MASSACHUSETTS INSTITUTE OF TECHNOLOGY
HAYSTACK OBSERVATORY
WESTFORD, MASSACHUSETTS 01886
25 May 2012

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

TO: Distribution
FROM: Alan R. Whitney, Dan L. Smythe, and Chester A. Ruszczyk
SUBJECT: Mark 5C Command Set Version 2.0

Note: The current version of the Mark 5C program is called '*drs'*, for the VLBI Data Recording Service and will adhere to the command set in this document.

1. *drs* program

The commands detailed in this memo are implemented by a program named *drs* and control the DIM functionality of the Mark 5C VLBI data recording system. The details concerning the operation of *drs* are available in documents at http://web.haystack.mit.edu/mark5/Mark5.html. The DOM functionality is not be handled by this application and will depend on the particular end use of the Mark 5C, e.g., a software correlator with fuseMk5 providing read access to the Conduant disk modules or using mark5 utilities of the software correlator, e.g. m5cp.

The startup command-line for *drs* is as follows:

drs –m [0|1|2|3|4] –e 'command'

drs supports the following command line options:

arg.	Parameter	default	description
-e	*command*		execute command immediately after starting
-f	*file*		execute the mark5c commands contained in *file* upon start
-h			display program version number, print help information, and quit
-m	*level*	1	set output message level to *level* (between 0 and 4)
-s	*number*	5	set maximum number of simultaneous connections to *number*
-b	*path*		set STREAMSTOR_BIB_PATH to *path*
-p	*path*		set the path where system programs can live (see § 8.4.2)

The order of the supplied arguments is random. Passing of unsupported arguments or parameter will cause the program to print help information and exit. The program will accept multiple -e arguments, each being run in order. Alternatively, a single -e argument can accept a string with multiple semicolon separated commands. Note that for such a string, enclosing quotes will be required to prevent the shell from interpreting the semicolon(s). For convenience, any number of commands that are to be run upon starting *drs* can be listed in a text file specified with the -f option. The order of these commands will be run sequentially. In both the -e and -f cases, each command will be executed to completion before continuing to the next; commands resulting in delayed completion shall cause startup to wait. Specifying the -b command line argument will force *drs* to use the setenv() function call, obviating the need for the environment variable to be set before starting mark5c. This is useful at boot time when the system environment variables are not yet accessible. Examples:

−> drs -s 3 -m 0 -e "disk state mask=1:0:1;MAC_list=00.50.8D.E9.36.F4;" &

−> drs -s 7 -m 1 -b /usr/local/share/streamstor/bib &

The behavior of *drs* should not depend on the filename of the executable (or links to it). Command line switches should not change the functionality of the software (i.e., there should be no FIFO mode).

2. Notes on DIM Command set

Note the following with respect to the command set:

1. Processing of all of the commands/queries expect the VSI-S communications protocol and command/response syntax. Later versions will also support *remote procedure calls* (RPCs).
2. Commands/queries are case *insensitive*.
3. Versions of program *drs* 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.
4. Commands that are not implemented have a (TBD) "To Be Done" next to the specific command and will be added in future versions. Note, DTS_id? provides the version number of the Mark5C Command Set that is supported as the last field.

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. For Field System compatibility, field length is limited to 32 characters except for the 'scan label' (see Section 6), which is limited to 64 characters.

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
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. Comments on '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.

Command	**Comment**
reset=erase	Forces record pointer to zero.
reset=erase_last_scan	Sets record pointer to beginning of the disk space occupied by the last scan (effectively erases the last scan).
record=on	Starts writing at current value of record pointer; advances record pointer as data are recorded.

Table 1: Commands that modify the record pointer

Command	**Comment**
record=on	Starts writing to Mark 5 disks at record pointer; increments record pointer as recording progresses.

Table 2: Commands affected by the record pointer

To determine the current value of the record pointer use the following query 'pointers?'.

5. Scan names, Scan Labels and Linux filenames

Mark5 defines a 'scan' as a continuously recorded set of data. Each scan is identified by a scan name, experiment name and station code, which are normally derived from the information in the associated VEX file used in the scheduling of the experiment (see http://lupus.gsfc.nasa.gov/vex/vex.html). An attempt to record a scan with a duplicate scan name on the same disk module will cause a trailing alphabetical character ('a-z', then 'A-Z') to be automatically appended to the scan name. If there are more than 52 scans with same user-specified name, the suffix sequence will repeat. Information about the experiment name, station code, bit-stream mask, and sample rate are stored in the associated directory entry.

A scan label is defined as the character string

<exp name>_<stn code>_<scan name>

where

<exp name> is the name of the experiment (e.g., 'grf103'); maximum 8 characters, but by convention corresponds to a standardized 6-character experiment name. If null, will be replaced with 'EXP'.

