, the disk module should be erased (by *SSErase* or *Mark5A*) *before* restarting *Mark5A* in disk-FIFO mode. To start Mark5A in disk-FIFO mode:

Mark5A –m 0 -d 1 &

The *'-m 0'* and *'&'* are optional, as usual. The *'-d 1'* initiates the special disk-FIFO mode of operation. When *Mark5A* is running in this mode, a 'status?' query will return 0x20 – 'Disk FIFO mode', and many of the normal Mark5A commands and queries will return errors or will not function properly. The 'in2net' function will use the scratch disk module as a FIFO.

7.3 Operation

Run *Mark5A* from *tstMark5A* or from the Field System, as usual. Except for the FIFO size, 'in2net' commands and queries operate as usual. A typical sequence of operation might be:

1. Start the receiving program (e.g. 'Net2file') on the target machine.
2. Set and check the formatter configuration and data mode with the 'mode=...' command and 'mode?' query, especially to verify that the input board is connected and the data are synchronized.
3. Connect to the target machine using an 'in2net=connect:…' command.
4. Start and stop each scan with 'in2net=on' and 'in2net=off'.

5. Log the target Mark 5A byte position before the start of each scan (or at the end of each scan) using an ‘in2net?’ query.

The byte log will be somewhat less precise than ‘in2net’ operating in normal mode (RAM FIFO) because there is up to one StreamStor block (62528 bytes) lost on each start/stop cycle. However ‘data_check?’ or ‘track_check?’ queries on the target machine starting near the logged positions will give exact answers.

To return to normal operation after completing disk-FIFO operations, shut down and restart *Mark5A* without *‘-d 1’*. The scratch disk module will also need to be erased (again) for normal operation.

8. Mark 5A Command/Query Summary (by Category)

8.1 General

DTS_id?	p. 26	Get system information (query only)
error?	p. 27	Get error number/message (query only)
OS_rev1?	p. 36	Get details of operating system (query only)
OS_rev2?	p. 37	Get more details of operating system (query only)
protect	p. 43	Remove erase protection for active module
recover	p. 46	Recover data which was overwritten or terminated abnormally during recording
reset	p. 48	Reset Mark 5 unit (command only)
SS_rev1?	p. 56	Get StreamStor firmware/software revision levels, part 1 (query only)
SS_rev2?	p. 57	Get StreamStor firmware/software revision levels, part 2 (query only)
status?	p. 59	Get system status (query only)
task_ID	p. 60	Set task ID (primarily for correlator use)

8.2 Record/Play

mode	p. 31	Set data recording/playback mode
play	p. 38	Play data from current or specified play pointer position
play_rate	p. 40	Set playback data rate; set tvg rate
record	p. 44	Turn recording on|off; assign scan label
scan_play	p. 52	Play scan specified by current value of scan_set parameters
scan_set	p. 53	Set scan for scan_check, scan_play, disk2file and disk2net
skip	p. 55	Skip forward|backward specified # of bytes during playback or net2out

8.3 Data Checking

data_check?	p. 15	Check data starting at position of current play pointer (query only)
scan_check?	p. 50	Get scan parameters (query only)
track_check?	p. 61	Check data on selected track (query only)
track_set	p. 63	Select tracks for monitoring with DQA or ‘track_check’

8.4 Data Transfer

disk2file	p. 22	Transfer data from Mark 5 to file
disk2net	p. 24	Transfer data from Mark 5 to network
file2disk	p. 28	Transfer data from file to Mark 5
in2net	p. 30	Transfer data directly from Mark 5 input to network
net_protocol	p. 35	Set network data-transfer protocol
net2disk	p. 33	Transfer data from network to Mark 5
net2out	p. 34	Transfer data directly from network to Mark 5 output

8.5 Bank Management

bank_info?	p. 12	Get bank information (query only)
bank_set	p. 13	Select active bank for recording or playback
bank_switch	p. 14	Enable|disable automatic bank-switching (not yet implemented)

8.6 Disk Info

dir_info?	p. 17	Get directory information (query only)
disk_model?	p. 18	Get disk model numbers (query only)
disk_serial?	p. 19	Get disk serial numbers (query only)
disk_size?	p. 20	Get disk sizes (query only)
disk_state	p. 21	Set/get Disk Module Status (DMS): last significant disk operation
disk_state_mask	p. 22	Set mask to enable changes in DMS
get_stats?	p. 29	Get disk-performance statistics (query only)
position?	p. 42	Get current record and play pointers (query only)
replaced_blks?	p. 47	Get number of replaced blocks during playback (query only)
rtime?	p. 49	Get remaining record time on current disk set (query only)
start_stats	p. 58	Start gathering disk-performance statistics.
VSN	p. 64	Write extended-VSN to permanent area

9. Mark 5A Command/Query Summary (Alphabetical)

bank_info?	p. 12	Get bank information (query only)
bank_set	p. 13	Select active bank for recording or playback
bank_switch	p. 14	Enable|disable automatic bank-switching (not yet implemented)
data_check	p. 15	Check data starting at position of current play pointer
dir_info?	p. 17	Get directory information (query only)
disk_model?	p. 18	Get disk model numbers (query only)
disk_serial?	p. 19	Get disk serial numbers (query only)
disk_size?	p. 20	Get disk sizes (query only)
disk_state	p. 21	Set/get Disk Module Status (DMS): last significant disk operation
disk_state_mask	p. 22	Set mask to enable changes in DMS
disk2file	p. 22	Transfer data from Mark 5 to file
disk2net	p. 24	Transfer data from Mark 5 to network
DTS_id?	p. 26	Get system information (query only)
error?	p. 27	Get error number/message (query only)
file2disk	p. 28	Transfer data from file to Mark 5
get_stats?	p. 29	Get disk-performance statistics (query only)
in2net	p. 30	Transfer data directly from Mark 5 input to network
mode	p. 31	Set data recording/playback mode
net_protocol	p. 35	Set network data-transfer protocol
net2disk	p. 33	Transfer data from network to Mark 5
net2out	p. 34	Transfer data directly from network to Mark 5 output
OS_rev1?	p. 36	Get details of operating system (query only)
OS_rev2?	p. 37	Get more details of operating system (query only)
play	p. 38	Play disk data from current or specified play pointer position
play_rate	p. 40	Set playback data rate; set tvg rate
position?	p. 42	Get current record and play pointers (query only)
protect	p. 43	Remove erase protection for active module
record	p. 44	Turn recording on|off; assign scan label
recover	p. 46	Recover data which was overwritten or terminated abnormally during recording
replaced_blks?	p. 47	Get number of replaced blocks during playback (query only)
reset	p. 48	Reset Mark 5 unit (command only)
rtime?	p. 49	Get remaining record time on current disk set (query only)
scan_check?	p. 50	Get scan parameters (query only)
scan_play	p. 52	Play scan specified by current value of scan_set parameters
scan_set	p. 53	Set scan for scan_check, scan_play, disk2file and disk2net

10. Mark 5A Command Set Details

This section contains a complete description of all Mark 5A commands/query in alphabetical order. Highlights in red are changes and updates from Revision 2.5.

bank_info – Get bank information (query only)

[command list]

Query syntax: bank_info? ;
Query response: !bank_info ? <return code> : <active bank> : <#bytes remaining> : <inactive bank> : <#bytes remaining> ;

Purpose: Returns information on both selected and unselected banks, including remaining space available.

Monitor-only parameters:

Parameter	Type	Values	Comments
<selected bank>	char		Currently selected bank; '-' if disk module is faulty; see Note 1.
<#bytes remaining>	int		Approximate #bytes remaining to be recorded on active module; =0 if no module selected or faulty module.
<other bank>	char		Unselected bank, if module is mounted and ready; if no module or faulty module, '-' is returned.
<#bytes remaining>	int		Approximate #bytes remaining to be recorded on inactive module; =0 if no module active or faulty module.

Notes:

1. If no modules are inserted, an error code 6 is returned.
2. The estimate of <#bytes remaining> is made without taking into account any slow or bad disks. When recording is not in progress, an 'rtime?' query gives a more precise estimate of the available space for the selected bank.

bank_info

bank_set – Select active bank for recording or playback

[command list]

Command syntax: bank_set = <bank> ;
Command response: ! bank_set = <return code> ;

Query syntax: bank_set? ;
Query response: ! bank_set ? <return code> : <active bank> : <active VSN> : <inactive bank> : <inactive VSN> ;

Purpose: When in bank mode, the selected bank becomes the 'active' bank for all Mark 5A activities.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<bank>	char	A | B | inc	A	'inc' increments to next bank in cyclical fashion around available bank; see Note 1.

Monitor-only parameters:

Parameter	Type	Values	Comments
<active bank>	char	A | B	'A' or 'B' if there is an active bank; '-' if no active bank
<active VSN>	char		VSN of active module, if any
<inactive bank>	char	B | A | -	'B' or 'A' if inactive bank is ready; '-' if module not ready
<inactive VSN>	char		VSN of inactive module, if any;

Notes:

1. If the requested bank is not the bank already selected, a completion code of '1' (delayed completion) is returned. Bank switching takes a variable amount of time up to about 3 seconds. While bank switching is in progress, many commands and queries will return a code of 5 (busy, try later) or 6 (conflicting request; in effect, neither bank is mounted during this transition). If an attempt to switch the bank fails (e.g. if there is no 'ready' disk module in the other bank), a 'status?' or "error?' query will return error 1006, "Bank change failed." A 'bank_set?' query will indicate whether the bank has changed. Switching banks can also generate other errors if there are problems with the target bank.
2. The 'bank_set' command may not be issued during recording or playback.
3. A 'bank_set?' query always returns the currently active module, which may change dynamically if automatic bank switching is enabled or if the operator changes banks using the keyswitches.

bank_set

bank_switch – Enable/disable automatic bank switching

[command list]

Command syntax: bank_switch = <auto-switch on/off> : [<mode>] ;
Command response: !bank_switch = <return code> ;

Query syntax: bank_mode? ;
Query response: !bank_mode ? <return code> : <auto-switch on/off> : [<mode>] ;

Purpose: Enable/disable automatic bank-switching for both record and playback.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<auto-switch mode>	char	off | on	off	If 'on', enables automatic bank-switching.
<mode>	char	-	-	Switching mode

Notes:

1. When automatic bank-switching is enabled, the following actions are triggered when recording hits end-of-media (say, on Bank A):
   a. Bank A stops recording and updates its directory.
   b. Bank B is selected as the 'active' bank (assumes Bank B is ready).
   c. Recording starts on Bank B and continues until a 'record=off' command is issued.
2. A similar sequence of events is executed during playback at the correlator; it is likely that some minor re-synchronization will be required by the correlator after the switching event.
3. During the bank-switching action, up to one second of data may be lost.
4. In the example above, if Bank B is not empty, the data on Bank B will be extended in the usual manner (i.e. no existing data on Bank B will be lost). In this case, automatic bank switching on playback will not work properly.
5. If the alternate Bank is not ready at the time switching is initiated, the recording or playback will stop.
6. The 'continuation segment' of the scan on the alternate disk module maintains the same scan label as the originating segment, except that the 'initial' and 'continuation' segments are identified by a trailing or preceding (respectively) '+' character added to the scan name subfield of the scan label when a 'scan_set?' query is executed.

bank_switch

data_check – Check data starting at position of current play pointer (query only)

[command list]

Query syntax: data_check? ;
Query response: !data_check ? <return code> : <data mode> : <data submode> : <data time> : <byte offset> : <track frame period> : <#bytes in frame> : <#missing bytes>;

Purpose: Reads a small amount of data starting at the present play pointer position and attempts to determine the details of the data, including recording mode and data time. For most purposes, the 'scan_check' command is more useful. Please be especially attentive to Note 1 for the track set that must be recorded.

