Update Mark-5 SoftwareUpdate Mark-5 Software
There is an updated version of a tar file now at:
http://web.haystack.edu/Mark5/Mark5A.tgz. You can download this file with a browser. A copy is at: ftp://web.haystack.edu/dist/mark5/Mark5A.tgz. You can download this file with ftp or with the script Mark5Update. This script uses ftp
to download the tar file from Haystack, unzips and untars the tar file, reinstalls the Jungo driver, and recompiles all the Mark-5 programs with various
error checks. This is intended to make upgrading easier and less prone to error.
After the latest upgrade, a copy of Mark5Update will be in the ~oper/bin directory for next time (but see notes below). Shut down all Mark-5 programs, then execute Mark5Update as root.
We'll try to keep this tar file updated whenever we make significant changes. You can see the date on this file using just the directory (in time order): http://web.haystack.edu/Mark5/.
If you can't or don't want to use Mark5Update, then make sure that no Mark-5 programs are running. Then the recommended untar and install procedure is (as root):

```
  tar  xzvpP  --same-owner  -f  Mark5A.tgz
```

with the path prefixed onto the file name if not the default. You can copy and paste this line into your Linux shell. If this untar does not generate any error
messages, then Mark5A (etc.) is ready to use, unless there is a Jungo driver update contained in this tar file or in an update that you skipped over (see notes below). If a Jungo driver re-installation is required after installing this tar file, then follow the procedure summarized in paragraph (3)(E) on page
3 of Mark5A Software. If you have trouble with individual programs such as tstMark5A, then see the frequently asked questions, Mark-5 FAQ, especially Q6. (But in some respects, both of these documents are out of date; sorry.) See additional notes below.
Note that this tar file contains nothing like a full installation. This update procedure works only on an already functional Mark-5 system.
Recent changes, additions, and fixes:
2003 October 08 (day 281); IMPORTANT: This version fixes a serious error in ROT-delayed play as used by Haystack-designed correlators. The previous version
(October 03) will NOT work with Haystack-designed correlators. This version of the Mark5A software substantially complies with version 2.6 of "Mark 5A command
set", and the 'DTS_id?' query with this version will return '2.6x', that is version 2.6 with extensions (x), for command set revision. This version of Mark5A was tested with input design revision (IDR) 0xb8 and output design revision (ODR) 0x19 and should work with all previous IDRs and ODRs but without
certain features as noted below. The 'DTS_id?' query shows this revision information. The current version of Mark5Update is September 29.
Bug fix: 'play=...' with a ROT-start argument (as used by Haystack-designed correlators) works again. Sorry.
Formal parsing (i.e., without -f 0 on the command line) has been made even more
formal: Mark5A now returns an error if a keyword is followed by neither '?' nor
'='. (Formerly a keyword alone (no '?' or '=') that was only a query was executed as a query, and a keyword that could be a command was executed as a command with defaults, if any. This caused some significant problems when a user
intended but forgot a '?'.) Informal parsing mode (i.e., with -f 0 on the command line) is unchanged.
The protection scheme has been modified as follows: A 'protect=off' command is

required immediately preceding a 'reset=...' or 'VSN=...' command, even if protection was already off. If not, an error return such as '!reset = 6 : Previous protect=off required ;' explains what is needed. This is an extension to version 2.6 of the writeup.
Bug fix: The default for 'scan_play=...' is now 'on' as in the writeup. (But note that the default for 'play=...' is 'off', and 'record=...' has no default.)

