MASSACHUSETTS INSTITUTE OF TECHNOLOGY

HAYSTACK OBSERVATORY

WESTFORD, MASSACHUSETTS 01886

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

13 November 2006

TO: Distribution
FROM: Alan R. Whitney, Dan L. Smythe and John A. Ball
SUBJECT: Mark 5B DIM command set (Revision 1.12)

Note: The current version of the Mark 5B program is called '*Mark5B'*, which will in a few months be replaced by a newly developed program called '*dimino'*. Both programs will adhere to the command set in this document, though *dimino* will also include additional capabilities.

1. *dimino* program

The commands detailed in this memo are implemented by a program named *dimino* (pronounced with the accent on 'DIM") and control the DIM functionality of the Mark 5B VLBI data system. The details concerning the operation of *dimino* are available in documents at http://web.haystack.mit.edu/mark5/Mark5.htm. The DOM functionality of the Mark 5B is controlled by a separate program called *domino*.

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

dimino –*m* [-1|0|1|2|3] –*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

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 DIM 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 *dimino* 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. 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

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

<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. Simplified Diagrams of Various Mark 5 Data Transfer Modes

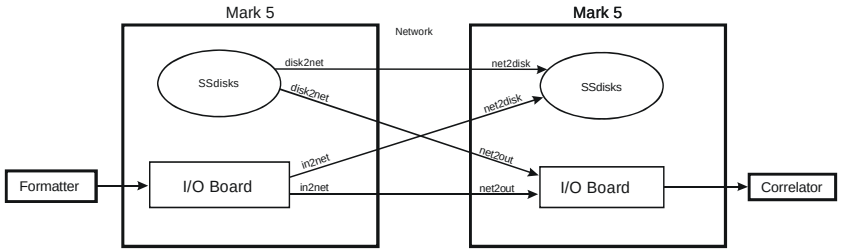

Figure 1: Mark 5 to Mark 5 transfer through network

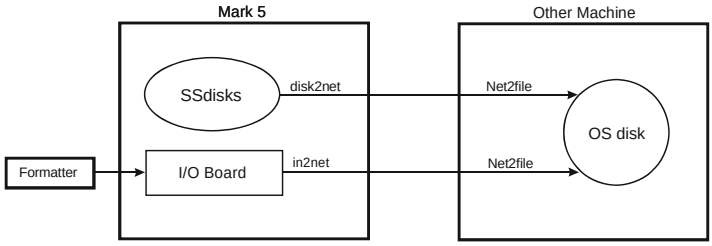

Figure 2: Mark 5 to file transfer through network

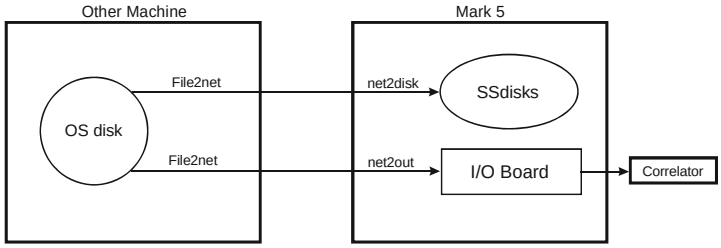

Figure 3: File to Mark 5 transfer through network

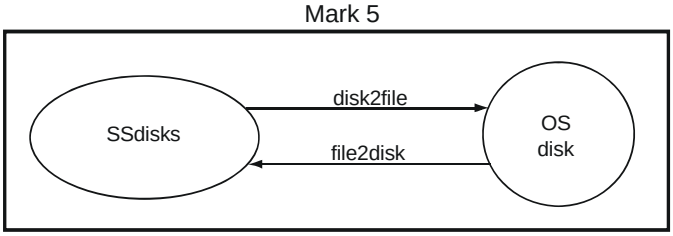

Figure 4: Internal Mark 5 to file transfer

5. Comments on 'Record pointer', 'Start-scan pointer' and 'Stop-scan Pointer'

Three different pointers are maintained by the Mark 5 system and it is important to understand what they are, what they mean, and how they are managed. The *record pointer* is associated only with recording data to the disks; the *start-scan* and *stop-scan pointer*s are used to control reading data from the disks.

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. Table 1 lists commands that modify the record pointer; Table 2 lists commands that are affected by the record pointer.

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.
file2disk	Data transfer from Linux disk to Mark 5: Starts writing to Mark 5 disks at current value of record pointer; record pointer advances as data are written.
net2disk	Data transfer from network to Mark 5: Starts writing to Mark 5 disks at current value of record pointer; record pointer advances as data are written.

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.
file2disk	Starts transfer to Mark 5 disks at record pointer; increments record pointer as data transfer progresses.
net2disk	Starts transfer to Mark 5 disks at record pointer; increments record pointer as data transfer progresses.