<stn code> is the station code (e.g., 'EF'); maximum 8 characters, but by convention corresponds to standardized 2-character codes. If null, will be replaced with 'ST'

<scan name> is the identifier for the scan (e.g., '254-1056'), usually assigned by the observation-scheduling program; max 31 characters, though may be augmented to 32 characters by automatically generated duplicate-breaking suffix character.

Maximum scan-label length, including embedded underscores and possible scan-name suffix character, is 50 characters. <experiment name>, <station code> and <scan name> may contain only standard alpha-numeric characters, except '+', '-' and '.' characters may also be used in <scan name>. All fields are case sensitive. No white space is allowed in any of these subfields. Lower-case characters in all subfields are preferred. An example scan label is:

grf103_ef_scan001

6. Dual-bank mode operation

The normal operation of the Mark 5 is in so-called 'bank' mode where only one disk module is active at any given time; bank mode operation is adequate for data rates up to 2048 Mbps. However, for recording at 4096 Mbps, which is possible with the Mark 5C, operating with a single module is marginal in terms of the data-rate capacity of a single module. In this case, it is recommended that the Mark 5C be operated in 'dual-bank' mode where two modules are active simultaneously and the data are spread across 16 disks, with each disk module comfortably operating at 2048 Mbps.

Rules for dual-bank mode operation:

1. A module-pair is initiated into dual-bank mode by issuing a 'personality=mark5c:dualbank' command. The 'personality=mark5c:dualbank' command requires that the two disk modules are mounted and ready in both banks. Bank A must contain eight disks; Bank B may have fewer, but will normally have the same number.
2. After the 'personality=mark5c:dualbank' command is completed, each disk module will be communicated with as a single module. If the modules were used in 'dual-bank' mode previously and are companion pairs normal operation may continue. If the modules are not companion modules or used in bank mode, in order to have data recorded on it the module MUST be erased using 'reset=erase' and

then the following information: 1) bank position of that module (A or B), and 2) VSN of the companion module in the opposite bank. (This information is written into a special directory entry and does not affect the area where the VSN of each module is stored.) On each subsequent occasion when the modules are mounted for record or read back operation, the location and identification of the modules are checked; only if the proper modules are mounted in the proper bank positions will *drs* place the system into dual-bank mode or allow any data read or write operations.

3. If only a single module of a non-bank module pair is ready, no operations involving recording or reading data are permitted, including data_check, scan_check, etc. A 'VSN?' query will return the VSN of the active module as well as the VSN of the missing companion module.
4. A module may be returned to normal 'bank' mode operation only by issuing a 'reset=erase' command for both companion modules while in 'dual-bank' mode, then placing the system in 'bank' mode, with the 'personality=mark5c:bank;' command, and the 'reset=erase' command issued again. Verification of the VSN should be checked before normal operation.

7. Mark 5 DIM Command/Query Summary (by Category)

7.1 General

DTS_id?	p. 18	Get system information (query only)
OS_rev?	p. 25	Get details of operating system (query only)
protect	p. 29	Write protection for active module
recover	p. 32	Recover record pointer which was reset abnormally during recording
reset	p. 33	Reset Mark 5 unit (command only)
SS_rev?	p. 38	Get StreamStor hardware/firmware/software information (query only)

7.2 System Setup and Monitoring

error?	p. 19	Get error number/message (query only)
mode	p. 23	Set data recording mode
status?	p. 41	Get system status (query only)
personality	p. 28	Set application personality
MAC_list	p. 23	Set list of MAC source addresses to accept
SS_ifconfig	p. 38	Set the StreamStor 10G NIC configuration (Not supported yet)

7.3 Data Checking

data_check?	p. 10	Check data starting at position of start-scan pointer (query only)
scan_check?	p. 35	Check data between start-scan and stop-scan pointers (query only)
scan_set	p. 36	Set start-scan and stop-scan pointers

7.4 Data Transfer

net_protocol	p. 24	Set network data-transfer protocol
record	p. 30	Turn recording on|off; assign scan label
fill_pattern	p. 21	Set StreamStor 32 bit fill pattern
packet	p. 30	Set packet acceptance criteria
disk2file	p. 17	Transfer data from Mark 5 to file

7.5 Bank Management

bank_info?	p. 8	Get bank information (query only)
bank_set	p. 9	Select active bank for recording or readback

7.6 Disk Info

dir_info?	p. 11	Get directory information (query only)
disk_model?	p. 12	Get disk model numbers (query only)
disk_serial?	p. 13	Get disk serial numbers (query only)
disk_size?	p. 14	Get disk sizes (query only)
disk_state	p. 15	Set Disk Module Status (DMS): last significant disk operation
disk_state_mask	p. 16	Set mask to enable changes in DMS
get_stats?	p. 21	Get disk-performance statistics (query only)
pointers?	p. 28	Get current byte values of pointers (query only)
rtime?	p. 34	Get remaining record time on current disk set (query only)
start_stats	p. 39	Start gathering disk-performance statistics.
VSN	p. 19	Write extended-VSN to permanent area