Bug fix: Several commands with error conditions were incorrectly returning '?' instead of '='.
Bug fix: A 'play=on' command following a previous 'play=arm' is not permitted to
have also a play pointer (byte number) (because the play pointer has already been used by the 'play=arm'). This case now returns an error.
2003 October 03 (day 276); IMPORTANT: Please read also the notes down through
the August-29 release below. This tar file contains a new release from Conduant,
sslinux60.tgz, dated September 24, which contains libraries that work with either version 2.x or 3.2 of the gcc compiler and its associated libraries. This
version of the Mark5A software substantially complies with version 2.6 of "Mark
5A command set", and the 'DTS_id?' query with this version will return '2.6x', that is version 2.6 with extensions (x), for command set revision. This version
of Mark5A was tested with input design revision (IDR) 0xb8 and output design revision (ODR) 0x19 and should work with all previous IDRs and ODRs but without
certain features as noted below. The 'DTS_id?' query shows this revision information. The current version of Mark5Update is September 29.
The cc5A compile script was updated to check which version of gcc (and g++) is present and update the symbolic link to the corresponding version of libssapi.a.
This was done to allow this version to be installed on Red-Hat Linux versions 8.x and 9.0 (as well as 7.x). This was checked in a Red-Hat 9.0 machine.
SSErase was rewritten to deal with write protection on either bank A or B. The operator will be asked whether to remove any write protection before erasing. Certain illegal situations now generate proper error messages. Additional information is now available from SSErase at msglev -1 (i.e., -m -1).
The 'play' and 'scan_play' commands and queries were rewritten to comply with version 2.6 of the writeup except that distinguishing between between 'arming' and 'armed' in the query responses is determined by calculated timings rather than feedback from the Conduant buffer.
Bug fix: In case a data transfer is in progress, 'scan_set=...' returns an error
and does nothing else (because it might change the data-transfer parameters).
Bug fix: 'bank_set=...' no longer wastes time by causing an unnecessary second call to CheckBanks() (and checkDisks() and readdir()).
Bug fix: A warning message from 'rtime?' is now only in case of a significantly
slow disk.
Bug fix: 'reset=erase' and 'reset=erase_last_scan' now also reset the reference
recording index used to determine the warning "Record pointer not incrementing."
This prevents some spurious warning messages.
The 'replaced_blks?' query in version 2.6 of the writeup is now implemented. It
works only when no data transfers (e.g., play or record) are in progress.
Bug fix (sort of): 'data_check?', 'track_check?', and 'scan_check?' now verify also the two required zero bits in the byte after sync in the frame header.

(This byte contains two BCD numbers in both Mark-4 and VLBA formats.) Although counter-intuitive, this allows 'track_check?' to find frame headers, previously
missed, in some partly flawed tracks with long strings of ones, which can be caused by no input to the video converter or no video converter connected to the
formatter for this track. This change is either good or bad depending on whether
you want 'track_check?' to complain about such partly flawed tracks (the track is written correctly; the data are flawed.)
Bug fix: Play_rate now correctly defaults to clock 9 MHz.
Bug fix: 'record=on', 'net2disk=...', and 'file2disk=...' now check for write protect and, if so, then return an error and do nothing else. (Previous versions
returned an error but then tried to write to disks anyway and became very confused.)
Bug fix: 'scan_check?' with "skipped" unknown now returns 0 but with '?'s to indicate this error (presumably a faulty scan).
Commands 'scan_set=...' and 'track_set=...' now have a 'dec' (for decrement) argument (as well as an 'inc' for increment). Both wrap or unwrap as appropriate. This is an extension of the specifications in writeup versions 2.5
and 2.6.
The maximum socbuf size was increased from 999424 to 3997696 bytes, where socbuf
is the second undocumented additional parameter to 'in2net=...' and 'disk2net...'. An error is generated by any attempt to exceed this maximum. (The
default socbuf size is unchanged at SOCBUF = 131072 bytes.) The larger size is intended to help "tuning" eVLBI transfers.
2003 September 29, IMPORTANT: An error was found in the Mark5Update script. The latest version, dated September 29, fixes a reboot problem. If you use the old version of this script, then your Mark 5 will work until you need to reboot,
then may fail because the Jungo driver may not be properly installed during the
reboot. This seems to depend on which version of Linux you have. You may also fix this by hand as follows: In the file /etc/rc.d/rc.local, change the line /sbin/insmod windrvr6 to /home/streamstor/linux/driver/redist/wdreg windrvr6. If
you execute this line by hand, then also execute the following line: chmod 0666
/dev/windrvr6.
2003 September 04 (day 247); IMPORTANT: Please read also the notes for the August-29 release below. The current version of Mark5Update is September 09. The
'DTS_id?' query with this version will return '2.5x' for command set revision, that is version 2.5 with extensions (x). Except for 'play', 'scan_play', and 'replaced_blks' however, this version closely complies with version 2.6 of "Mark
5A command set". It was tested with input design revision (IDR) 0xb8 and output
design revision (ODR) 0x19 and should work with all previous IDRs and ODRs but without certain features as noted below. The 'DTS_id?' query shows this revision
information.
This release contains a new sspxf.bib file (September 04) from Conduant, which fixes two problems: (1) Bank B now works with delayed play as used at correlators, and (2) bank A's "Ready" light does not fib about being ready on startup.
Bug fix: 'File2disk' now sets source file name default to "save.data".
Bug fix: In 'disk2net' and 'disk2file', if the starting and ending byte