Monitor-only parameters:

Parameter	Type	Values	Comments
<data mode>	char	st | mark4 | vlba | tvg | SS	See 'mode' command for explanation of data modes; 'tvg' corresponds to VSI test pattern;' SS' corresponds to StreamStor test pattern '?' indicates unknown format.
<data submode>	char	8 | 16 | 32 | 64 | mark4 | vlba	'8|16|32|64' if <data mode> is 'mark4' or 'vlba'; 'mark4|vlba' if <data mode> is 'st' When <data mode>='tvg', returns special diagnostic info - see Note 6.
<data time>	time		Time tag from *next* 'track' frame header beyond current play pointer. See Note 5 of 'scan_check'. When <data mode>='tvg', returns special diagnostic info - see Note 6.
<byte offset>	int		Byte offset from current play pointer to beginning of *next* 'track' frame header. When <data mode>='tvg', returns special diagnostic info - see Note 6.
<track frame period>	time		Time tag difference between adjacent track frames; allows sample-rate determination
<#bytes in frame>	int		Total #bytes in recording between track frame headers. This is a useful (if somewhat redundant) number. For 'st:mark4' mode: should always be 90,000 (i.e. 32*2500*9/8); for 'st:vlba' mode, should always be 90,720 (i.e. 32*2520*9/8). For mode 'mark4:#trks', will be (#trks*2500); for mode 'vlba:#trks', will be (#trks*2520), where '#trks' is 8, 16, 32, or 64.
<#missing bytes>	int	bytes	Number of missing bytes between last and current 'data_check'; Should be =0 if immediately previous 'data_check' was within same scan Meaningless if immediately previous 'data-check was in a different scan, or if data are not formatted VLBI data. Null if <#missing bytes> cannot be calculated. See Notes 4 and 5; see also Note 5 in 'scan_check'

Notes:

1. Starting at the present play-pointer position, the 'data_check' query searches through all possible recording modes until it can make sense of the data, then reports what it has found; 'mark4:xx', 'vlba:xx', 'st:mark4' and 'st:vlba' modes must have been recorded data from a VLBA or Mark 4 formatter. In order for the 'data_check' command to be successful with data recorded from a VLBA or Mark 4 formatter, a minimum set of tracks must be recorded according to the following table:

mode:submode	Minimum set of Mark4/VLBA tracks that must be active
'mark4:8' or 'vlba:8' - 8 tks	2-16 even (headstack 1)
'mark4:16' or 'vlba:16' - 16 tks	2-16 even or 18-33 even (headstack 1)
'mark4:32' or 'vlba:32' or any 'st' mode - 32 tks	2-9 or 10-17 or 18-25 or 26-33 (headstack 1)
'mark4:64' or 'vlba:64' - 64 tks	2-9 or 10-17 or 18-25 or 26-33 (headstack 1 or headstack 2)

data_check

2. The ‘data_check’ query will be honored only if record and play are both off.
3. The ‘data_check’ query does not affect the play pointer.
4. A blank will be returned in the <#missing bytes> field if the # of missing bytes cannot be calculated; for example, if the data are tvg data or other non-VLBI-format test data, the <#missing bytes> parameter is meaningless.
5. Regarding the ‘data time’ value returned by the ‘data_check?’, ‘scan_check?’ and ‘track_check?’ queries: The Mark 4 time-tags contain the day-of-year (DOY) but only the final digit of the year; the VLBA time-tags contain, instead, the last 3 digits of the Julian day number (misnamed MJD). To show the year and DOY in the returned values of ‘data time’ requires some assumptions. For Mark 4, we assume the most recent year consistent with the unit-year and DOY written in the Mark 4 time-tag; this algorithm reports the proper year provided the data were taken no more than 10 years ago. For VLBA, we assume the most recent Julian Day Number (JDN) consistent with the last 3 digits available in the VLBA time-tag; this algorithm reports the proper year provided the data were taken no more than 1000 days ago.
6. When the <data mode> is determined to be ‘tvg’ or ‘SS’, three integer diagnostic parameters are returned following <data mode>. A buffer of data is read (typically ~1MB) from the disks at the present play pointer position, which is analyzed. The following information is returned:
    a. Position of first 32-bit word (starting from zero) in buffer containing first valid word in the ‘tvg’ or ‘SS’ sequence.
    b. Position of first 32-bit word which is not in the proper order of the ‘tvg’ or ‘SS’ sequence.
    c. Size of block read.

    For a properly operating system, the first number will be 0 and the 2nd and 3rd numbers will have the same values.

data_check

dir_info – Get directory information (query only)

[command list]

Query syntax: dir_info? ;
Query response: !dir_info ? <return code> : <number of scans> : <total bytes recorded> : <total bytes available> ;

Purpose: Returns information from the data directory, including number of scans, total bytes recorded and remaining bytes available.

Monitor-only parameters:

Parameter	Type	Values	Comments
<number of scans>	int		Returns number of scans currently in the data directory.
<total bytes recorded>	int		Sum over all recorded scans
<total bytes available>	int		Sum of total available disk space (unrecorded plus recorded)

Notes:

3. The scan directory is automatically stored each time data are recorded to the disks.

dir_info

disk_model – Get disk model numbers (query only)

[command list]

Query syntax: disk_model? ;
Query response: !disk_model ? <return code> : <disk model#> : <disk model#> : ……;

Purpose: Returns a list of model numbers currently mounted disks.

Monitor-only parameters:

Parameter	Type	Values	Comments
<disk model#>	literal ASCII		Returned in order of drive number (0=0M, 1=0S, 2=1M, 3=1S,….,14=7M, 15=7S); a blank field is returned for an empty slot.

disk_model

disk_serial – Get disk serial numbers (query only)

[command list]

Query syntax: disk_serial? ;
Query response: !disk_serial ? <return code> : <disk serial#> : <disk seriall#> : ……;;

Purpose: Returns a list of serial numbers of currently mounted disks.

Monitor-only parameters:

Parameter	Type	Values	Comments
<disk serial#>	literal ASCII		Returned in order of drive number (0=0M, 1=0S, 2=1M, 3=1S,….,14=7M, 15=7S); A blank field is returned for an empty slot.

disk_serial

disk_size – Get disk sizes (query only)

[command list]

Query syntax: disk_size? ;
Query response: !disk_size ? <return code> : <disk size> : <disk size> : ……;

Purpose: Returns a list capacities of currently mounted disks.

Monitor-only parameters:

Parameter	Type	Values	Comments
<disk size>	int	bytes	Returned in order of drive number (0=0M, 1=0S, 2=1M, 3=1S,….,14=7M, 15=7S); A blank field is returned for an empty slot.

disk_size

disk_state –Set/get Disk Module Status (DMS): last significant disk operation

[command list]

Command syntax: disk_state = <DMS> ;
Command response: !disk_state = <return code> : <DMS> ;

Query syntax: disk_state? ;
Query response: !disk_state? <return code> : <active bank> : <active-bank DMS> : <inactive bank> : <inactive-bank DMS> ;

Purpose: Set/get Disk Module Status (DMS), which logs the last significant operation that happened on the disk module.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<DMS> (disk Module Status)	char	recorded | played | erased | unknown | error	none	To be used only if automatically-set DMS parameter is to be overwritten. Requires a preceding 'protect=off' and affects only the active module. Current value of 'disk_state_mask' is ignored. See also Note 3

Monitor-only parameters:

Parameter	Type	Values	Comments
<active bank>	char	A | B	Currently selected bank; '-' if disk module is faulty; see Note 2.
<active-bank DMS>	char	recorded | played | erased | unknown | error	recorded – last significant operation was record or a record-like function (net2disk or file2disk). played – last significant operation was play; disk2net and disk2file do not affect DMS. erased – last significant operation was erase or conditioning, either from 'reset=erase' or SSErase. unknown – last significant operation was performed with version of Mark5A or SSErase prior to implementation of the DMS function. error – error occurred; for example, an interrupted conditioning attempt or a failure during one of the significant operations above
<inactive bank>	char	B | A | -	Unselected bank, if module is mounted and ready; if no module or faulty module, '-' is returned.
<inactive-bank DMS>	char	recorded | played | erased | unknown | error	See above.

Notes:

1. The 'disk_state' and 'disk_state_mask' commands were requested by NRAO and are designed primarily for use at a correlator.
2. The DMS represents the last significant operation that occurred on a disk module. It is designed to distinguish between disk modules waiting to be correlated, have been correlated, or have no data (erased) and ready to be recorded. The DMS is saved on the disk module in the same area as the permanent VSN so that the DMS from both active and inactive disk banks are accessible. Commands scan_check, data_check, track_check, disk2net, and disk2file, do not affect DMS.
3. Normally, the setting of the DMS parameter happens automatically. However, the <disk_state=...> command is provided to manually overwrite the current DMS parameter. This command requires a preceding 'protect=off' and affects only the active module. A 'disk_state=…" command ignores the current value of the disk_state_mask (see 'disk_state_mask' command).
4. If no modules are inserted, an error code 6 is returned.

disk_state

disk_state_mask – Set mask to enable changes in DMS

[command list]

Command syntax: disk_state_mask = <erase_mask_enable> : <play_mask_enable> : <record_mask_enable>;
Command response: !disk_state_mask = <return code> : <erase_mask_enable> : <play_mask_enable> : <record_mask_enable> ;

Query syntax: disk_state_mask? ;
Query response: !disk_state_mask? <return code> : <erase_mask_enable> : <play_mask_enable> : <record_mask_enable> ;

Purpose: Set mask to enable changes in DMS.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<erase_mask_enable>	int	0 | 1	1	0 – disable an erase operation from modifying the DMS. 1 – enable erase operation to modify the DMS.
<play_mask_enable>	int	0 | 1	1	0 – disable a play operation from modifying the DMS. 1 – enable play operation to modify the DMS.
<record_mask_enable>	int	0 | 1	1	0 – disable a record operation from modifying the DMS. 1 – enable record operation to modify the DMS.