Table 2: Commands affected by the record pointer

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

5.2 Start-scan and Stop-scan pointers

The 'start-scan' and 'stop-scan' pointers specify the start and end points for reading all or part of pre-recorded scan for data checking or data-transfer purposes. By default, these pointers are normally set to the beginning and end of a block of continuously recorded data, but for special purposes may be set to include only a portion of the recorded scan. Table 3 lists commands that modify these pointers; Table 4 lists commands that are affected by these pointers.

Command	Comment
reset=erase	Resets start-scan and stop-scan pointers to zero.
reset=erase_last_scan	Sets start-scan pointer to beginning of 'new' last scan; sets stop-scan pointer to end of 'new' last scan.
record=off	Sets start-scan pointer to beginning of scan just recorded; sets stop-scan pointer to end of scan just recorded.
file2disk	Sets start-scan pointer to beginning of scan just transferred; sets stop-scan pointer to end of scan just transferred.
net2disk	Sets start-scan pointer to beginning of scan just transferred; sets stop-scan pointer to end of scan just transferred.
scan_set	Sets start-scan and stop-scan pointers to a data range within a scan as specified.

Table 3: Commands that modify the start-scan and stop-scan pointers

Command	Comment
data_check	Reads and checks a small amount of data beginning at start-scan pointer
scan_check	Checks small amount just after start-scan pointer and before end pointer.
disk2file	Unless specific start/stop byte numbers are specified, transfers data between start-scan and stop-scan pointers
disk2net	Unless specific start/stop byte numbers are specified, transfers data between start-scan and stop-scan pointers

Table 4: Commands affected by the start-scan and stop-scan pointers

A 'scan_set?' or 'pointers?' query returns information about the current value of the start-scan and stop-scan pointers.

6. 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 ‘STN’

<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

When a Mark 5B scan (or portion of a scan) is copied to a Linux file with *disk2file*, a Linux filename compatible with the internationally agreed e-VLBI filenaming convention (reference http://www.haystack.edu/tech/vlbi/evlbi/memo.html memo #49) is assigned as

‘<scan label>_bm=<bit-stream mask>.m5b’
(example: ‘grf103_ef_scan001_bm=0x0000ffff.m5b’)

Linux files to be transferred to a Mark 5B disk via the ‘*file2disk’* should have filenames corresponding to the standardized format described above so that the associated Mark 5B directory entries can be properly filled.

Note: The <scan name> is equivalent to what is called <scan_ID> in VEX files, except the set of legal characters in <scan name> is more restricted and must be observed.

7. Non-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 1024Mbps. However, for recording at 2048 Mbps, which is possible with the Mark 5B+, 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 5 be operated in ‘non-bank’ mode where two modules are active simultaneously and the data are spread across 16 disks, with each disk module comfortably operating at 1024Mbps.

Rules for non-bank mode operation:

1. A module-pair is initiated into non-bank mode by issuing a ‘reset=nberase’ command. The ‘reset=nberase’ 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 ‘reset=nberase’ command is completed, each module will have recorded on it (until the module is again erased) 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 readback operation, the location and identification of the modules are checked; only if the proper modules are mounted in the proper

bank positions will *dimino* place the system into non-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.

8. 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.

8.1 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.

8.2 Setup

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

dimino –m 0 -d 1 &

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

8.3 Operation

Run *dimino* from *rundim* 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:

Start the receiving program (e.g. 'Net2file') on the target machine.

1. 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.
2. Connect to the target machine using an 'in2net=connect:…' command.
3. Start and stop each scan with 'in2net=on' and 'in2net=off'.
4. Log the target Mark 5 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 a 'data_check?' query 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 *dimino* without *'-d 1'*. The scratch disk module will also need to be erased (again) for normal operation.

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

9.1 General

	Common to Mk5A DIM		
DTS_id?	●	p. 30	Get system information (query only)
OS_rev?	●	p. 38	Get details of operating system (query only)
protect	●	p. 40	Set/remove erase protection for active module
recover	●	p. 43	Recover record pointer which was reset abnormally during recording
reset	●	p. 44	Reset Mark 5 unit (command only)
SS_rev?	●	p. 49	Get StreamStor firmware/software revision levels (query only)

9.2 System Setup and Monitoring

	Common to Mk5A DIM		
1pps_source		p. 12	Select source of 1pps synchronization tick
clock_set		p. 16	Specify frequency and source of the CLOCK driving the DIM
DOT?		p. 27	Get DOT (Data Observe Time) clock information (query only)
DOT_inc		p. 28	Increment DOT clock
DOT_set		p. 29	Set DOT clock on next external 1pps tick
error?	●	p. 31	Get error number/message (query only)
mode	●	p. 35	Set data recording/readback mode
status?	●	p. 51	Get system status (query only)