numbers
are not integer multiples of 8, then they are truncated down and handled correctly.
The inline copy of the directory is no longer written, but we still try to read
it (e.g., at a correlator) if the separate directory can't be read.
2003 August 29 (day 241); IMPORTANT: This version is in a renamed tar file, namely Mark5A.tgz (rather than Mark5A.tar.gz), and a new version of Mark5Update
(September 02 or 09) is needed to download and install it. Do not try to use the
previous version of Mark5Update (June 02) with this tar file. This new Mark5Update script (September 02 or 09) contains several necessary changes, including editing /etc/rc.d/rc.local to accomodate the renamed Jungo driver, that is windrvr6 instead of windrvr. This all happens automatically using this new script.
This tar file contains a new release from Conduant, sslinux60Beta, dated August
27. (But note that some older releases had the same name but now have "old" suffixes.)
This version of the software agrees with version 2.5 of the writeup but with extensions (2.5x) as noted below. It was tested with input design revision (IDR)
0xb8 and output design revision (ODR) 0x19 and should work with all previous IDRs and ODRs but without certain features as noted below. The 'DTS_id?' query shows this revision information.
The stand-alone SSErase program with conditioning (-c 1) now prints the disk statistics obtained during conditioning. This might allow identification of faulty disks, but a suspicious disk should always be retested because disks might have been improved by the conditioning process---that's the point of doing
it. Conditioning is a process rather different from normal recording and playing. To use the statistics after conditioning, try to compare one disk with
another in the same pack or at least after the same conditioning process.
The SSErase program has also been modified to check for write protection (which
applies only to a disk pack) and ask the user whether to remove the write protection, as is necessary, of course, for erasure or conditioning.
Note that SSErase does not have a reset at the end as, for example, Mark5A does.
This allows erasing multiple disk packs sequentially without waiting for the normal resetting and checking processes that accompany XLROpen. The stand-alone
program SSReset performs the reset that is missing from SSErase and so causes resetting and checking at the next startup of Mark5A or any other SS-aware program.
An optional experiment name and source name (or source name and experiment name)
have been added as two additional parameters on 'record=on...' (no embedded spaces). (This is labeled as "NYI" in version 2.5 of the writeup.) DirList, 'scan_set?', and 'scan_check?' now show these added parameters separated by spaces from the scan name. DirList was modified to leave room to print the added
parameters.
The 'scan_set=...' command can now use abbreviated scan names. If its first parameter is all numeric, then 'scan_set' will first try to interpret this parameter as a scan number. Failing that (e.g., not that many scans), then 'scan_set' will find the first scan whose scan name, experiment name, or source
name contains the string in this first parameter. (Like almost everything else in Mark5A, this search is case insensitive.) This is an extension of the