Notes:

1. The disk_state_mask is intended to prevent accidental changes in the DMS. When a module is at a station, the disk_state_mask setting of 1:0:1 would disable a play operation from modifying the DMS. Likewise, at a correlator one might want to disable the record_mask_enable.

disk_state_mask

disk2file – Transfer data from Mark 5 to file

[command list]

Command syntax: disk2file = <destination filename> : [<start byte#>] : [<end byte#>] : <option> ;
Command response: !disk2file = <return code> ;

Query syntax: disk2file? ;
Query response: !disk2file ? <return code> : <status> : <destination filename> : <start byte#> : <current byte#> : <end byte#> : <option> ;

Purpose: Initiates a data transfer from the Mark 5 data disk to an ordinary file.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<dest filename>	literal ASCII	no spaces allowed	See Comments	Default is scan label as reported by 'scan_set', with 'm5a' suffix attached (e.g. 'grf103_ef_scan035.m5a'). Filename must include path if path is not default.
<start byte#>	int | null		See Comments	Absolute byte#; if null, defaults to <start scan_play> position as set and/or reported by scan_set
<end byte#>	int | null		See Comments>	Absolute end byte#; if preceded by '+', increment from <start byte#> by specified value; if null, defaults to <end scan_play> position as set and/or reported by scan_set.
<option>	char	n | w | a	n	n – create file; error if existing file w –erase existing file, if any; create new file. a – create file if necessary, or append to existing file

Monitor-only parameters:

Parameter	Type	Values	Comments
<dest filename>			Destination filename (returned even if filename was defaulted in corresponding 'disk2file' command)
<status>	char	active | inactive	Current status of transfer
<current byte#>	int		Current byte number being transferred

Notes:

1. To abort data transfer: The 'reset=abort' command may be used to abort an active disk2file data transfer. See 'reset' command for details.
2. When <status> is 'inactive', a 'disk2file?' query returns the <dest filename> of the last transferred scan, if any.

disk2file

disk2net – Transfer data from Mark 5 to network [command list]

Command syntax: disk2net = connect : <target hostname> ;
disk2net = on : [<start byte#>] : [<end byte#>] ;
disk2net = disconnect ;
Command response: !disk2net = <return code>;

Query syntax: disk2net? ;
Query response: !disk2net ? <return code> : <status> : <target hostname> : <start byte#> : <current byte#> : <end byte#> ;

Purpose: Initiates a data transfer from Mark 5 data disks to a remote Mark 5 system.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<control>	char	connect | on | disconnect		'connect' – connect to socket on receiving Mark 5 system 'on' – start data transfer 'disconnect' – disconnect socket See Notes.
<target hostname>	char		localhost or previously set name	Required only on if <control>='connect'..
<start byte#>	int | null		See Comments	Absolute byte#; if null, defaults to <start scan_play> position as set and/or reported by scan_set
<end byte#>	int | null		See Comments	Absolute end byte#; if preceded by '+', increment from <start byte#> by specified value; if null, defaults to <end scan_play> position as set and/or reported by scan_set.

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	connected | active | inactive	Current status of transfer
<target hostname>	char		
<current byte#>	int		Current byte number being transferred

Notes:

1. To set up connection: First, issue 'open' to the *receiving* system ('net2disk=open' or 'net2out=open' to Mark 5, or Net2file as standalone program; then issue 'connect' to the *sending* system ('in2net=connect:..' or 'disk2net=connect:…' to Mark 5).
2. To start data transfer: Issue 'on' to *sending* system ('in2net=on' or 'disk2net=on' to Mark 5). A 'disk2net' transfer will stop automatically after the specified number of bytes are sent.
3. To stop data transfer: Issue 'off' to the sending system ('in2net=off' to Mark 5). After each transfer has been stopped or completed, another transfer may be initiated (see Note 2).
4. To close connection: First, issue 'disconnect' to the sender ('in2net=disconnect' or disk2net=disconnect' to Mark5'). A 'disk2net=disconnect' command issued before the specified number of bytes are transferred will abort the transfer and close the connection. Then, 'close' the receiver ('net2disk=close' or 'net2out=close' to Mark 5; Net2file ends). Net2file ends automatically on a 'disconnect'. After a 'net2disk' transfer, the data on disk are not ready for use until after a 'net2disk=close' command has been issued.

disk2net

5. To abort data transfer: The ‘reset=abort’ command may be used to abort an active disk2net data transfer. A subsequent ‘disk2net=disconnect’ must be issued to close the socket and put the Mark 5A back into idle/bypass mode. See ‘reset’ command for details.
6. Only one data transfer activity may be active at any given time. That is, among ‘record=on’, ‘play=on’, ‘in2net=..’, ‘disk2net=..’, ‘net2disk=..’, ‘net2out=..’, ‘disk2file’=..’, ‘file2disk=..’, ‘data_check’ and ‘track_check’ ‘scan_check’, only one may be active at any given time.
7. Note that the network protocol parameters are set by the ‘net_protocol’ command.

disk2net

DTS_id – Get system information (query only)

[command list]

Query syntax: DTS_id? ;

Query response: !DTS_id ? <return code> : <system type> : <software revision date> : <media type> : <serial number> : <#DIM ports> : <#DOM ports> : <command set revision> : <Input design revision> : <Output design revision> ;

Purpose: Get Mark 5 system information

Monitor-only parameters:

Parameter	Type	Values	Comments
<system type>	char	mark5A	
<software revision date>	time		Date stamp on current version of Mark 5 software
<media type>	int	1	Per VSI-S spec: 1 – magnetic disk [0 – magnetic tape; 2 – real-time (non-recording)]
<serial number>	ASCII		System serial number; generally is in the form 'mark5-xx' where xx is the system serial number
<#DIM ports>	int	1	Number of DIM ports in this DTS
<#DOM ports>	int	1	Number of DOM ports in this DTS
<command set revision>	char		Mark 5 command set revision level corresponding to this software release (e.g. '2.3a')
<Input design revision>	int		Revision level of Input section of Mark 5A I/O board
<Output design revision>	int		Revision level of Output section of Mark 5A I/O board

DTS_id

error – Get error number/message (query only)

[command list]

Query syntax: error? ;
Query response: !error ? <return code> : <error#> : <error message> ;

Purpose: Get error number causing bit 1 of 'status' query return to be set

Monitor-only parameters:

Parameter	Type	Values	Comments
<error#>	int		Error number associated with 'status'query return bit 1
<error message>	literal ASCII		Associate error message, if any

Notes:

1. Most errors are 'remembered' (even if printed with debug) and printed (and cleared) by either a 'status?' or 'error?' query. Thus, errors may be remembered even after they have been corrected..

error

file2disk – Transfer data from file to Mark 5

[command list]

Command syntax: file2disk = <source filename> : <start byte#> : <end byte#> : [<destination scan label>] ;
Command response: !file2disk = <return code>;

Query syntax: file2disk? ;
Query response: !file2disk ? <return code> : <status> : <source filename> : <start byte#> : <current byte#> : <end byte#> : <scan#> : <destination scan label> ;

Purpose: Initiate data transfer from file to Mark 5 data disks

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<source filename>	literal ASCII	no spaces allowed	'save.data' or last value	Filename must include path if not default.
<start byte#>	int		0	Absolute byte number; if unspecified, assumed to be zero
<end byte#>	int		0	If =0, will copy to end of file
<dest scan label>	literal ASCII		<source filename> sans filename suffix, if any	Scan label saved to Mark 5 scan directory

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	active | inactive	Current status of transfer
<current byte#>	int		Current source byte# being transferred
<scan#>	int		Sequential scan number on disk module
<dest scan label>	ASCII		Scan label

Notes:

1. To abort data transfer: The 'reset=abort' command may be used to abort an active file2disk data transfer. See 'reset' command for details.
2. When <status> is 'inactive', a 'file2disk?' query returns <dest scan label> of the last transferred scan, if any.

file2disk

get_stats – Get disk performance statistics (query only)

[command list]

Query syntax: get_stats? ;
Query response: !get_stats ? <return code> : < drive number> : <bin 0 count> : <bin 1 count> :….: <bin 7 count> : <replaced-block count> ;

Purpose: Get detailed performance statistics on individual Mark 5 data disks

Monitor-only parameters:

Parameter	Type	Values	Comments
<drive number>	int		0=0M, 1=0S, 2=1M, 3=1S,….,14=7M, 15=7S
<bin 0 count>	int		Number of drive transactions falling in its bin 0 (see 'start_stats' command for explanation)
<bin 1 count>	int		Number of drive transactions falling in its bin 1
<bin 2 count>	int		Number of drive transactions falling in its bin 2
<bin 3 count>	int		Number of drive transactions falling in its bin 3
<bin 4 count>	int		Number of drive transactions falling in its bin 4
<bin 5 count>	int		Number of drive transactions falling in its bin 5
<bin 6 count>	int		Number of drive transactions falling in its bin 6
<bin 7 count>	int		Number of drive transactions falling in its bin 7
<replaced-block count>	int		Number of 65KB (actually 0xFFF8 bytes) data blocks unavailable on playback from this drive; these blocks have been replaced with fill pattern with even parity. See 'replaced_blks?' query for more information.

Notes:

1. Each subsequent 'get_stats' query returns current performance statistics for the next mounted drive; recycles through mounted drives. Bin counts are not cleared. See details in Notes on 'start_stats' command.
2. The 'get_stats' query may not be issued during active recording or playback.
3. Drive statistics and replaced-block counts are cleared and re-started whenever a new disk module is mounted or a 'start_stats' command is issued.
4. The 8 bin counts in the 8 bins correspond to drive-response (transaction completion) times, with response time increasing from left to right. A good disk will have large numbers in bins 0 and 1 and small numbers (or 0) in the last few bins. See 'start_stats for additional information.

get_stats

in2net – Transfer data directly from Mark 5 input to network

[command list]

Command syntax: in2net = <control> : <remote hostname> ;
Command response: !in2net = <return code> ;

Query syntax: in2net? ;
Query response: !in2net ? <return code> : <status> : <remote hostname> : <#bytes received> : <#bytes in buffer> ;

Purpose: Control direct data transfer from Mark 5 input to network; bypass disks

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<control>	char	connect | on | off | disconnect		'connect' – connect to socket on receiving Mark 5 system; initially, data transfer is off. 'on' – start data transfer 'off' – end data transfer 'disconnect' – disconnect socket See Notes with 'disk2net'
<remote hostname>	char		localhost	Required only on first 'connect'; otherwise ignored

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	inactive | connected | sending	
<remote hostname>			
<#bytes received>	int		#bytes received at the Input since 'connect' and while status is 'sending'
<#bytes in buffer>	int		#bytes remaining in buffer, waiting to be sent

Notes:

1. Important: Due to current software problem, a scratch disk is required in Bank A for in2net operation; will be fixed in future update.
2. See Notes with 'disk2net' command for usage rules and restrictions.
3. If the data rate is too fast for the network to handle, the FIFO will eventually overflow; this will be reported by either a 'status?' query or an 'in2net?' query with an error message.
4. After 'in2net=off', but before 'in2net=disconnect', <#bytes received> shows the approximate total #bytes transferred from the input source; the #bytes currently sent out through the network is ~<#bytes received> minus <#bytes in buffer>. As <#bytes in buffer> drains to zero (as remaining data is sent out over the network), <#bytes received> becomes somewhat more precise.
5. If 'in2net=disconnect' is issued while <#bytes in buffer> is >0, data will be lost.
6. For operation in special disk-FIFO mode, see Section 7.
7. Note that the network protocol parameters are set by the 'net_protocol' command.

in2net

mode – Set data recording/playback mode

[command list]

Command syntax: mode = <data mode> : <data submode> : [<output data mode> : <output submode>] ;
Command response: !mode = <return code> ;

Query syntax: mode? ;
Query response: !mode ? <return code> : <data mode> : <data submode> : <output mode> : <output submode> : <sync status> : <#sync attempts> ;

Purpose: Set the recording and playback mode of the Mark 5 I/O card.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<data mode>	char	mark4 | vlba st | tvg | mark5a+n	st	'mark4' or 'vlba': strips and restores parity bits. 'st' ('straight-through') mode records 32 input 'tracks' directly 'tvg' – takes data from internal TVG – see Note 8. A null field is special case for correlator. See Note 5. For Mark 5A+ operation, 'n' is track-map # to be used; see Note 14.
<data submode>	char	8 | 16 | 32 | 64 | mark4 | vlba	See Note 2 and Note 14	8,16,32,64 is relevant only for 'mark4', 'vlba' and 'mark5a+' modes and corresponds to number of tracks. 'mark4' and 'vlba' relevant only for 'st' mode. Not relevant when <data mode> is 'tvg'. A null field is special case for correlator. See Note 5.
<output mode>	char	mark4 | vlba | st	<data mode>	Optional: For correlator or diagnostic use only: Forces the Output Section of the Mark 5A I/O board into specified mode and submode independently of the Input Section – see Note 5.
<output submode>	char	8 | 16 | 32 | 64 | mark4 | vlba	<data submode>	Optional: For correlator or diagnostic use only – see Note 5.