9.3 Data Checking

	Common to Mk5A DIM		
data_check?	●	p. 17	Check data starting at position of start-scan pointer (query only)
scan_check?	●	p. 46	Check data between start-scan and stop-scan pointers (query only)
scan_set	●	p. 47	Set start-scan and stop-scan pointers
TVR		p. 52	Start TVR testing

9.4 Data Transfer

	Common to Mk5A DIM		
disk2file	●	p. 24	Transfer data between start-scan and stop-scan pointers from Mark 5 to file
disk2net	●	p. 25	Transfer data between start-scan and stop-scan pointers from Mark 5 to network
file2disk	●	p. 32	Transfer data from file to Mark 5
in2net	●	p. 34	Transfer data directly from Mark 5 input to network
net_protocol	●	p. 37	Set network data-transfer protocol
net2disk	●	p. 36	Transfer data from network to Mark 5
record	●	p. 41	Turn recording on|off; assign scan label

9.5 Bank Management

	Common to Mk5A DIM		
bank_info?	●	p. 13	Get bank information (query only)
bank_set	●	p. 14	Select active bank for recording or readback
bank_switch	●	p. 15	Enable|disable automatic bank-switching (not yet implemented)

9.6 Disk Info

	Common to Mk5A DIM		
dir_info?	●	p. 18	Get directory information (query only)
disk_model?	●	p. 19	Get disk model numbers (query only)
disk_serial?	●	p. 20	Get disk serial numbers (query only)
disk_size?	●	p. 21	Get disk sizes (query only)
disk_state	●	p. 22	Set/get Disk Module Status (DMS): last significant disk operation
disk_state_mask	●	p. 23	Set mask to enable changes in DMS
get_stats?	●	p. 33	Get disk-performance statistics (query only)
pointers?	●	p. 39	Get current byte values of pointers (query only)
rtime?	●	p. 45	Get remaining record time on current disk set (query only)
start_stats	●	p. 50	Start gathering disk-performance statistics.
VSN	●	p. 53	Read/write extended-VSN

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

	Common to Mk5A DIM		
1pps_source		p. 12	Select source of 1pps synchronization tick
bank_info?	●	p. 13	Get bank information (query only)
bank_set	●	p. 14	Select active bank for recording or readback
bank_switch	●	p. 15	Enable|disable automatic bank-switching (not yet implemented)
clock_set		p. 16	Specify frequency and source of the CLOCK driving the DIM
data_check?	●	p. 17	Check data starting at position of start-scan pointer (query only)
dir_info?	●	p. 18	Get directory information (query only)
disk_model?	●	p. 19	Get disk model numbers (query only)
disk_serial?	●	p. 20	Get disk serial numbers (query only)
disk_size?	●	p. 21	Get disk sizes (query only)
disk_state	●	p. 22	Set/get Disk Module Status (DMS): last significant disk operation
disk_state_mask	●	p. 23	Set mask to enable changes in DMS
disk2file	●	p. 24	Transfer data between start-scan and stop-scan pointers from Mark 5 to file
disk2net	●	p. 25	Transfer data between start-scan and stop-scan pointers from Mark 5 to network
DOT?		p. 27	Get DOT (Data Observe Time) clock information (query only)
DOT_inc		p. 28	Increment DOT clock
DOT_set		p. 29	Set DOT clock on next external 1pps tick
DTS_id?	●	p. 30	Get system information (query only)
error?	●	p. 31	Get error number/message (query only)
file2disk	●	p. 32	Transfer data from file to Mark 5
get_stats?	●	p. 33	Get disk-performance statistics (query only)
in2net	●	p. 34	Transfer data directly from Mark 5 input to network
mode	●	p. 35	Set data recording mode
net2disk	●	p. 36	Transfer data from network to Mark 5
net_protocol	●	p. 37	Set network data-transfer protocol
OS_rev?	●	p. 38	Get details of operating system (query only)
pointers?	●	p. 39	Get current byte values of pointers (query only)
protect	●	p. 40	Set/remove erase protection for active module
record	●	p. 41	Turn recording on|off; assign scan label
recover	●	p. 43	Recover record pointer which was reset abnormally during recording
reset	●	p. 44	Reset Mark 5 unit (command only)
rtime?	●	p. 45	Get remaining record time on current disk set (query only)
scan_check?	●	p. 46	Check data between start-scan and stop-scan pointers (query only)
scan_set	●	p. 47	Set start-scan and stop-scan pointers
SS_rev?	●	p. 49	Get StreamStor firmware/software revision levels (query only)

start_stats	●	p. 50	Start gathering disk-performance statistics.
status?	●	p. 51	Get system status (query only)
TVR		p. 52	Start TVR testing
VSN	●	p. 53	Read/write extended-VSN

11. Mark 5B DIM Command Set Details

This section contains a complete description of all Mark 5B commands/query in alphabetical order. Highlights in red are changes and updates from Revision 1.0.