8. Mark 5 DIM Command/Query Summary (Alphabetical)

9. Mark 5C DIM Command Set Details

This section contains a complete description of all Mark 5C commands/query in alphabetical order.

bank_info – Get bank information (query only)

[command list]

Query syntax: bank_info? ;
Query response: !bank_info ? <return code> : <selected bank> : <#bytes remaining> : <other 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	A | B | db	Currently selected bank, or 'db' if operating in dual-bank, or non-bank, mode. '-' if disk module is faulty; see Note 1.
<#bytes remaining>	int		Approximate #bytes remaining to be recorded on active module or on a non-bank-mode module pair. =0 if no module selected or faulty module.
<other bank>	char		Bank mode: Unselected bank, if module in unselected bank is mounted and ready; if no module or faulty module, '-' is returned. 'db' mode: returned null
<#bytes remaining>	int		Bank mode: Approximate #bytes remaining to be recorded on inactive module; =0 if no module active, faulty module. 'db' mode: returned null

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 readback

[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 5 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. 'bank_set command will generate an error when operating in 'db' mode; see Note 2.

Monitor-only parameters:

Parameter	Type	Values	Comments
<active bank>	char	A | B | db	'A' or 'B' if there is an active bank; '-' if no active bank; 'db' if operating in 'dual-bank mode'.
<active VSN>	char		VSN of active module, if any; if operating in 'db' mode, VSN of module in Bank A.
<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; if operating in 'db' mode, VSN of module in Bank B

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 selected 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. When operating in 'db' (i.e., dual-bank) mode, a 'bank_set' command is illegal and will generate an error; a 'bank_set?' query is allowed to gather information. The system will switch automatically to 'db' mode if (and only if) both 'db' modules are properly mounted and ready.
3. The 'bank_set' command may not be issued during recording or readback (will return an error).
4. When operating in bank mode, a 'bank_set?' query always returns the currently active module.

data_check – Check data starting at position of start-scan pointer (query only)

[command list]

Query syntax: data_check? ;
Query response: !data_check ? <return code> : <data source> : <start time> : <date code> : <frame#> : <frame header period> : <total recording rate> : <byte offset> : <#missing bytes> ;

Purpose: Reads a small amount of data starting at the start-scan pointer position and attempts to determine the details of the data, including mode and data time. For most purposes, the 'scan_check' command is more useful.

Monitor-only parameters:

Parameter	Type	Values	Comments
<data source>	char	ext | ?	ext – data from Mark 5C DIM input port ? – data not in Mark5B emulation or VDIF Mark 5C format; all subsequent return fields will be null
<start time>	time		Time tag at first disk frame header. See Note 4 below.
<date code>	int		3-digit date code written in first disk frame header (module 1000 value of Modified Julian Day)
<frame#>	int		Extracted from first disk frame header; frame# is always zero on second tick
<frame header period>	time		Each disk frame is now dependent on the data type, e.g. Mark5B or VDIF
<total recording rate>	real	(Mbps)	
<byte offset>	int		Byte offset from start-scan pointer to first disk frame header.
<#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 Note 5.

Notes:

1. Starting at the start-scan pointer position, the 'data_check' query searches to find the first valid disk frame header.
2. The 'data_check' query will be honored only if record is off.
3. The 'data_check' query does not affect the start-scan pointer.
4. Regarding the <start time> value returned by the 'data_check?' and, 'scan_check?' queries: The year and DOY reported in <start time> represent the most recent date consistent with the 3-digit <date code> in the frame header time tag (modulo 1000 value of Modified Julian Day as defined in VLBA tape-format header); this algorithm reports the proper year and DOY 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.

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 total module size.

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

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 in currently selected disk module.

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, etc.); a blank field is returned for an empty slot. When operating in 'db' mode, disks in banks A and B are treated as a single module.

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 in currently selected disk module.

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, etc.); A blank field is returned for an empty slot. When operating in 'db' mode, disks in banks A and B are treated as a single module.

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 individual capacities of currently selected module.

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, etc.); A blank field is returned for an empty slot. When operating in 'db' mode, disks in banks A and B are treated as a single module.

disk_size

disk_state – Set 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.

Monitor-only parameters:

Parameter	Type	Values	Comments
<active bank>	char	A | B| db	Currently selected bank; '-' if disk module is judged faulty.
<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 playback; 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 *drs* 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. Normally, the setting of the DMS parameter happens automatically whenever a record, play or erase command is issued. 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).
2. The DMS logs 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, disk2net, and disk2file, do not affect DMS.
3. The 'disk_state' and 'disk_state_mask' commands were requested by NRAO and are designed primarily for use at a correlator.
4. If no modules are inserted, an error code 6 is returned.
5. When operating in 'db' mode, disks in banks A and B are treated as a single module.