specifications in writeup version 2.5.
The start byte number that is set and reported by 'scan_set' was augmented as follows: It defaults to the start of the scan or can be set by 's', 's+', 'c', or 'e' as before. Or it can be set, instead, by an offset byte number, preceded
by '+' (no space), to be added to the start of the scan, or preceded by '-' (no
space), to be subtracted from the end of the scan to give the start byte number.
Or it can be set by a time in standard notation (no embedded spaces): 'nnndnnhnnmnns'. Any of these time units can be omitted, and any omitted parameters to the right of any prescribed parameters are taken to be zero. As an
example, '5h' is the same as '5h0m0s'. Any omitted parameters to the left of any
prescribed parameters mean to ignore the corresponding time units in the data. Mark5A will then, if possible, set the start byte number to the first occurence
of this time within the prescribed scan. (But note that '2m', which means '2m0s', will not start at 4m, 6m, and so forth.) Alternatively, a '+' preceding
such a time (no space) tells Mark5A, instead, to just add this time onto the start of the scan to give the start byte number. The play pointer also is set to
this same start byte number by the scan_set command, but the play pointer can be
changed by the play command, and it increases while playing. These are changes in the specifications from writeup version 2.5.
There is now also an additional parameter on 'scan_set', the end byte number, which defaults to the end of the prescribed scan. Or the end byte number can be
set by an offset byte number, preceded by '+' (no space), to be added to the start byte number (determined as above---not necessarily the start of the scan),
or preceded by '-' (no space) to be subtracted from the end of the scan. As another alternative, the end byte number can be offset from the start byte number by a time parameter in the same time format as above. And if a '+' (no space) preceds this time parameter, then the time offset is added to the start byte number determined as above. If a '-' (no space) preceds this time parameter, then the time offset is subtracted from the end of the scan. These are changes in the specifications from writeup version 2.5.
These starting and ending byte numbers as set and reported by 'scan_set' are now
used in scan_play. This is a change in the specifications from writeup version 2.5.
In 'disk2net' and 'disk2file', the defaults for the start byte and end byte numbers are now the start and end byte numbers as set and reported by 'scan_set'. And, in 'disk2file', the destination file name defaults to the name
of this scan (in the current directory). These defaults can, of course, be overwritten by explicit parameters. These are changes in the specifications from
writeup version 2.5 and also changes from the previous update (below).
Bug fixes: 'File2net' and 'file2disk=...' now correctly translate 0 for the length of a file into the actual length even for files over 3 Gbyte, and they also correctly deal with non-zero file starting bytes.
Bug fix: In 'file2disk', the default scan name is the file name without the path.
The 'bank_set=...' command was rewritten. If the bank that you ask for is not the bank already selected, then this command returns a code of 1 meaning delayed
completion. Bank changing takes a variable amount of time up to almost 3

seconds. While bank changing is in progress, many commands and queries return a code of 5 meaning busy---try again later---or 6 meaning conflicting request. (In effect, neither bank is mounted during this transition.) If an attempt to change the bank fails (e.g., if there is no ready disk pack in the other bank), then a 'status?' or 'error?' query will show error 1006, "Bank change failed." And a 'bank_set?' query will show whether the bank has changed. Changing banks can also generate other errors if there are problems with the new bank.
Bug fix: A signal handler for SIGPIPE now handles a broken pipe (e.g., talking to the Field System) to keep Mark5A alive. This generates error 1005, "Broken pipe".
Bug fix: Now 'disk2file?' query works again.
Bug fix: Now 'disk2net?' query does "waiting" correctly.
Bug fix: Debuggery was added for pthread_create() failure, and pthread_detach() is now used to avoid leaking thread resources. This bug fix is needed at correlators that use delayed play (i.e., all Haystack-designed correlators).
Bug fix: 'Disk_serial', 'disk_model', and 'disk_size' now work without generating error messages even during recording or playing.
Bug fix: 'VSN?' query now correctly checks disk serial numbers against its stored serial numbers and reports any errors (except that these can't be checked after an SSErase). There is also added information in case of a mismatch.
Bug fix: Reset=erase now uses XLRErase() (instead of XLRRecord() and XLRStop()) and also clears the user-directory area, which clears the remembered serial numbers. (There is no effect on VSN.) This fix is intended to eliminate the problem of remembered old data that were recorded with previous SS versions.
Bug fix: In case of a bad or missing disk, the check-directory flag is turned back on when bank or disks are changed.
Bug fix: On 'play=off:', the play pointer is set to this even if playing ended at a different byte number. And 'play=off' without a updates the play pointer only if playing was in progress and this command stopped it.
The 'rtime?' query was modified to calculate a better approximation to the remaining time and capacity using the effectively reduced capacity of the disk set in case one or more of the disks is slow enough to impair performance. At this release, however, this better approximation does not work during recording:
Expect more accurate numbers from 'rtime?' only when idle, and expect less accurate numbers during recording. Note also that this situation---one or more slow or faulty disks---also increases the rate at which the remaining disk space is used up and reduces the maximum data rate that the disk set can accept. (The 'dir_info?' query was not changed.)
The stand-alone DirList program was modified to accept an alternative input file name. With no command-line arguments, DirList reads and lists /var/dir/Mark5A as before. There are now two setable parameters: -m sets the debug message level, msglev (range -1 to 3, default 1), and -f sets the file name, default /var/dir/Mark5A. This allows reading a copy of the directory file as sent, perhaps by email, from stations.
The delayed-play scheme used at Haystack-designed correlators was changed to use the new "play-arm" option from Conduant. The SS buffer is set to start filling a few seconds in advance of the desired start ROT; then a play trigger starts actual playing with only sub-millisecond delays. This is intended to almost