Monitor-only parameters:

Parameter	Type	Values	Comments
<sync status>	char	s | -	's' indicates Output Section of I/O board is sync'ed; '-' indicates not sync'ed. See Note 9
<#sync attempts>	int		Number of sync attempts by output section. Relevant only for 'mark4' and 'vlba' modes only. See Note 10.

Notes:

1. The 'mode=' command sets both the input and output modes to be the same unless overridden by <output data mode> and <output submode> parameters.
2. Power-on default <data mode>:<data submode> is 'st:mark4'. For <data mode> of 'st', default <data submode> is 'mark4; for <data mode> of 'mark4' or 'vlba', default <data submode> is '32'.
3. In 'mark4' or 'vlba' mode, the Mark 5A strips parity on record and restores it on playback to save storage space. If the number of tracks is 8, 16 or 64, the Mark 5 I/O does the necessary multiplexing/demultiplexing to always fully utilize all FPDP 32 bit streams driving the disk array. In 'st' ('straight-through') mode, the input data are recorded and played back with no processing.

4. **In 'mark4:xx' mode, the station ID (set by jumpers in the Mark 4 DAS rack) must be an even number**. Attempting to record in 'mark4' mode with an odd station ID will result in an error. This is due to the fact that, with parity stripped, an odd station ID considerably complicates the job of properly recovering synchronization during playback, and is therefore not allowed.
5. At a correlator, where there is normally nothing connected to the Mark 5A input, it is suggested that the desired playback mode be specified in <output mode> and <output submode> and that <data mode> and <data submode> both be null fields. This will cause the input section of the I/O board to be set to default ('st:mark4') mode and prevents spurious error messages from appearing regarding the input station ID.
6. The only reason to distinguish between 'st:mark4' and 'st:vlba' modes is to allow the 'play_rate' command' to properly set the internal clock generator for a specified data rate; the setting is slightly different for the Mark4 and VLBA cases.
7. The tracks expected from a Mark4 or VLBA formatter in the various modes are as follows:

mode:submode	Recorded formatter track#'s	FPDP bit streams
'mark4:8' or 'vlba:8'	2-17 even (headstack 1)	Trk 2 to FPDP streams 0,8,16,24; trk 4 to 1,9,17,25; etc.
'mark4:16' or 'vlba:16'	2-33 even (headstack 1)	Trk 2 to FPDP streams 0,16; trk 4 to 1,17; etc.
'mark4:32' or 'vlba:32'	2-33 all (headstack 1)	Correspond to FPDP bit streams 0-31, respectively
'mark4:64' or 'vlba:64'	2-33 (headstacks 1 and 2)	Trks 2 from both hdstks mux'ed to FPDP bit stream 0, etc.
'st' (any submode)	2-33 all (headstack 1)	Correspond to FPDP bit streams 0-31, respectively

8. In all modes except 'tvg' mode, the data clock is provided by the external data source. In 'tvg' mode, the clock-rate is set by 'play_rate' command.
9. The 'sync status' parameter is relevant only in output mode 'mark4' or 'vlba' where parity must be restored. If 'sync'ed', the I/O board has properly synchronized to the data frames and is properly de-multiplexing and restoring parity.
10. The '# of sync attempts' returned value in the 'mode=' command counts the number of sync attempts the Mark 5A I/O board output section had to make before parity-stripped data ('mark4' or 'vlba') was re-sync'ed, as necessary for parity re-insertion. A large number indicates a problem, perhaps in the output clock or the data itself. The counter is reset to zero on a subsequent 'mode=' command.
11. If in 'tvg' mode, TVG is operated at clock-rate set by 'play_rate' command.
12. When <data mode> is 'mark4' or 'vlba', the NRZM output coding present on the data received from the formatter is converted to NRZL for transmission to the FPDP bus, and hence for recording on disk or transmission over a network. On output, the inverse operation (conversion back to NRZM) is performed, so that the output data on the Mark5A I/O Panel are in the NRZM format for input to the correlator. When operating in <st> or <tvg> mode, no coding conversions are done.
13. When operating in tvg mode, the 32-bit-wide tvg pattern is written directly to the disk with no tape-frame headers or synchronization information of any sort. Furthermore, the tvg runs continuously with no forced resets unless an external 1pps signal is connected to J11 on the Mark 5A I/O board, in which case the tvg is asynchronously reset on each 1pps tick.
14. Operation in Mark 5A+ mode allows a Mark 5B disk module to be read on the Mark 5A and create VLBA output tracks; the Mark 5B data are read, transformed into VLBA track format (including the addition of parity bits), and VLBA-format headers are inserted. The Mark 5A must have the proper Xilinx code and software upgrades installed for this mode to work. The track mapping option ('n' in 'mark5a+n'), as well as the number of output tracks, must be specified. Details are given in Mark 5 memo #39 available at http://www.haystack.edu/tech/vlbi/mark5/memo.html. Note that the VLBA auxiliary data field will be identically zero for all Mark 5A+ playback.

mode

net2disk – Transfer data from network to disks

[command list]

Command syntax: net2disk = <control> : <scan label> : [<experiment name>] : [<station code>] ;
Command response: !net2disk = <return code> ;

Query syntax: net2disk? ;
Query response: !net2disk ? <return code> : <status> : <scan#> : <scan label> ;

Purpose: Enable data transfer from network to local disks

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<control>	char	open | close		'open' or 'close' socket
<scan label>	literal ASCII	See Note 5		Scan label to be assigned to this data; if not specified, defaults to 'net2disk'
<experiment name>	ASCII	See Note 5		(Optional) Experiment name; ignored if <record on/off> is 'off'; intended for use only for backwards compatibility - see Note 3.
<station code>	ASCII	See Note 5		(Optional) Station code; ignored if <record on/off> is 'off'; intended for use only for backwards compatibility - see Note 3.

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	active | inactive | waiting	Current status of transfer
<scan#>	int		Sequential scan number on disk module
<scan label>	ASCII		Assigned scan label

Notes:

1. See Notes with 'disk2net' command for usage rules and restrictions.
2. When <status> is 'inactive', a 'net2disk?' query returns <scan label> of the last transferred scan, if any.
3. Normally, the <scan label> field is provided in the standardized format specified in Section 6, which includes <experiment name> and <station code> subfields. For backwards compatibility with older versions of the Field System: if <experiment name> and/or <station code> are non-null, the <scan label> parameter is assumed to actually be just the scan name; a scan label of the form '<experiment name>_<station code>_<scan label> is then constructed.
4. Note that the network protocol parameters are set by the 'net_protocol' command.
5. The total number of character in the <scan label>, including field-separator underscore characters, may not exceed 63.

net2disk

net2out – Transfer data directly from network to Mark 5 output

[command list]

Command syntax: net2out = <control> ;
Command response: !net2out = <return code> ;

Query syntax: net2out? ;
Query response: !net2out ? <return code> : <status> : <nowbyte> ;

Purpose: Enable data transfer from network to Mark 5 output; bypass disks

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<control>	char	open | close	-	'open' or 'close' socket

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	active | inactive | waiting | paused	active – connected and data flowing inactive – no socket open; doing nothing waiting – socket open and waiting for a connection paused – socket connected but no data waiting to be read See Note 3.
<nowbyte>	int	-	Total number of bytes transferred to the output since 'open'; the 'open' command resets <nowbyte> to 0.

Notes:

1. See Notes with 'disk2net' command for usage rules and restrictions.
2. Note that the network protocol parameters are set by the 'net_protocol' command.
3. Note that none of the status returns necessarily indicate an error – depends on context. At modest data-transfer speed, <status> may be "paused" (indicating no data waiting to be read) most of the time even if <nowbyte> is incrementing; incrementing <nowbyte> indicates that data are flowing.

net2out

net_protocol – Set network data-transfer protocol

[command list]

Command syntax: net_protocol = <protocol> : [<socbuf size>] : [<workbuf size>] : [<nbuf>] ;
Command response: !net_protocol = <return code> ;

Query syntax: net_protocol? ;
Query response: !net_protocol? <return code> : <protocol> : <socbuf size> : <workbuf size> : <nbuf> ;

Purpose: Set network data-transfer protocol

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<protocol>	char	tcp | udp	tcp	
<socbuf size>	int	bytes	Linux OS buffer size	Used as receive buffer size in 'net2out' and 'net2disk'. Used as send buffer size in 'in2net' and 'disk2net'. Defaults to 0, which causes use of Linux OS defaults buffer size.
<workbuf size>	int	bytes	131072 or last value set	Used as buffer size to send to data sockets in 'in2net' and 'disk2net'.
<nbuf>	int	1-16	8	Number of blocks, each of size <workbuf size>, allocated in a circular FIFO; see Note 2

Notes:
1. Query returns protocol and buffer sizes currently in force.
2. <nbuf> times <workbuf size> must not exceed 134,217,728 bytes.

net_protocol

OS_rev1 – Get details of operating system (query only)

[command list]

Query syntax: OS_rev1? ;
Query response: !OS_rev1 ? <return code> : <OS info, part 1> ;

Purpose: Get detailed information about operating system.

Monitor-only parameters:

Parameter	Type	Values	Comments
<OS info, part 1>	literal ASCII		Primarily for diagnostic purposes

OS_rev1

OS_rev2 – Get more details of operating system (query only)

[command list]

Query syntax: OS_rev2? ;
Query response: !OS_rev2 ? <return code> <OS infor, part 2> ;

Purpose: Get more detailed information about operating system.