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 disk state mask (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 Mark5 to file

[command list]

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

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

Purpose: Transfer data between start-scan and stop-scan pointers from Mark5 disk module to file.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<dest filename>	ASCII	A | B | inc		Default <dest filename> is as specified in Section 6 (i.e. '<scan label>_bm=<bit mask>.fmt') (see Note 6). Filename must include path if path is not default (see Note 5).
<start byte#>	int			Absolute byte#; if null, defaults to start-scan pointer. See Notes 1 and 2.
<end byte#>	int			Absolute end byte#; if preceded by '+', increment from <start byte#> by specified value; if null, defaults to stop-scan ponter. See Notes 1 and 2.
<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>	char		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. The 'scan_set' command is a convenient way to set the <start byte#> and <stop byte#>.
2. If <start byte#> and <end byte#> are null, the range of data defined by 'scan_set' will be transferred.
3. To abort data transfer: The 'reset=abort' command may be used to abort an active disk2file data transfer. See 'reset' command for details
4. When <status> is 'inactive', a 'disk2file?' query returns the <dest filename> of the last transferred scan, if any.
5. Default path is the Linux default, which is the directory from which *drs* was started.
6. The extension of the saved file defined by the mode specification, e.g. mark5b =.m5b, vdif=vdf, unknown=unk

disk2file

DTS_id – Get system information (query only)

[command list]

Query syntax: DTS_id? ;
Query response: !DTS_id ? <return code> : <system type> : <software version number> : <serial number> : <command set revision>;

Purpose: Get Mark 5 system information

Monitor-only parameters:

Parameter	Type	Values	Comments
<system type>	char	Mark5C	
<software version number>	char		Version number for current version of *drs*
<serial number>	ASCII		System serial number; generally is in the form 'mark5-xx' where xx is the system serial number
<command set revision>	char		Mark 5C DIM command set revision level corresponding to this software release (e.g., '2.0')

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

fill_pattern – Set StreamStor 32-bit fill pattern

[command list]

Command syntax: fill_pattern = <pattern> ;
Command response: !fill_pattern = <return code>;

Query syntax: fill_pattern? ;
Query response: !fill_pattern ? <return code> : <pattern> ;

Purpose: Set the 32-bit fill pattern that replaces data that cannot be recovered.

Settable parameters:

Parameter	Type	Values	Comments
<fill_pattern>	hex	0x0 – 0xffffff	The fill pattern

Notes:

1. This fill pattern is used both at record time to replace invalid data when recording in PSN monitor mode 1 or 2.

2. If failures occur at both record time (bad packets) and read time (disk went bad), then it is possible that two different fill patterns need to be detected, and the duration of the bad-packet fill pattern could be less than the length of a packet data frame. The use of two different fill patterns has the potential advantage of distinguishing between record and playback problems.

3. If the same fill pattern is used at record time and read time, the 64-bit data granularity will ensure that the fill patterns are always in phase with each other.

fill_pattern

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 readback 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 readback.
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.
5. When operating in 'db' mode, disks in banks A and B are treated as a single module.

get_stats

MAC_list – Set list of MAC source addresses to accept

[command list]

Command syntax: MAC_list = <action> : [<MAC #1>] : [<action>]: [<MAC #2>] : … ;
Command response: !mode = <return code> ;

Query syntax: MAC_list? ;
Query response: !MAC_list ? <return code> : <state>: [<MAC #1>] : [<state>]:[<MAC #2>] : … ;

Purpose: Set the list of MAC source addresses (up to 16) from which to accept data.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<action>	ACII	add| delete | disable | enable | flush	-	add – add a MAC source address to the filter and enable filter checking delete – delete a MAC source address that matches MAC #n passed after action disable – disable a MAC source address that matches MAC #n, but leave it in the filter bank enable – enables a disabled a MAC source address that matches MAC #n in the filter bank flush – remove all MAC source address from filter, and disable MAC filter checking
<MAC #n>	ASCII	hex		Source MAC Addresses used for packet filtering [1]

Monitor parameters:

Parameter	Type	Values	Comments
<state>	ASCII	enabled | disabled	enabled – that the MAC address following this state is enabled in the source address filtering disabled – that the MAC address following this state is disabled in the source address filtering
<MAC #n>	ASCII	XX.XX.XX.XX.XX	Destination MAC Addresses used for packet filtering

Notes:

1. All MAC addresses should be period separated 8-bit hextets, e.g., 00.50.8d.e9.36.f4. Traditionally colons are used to separate the digits, but this is incompatible with VSI-S.