1pps_source – Select source of 1pps synchronization tick

[command list]

Command syntax: 1pps_source = <1pps source> ;
Command response: ! 1pps_source = <return code> ;

Query syntax: 1pps_source? ;
Query response: ! 1pps_source ? <return code> : <1pps source> ;

Purpose: Select source of 1pps which will be used to synchronize the Mark 5B.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<1pps_source>	char	altA | altB | vsi	vsi	'altA' = AltA1PPS (LVDS signal on 14-pin VSI-H connector; rising edge) 'altB' = AltB1PPS (LVTTL signal on SMA connector on back panel; rising edge) 'vsi' = VSI (LVDS signal on 80-pin VSI-H connector; rising edge)

Monitor-only parameters:

Parameter	Type	Values	Comments
<1pps_source >	char	altA | altB | vsi	

Notes:
1. -

1pps_source

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 | nb	Currently selected bank, or 'nb' if operating in 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. 'nb' mode: returned null
<#bytes remaining>	int		Bank mode: Approximate #bytes remaining to be recorded on inactive module; =0 if no module active, faulty module. 'nb' 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 'nb' mode; see Note 2.

Monitor-only parameters:

Parameter	Type	Values	Comments
<active bank>	char	A | B | nb	'A' or 'B' if there is an active bank; '-' if no active bank; 'nb' if operating in 'non-bank mode'.
<active VSN>	char		VSN of active module, if any; if operating in 'nb' 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 'nb' 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 'nb' (i.e. non-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 'nb' mode if (and only if) both 'nb' 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.

bank_set

bank_switch – Enable/disable automatic bank switching (NYI)

[command list]

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

Query syntax: bank_switch? ;
Query response: !bank_switch ? <return code> : <auto-switch on/off> ;

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

Settable parameters:

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

Notes:

1. 'bank_switch' command will return error if system is operating in 'nb' (non-bank) mode.
2. 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.
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 readback will not work properly.
5. If the alternate Bank is not ready at the time switching is initiated, the recording or readback 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

clock_set – Specify CLOCK parameters

[command list]

Command syntax: clock_set = <clock frequency> : <clock_source> : [<clock-generator freq>] ;
Command response: !clock_set = <return code> ;

Query syntax: clock_set? ;
Query response: !clock_set ? <return code> : <clock frequency> : <clock source> : <clock-generator freq> ;

Purpose: Specify the frequency and source of the CLOCK driving the DIM

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<clock frequency>	int	2 | 4 | 8 | 16 | 32 | 64	-	DOT clock will advance according to specified frequency; see Note 1. Certain restrictions apply when <clock frequency> = 64; see Note 2. Must be specified before a 'DOT_set' command is issued – see Note 3.
<clock_source>	int	ext | int	ext	ext – VSI clock int – Use clock from internal clock-frequency generator
<clock-generator freq>	real	0 – 40 (MHz)	<clock frequency> (but max 40MHz)	Frequency to which internal clock generator is set. See Note 1. Note that clock generator can be set a maximum of 40MHz.

Monitor-only parameters:

Parameter	Type	Values	Comments
<clock frequency>	int		

Notes:

1. The DOT clock timekeeping will advance by counting clock cycles according to the specified value of <clock frequency>, regardless of the actual clock frequency. For example, if <clock frequency> is specified as 32 MHz, but the actual value is 16 MHz, the length of a DOT second with be 32,000,000 clock cycles, occupying 2 wall-clock seconds. Occasionally, such a mis-setting may be deliberate, mostly for testing purposes; for example, setting <clock frequency> to 64MHz when the actual frequency of the clock is 32MHz would allow testing 64MHz functions of the DIM at half-speed. Likewise, if the clock source is chosen to be the internal clock generator, the specified <clock frequency> need not necessarily correspond; the ratio of the <clock-generator freq> to <clock frequency> will determine the DOT clock rate relative to actual wall clock rate (e.g. OS clock rate).
2. If the <clock_frequency> is 64 MHz, the combination of bit-stream mask and decimation must be set so as not to exceed 1024Mbps aggregate bit rate, except for Mark 5B+.
3. A 'clock_set' command issue after a 'DOT_set' command will cause the value of the DOT clock to become indeterminant.
4. A 'DOT_set' command issued before a 'Clock_set' command will cause an error.
5. 'clock_frq' may be used as a synonym for 'clock_set' for VSI compatibility.

clock_set

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 | tvg | ?	ext – data from Mark 5B DIM input port tvg – data from internal tvg (as indicated by bit in disk frame header) ? – data not in Mark 5B format (might be Mark5A, for example); 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 frames contains a 16-byte header followed by 10000 bytes of data
<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 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:
1. 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 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 'nb' 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 'nb' 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 'nb' mode, disks in banks A and B are treated as a single module.