Monitor-only parameters:

Parameter	Type	Values	Comments
<OS info, part 2>	literal ASCII		Primarily for diagnostic purposes

OS_rev2

play – Play data from from current play pointer position

[command list]

Command syntax: play = <play arm/on/off> : [<start play pointer>] : [<ROT start>];
Command response: !play = <return code>;

Query syntax: play? ;
Query response: !play ? <return code> : <status> ;

Purpose: Initiate playback from disk data at current or specified play-pointer position.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<play arm/on/off>	char	arm | on | off	off	'arm' - causes Mark5A to pre-fill buffer and prepare to play at position specified by field 2 – see Note 2. 'on' – causes playback to start at position specified by field 2. If field 2 is null, starts playback from current play pointer; Field 2 should be null for 'play=on' command following a successful 'play=arm'. 'off' = stops playback (if active) and unconditionally updates playback pointer to current play position or, if field 2 is non-null, to the position specified. See also Note 1. Cannot be issued while 'record' is 'on' (error). In all play modes, all 64 output tracks are active; if fewer than 64 tracks were recorded, the recorded track set is duplicated to unused output tracks; see Note 2.
<start play pointer>	int	>=0	current play pntr	Absolute byte number in recorded data stream; if null field, maintains current value; if <start play pointer> and <ROT start> are both null fields, play starts at current play pointer value.
<ROT start>	int	#sysclks		For use with Mark 4 correlator only: cause play to start after specified number of sysclk periods (counting from beginning of year); sysclk frequency is normally 32MHz, so these can be very large numbers.

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	arming | armed | on | off | halted | waiting	'arming' – arming is in progress 'armed' – system is armed and ready to start play 'on' – playback active 'off' – playback inactive 'halted' – playback stopped due to reaching end-of-media or (end-of-scan when playback initiated by 'scan_play') 'waiting' – delayed start of playback (special mode for correlator only)

Notes:

1. After play is turned 'on', the user should periodically query 'status' for details; if playback stops on its own accord (due to end-of-media, etc.), this will be reflected in the response to the 'status' query as 'halted', and a 'play' query will show the status as well; a subsequent command to turn play 'off' or 'on' will reset the relevant bits (9-8) in the 'status' response.
2. The 'play=arm' command causes the Mark 5A to prefill its buffers according to the prescribed position so that playing will start almost instantaneously after a subsequent 'play=on' command is issued; this is intended primarily for use at a correlator. The amount of time need to prefill the buffer can range from a few tens of msec to a few seconds. If all disks are good and all data have been recorded properly, the time will be relatively short; however, if difficulties with disks or recorded data are encountered during the prefill period, up to several seconds may

be required. A ‘play?’ query should be issued to verify the system is armed for playback before issuing a ‘play=on’ command. A ‘play=on’ without a preceding ‘play=arm’ will begin play, but after an indeterminate delay.

3. During playback initiated by a ‘scan_play’ command, a ‘play?’ query will indicated the playback status.
4. When playing back in a mode with fewer than 64 tracks, groups of tracks are duplicated so that all 64 track outputs are always active, as follows:

mode	Primary playback tracks	Duplicated playback tracks
‘mark4:8’ or ‘vlba:8’	2-16 even (headstack 1)	Duplicated to 3-17 odd, 18-32 even, 19-33 even on hdstk1; hdstk2 is duplicate of hdstk1
‘mark4:16’ or ‘vlba:16’ - 16	2-33 even (headstack 1)	Duplicated to 2-33 even on hdstk1; hdstk2 is duplicate of hdstk1
‘mark4:32’ or ‘vlba:32’ - 32 tks	2-33 all (headstack 1)	Headstack 1 output is duplicated to Headstack 2
‘mark4:64’ or ‘vlba:64’ - 64 tks	2-33 (headstacks 1 and 2)	None
‘st’ (any submode) – 32 tks	2-33 all (headstack 1)	Headstack 1 output is duplicated to Headstack 2
tvg	equivalent to tracks 2-33	Headstack 1 output is duplicated to Headstack 2

Playback of ‘Mark5A+n:k’ data is to same set of output tracks as for ‘vlba:k’ data.

5. Note that record/play pointers may have values as large as ~$2x10^{13}$ (~44 bits), so pointer arithmetic must be handled appropriately.
6. Playback clock rate is set by the ‘play_rate’ command
7. When playing, the playback pointer will update to show the approximate position. If the playback pointer is noted not to be incrementing, an error flag is set in the ‘status?’ query which can be used as a first order check of proper playback.

play_rate – Set playback data rate; set tvg rate

[command list]

Command syntax: play_rate = <play rate reference> : <rate> ;
Command response: !play_rate = <return code> :;

Query syntax: play_rate? ;
Query response: !play_rate ? <return code> : <track data rate> : <track clock rate> : <clockgen freq> ;

Purpose: Set the playback rate (specified as <track data rate>, <track clock rate> or <clock generator frequency>.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<play rate reference>	char	data | clock | clockgen | ext	clock	'data' – set output track data rate (not including parity) to specified value. 'clock' – set output track clock rate (including parity) to specified value. 'clockgen' – set clock generator chip to specified frequency; max is 40 MHz.. 'ext' – external clock select See also Notes below.
<rate>	real	MHz	8	>0 – set rate to specified value; freq resolution of clock generator chip is ~20 mHz. If in 'tvg' mode, sets on-board clock generator rate regardless of value of <play rate reference>; see Notes.

Monitor-only parameters:

Parameter	Type	Values	Comments
<track data rate>	real	Mbps	Track data rate (without parity); =0 if external clock selected
<track clock rate>	real	MHz	Track clock rate; see Note 1 for relationship to <track data rate>
<clockgen freq>	real	MHz	Internal clock generator frequency; see Note 1 for relationship to <track data rate>

Notes:

1. For a given operating mode, the relationships between <track data rate>, <track clock rate> and <clockgen freq> are as follows:

mode:submode	<track data rate> (Mbps)	Typical 'standard' values of <track data rate>	Corresponding <track clock rate> (MHz)	Corresponding <clockgen> frq (MHz)	Total playback data rate (Mbps)
st:mark4	f	2 | 4 | 8 | 16	9/8*f	9/8*f	32*9/8*f
st:vlba	f	2 | 4 | 8	9/8*f	9/8*f	32*9/8*f
mark4:8	f	2 | 4 | 8 | 16	9/8*f	9/8*f	8*f
mark4:16	f	2 | 4 | 8 | 16	9/8*f	9/8*f	16*f
mark4:32	f	2 | 4 | 8 | 16	9/8*f	9/8*f	32*f
mark4:64	f	2 | 4 | 8 | 16	9/8*f	2*9/8*f	64*f
vlba:8 *or* mark5a+n:8	f	2 | 4 | 8	1.008*9/8*f	1.008*9/8*f	1.008*8*f
vlba:16 *or* mark5a+n:16	f	2 | 4 | 8	1.008*9/8*f	1.008*9/8*f	1.008*16*f
vlba:32 *or* mark5a+n:32	f	2 | 4 | 8	1.008*9/8*f	1.008*9/8*f	1.008*32*f
vlba:64 *or* mark5a+n:64	f	2 | 4 | 8	1.008*9/8*f	2*1.008*9/8*f	1.008*64*f
tvg (see Note 4)	f	any up to 40	f	f	32*f

play_rate

2. Upon a ‘mode’ change, the Mark 5A software automatically makes any necessary adjustments to the clock generator to meet the current <track data rate> value (e.g. as returned by a ‘play_rate?’ query).
3. The value of the ‘play_rate’ parameters has no effect when recording data from an external source; the recording rate is strictly determined by the operating mode and input clock frequency. However, when using the ‘rtime?’ query to determine the remaining available recording time, the ‘play_rate’ parameters must correspond to the input data rate.
4. When recording or playing tvg data, <rate> record/playback aggregate rate is always 32*<rate>; set <rate> equal to 32 for tvg record/playback at 1024Mbps.
5. The maximum clock generator rate is 40 MHz, which results in corresponding maximum <track data rates> and <track clock rates> as follows:

data mode:submode	**<track data rate> (Mbps)**	**<track clock rate> (MHz)**
st:mark4	35.56	40
st:vlba	35.27	40
mark4:8	35.56	40
mark4:16	35.56	40
mark4:32	35.56	40
mark4:64	17.78	20
vlba:8 *or* mark5a+n:8	35.27	40
vlba:16 *or* mark5a+n:16	35.27	40
vlba:32 *or* mark5a+n:32	35.27	40
vlba:64 *or* mark5a+n:64	17.64	20
tvg	40	40

play_rate

position – Get current record and play pointers (query only)

[command list]

Query syntax: position? ;
Query response: !position? <record pointer> : <play pointer>;

Purpose: Get current value of record and play pointers.

Monitor-only parameters:

Parameter	Type	Values	Comments
<record pointer>	int	bytes	If stopped, returns position at which 'record=on' command will begin recording (always appends to existing); if recording, returns current record position.
<play pointer>	int	bytes	If stopped, returns position at which 'playback=on' command will begin playing; can never be greater than current record position.

Notes:

1. Note that record/play pointers may have values as large as ~$2x10^{13}$ (~44 bits), so pointer arithmetic must be handled appropriately.
2. When recording or playing, the corresponding pointer will be updated to show the approximate position. If the respective pointer is noted not to be incrementing during recording or playing, an error flag is set in the 'status?' query which can be used as a first order check of proper operation.

position

protect – Set write protection for active module

[command list]

Command syntax: protect = <on | off> ;
Command response: !protect = <return code> ;

Query syntax: protect? ;
Query response: !protect? <return code> : <protect on/off>;

Purpose: Set write protection on/off for active disk module

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<off>	char	on | off	off	

Notes:

1. A ‘protect=on’ command prevents any additional writing to module.
2. A ‘protect=off” command allows writing to a module.
3. A ‘protect=off’ command is required to *immediately* precede a ‘reset=erase’, ‘reset=erase_last_scan’ or ‘VSN=…’ command, even if protection is already off. This protects the module from any accidental erasure or rewriting of the VSN.

protect

record – Record data from Mark 5 input to disks

[command list]

Command syntax: record = <record on/off> : <scan label> : [<experiment name>] : [<station code>] ;
Command response: !record = <return code> ;

Query syntax: record? ;
Query response: !record ? <return code> : <status>: <scan#> : <scan label> ;

Purpose: Turn recording on|off; assign scan label

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<record on/off>	char	on | off		'on' automatically appends to the end of the existing recording; 'bypass' is active while recording is 'on'. 'off' stops recording and leaves system in 'bypass' mode.
<scan label>	ASCII	See Note 8		Relevant only if record is 'on';. '+' character not allowed. No checking is done for duplicate scan labels. See also Note 7.
<experiment name>	ASCII	See Note 8		(Optional) Experiment name; ignored if <record on/off> is 'off'; intended for use only for backwards compatibility - see Note 7.
<station code>	ASCII	See Note 8		(Optional) Station code; ignored if <record on/off> is 'off'; intended for use only for backwards compatibility - see Note 7.

Monitor-only parameters:

Parameter	Type	Values	Comments
<scan#>	int		Sequential scan number; starts at 1 for first recorded scan.
<status>	char	on | off | halted | throttled | overflow | waiting	'halted' indicates end-of-media was encountered while recording. 'throttled', 'overflow' and 'waiting' are all error conditions.
<scan label>	ASCII		Scan label - see Notes 7 and 8.