2. A MAC address of 00.00.00.00.00.00 can be specified to accept valid data from any source.

MAC_list

mode – Set data recording mode

[command list]

Command syntax: mode = <data source> : <bit-stream mask> : [<decimation ratio>] ;
Command response: !mode = <return code> ;

Query syntax: mode? ;
Query response: !mode ? <return code> : <data source> : <bit-stream mask> : <decimation ratio> ;

Purpose: Set the recording mode of the Mark 5C DIM

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<data source>	char	vdif | mark5b | unk	mark5b	'vdif' – record in vdif format with Mark5C profile 'mark5b' – record in mark5b compatibility mode 'unk' – record in unknown mode meaning that no header checking is performed
<data submode1>	int hex	>0 != 0x0		(vdif) number of channels (mark5b) bit-stream mask
<data submode2>	int	1|2| 4|8|16	1	(mark5b) decimation ratio (vdif) not used

Monitor-only parameters:

Parameter	Type	Values	Comments
<data source>	int	vdif | mark5b|unk	Format of data stored on disk module
<data submode1>	int /hex	>0 / !=0x0	Channels per packet / bit stream mask
<data submode2>	int	- / 1|2|4|8|16	Decimation ratio

Notes:

1. For VDIF formats the parameters required to completely specify the setup, scan information is split between the commands 'mode' and 'packet'. The parameters set by 'packet' are low-level and used by the StreamStor card to determine the acceptance criteria for packets. The parameters set by 'mode' are used for the generation of the scan directory listing to aid in the unpacking of data and are not used by the StreamStor card.

net_protocol – Set network data-transfer protocol

[command list]

Command syntax: net_protocol = <protocol>;
Command response: !net_protocol = <return code> ;

Query syntax: net_protocol? ;
Query response: !net_protocol? <return code> : <protocol> ;

Purpose: Set network data-transfer protocol for when the personality is set to file, unused otherwise.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<protocol>	char	udp | L2	udp	'udp' - the UDP transport protocol is expected on the incoming data stream 'L2' - the VLBI data frame will be directly encapsulated in a Layer 2 MAC packet

Monitor-only parameters:

Parameter	Type	Values	Comments
<protocol>	char	udp | L2	'udp' - the UDP transport protocol is expected on the incoming data stream 'L2' - the VLBI data frame will be directly encapsulated in a Layer 2 MAC packet

Notes:

1. This is command is primarily used with personality set to file specifying the transport protocol of the incoming data stream. When the personality is set to Mark5C, the packet offsets will determine where and how much of the VLBI payload to record.

net_protocol

OS_rev – Get details of operating system (query only)

[command list]

Query syntax: OS_rev? ;
Query response: !OS_rev? <return code> : <OS field1> : <OS field2>: ….. : <OS fieldn> ;

Purpose: Get detailed information about operating system.

Monitor-only parameters:

Parameter	Type	Values	Comments
<OSfield1> through <OSfieldn>	literal ASCII		Primarily for diagnostic purposes. The character stream returned from OS, which is very long, is divided into 32-character fields separated by colons to stay within Field System limits. See Notes.

Notes:

1. ‘OS_rev?’ is a replacement for the old ‘OS_rev1?’ and ‘OS_rev2?’ queries; all three of these queries are now synonyms.

OS_rev

packet – Set packet acceptance criteria

[command list]

Command syntax: packet = <DPOFST> : <DFOFST> : <length> : <PSN Mode> : <PSNOFST> ;
Command response: !packet = <return code>;

Query syntax: packet? ;
Query response: !packet? <return code> : <DPOFST> : <DFOFST> : <length> : <PSN Mode> : <PSNOFST> ;

Purpose: Set / get the packet acceptance criteria.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
< DPOFST >	int	≥ 0	0	payload byte offset from beginning of payload to first recorded data
< DFOFST >	int	≥ 0	0	payload byte offset to beginning of recording
< length >	int	> 0	5008	number of bytes to record per packet (see Note 1)
< PSN Mode >	int	0 | 1 | 2	0	Packet Serial Number (PSN) monitor mode (see Note 2,3)
< PSNOFST >	int	≥ 0	0	payload byte offset from beginning of payload to PSN (for PSN monitor mode 1 or 2)

M onitor-only parameters:

Parameter	Type	Values	Comments
< DPOFST >	int	≥ 0	payload byte offset from beginning of payload to first recorded data
< DFOFST >	int	≥ 0	payload byte offset to beginning of recording
< length >	int	> 0	number of bytes to record per packet (see Note 1)
< PSN Mode >	int	0 | 1 | 2	Packet Serial Number (PSN) monitor mode (see Note 2)
< PSNOFST >	int	≥ 0	payload byte offset from beginning of payload to PSN (for PSN monitor mode 1 or 2)

Notes:

1. The length of data to be recorded must be a multiple of 8 bytes.
2. PSN-mode 0 will disable packet serial number checking and record all data in the order received. PSN-monitor mode 1 will replace invalid packets with the specified fill pattern and guarantee order. PSN-monitor mode 2 will prevent packets from being written to disk if the most significant bit is set.
3. PSN Mode 0 is the only mode supported by the 10DB firmware at this time.

personality – Set application personality

[command list]

Command syntax: personality = <type> : <root > ;
Command response: !personality = <return code> ;

Query syntax: personality? ;
Query response: !personality? <return code> : < type >: < root > ;

Purpose: Set the application personality (i.e., the emulation mode), which configures the StreamStor card in the proper configuration.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
< type >	char	mark5C | file	NULL	Mark5C– Normal operating mode (default) file – write data to the OS system disk, not streamstor (see Note 3)
< root >	char	ascii bank | dualbank	NULL	For <type> file this is the root file system path on where to store the incoming scans. (see Note 3) For <type> Mark5C sets up the StreamStor disk module to store data in either bank or dual-bank mode.

Monitor-only parameters:

Parameter	Type	Values	Comments
< type >	char	mark5C | file	Mark5C– Normal operating mode (default) file – write data to the OS system disk, not streamstor (see Note 3)
< root >	char	system path bank | dualbank	For <type> file this is the root file system path on where to store the incoming scans. (see Note 3) For <type> Mark5C sets up the StreamStor disk module to store data in either bank or dualbank mode.

Notes:

1. A personality is defined as a set of functions bound to the commands and queries described in this document. Nothing is to preclude the various personalities from sharing a subset of functionality. The implementation of *drs* should make it easy to add new personalities to the program.
2. This command cannot be issued while a delayed completion operation is in effect or while data is being recorded on any type of medium.
3. The file personality causes incoming data received through the system NIC to be written to a file system (e.g., RAID Array). The optional parameter should be a directory specifying the root of the file system to write. Files written to this directory will have systematically determined file names bearing close resemblance to scan names on Mark5 Modules. Additionally, a file containing the equivalent of a Mark5 scan list will be created in the specified directory.
4. Implementation of this command is optional; its absence will not imply non-conformance with the Mark5C software specification.

personality

pointers – Get current values of pointers (query only)

[command list]

Query syntax: pointers? ;
Query response: !pointers? <return code> : <record pointer> : <start-scan pointer> : <stop-scan pointer> ;

Purpose: Get current value of record, start-scan and stop-scan 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.
<start-scan pointer>	int	bytes	Current value of <start-scan pointer>
<stop-scan pointer>	int	bytes	Current value of <stop-scan pointer>; '-' if undefined.

Notes:

1. Note that the returned byte numbers may have values as large as ~$2x10^{13}$ (~44 bits), so pointer arithmetic must be handled appropriately.
2. When recording, the <record pointer> will be updated to show the approximate current recording position. If the record pointer is noted not to be incrementing during recording, unlike previous version of Mark5, an error flag is NOT set in 'status?. This is a result of being unable to predict whether a digital backend is continuously transmitting over the 10G Ethernet cable.

pointers

protect – Write protection for active module

[command list]

Command syntax: protect = <state> ;
Command response: !protect = <return code> ;

Query syntax: protect? ;
Query response: !protect? <return code> : <state> ;

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

Settable parameters:

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

Monitor-only parameters:

Parameter	Type	Values	Comments
<state>	char	on | 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* proceed 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.

record – Turn recording on|off; assign scan label

[command list]

Command syntax: record = <record on/off> : <scan label/name> : [<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 name, experiment name and station code.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<record on/off>	char	on | off		'on' automatically appends to the end of the existing recording. 'off' stops recording and leaves system in 'idle' mode.
<scan name>	ASCII	32 chars max		Relevant only if record is 'on'. If in <scan label> format, field is parsed for <exp name>, <station code> and <scan name>. Otherwise, interpreted as <scan name>, in which case <experiment name> and <station code> should be specified separately. If <scan name> is duplicate of already-recorded scan, a suffix will be added to the <scan name> part of the <scan label> (see Note 6).
<experiment name>	ASCII	8 chars max		Experiment name; ignored if <record on/off> is 'off'
<station code>	ASCII	8 chars max		Station code; ignored if <record on/off> is 'off'

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	on | off | halted | overflow	'halted' indicates end-of-media was encountered while recording. 'overflow' is the error condition.
<scan#>	int		Sequential scan number; starts at 1 for first recorded scan.
<scan label>	ASCII		Scan label – (see Notes 5 & 6). See Section 6 for definition of scan label.