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.

Monitor-only parameters:

Parameter	Type	Values	Comments
<active bank>	char	A | B	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; 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 *dimino* 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 'nb' 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 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: Transfer data between start-scan and stop-scan pointers from Mark 5 to file.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<dest filename>	literal ASCII	no spaces allowed	See Comments	Default <dest filename> is as specified in Section 6 (i.e. '<scan label>_bm=<bit mask>.m5b'). Filename must include path if path is not default (see Note 5).
<start byte#>	int | null		See Notes	Absolute byte#; if null, defaults to start-scan pointer. See Notes 1 and 2.
<end byte#>	int | null		See Notes	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>			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 *dimino* or *Mark 5B* was started.

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: Transfer data between start-scan and stop-scan pointers from Mark 5 to network.

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 Note 1	Absolute byte#; if null, defaults to start-scan pointer. See Note 1.
<end byte#>	int | null		See Note 1	Absolute end byte#; if preceded by '+', increment from <start byte#> by specified value; if null, defaults to start-scan pointer. See Note 1.

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. 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 scan defined by 'scan_set' will be transferred.
3. 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).
4. 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.
5. 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).
6. 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

disk2net

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.

7. 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 5B back into idle. See ‘reset’ command for details.
8. Only one data transfer activity may be active at any given time. That is, among ‘record=on’, ‘in2net=..’, ‘disk2net=..’, ‘net2disk=..’, ‘net2out=..’, ‘disk2file’=..’, ‘file2disk=..’, ‘data_check’ and ‘scan_check’, only one may be active at any given time.
9. Note that the network protocol parameters are set by the ‘net_protocol’ command.

disk2net

DOT – Get DOT clock information (query only)

[command list]

Query syntax: DOT? ;
Query response: !DOT ? <return code> : <current DOT reading> : <sync status> : <FHG status> : <current OS time> : <DOT-OS difference> ;

Purpose: Get DOT clock information

Monitor-only parameters:

Parameter	Type	Values	Comments
<current DOT reading>	time		Current value of DOT clock.
<sync status>	char	not_synced | syncerr_eq_0 | syncerr_le_3 | syncerr_gt_3	'not_synced' – DOT 1pps generator has not yet been sync'ed. See Note 1. 'syncerr_eq_0' - DOT 1pps tick is exactly coincident with selected external 1pps tick. See Note 2. 'syncerr_le_3' – DOT 1pps tick within +/-2 clock cycles of selected external 1pps tick. 'syncerr_gt_3' – DOT 1pps tick more +/-3 clock cycles from selected external 1pps tick
<FHG status>	char	FHG_off | FHG_on	'FHG_off' – Frame Header Generator is not running (<current DOT reading> is software estimate) 'FHG_on' – FHG is running (<current DOT reading> is read from hardware FHG). See Note 3.
<current OS time>	time		Corresponding OS time
<DOT-OS difference>	time		<current DOT reading> minus <current OS time>

Notes:

1. A <sync status> of 'not_synced' indicates that the DOT hardware 1pps generator has not been sync'ed to an external 1pps tick. All other <sync status> returns indicate that the DOT 1pps generator has been sync'ed with a 'DOT_set' command. See also Note 2 of 'DOT_set' command.
2. The <syncerr_eq/le/gt_n> status returns are only relevant provided the selected external 1pps that was used to sync the DOT clock remains connected, selected and active and indicates quantitatively the current relationship between the DOT 1pps and the external 1pps to which the DOT clock was originally sync'ed. Note that if the external 1pps tick is asynchronous to the data clock (e.g. comes from a GPS receiver), one would not expect that exact synchronization would be maintained. Caution: the 1pps tick on the VSI-80 connector from a VSI Mark 4 formatter has a slow leading edge and may not always pass the <exact sync> test, though it should always pass the <approx sync> test.
3. DOT 1pps ticks are always counted by dimino to keep higher-order time; when data collection is inactive, the sub-second part of <current DOT reading> is estimated by software measurement of the time interval from the last DOT 1pps second tick. The hardware Frame Header Generator (FHG) runs only during active data collection and creates the Disk Frame Headers inserted periodically into the data stream transmitted to the StreamStor card; during this time, the <current DOT reading> is read directly from the FHG. The FHG is set up according to the parameters of each recording and is started at the beginning of each recording. The time resolution of the FHG is the Mark 5B disk-frame-header period, which is given by 80/(total-rate in Mbps) milliseconds, where the total data rate is determine by three parameters: clock rate (set by 'clock_set' command), bit-stream mask and decimation ratio (both set by 'mode' command); for example, for a total data rate of 1024Mbps, the time resolution of the FHG is ~78 *micro*seconds. For more details see memo 'Data Input Module Mark 5B I/O Board Theory of Operation'.