Notes:

1. The formatter output track numbers that are actually recorded in each mode are as follows (see also Mark 5 memo 11.1):

mode:submode	Recorded track#'s	FPDP bit streams
'mark4:8' or 'vlba:8' – 8 tks	2-16 even (headstack 1)	Trk 2 to FPDP streams 0,8,16,24; trk 4 to 1,9,17,25; etc.
'mark4:16' or 'vlba:16' – 16 tks	2-33 even (headstack 1)	Trk 2 to FPDP streams 0,16; trk 4 to 1,17; etc.
'mark4:32' or 'vlba:32' – 32 tks	2-33 all (headstack 1)	Correspond to FPDP bit streams 0-31, respectively
'mark4:64' or 'vlba:64' – 64 tks	2-33 (headstacks 1 and 2)	Trks 2 from both hdstks mux'ed to FPDP bit stream 0, etc.
'st' (any submode) – 32 tks	2-33 all (headstack 1)	Correspond to FPDP bit streams 0-31, respectively
tvg	equivalent to tracks 2-33	TVG data; correspond to FPDP bit streams 0-31, respectively,

2. The recording rate is controlled by the track clock from the formatter *except* in 'tvg' mode. The on-board TVG is driven by the same clock generator that sets the output clock rate during playback; therefore, when 'tvg' mode is active in record or bypass, the TVG is driven at the clock generator frequency, which is set by the 'play_rate' command.

record

3. After record is turned ‘on’, the user should periodically query ‘status’ for details; if recording stops on its own accord (due to end-of-media, etc.), this will be reflected in the response to the ‘status’ query as ‘recording stopped’, and a ‘record’ query will show the status as ‘halted’; a subsequent command to turn record ‘off’ or ‘on’ will reset the relevant bits (5-4) in the ‘status’ response.
4. When recording, the record pointer will update to show the approximate position. If the record pointer is noted not to be incrementing, an error flag is set in the ‘status?’ query which can be used as a first order check of proper recording.
5. When <status> is ‘off’, a ‘record?’ query returns the <scan label> of the last recorded scan, if any.
6. Typical causes for status errors:
    a. “throttled” – data rate from formatter is too fast for disks to keep up (flag received by I/O board from SS)
    b. “overflow” – FIFO overflow on StreamStor card (reported by SS card)
    c. “waiting” – formatter clock has stopped or is faulty
7. Normally, the <scan label> field is provided in the standardized format specified in Section 6, which includes <experiment name> and <station code> subfields. For backwards compatibility with older versions of the Field System: if <experiment name> and/or <station code> are non-null, the <scan label> parameter is assumed to actually be just the scan name; a scan label of the form ‘<experiment name>_<station code>_<scan label> is then constructed.
8. The total number of characters in the <scan label>, including field-separator underscore characters, may not exceed 63.

record

recover – Recover data which was overwritten or terminated abnormally during recording [command list]

Command syntax: recover = <recovery mode> ;
Command response: !recover = <return code> : <recovery mode>;

Query syntax: recover? ;
Query response: !recover ? <return code> ;

Purpose: Recover data which was overwritten or which was terminated abnormally during recording.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<recovery mode>	int	0 | 1 | 2	1	0 – attempt to recover data from scan that was terminated abnormally during recording; see Note 1. 1 – attempt to recover from accidental use of 'sstest' or 'WRSpeed Test'; see Note 2. 2 – attempt to recover from StreamStor abnormality; see Note 3.

Notes:

1. A scan terminated abnormally during recording (for example, by a power failure or a keyswitch being accidentally turned to the 'off' position) will not be accessible unless special actions are taken to recover it by forcing the record pointer to the end of recorded data; the scan will be overwritten if a new 'record=on' command is issued before a recovery attempt is made. It is suggested that a 'record=off' command be tried before a 'recover=0' command; this will not cause any harm and might fix the problem by itself. It has also been reported that success with 'recover=0' is demonstrably higher if a 'scan_set' command to select a scan [seemingly any scan, but perhaps preferably to the last (incomplete) scan] is issued before the 'recover=0' is attempted.
2. The utility programs '*sstest*' and '*WRSpeedTest*' will overwrite any existing data near the beginning of a disk module, but will prevent access to user data recorded beyond that point. A 'recover=1' command will attempt to recover the data beyond the overwritten section; the overwritten data are irrecoverable.
3. (Most common) Try recover=2 to recover data that were erased, or if the record pointer has been set to a point near the beginning (often to zero).

recover

replaced_blks – Get number of replaced blocks on playback (query only)

[command list]

Query syntax: replaced_blks? ;
Query response: !replaced_blks? <return code> : <disk 0> : <disk 1> : <disk 2> : <disk 3> : < disk 4> : <disk 5> : <disk 6> : <disk 7> : <total replaced blks> ;

Purpose: Get number of replaced blocks during playback on disk-by-disk basis.

Monitor-only parameters:

Parameter	Type	Values	Comments
<disk 0>	int		Number of replaced blocks on disk 0
<disk 1>	int		Number of replaced blocks on disk 1
<disk 2>	int		Number of replaced blocks on disk 2
<disk 3>	int		Number of replaced blocks on disk 3
<disk 4>	int		Number of replaced blocks on disk 4
<disk 5>	int		Number of replaced blocks on disk 5
<disk 6>	int		Number of replaced blocks on disk 6
<disk 7>	int		Number of replaced blocks on disk 7
<total replaced blks>	int		Total number of replaced blocks.

Notes:

1. If a disk is unable to provide a requested 65KB (actually 0xfff8 bytes) block of data within the allowed time limits, due to a slow or failed drive, the Mark 5A replaces the requested data block with a data block with even parity that can be detected by as invalid by a correlator. See 'Mark 5A User's Manual' for details.
2. Drive statistics and replaced-block counts are cleared and re-started whenever a new disk module is mounted or a 'start_stats' command is issued. Replaced-block counts restart on each 'play=on' or 'scan_play=on' command.
3. If the case of a totally failed drive, the replaced-block count for that drive will be 0 since the StreamStor ceases to ask for data from that drive, but the <total replaced blks> will be accurate. Statistics gathered from the 'get_stats?' query should be used to help diagnose the failed drive.
4. Replaced-block statistics are updated only after playback has ceased (i.e. replaced-block statistics are not updated during playback).

replaced_blks

reset – Reset Mark 5 unit (command only)

[command list]

Command syntax: reset = <control> ;
Command response: !reset = <return code> ;

Purpose: Reset system; mount/dismount disks

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<control>	char	erase | erase_last_scan | abort		'erase' intiates the setting of record and play pointers to zero (i.e. beginning of media); effectively erasing media; 'erase_last_scan' erases the last recorded scan; sets record pointer to end-of-scan just prior to erased scan; sets play pointer to beginning of scan just prior to erased scan. 'abort' aborts active disk2net, disk2file or file2disk transfers (only) – See Note 2 System is always left in 'bypass' mode after any reset command. See Note 1.

Notes:

1. The former 'reset=mount' and 'reset=dismount' commands are no longer supported; the keyswitches associated with the disk modules are used for all mount and dismount operations.
2. 'reset=abort' returns immediately, but there may be a delay of up to two seconds before the data transfer stops. During this delay, a 'status?' query will show what is happening. The 'reset=abort' command simulates the end of data by setting 'nowbyte=endbyte', which then executes a normal termination.
3. A 'protect=off' command is required *immediately* prior to a 'reset=erase' or 'reset=erase_last_scan' command, even if protection is already off.

rtime – Get remaining record time on current disk set (query only)

[command list]

Query syntax: rtime? ;
Query response: !rtime ? <return code> : <remaining time> : <remaining GB> : <remaining percent> : <mode> : <submode> : <track data rate> : <total recording rate> ;

Purpose: Get remaining record time of current disk set; assumes recording will be in the mode currently set by the 'mode' command and data rate set by 'play_rate' command.

Monitor-only parameters:

Parameter	Type	Values	Comments
<remaining time>	real	seconds	Approximate remaining record time for current 'mode' and 'play_rate' parameters; Requires that 'play_rate' be set to current record rate – see Notes.
<remaining GB>	real	GB	GB remaining on current disk set (1 GB = 10^9 bytes)
<remaining percent>	real	0-100	Remaining percentage of disk space still available
<mode>	char	mark4 | vlba | st | tvg	Mode assumed in calculation of <remaining time>. See 'mode' command.
<submode>	char	8 | 16 | 32 | 64 | mark4 | vlba	Submode assumed in calculation of <remaining time>
<track data rate>	real	MHz	Track data rate assumed in calculation of <remaining time>; see 'play_rate' command
<total recording rate>	real	Mbps	Total data rate to disks; based on assumed mode, submode and track data rate

Notes:

1. Since recording rate is controlled by an external clock (except in 'tvg' mode), the Mark 5A has no knowledge of the record data rate. However, if 'play_rate' is set to match the recording rate, the remaining recording time can be accurately estimated.
2. Each 'rtime?' query returns an updated estimate during recording; a somewhat more accurate estimate is obtained when recording is stopped and the effects of any slow or bad disks can be more accurately measured.

rtime

scan_check – Get scan parameters (query only)

[command list]

Query syntax: scan_check? ;
Query response: !scan_check ? <return code> : <scan#> : <scan label> : <data mode> : <data submode> : <start time> : <scan length> : <track data rate> : <#missing bytes>;

Purpose: Determine parameters of recorded scan specified by current scan pointer (e.g. value of 'scan_set'). Please be especially attentive to Note 1 of 'data_check' for the track set that must be recorded.

Monitor-only parameters:

Parameter	Type	Values	Comments
<scan#>	int		Start at 1 for first recorded scan
<scan label>	literal ASCII		
<data mode>	char	st | mark4 | vlba | tvg | SS	See 'mode' command for explanation of data modes; 'tvg' corresponds to VSI test pattern;' SS' corresponds to StreamStor test pattern
<data submode>	char	8 | 16 | 32 | 64 | mark4 | vlba	'8|16|32|64' if <data mode> is 'mark4' or 'vlba'; 'mark4|vlba' if <data mode> is 'st'
<start time>	time		Time tag at first frame header in scan. See Note 5 below.
<scan length>	time		Scan length in seconds
<track data rate>	real	Mbps	Excludes parity bits; will always be 0.125, 0.25, 0.5, 1, 2, 4, 8 or 16 (Mbps)
<#missing bytes>	int	See Note 5	Should always be =0 for normally recorded data. >0 indicates #bytes that have been dropped somewhere within scan <0 indicates #bytes that have been added somewhere within scan

Notes:

1. The 'scan_check' query will be honored only if record and play are both off.
2. The 'scan_check' query does not affect the play pointer.
3. The 'scan_check' query essentially executes a 'data_check' at the beginning of a scan, followed by a 'data_check' at the end of the scan. This allows information about the selected scan to be conveniently determined.
4. Regarding the 'data time' value returned by the 'data_check?', 'scan_check?' and 'track_check?' queries: The Mark 4 time-tags contain the day-of-year (DOY) but only the final digit of the year; the VLBA time-tags contain, instead, the last 3 digits of the Julian day number (misnamed MJD). To show the year and DOY in the returned values of 'data time' requires some assumptions. For Mark 4, we assume the most recent year consistent with the unit-year and DOY written in the Mark 4 time-tag; this algorithm reports the proper year provided the data were taken no more than 10 years ago. For VLBA, we assume the most recent Julian Day Number (JDN) consistent with the last 3 digits available in the VLBA time-tag; this algorithm reports the proper year provided the data were taken no more than 1000 days ago.
5. The <#missing bytes> parameter is calculated as the difference the expected number of bytes between two samples of recorded data based on embedded time tags and the actual observed number of bytes between the same time tags. The reported number is the *total* number of bytes missing (or added) between the two sample points.

scan_check

6. When the <data mode> is determined to be ‘tvg’ or ‘SS’, three integer diagnostic parameters are returned following <data mode>. A buffer of data is read (typically ~1MB) from the disks at the present play pointer position, which is analyzed. The following information is returned:
    a. Position of first 32-bit word (starting from zero) in buffer containing first valid word in the ‘tvg’ or ‘SS’ sequence.
    b. Position of first 32-bit word which is not in the proper order of the ‘tvg’ or ‘SS’ sequence.
    c. Size of block read.