Notes:

1. 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.
2. 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.
3. When <status> is 'off', a 'record?' query returns the <scan label> of the last recorded scan, if any.
4. Typical causes for status errors is an "overflow" – FIFO overflow on StreamStor 10G Ethernet daughter card
5. The <scan label> field is created in the standardized format specified in Section 6, namely '<exp name>_<station code>_<scan name>'. If <experiment name> and/or <station code> are null, they will be replaced with 'EXP' and 'ST', respectively.

record

6. An attempt to record a scan with a duplicate scan name on the same disk module will cause a trailing alphabetical character (‘a-z’, then ‘A-Z’) to be automatically appended to the scan name (example: ‘312-1245a’). If more than 52 scans with same user-specified name, the suffix sequence will repeat.

recover – Recover record pointer which was reset 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 record pointer which was reset 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

reset – Reset Mark 5 unit (command only)

[command list]

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

Purpose: Reset system; erase, mount, or dismount disk modules.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<control>	char	erase | erase_last_scan | abort | mount | dismount		'erase'sets record, start-scan and stop-scan pointers to zero (i.e., beginning of media); effectively erasing media; (see Note 1) 'erase_last_scan' erases the last recorded scan; sets record pointer to end-of-scan just prior to erased scan; sets start-scan and stop-scan pointer to beginning and end, respectively, of scan just prior to erased scan. 'abort' aborts active disk2net, disk2file or file2disk transfers (only) ; (see Note 3) System is always left in 'idle' mode after any reset command. (See Note 2). 'mount' causes the module in bank <bank> to be mounted. 'dismount' causes the module in bank <bank> to be dismounted.
<bank>	char	A | B		used only for mount and dismount options.

Notes:

1. The 'reset=erase' command is used for both 'bank' and 'dualbank' personalities. By setting the 'personality' dictates how the disk modules inserted in the Mark5C will be erased. Bank A must contain eight disks; bank B may have fewer, though it will normally have the same number. After the 'reset=erase' command is completed, each module will have recorded on it (until the module is again erased) the following information: 1) bank position of the module (A or B), and 2) VSN of the module in the opposite bank. Each subsequent occasion when the modules are mounted for record or readback operation, the location and identification of the modules is checked; only if the proper modules are mounted in the proper positions will *drs* place the system into dual-bank mode or allow any read or write 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> : <data source> : <bit-stream mask> : <decimation ratio> : <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.

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
<data source>	char	ext | tvg	Assumed to be same as specified in last 'mode' command
<bit-stream mask>	hex		Assumed to be same as specified in last 'mode' command
<decimation ratio>	int		Assumed to be same as specified in last 'mode' command
<total recording rate>	real	Mbps	Net recording rate assumed in calculation of <remaining time>, based on current clock frequency and 'mode' parameters

Notes:

1. 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 – Check recorded data between start-scan and stop-scan pointers (query only) [command list]

Query syntax: scan_check? ;
Query response: !scan_check ? <return code> : <scan#> : <scan label> : <data type> : <date code> : <start time> : <scan length> : <total recording rate> : <#missing bytes> ;

Purpose: Check recorded data between the start-scan and stop-scan pointers (e.g., returned by 'pointers?' query).

Monitor-only parameters:

Parameter	Type	Values	Comments
<scan#>	int		Start at 1 for first recorded scan
<scan label>	ASCII		
<data type>	char	vdif| mark5b | unk | tvg | SS	vdif – Mark 5C format, but undetermined data type mark5b – Mark5B emulation mode data unk – Unknow data format; all subsequent return fields are null. tvg – undecimated 32-bit-wide tvg data (see Note 4) SS – raw StreamStor test pattern data
<date code>	int		3-digit date code written in first disk frame header
<start time>	time		Time tag at first frame header in scan. See Note 5.
<scan length>	time		
<total recording rate>	real	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 start-scan or stop-scan pointers.
3. The 'scan_check' query essentially executes a 'data_check' starting at the start-scan pointer, followed by a 'data_check' just prior to the stop-scan pointer. This allows information about the selected scan to be conveniently determined.
4. Only tvg data that were recorded with a bit-stream mask of 0xffffffff and no decimation will be recognized.
5. Regarding the <start time> value returned by the 'data_check?' and, 'scan_check?' queries: The year and DOY reported in <start time> represent the most recent date consistent with the 3-digit <date code> in the frame header time tag (modulo 1000 value of Modified Julian Day as defined in VLBA tape-format header); this algorithm reports the proper year and DOY provided the data were taken no more than 1000 days ago.
6. 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

scan_set – Set start-scan and stop-scan pointers

[command list]

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

Query syntax: scan_set? ;
Query response: !scan_set? <return code> : <scan label> : <start scan> : <stop scan> ;

Purpose: Set start-scan and stop-scan pointers for data_check, scan_check, 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.
<start read>	int	+<bytes> | -<bytes>	s	+<bytes>: offset number of bytes from beginning of scan. -<bytes>: offset number of bytes from end of scan
<stop read>	int	+<bytes> | -<bytes>	end-of scan	+<bytes>: offset bytes from <start scan> position -<bytes>: offset bytes from end of scan

Monitor-only parameters:

Parameter	Type	Values	Comments
<scan label>	ASCII		Scan label of scan matching <search string>
<start play>	bytes		Start byte position of scan.
<stop play>	bytes		End byte position of scan.