DOT_inc – Increment DOT clock

[command list]

Command syntax: DOT_inc = <inc> ;
Command response: ! DOT_inc = <return code> ;

Query syntax: DOT_inc? ;
Query response: ! DOT_inc? <return code> : <inc> ;

Purpose: Increment DOT clock time by specified number of seconds

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<inc>	int		0	Number of seconds to increment DOT clock (may be positive or negative). >0 will advance the DOT clock setting; <0 will retard the DOT clock setting

Monitor-only parameters:

Parameter	Type	Values	Comments
<inc>	int		

Notes:

1. The DOT_inc command should be used to adjust an error in the DOT clock only after the DOT clock has been synchronized to an external 1pps tick with the 'DOT_set' command.

DOT_inc

DOT_set – Set DOT clock on next external 1pps tick

[command list]

Command syntax: dot_set = <time> : [<option>] ;
Command response: !dot_set = <return code> ;

Query syntax: dot_set? ;
Query response: !dot_set ? <return code> : <time> : <option> : <time offset> ;

Purpose: Set initial value of Mark 5B DOT clock on next tick of selected external 1pps source

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<time>	time	null | time	OS time	DOT clock can only be set to an integer second value. See Note 1. If null, sets DOT clock according to current OS time.
<option>	char	null | force	null	If 'force', 1pps generator will be re-synced even though the <DOT_synced> status returned by 'DOT?' already indicates it is sync'ed. See Note 2. If null, 1pps generator will be synced only if <DOT_synced> status indicates it is not already sync'ed.

Monitor-only parameters:

Parameter	Type	Values	Comments
<time>	time		Includes all inferred higher-order time (e.g., year, DOY, etc) that was not explicit in command value of <time>
<option>	char		
<time offset>	time		Estimated interval between time from receipt of 'DOT_set' command to 1pps tick that set the DOT clock

Notes:

1. If <time> does not specify higher-order time (i.e. year, day, etc), the current values from the OS time are used. Of course, the DOT clock should be carefully checked with the 'DOT?' query after setting to verify that it is properly set.
2. Because the Mark 5B keeps higher-order time (above 1sec) in software, higher-order time will be lost *dimino* is restarted or the system is re-booted, but the DOT hardware 1pps generator will not lose sync as long as power and data clock (on the VSI 80-pin connector) are maintained; a re-issued 'DOT_set' command will set higher-order time without disturbing the existing 1pps synchronization unless the 'force' option is specified to force re-synchronization of the hardware 1pps generator.
3. After the 'DOT_set' command is issued, a 'DOT?' query should be subsequently issued to verify the proper time setting. If necessary, the 'DOT_inc' command may be used to adjust the DOT clock to the correct second.
4. If the DOT clock is set from OS time, care must be taken that the OS clock is reasonably well aligned with the external 1pps tick (to within a few tens of milliseconds, at worst). Otherwise, there is no requirement on the OS timekeeping apart from Note 2 above.
5. Following the setting of the DOT clock, the DOT clock keeps time based strictly on the selected CLOCK as specified in the 'clock_set' command and runs completely independently of the OS time; furthermore, the external 1pps is not used for any further operations other than cross-checking against the internally generated DOT1PPS interrupt and, in principle, can be disconnected.
6. Normally the DOT_set operation is only done once at the beginning of an experiment. See also Note 1 for 'DOT?' query.
7. Any change in 'clock_set' parameters enacted after a 'DOT_set' command will cause the value of the DOT clock to become indeterminate.
8. A 'DOT_set' command issued before a 'clock_set' command will cause an error.

DOT_set

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	mark5b	
<software revision date>	time		Date stamp on current version of *dimino* source code (*.c) files
<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 5B DIM command set revision level corresponding to this software release (e.g. '1.0')
<DIM design revision>	int		Revision level of DIM FPGA design on Mark 5B 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 5B

[command list]

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

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

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

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<source filename>	ASCII	no spaces allowed	'save.data' or last value	If not in standardized filename format (see Section 6), must specify at least <scan label> and recommend specifying <bit-stream mask> as well. See Note 1. Filename must include path if not default (see Note 5).
<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
<scan label>	ASCII	64 chars max	Extracted from <source filename>	Required if <source filename> is not in standardized format (see Section 6). Example: 'exp53_ef_scan123'
<bit-stream mask>	hex		0	Should be specified if <scan label> is specified. See Note 1.