   For a properly operating system, the first number will be 0 and the 2nd and 3rd numbers will have the same values.

scan_check

scan_play – Play scan specified by current value of scan_set parameters

[command list]

Command syntax: scan_play = <arm/on/off>;
Command response: !scan_play = <return code> ;

Query syntax: scan_play? ;
Query response: !scan_play ? <return code> : <status>;

Purpose: Play scan specified by current value of scan_set parameters

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<arm/on/off>	char	arm | on | off	on	'arm' – causes Mark5A to pre-fill buffer and prepare to play at current <start scan_play> position – see Note 2. 'on' – starts playing at current <start scan_play> position. 'off' – stops playing at current position; does not affect <start scan_play> position

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	arming | armed | active | inactive | halted	'arming' – arming is in progress 'armed' – system is armed and ready to play 'active' –playback is active 'inactive' – playback is inactive 'halted' indicates end-of-scan encountered; requires 'play-off' command to update play pointer. See Notes.

Notes:

1. 'scan_play' starts playback at the <start scan_play> position as set and/or reported by 'scan_set' and ends at the corresponding <end scan_play> position.
2. The 'scan_play=arm' command causes the Mark 5A to prefill its buffers so that the playing will start almost instantaneously after a subsequent 'scan_play=on' command is issued and is intended primarily for use at a correlator. The amount of time need to prefill the buffer can range from a few tens of msec to a few seconds. If all disks are good and all data have been recorded properly, the time will be relatively short; however, if difficulties with disks or recorded data are encountered during the prefill period, up to several seconds may be required. A 'scan_play?' query should be issued to verify the system is armed for playback before issuing a 'scan_play=on' command. A 'scan_play=on' without a preceding 'scan_play=arm' will begin play, but after an indeterminate delay.
3. At end of scan_play, a 'play?' query will return 'halted'. May also be stopped by 'play=off' command; play pointer will be updated to stop position. Scan pointer is not affected.
4. During playback initiated by a 'scan_play' command, a 'play?' query will indicate the playback status.

scan_play

scan_set – Set scan for scan_check, scan_play, disk2file and disk2net

[command list]

Command syntax: scan_set = <search string> : [<start scan_play>] : [<end scan_play>] ;
Command response: !scan_set = <return code> ;

Query syntax: scan_set? ;
Query response: !scan_set? <return code> : <scan#> : <scan label> : <start scan_play byte#> : <end scan_play byte#> ;

Purpose: Set scan for scan_check, scan_play, disk2file and disk2net.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<search string>	int or ASCII	scan number | scan label | 'inc' | 'dec' | 'next'	last recorded scan	First attempts to interpret as scan number (first scan is number 1); if not numeric or no match, attempts to match all or part of existing scan label, case insensitive (see Note 1). 'inc' increments to next scan; cycles back to first scan at end; 'dec' decrements to previous scan. 'next' finds next scan with previous value of <search string>. If null field, defaults to last fully recorded scan (e.g. if recording is in progress, defaults to previous scan).
<start scan_play>	char | time | int	s | c | e | s+ | <time> | +<time> | -<time> | +<bytes> | -<bytes>	s	s|c|e|s+: Set start play position to 'start', 'center', 'end' (actually ~1MB before end) of scan, or specified <time> within scan; this is convenient if you want to do a subsequent 'data_check' or 'track_check' at a prescribed position. 's+' sets play pointer to 65536 bytes past the start of the scan. <time>: time within scan: see Notes 2 & 3 +<time>: offset time from beginning of scan (i.e. '+30s' will start 30 seconds from beginning of scan) -<time>: offset time from end of scan (i.e. '-30s' will start 30 seconds before end of scan) +<bytes>: offset number of bytes from beginning of scan. -<bytes>: offset number of bytes from end of scan
<end scan_play>	time | int	<time> | +<time> | -<time> | +<bytes> | -<bytes>	end-of scan	<time>: Time at which to end playback; see Notes 2 & 3. If preceded by '+', indicates duration of data (in record-clock time) from <start scan_play> time. +<time>: offset time from <start scan_play> position. -<time>: offset time from end-of-scan +<bytes>: offset bytes from <start scan_play> position -<bytes>: offset bytes from end of scan

Monitor-only parameters:

Parameter	Type	Values	Comments
<scan#>	int		Returns current 'scan pointer', which relates only to the 'scan_play' command and to the 'scan_check" and 'scan_dir' queries. Followed by '+' if scan automatically switched to another disk module (not yet implemented). Prefaced by '+' if scan is continuation from another disk module (not yet implemented).
<scan label>>	ASCII		Scan label to which scan_set is currently pointed.
<start scan_play byte#>	int	bytes	Absolute byte position to start scan_check, scan_play, disk2file or disk2net.
<end scan_play byte#>	int	bytes	Absolute byte position to stop scan_check, scan_play, disk2file or disk2net.

scan_set

Notes:

1. If <search string> is all numeric, scan_set will first try to interpret it as a scan number. If it is not all numeric or the scan number does not exist, scan_set will find the first scan label that matches all or part of the corresponding non-null subfields in <search string>; null subfields in <search string> match all scans. All searches start from the first scan except if 'scan_set=next'; if 'scan_set' is already pointing at last scan, then 'scan_set=next' will start search at first scan. Searches are case insensitive. Examples:

<search string>	**Matches**
105	Scan #105, if it exists; otherwise, first scan label containing '105' anywhere (e.g. 'grf103_ef_123-1056')
grf103_	First scan label with 1st subfield containing 'grf103'
_EF	First scan label with 2nd subfield containing 'EF' (searches are case insensitive)
__1056	First scan label with 3rd subfield containing '1056'
_ef_1056	First scan label with 2nd subfield containing 'ef' and 3rd subfield containing '1056'

2. When 'record=off' is issued or end-of-media (following a 'record=on') is encountered, the default scan playback parameters are set to playback the entire just-recorded scan.
3. If the <start scan_play> or <end scan_play> parameter is a <time> value, this time must be specified with sufficient significance to resolve any ambiguity. For example, '30s' would set the scan_play pointer to start at the first '30s' mark in the scan (regardless of the value of the minute). If a calculated byte position is outside the bounds of a scan, an error code '0', but the default will be retained and an error code will be posted, which can be recovered by an 'error?' or 'status?' query.
4. A 'scan_set=' command is not allowed during active data transfers.

skip – Skip forward/backwards specified # of bytes during playback or net2out

[command list]

Command syntax: skip = <requested skip> ;
Command response: !skip = <return code> ;

Query syntax: skip? ;
Query response: !skip ? <return code> : <actual skip> ;

Purpose: Skip forward/backwards specified # of bytes during playback or net2out

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<requested skip>	int	multiple of 8		>0 – skip forward; <0 – skip backward; must be multiple of 8 bytes Used to synchronize data to correlator – see Notes. If not playing, increments start-playback position.

Monitor-only parameters:

Parameter	Type	Values	Comments
<actual skip>	int		Actual value of skip executed. See Notes.

Notes:

1. During playback or net2out, the 'skip' command will synchronously skip over a prescribed amount of data, either positive or negative, and is intended for synchronizing the data to the correlator. A skip of any size may be requested, however the actual skip executed is limited to be within the data currently within the SS on-board 512MB buffer; the size of the actual executed skip can be determined with a subsequent 'skip?' query. Subsequent 'skip' commands can then be used to make up the remainder of the total desired skip, if necessary. During normal playback, a maximum forward or backward skip of ~256MB is possible (except immediately after starting playback and possibly after a preceding large skip), which corresponds to ~2 seconds of data at 1 Gbps and longer at slower playback rates. Normally, it should be possible to control the position and timing of the start of playback so that skips larger than the available buffer size are not necessary.

SS_rev1 – Get StreamStor firmware/software revision levels, part 1 (query only)

[command list]

Query syntax: SS_rev1? ;
Query response: !SS_rev1 ? <return code> : <SS info 1> ;

Purpose: Get information on StreamStor firmware/software revision levels.

Monitor-only parameters:

Parameter	Type	Values	Comments
<SS info 1>	literal ASCII		Primarily for diagnostic purposes.

SS_rev1

SS_rev2 – Get Streamstor firmware/software revision levels, part 2 (query only)

[command list]

Query syntax: SS_rev2? ;
Query response: !SS-rev2 ? <return code> : <SS info 2> ;

Purpose: Get more information on StreamStor firmware/software revision levels.

Monitor-only parameters:

Parameter	Type	Values	Comments
<SS info 2>	literal ASCII	-	Primarily for diagnostic purposes.