eliminate the need for skips.
The 'get_stats' function now also shows a count of the number of SS blocks that
were replaced with fill pattern during the previous playing. This count, for each disk, appears as the last number on the printed line. Anything other than 0
for this count indicates a problem with this disk. This is an extension of the specifications in writeup version 2.5.
The 'in2net' function has given us a lot of trouble and was rewritten. Among several changes, clocking in the input board is now turned off and on by in2net
using "notclock" to define the start (on) and end (off) of each scan. Version 2.5 of the writeup is still correct except that the "#bytes sent" as returned by
the 'in2net?' query is now the number of bytes received (including the bytes in
the FIFO waiting to be sent) since 'in2net=connect'. And the 'in2net?' query has
an added int parameter, the number of bytes in the FIFO waiting to be sent. After 'in2net=off', you should wait for this last parameter to drain to 0 before
'in2net=disconnect' to avoid losing data.
After 'in2net=off' but before 'in2net=disconnect', "#bytes received" (formerly "#bytes sent") can be logged as an index into the received file or scan at the target computer. This is a good approximation even if the "#bytes remaining in buffer" is not 0 and becomes exact when this number drains to 0. This is an extension of the specifications in writeup version 2.5.
Proper operation of the 'in2net=off' feature requires version 0xb7 or later of the "Input design revision" (IDR, which is the next-to-last parameter returned by the 'DTS_id?' query). Otherwise you'll need to turn off the formatter clock between scans to stop data flow to the I/O board. (The FS can do this.)
Bug NOT yet fixed: 'in2net' wants a scratch disk pack in bank A even though it doesn't need or use it.
2003 June 12 (day 163); This version agrees with writeup version 2.5 but with
extensions (2.5x) as noted below. It was tested with input design revision (IDR)
0xb8 and output design revision (ODR) 0x19 and should work with all previous IDRs and ODRs but without certain features as noted below. The query 'DTS_id?' shows this revision information.
The 'dir_info?' query can now be used during record, play, and other data transfers.
The stand-alone SSErase program was augmented to do disk conditioning. With no command-line arguments, SSErase does a simple erase as before. There are now two
setable parameters: -m sets the msglev (range -1 to 3, default 1), and -c sets conditioning (0 for FALSE, 1 TRUE, default FALSE). So, for example:

```
  SSErase  -m 0  -c 1  &
```

goes through the long conditioning process on whatever disks it can access, up to 16 at a time in both banks. With msglev set to 0, debug prints about every minute to show what's happening.
Conditioning disks is recommended before recording especially if they are to be
recorded at or near their maximum data rate. Conditioning amounts to a write-read-write cycle through the whole set of disks, but here it is done in two rather than three passes. Printed with debuggery and counting down twice is
the number of bytes per bus. With an 8-disk pack, for example, this count starts
at a quarter of the total capacity, which is twice the capacity of a single disk. Conditioning one 120-Gbyte disk takes about 101 minutes; two 120-Gbyte disks (as two masters), 103 minutes; four 120-Gbyte disks (as four masters),