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
<bit-stream mask>	hex		Bit-stream mask

Notes:

1. If <source filename> is in the standardized format for Mark 5B (see Section 6), *dimino* will parse the constituent fields to determine <experiment name>, <station code>, <scan name> and <bit-stream mask>. If source filename does not include a scan label in the proper format, <scan label> must be specified. If <scan label> is specified, it is recommended that <bit-stream mask> also be specified so that the Mark 5B directory entry can be properly completed.
2. The data in the source file must be in Mark 5B data format.
3. To abort data transfer: The 'reset=abort' command may be used to abort an active file2disk data transfer. See 'reset' command for details.
4. When <status> is 'inactive', a 'file2disk?' query returns <source filename> of the last transferred scan, if any.
5. Default path is the Linux default, which is the directory from which *dimino* or *Mark 5B* was started.

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 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 'nb' mode, disks in banks A and B are treated as a single module.

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 a 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 mode

[command list]

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

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

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

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<data source>	int	ext | tvg | ramp	ext	'ext' – data on VSI 80-pin connector 'tvg' – internal Test Vector Generator; see Note 1 'ramp' – internal ramp generator; see Note 2
<bit-stream mask>	hex	See Note 1	0xffffffff	
<decimation ratio>	int	1, 2, 4, 8, 16	1	Specifies ratio of data clock to recorded sample rate; if <data source> is 'tvg' or 'ramp', value of '1' should always be used - see Notes 1 and 2
<FPDP mode>	int	1 | 2	See Note 3	*For diagnostic use only*: Sets FPDP mode (FPDP1 or FPDP2). See Note 3

Monitor-only parameters:

Parameter	Type	Values	Comments
<data source>	int		
<bit-stream mask>	hex		
<decimation ratio>	int		
<FPDP mode>	int	1 | 2	

Notes:

1. As per the VSI-H specification, the tvg pattern resets at every second tick. The bit-stream mask selects which bits of the tvg pattern are recorded; a decimation value other than '1' should not be used except for special diagnostic testing as the resulting recorded data may not be recognized as tvg pattern by 'data_check' or 'scan_check'.
2. A 'ramp' pattern replaces the tvg with a 32-bit counter that starts at zero on the next second tick and increments each clock tick for 100 seconds before resetting to zero again. The bit-stream mask selects which bits of the ramp pattern are recorded; a decimation value other than '1' should not be used except for special diagnostic testing as the resulting recorded data may not be recognized as a ramp pattern by 'data_check' or 'scan_check'.
3. Mark 5B only supports FPDP1; any attempt to set to FPDP2 will cause an error. Mark 5B+ always defaults to FPDP2, but may be forced to FPDP1 for test purposes; maximum aggregate date rate for FPDP1 is 1024Mbps - an attempt to record at 2048Mbps in FPDP1 will cause error.

mode

net2disk – Transfer data from network to disks

[command list]

Command syntax: net2disk = <control> : <scan label> : <bit-stream mask> ;
Command response: !net2disk = <return code> ;

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

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			Scan label to be assigned to this data; if not specified, defaults to 'EXP_STN_net2disk' See Section 6 for format of scan label.
<bit-stream mask>	hex		0	<bit-stream mask> associated with data. See Note 1.

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
<bit-stream mask>	hex		Assigned bit-stream mask

Notes:
1. The <bit-stream> mask should always specified so that the Mark 5B directory entry can be properly completed.
2. See Notes with 'disk2net' command for usage rules and restrictions.
3. When <status> is 'inactive', a 'net2disk?' query returns <scan label> of the last transferred scan, if any.
4. Note that the network protocol parameters are set by the 'net_protocol' command.

net2disk

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_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

pointers – Get current value of record, start-scan and stop-scan pointers (query only)

[command list]

Query syntax: pointers? ;
Query response: !pointers? <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, an error flag is set in the 'status?' query which can be used as a first order check of proper operation.

pointers

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 – 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 | throttled | overflow | waiting	'halted' indicates end-of-media was encountered while recording. 'throttled', 'overflow' and 'waiting' are all error conditions.
<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:
    a. "throttled" – data rate from Mark 5B I/O card is too fast for disks to keep up (flag received by I/O board from StreamStor card)
    b. "overflow" – FIFO overflow on Mark 5B I/O card
    c. "waiting" – CLOCK has stopped or is faulty

record

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 'STN', respectively.
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.