SS_rev2

start_stats – Start gathering disk-performance statistics

[command list]

Command syntax: start_stats = [<t0> : <t1> :….: <t6>] ;
Command response: !start_stats = <return code> ;

Query syntax: start_stats? ;
Query response: !start_stats ? <return code> : <t0> : <t1> :….: <t6> ;

Purpose: Start gather disk performance statistics

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<tn>	time		0.001125s 0.00225s 0.0045s 0.009s 0.018s 0.036s 0.072s	Clears and restarts gathering of drive statistics. See Notes. Seven optional values define 8 bins corresponding to drive-response (i.e. transaction completion) times; values must increase monotonically; a separate set of bins is maintained for each mounted drive. The count in a bin is incremented according to the following rules, where 't' is drive-response time of a single read or write transaction: Bin 0: t<t0 Bin 1: t0<t<t1 . Bin 6: t5<t<t6 Bin 7: t>t6

Notes:

1. Drive statistics and replaced-block counts are cleared and re-started whenever a new disk module is mounted or a 'start_stats' command is issued. Read drive statistics with 'get_stats' query. Bin values are common for all drives. Each count within a bin represents a transfer of 65528 bytes (2^{16}-8).
2. The 'start_stats' command may not be issued during active recording or playback.

start_stats

status – Get system status (query only)

[command list]

Query syntax: status? ;
Query response: !status ? <return code> : <status word> ;

Purpose: Get general system status

Monitor-only parameters:

Parameter	Type	Values	Comments
<status word>	hex	-	Bit 0 – (0x0001) system 'ready' Bit 1 – (0x0002) error message(s) pending; (message may be appended); messages may be queued; error is cleared by this command. See also 'error?' query Bit 2 – (0x0004) not used Bit 3 – (0x0008) one or more 'delayed-completion' commands are pending. Also set whenever any data-transfer activity, such as recording, playing, or transfer to or from disk or net, is active or waiting. --- Bit 4 – (0x0010) one or more 'delayed-completion' queries are pending Bit 5 – (0x0020) Disk-FIFO mode Bit 6 - (0x0040) record 'on' Bit 7 - (0x0080) media full (recording halted) --- Bit 8 - (0x0100) playback 'on' Bit 9 - (0x0200) end-of-scan or end-of-media (playback halted) Bit 10 – (0x0400) recording can't keep up; some lost data Bit 11 – (0x0800) not used --- Bit 12 – (0x1000) disk2file active Bit 13 – (0x2000) file2disk active Bit 14 – (0x4000) disk2net active Bit 15 – (0x8000) net2disk active or waiting --- Bit 16 – (0x10000) in2net sending (on) Bit 17 – (0x20000) net2out active or waiting Bit 18 – (0x40000) not used Bit 19 – (0x80000) not used --- Bits 20-27 are set properly even if a data transfer is in progress. Bit 20 – (0x100000) Bank A selected Bit 21 – (0x200000) Bank A ready Bit 22 – (0x400000) Bank A media full or faulty (not writable) Bit 23 – (0x800000) Bank A write protected --- Bit 24 – (0x1000000) Bank B selected Bit 25 – (0x2000000) Bank B ready Bit 26 – (0x4000000) Bank B media full or faulty (not writable) Bit 27 – (0x8000000) Bank B write protected

status

task_ID – Set task ID (primarily for correlator use)

[command list]

Command syntax: task_ID = <task_ID> ;
Command response: !task_ID = <return code> ;

Query syntax: task_ID? ;
Query response: !task_ID ? <return code> : <task_ID> ;

Purpose: Set task ID (primarily for correlator use)

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<task_ID>	int			For use with Mark 4 correlator only: Causes Mark 5 system to listen to only ROT broadcasts with the corresponding 'task ID'. See Notes.

Notes:

1. The 'task_ID' command is used in conjunction with the 'play' command for accurate synchronization of Mark 5 playback-start with correlator ROT clock.

task_ID

track_check – Check data on selected track (query only)

[command list]

Query syntax: track_check? ;
Query response: !track_check ? <return code> : <data mode> : <data submode> : <data time> : <byte offset> : <track frame period> : <track data rate> : <decoded track#> : <#missing bytes>;

Purpose: Check recorded track which, on playback, will output data to track pointed to by current 'track_set' value.

Monitor-only parameters:

Parameter	Type	Values	Comments
<data mode>	char	st | mark4 | vlba | tvg | SS	See 'mode' command for explanation of data modes; 'tvg' corresponds to VSI test pattern;' SS' corresponds to StreamStor test pattern '?' indicates unknown format. **Note** – If a Mark 5B module is present, a 'data_check' is done instead – See Note 6
<data submode>	char	8 | 16 | 32 | 64 | mark4 | vlba	'8|16|32|64' if <data mode> is 'mark4' or 'vlba'; 'mark4|vlba' if <data mode> is 'st'
<data time>	time		Time tag from next 'track' frame header beyond current play pointer. See Note 5 of 'scan_check'.
<byte offset>	int	bytes-	Byte offset from current play pointer to beginning of next 'track' frame header of target track
<track frame period>	time		Time tag difference between adjacent track frames; allows original track data rate to be determined.
<track data rate>	real	MHz	Track data rate of source data from formatter.
<decoded track#>	int		Track# decoded from auxiliary data field of target track; followed by 'D' if track is a 'duplicated' track; followed by'?' if unallowed track# in this position. See Note 3.
<#missing bytes>	int	bytes	Number of missing bytes between last and current 'track_check'; Should be =0 if immediately previous 'track_check' was within same scan Meaningless if immediately previous 'track_check' was in a different scan. See Note 4. See also Note 6 in 'scan_check'

Notes:

1. The 'track_check' query will be honored only if record and play are both off.
2. The 'track_check' query checks data beginning at the current position of the play pointer; the play pointer is not affected.
3. The 'track_check' query targets the first of the two selected 'track_set' tracks and executes the following actions:
   a. Determines the data mode/submode based on the format of the disk data.
   b. If the target track is a track which is actually recorded in this mode/submode (see 'mode' command Notes), several frames of data are collected from the expected position of this track in the disk data. If the target track is not recorded, the data are collected from the position of the recorded track number which, during playback, is duplicated onto the target track (see 'play' command Notes) in this mode/submode.
   c. A 'track frame header' is extracted from the collected data and the embedded <data time> and <track#> information is decoded. Note that the <decoded track#> will match the target track only in the case in which the target track was actually recorded.
4. Further analysis is done to determine the <track frame period> and <#missing bytes>. A 'blank' is returned in the <#missing bytes> field if the # of missing bytes cannot be calculated.

track_check

5. Regarding the ‘data time’ value returned by the ‘data_check?’, ‘scan_check?’ and ‘track_check?’ queries: The Mark 4 time-tags contain the day-of-year (DOY) but only the final digit of the year; the VLBA time-tags contain, instead, the last 3 digits of the Julian day number (misnamed MJD). To show the year and DOY in the returned values of ‘data time’ requires some assumptions. For Mark 4, we assume the most recent year consistent with the unit-year and DOY written in the Mark 4 time-tag; this algorithm reports the proper year provided the data were taken no more than 10 years ago. For VLBA, we assume the most recent Julian Day Number (JDN) consistent with the last 3 digits available in the VLBA time-tag; this algorithm reports the proper year provided the data were taken no more than 1000 days ago.
6. When a Mark 5B module is inserted into the Mark 5A (operating in so-called ‘Mark 5A+’ mode), a ‘track_check?’ query is meaningless since Mark 5B has no notion of ‘tracks’. Instead, a ‘track_check?’ query performs a ‘data_check’. This is done to maintain backward compatibility with existing Mark 4 correlator software.

track_check

track_set – Select tracks for monitoring with DQA or 'track_check'

[command list]

Command syntax: track_set = <track A> : <track B> ;
Command response: !track_set = <return code> ;

Query syntax: track_set? ;
Query response: !track_set ? <return code> : <track A> : <track B> ;

Purpose: The 'track_set' command serves a two-fold purpose: 1) to select two tracks to be output to the Mark 4 decoder or VLBA DQA and 2) to select the track examined by the 'track_check' query.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<track A>	int/char	2-33 (hdstk 1) 102-133 (hdstk 2) | inc | dec	15	Track selected to be sent to DQA/decoder channel A; track to be analyzed by 'track_check'. Default is headstack 1; add 100 for headstack 2, if present. Track numbers follow the 'VLBA' convention; i.e. 2-33 for headstack 1, 102-133 for headstack 2. 'inc' increments current value – see Note 3; 'dec' decrements current value. If null field, current value is maintained.
<track B>	int/char	2-33 (hdstk 1) 102-133 (hdstk 2) | inc | dec	16	Track selected to be sent to DQA/decoder channel B. 'inc' increments current value – see Note 3; 'dec' decrements current value. If null field, current value is maintained.

Notes:

1. Note that tracks are duplicated according to the table in the Notes with the 'play' command. Any of the 'primary' or 'duplicated' tracks may be selected to go to the DQA/decoder.
2. <track A> is also used as the track to be examined by the 'track_check' query and should correspond to a track that is actually recorded in the selected data mode (see table with 'record' command).
3. The 'inc' value increments the current selected track value by one; cycles through all 32 tracks on each headstack, then begins again. This is a convenient method of cycling through all tracks during system testing.

track_set

VSN – Write extended-VSN to permanent area

[command list]

Command syntax: VSN = <VSN> ;
Command response: !VSN = <return code> ;

Query syntax: VSN? ;
Query response: !VSN ? <return code> : <extended VSN> : <status> :
[: <disk#> : <original S/N> : <new S/N> : 'Disk serial-number mismatch'] ;

Purpose: Write module extended-VSN (volume serial number) to permanent area on module

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
VSN	char			Permanent 8-character VSN, analogous to tape VSN, which survives 'reset=erase' command and module conditioning (example: 'MPI-0153'). VSN format rules are enforced – see Note 4. The module capacity and maximum data rate for the extended-VSN are calculated and appended to the VSN to create the 'extended-VSN' (example 'MPI-0153/960/1024')

Monitor-only parameters:

Parameter	Type	Allowed values	Comments
<extended VSN>	char		Example: 'MPI-0153/960/1024'; see Notes 4 and 5.
<status>	char	OK | Unknown | Fail	OK – disk serial #'s on current set of disks matches serial #'s when VSN was last written. Unknown – disk serial #'s have not been written Fail – current disk serial #'s do not match serial #'s when VSN was last written. See Note 6.
Following parameters are returned only if <status> is 'Fail":			
<disk#>	int	0-7	First disk# in module in which there is a serial-number discrepancy
<original S/N>	char		Serial number of disk in position <disk#> when VSN was written
<new S/N>	char		Serial number of disk now in position <disk#>
'Disk serial-number mismatch'	char		Warning message

Notes:

1. The 'VSN=..' command is normally issued only when the module is first procured or assembled, or when the disk configuration is changed. The serial numbers of the resident disks are noted.
2. The 'VSN?' query compares the serial numbers of the original disks to the serial numbers of the currently-resident disks and reports only the first discrepancy. Issuing a 'VSN=…' command or a 'reset=erase' command will update the disk-serial# list to the currently-resident disks.
3. A 'protect=off' command is required *immediately* preceding a 'VSN=' command, even if protection is already off.
4. The format of the extended-VSN is "VSN/capacity(GB)/maxdatarate(Mbps)" – example 'MPI-0153/960/1024'. The following rules are enforced by the Mark 5A software:
   a. VSN – Must be 8 characters in length and in format "ownerID-serial#" (for parallel-ATA modules) or "ownerID+serial#" (for serial-ATA modules, when they become available)

VSN

   b. ownerID – 2 to 6 upper-case alphabetic characters (A-Z). The ‘ownerID’ must be registered with Jon Romney at NRAO (jromney@nrao.edu) to prevent duplicates. Numeric characters are not allowed. Any lower-case characters will automatically be converted to upper case.
   c. serial# - numeric module serial number, with leading zeroes as necessary to make the VSN exactly 8 characters long. Alphabetic characters are not allowed in the serial#.
5. Mark5A will compute the capacity of the module in GB and the maximum data rate in Mbps (number of disks times 128 Mbps) and append these to the VSN to create the extended VSN. Module capacity in GB is calculated as capacity of the smallest disk, rounded down to nearest 10GB, and multiplied by the number of disks in the module.
6. The recorded disk serial #’s are updated each time a scan is recorded.

VSN