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 start-scan and stop-scan pointers are set to span the entire just-recorded scan.
3. A 'scan_set=' command is not allowed during active data transfers.
4. The specified values of <start scan> and <stop scan> must be within the target scan.
5. The 'pointers' query can be issued at any time to retrieve the current value of the start-scan and stop-scan pointers.

scan_set

SS_ifconfig – Set StreamStor 10G NIC configuration (TBD)

[command list]

Command syntax: SS_ifconfig = <state> : <MTU> : <Mode> : <MAC Address> ;
Command response: !SS_ifconfig = <return code> ;

Query syntax: SS_ifconfig? ;
Query response: !SS_ifconfig ? <return code> : <state> : <MTU> : <Mode> : <MAC Address> ;

Purpose: Get and set StreamStor network configuration.

Settable parameters:

Parameter	Type	Values	Comments
<state>	char	active | inactive	enable 10G interface disable 10G interface.
< MTU>	int	64 ≤ X ≤ 9000	maximum packet size (bytes) to accept, default 9000
<mode>	char	normal | all	apply packet filtering (default) accept all packets
<MAC Address>	ASCII		see Note 2

Notes:

1. All parameters specified by this command apply only to the 10G Ethernet interface on the StreamStor daughter card.
2. The format for the MAC address shall be a period separated 8 bit hextet, e.g., 00.50.8d.e9.36.f4 .

SS_ifconfig

SS_rev – Get StreamStor hardware/firmware/software information

[command list]

Query syntax: SS_rev? ;
Query response: !SS_rev ? <return code> : <SS field1> : <SS field2> : ….. : <SS fieldn> ;

Purpose: Get information on StreamStor hardware/firmware/software model and version information.

Monitor-only parameters:

Parameter	Type	Values	Comments
<SSfield1> through <SSfieldn>	literal ASCII		Primarily for diagnostic purposes. The character stream returned from StreamStor, which is very long, is divided into 32-character fields separated by colons to stay within Field System limits. See Notes.

Notes:

1. 'SS_rev?' is a replacement for the old 'SS_rev1?' and 'SS_rev2?' queries; all three of these queries are now synonyms.

SS_rev

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 gathering 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 readback.

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) system command has control of StreamStor Bit 3 – (0x0008) one or more 'delayed-completion' commands are pending. Also set whenever any data-transfer activity, such as recording. --- Bit 4 – (0x0010) one or more 'delayed-completion' queries are pending Bit 5 – (0x0020) not used Bit 6 - (0x0040) record 'on' Bit 7 - (0x0080) media full (recording halted) --- Bit 8 - (0x0100) not used Bit 9 - (0x0200) not used Bit 10 – (0x0400) recording can't keep up; some lost data Bit 11 – (0x0800) not used --- Bit 12 – (0x1000) not used Bit 13 – (0x2000) not used Bit 14 – (0x4000) not used Bit 15 – (0x8000) not used --- Bit 16 – (0x10000) not used Bit 17 – (0x20000) not used Bit 18 – (0x40000) DIM ready to record 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

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'] :
<companion extended VSN> : <companion bank> ;

Purpose: Write module extended-VSN (volume serial number) to permanent area on active 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 5. 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'). For non-bank-mode modules, see Note 8.

Monitor-only parameters:

Parameter	Type	Allowed values	Comments
<extended VSN>	char		Example: 'MPI-0153/960/1024'; see Notes 5 and 6. For non-bank-mode modules, see Note 7.
<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 7.
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
<companion extended VSN>	char		If dual-bank mode module, returns VSN of companion non-bank module. See Note 9.
<companion bank>	char	B | A	Bank position of companion dual-bank module

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.

VSN

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 *drs*:
   a. VSN – Must be 8 characters in length and in format “ownerID-serial#” (for parallel-ATA modules) or “ownerID+serial#” (for serial-ATA modules)
   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. *drs* 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.
7. A “VSN=” command may not be issued to any module which has been initialized in non-bank mode.
8. When a non-bank-mode pair of modules is mounted and the unit is operating in non-bank mode, a “VSN?” query will return the VSN of both modules as indicated in the return parameters.
9. When only a single module of a non-bank-mode module-pair is mounted, a “VSN?” query will return the both the VSN of the mounted module plus the VSN and bank position of its unmounted companion; however, no reading or writing of data will be allowed in this situation.

VSN