record

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

Purpose: Reset system; mount/dismount disks

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<control>	char	erase | nberase erase_last_scan | abort		'erase'sets record, start-scan and stop-scan pointers to zero (i.e. beginning of media); effectively erasing media; 'nberase' is a 'non-bank' erase that performs the same actions as the 'erase' command except the modules in Banks A and B are initialized as a single unit and recording takes place across the disks in both modules; normally used only for 2Gbps operation with Mark 5B+. '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 2 System is always left in 'idle' 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.
4. The 'reset=nberase' command requires that disk modules are mounted and ready in both banks. Bank A must contain eight disks; bank B may have fewer, though it will normally have the same number. After the 'reset=nberase' 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 *dimino* place the system into non-bank mode or allow any read or write operations.

reset

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 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
<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>	literal ASCII		
<data type>	char	- | tvg | SS | ?	dash – Mark 5B format, but undetermined data type (probably real data) tvg – undecimated 32-bit-wide tvg data (see Note 4) SS – raw StreamStor test pattern data ? – recording not in Mark 5B format (might be Mark 5A, for example); all subsequent return fields are null.
<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 for data readback

[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>	char | time | int	s | c | e | s+ | <time> | +<time> | -<time> | +<bytes> | -<bytes>	s	s|c|e|s+: Set start scan 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' at a prescribed position. 's+' sets the start-scan 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
<stop read>	time | int	<time> | +<time> | -<time> | +<bytes> | -<bytes>	end-of scan	<time>: Time at which to end readback; see Notes 2 & 3. If preceded by '+', indicates duration of data (in record-clock time) from <start scan> time. +<time>: offset time from <start scan> position. -<time>: offset time from 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>	time		Scan label of scan matching <search string>
<start play>	time		Time at beginning of specified data range.
<stop play>	time		Time at end of specified data range.

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. If the <start scan> or <stop scan> parameter is a <time> value, this time must be specified with sufficient significance to resolve any ambiguity within the scan. For example, '30s' would set the start-scan 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.
5. The specified values of <start scan> and <stop scan> must be within the target scan.
6. 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_rev – Get StreamStor firmware/software revision levels (query only)

[command list]

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

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

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:

2. ‘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 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 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) 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) readback 'on' Bit 9 - (0x0200) end-of-scan or end-of-media (readback 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) DIM ready to record Bit 19 – (0x80000) DOM ready to play --- 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

TVR – Start TVR testing

[command list]

Command syntax: TVR = <tvr mask> ;
Command response: ! TVR = <return code>

Query syntax: TVR? ;
Query response: ! TVR? <return code> : <status> : <tvr mask> : <tvr error> ;

Purpose: Start TVR testing

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<tvr mask>	hex		Current bit-stream mask	Defines bit streams to be tested; reset <tvr error> flag to 0.

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	int	0 | 1	0 – TVR not running 1 – TVR running
<tvr mask>	hex		
<tvr error>	int	0 | 1	0 – no errors detected 1 – at least one error detected; reset <tvr error> flag to 0

Notes:

1. The TVR receives data from the VSI input and can be used only when the ext (VSI) clock has been selected by the 'clock_set' command. After issuing the 'TVR' command, the TVR will start operation on the next DOT 1pps tick. Only the bit-streams specified in the <tvr mask> will be tested; only undecimated TVG data can be tested. The tested bit streams must appear on the VSI input bit-streams in the positions expected for full 32-bit TVG pattern. A 'TVR?' query may be issued anytime after the TVR has started. Any single bit error on any of the selected bit-streams sets the <tvr error> status bit to '1', then and clears the <tvr error> in anticipation of the next 'tvr?' query.

TVR

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 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'). For non-bank-mode modules, see Note 7.

Monitor-only parameters:

Parameter	Type	Allowed values	Comments
<extended VSN>	char		Example: 'MPI-0153/960/1024'; see Notes 4 and 5. 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 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
<companion extended VSN>	char		If non-bank-mode module, returns VSN of companion non-bank module. See Note 8.
<companion bank>	char	B | A	Bank position of companion non-bank module

Notes:The 'VSN=..' command is normally issued only when the module is first procured or assembled, or when the disk configuration is changed. he serial numbers of the resident disks are noted.

1. 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.
2. A 'protect=off' command is required *immediately* preceding a 'VSN=' command, even if protection is already off.

VSN

3. The format of the extended-VSN is "VSN/capacity(GB)/maxdatarate(Mbps)" – example 'MPI-0153/960/1024'. The following rules are enforced by the *dimino*:
   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)
   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#.
4. *dimino* 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.
5. The recorded disk serial #'s are updated each time a scan is recorded.
6. A "VSN=.." command may not be issued to any module which has been initialized in non-bank mode.
7. 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.
8. 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