111
minutes; eight 120-Gbyte disks (an eight pack), 157 minutes; and sixteen 120-Gbyte disks (two eight packs), 278 minutes. 200-Gbyte disks take about 5/3rds as long---no surprise. Conditioning one 200-Gbyte disk takes about 160 minutes; eight 200-Gbyte disks, 286 minutes; and sixteen 200-Gbyte disks, 465 minutes. These times are approximate for the ideal case, but, if any of the disks has a problem, times can be much longer. After conditioning two disk packs
at a time (e.g., more than eight disks), or in any case where you change the disks into a different configuration, then you'll also need to erase each disk pack separately before recording.
When the recording speed (e.g., formatter clock) and mode are such that the disks can't keep up, either because of an improper setup or a faulty disk or disks, then the recording will be "throttled" (some data are lost), and a flag indicating this throttling (called a suspend flag because recording is momentarily suspended) is now available to Mark5A. This flag causes error 1003,
"Recording throttled (can't keep up)" and, on 'status?', bit 0x400 set. In marginal conditions, this flag may flicker on and off, and this error, as with most errors, may be remembered after the error condition is fixed. Clear the error with 'status?' or 'error?'. This feature requires version 0x19 or later of
the "Output design revision" (ODR, which is the last parameter returned by 'DTS_id?'). The added status bit 0x400 is an extension of the specifications in
writeup version 2.5.
Bug fix: With ODR 0x19 and later, a 'mode?' query reports the correct output mode and output submode.
In disk2net and disk2file, if the end byte number is preceded by a + (no space),
then this number will be interpreted, instead, as the byte length of the transfer. This is equivalent to saying that this number will then be added to the start byte number to give the effective end byte number. This is an extension of the specifications in writeup version 2.5.
In disk2file, if the destination filename (minus path) matches an existing scan
name, then the start and end byte numbers default to the start and end of this scan. These defaults can, of course, be overwritten by explicit numbers. This is
an extension of the specifications in writeup version 2.5. We recommend that you
do not use this feature because it will be changed in the next release.
2003 June 4 (day 155); This tar file contains a new partial release from Conduant (May 28), which fixes some problems with playing or reading with missing or failing disks. See notes below on fill pattern.
The VSN command was augmented to record the list of serial numbers when the VSN
command is issued and also at recording. The VSN query was augmented to cross check this list of serial numbers against the disks actually present and return
an error, "Disk serial-number mismatch," if they differ.
Bug fix: A 'status?' query now distinguishes between net2disk and net2out.
Bug fix: 'get_stats?' doesn't generate spurious XLR errors on missing slave disks but does report disk number.
Bug fix: 'reset=abort' now actually works as advertised.
2003 May 24 (day 144); An interim update from Conduant (May 23) is in this tar file. This fixes a bug in recording on bank B that affected the directory on
bank A.
To deal with a missing or faulty disk in a disk pack, this version sets a Conduant option that allows reading and playing in spite of this problem. Fill pattern replaces the missing data. (See notes on fill pattern below.) This

might
allow correlators to salvage some of the data from such a faulty disk pack.
Data_check, track_check, and scan_check were modified to deal with certain
kinds
of disk-reading errors. If at least one valid frame header can be found (but
not
all five), then these queries report all the known information but without the
track-frame period in time or in bytes. These two parameters are replaced by
'?'s to indicate unknown and probably an error. This should allow correlators
to
try to use such flawed data.
Bug (re)fix: net2disk and net2out do not generate incorrect error 1001 after
the
other machine has disconnected.
2003 May 20 (day 140); This is a very minor update to fix an error in the
makefile for the Jungo driver (in $SS/driver).
The 'position?' query was improved: It can now be used during playing, and its
play pointer gives an (approximate) answer during playing in progress.
Also during playing, polling now sets an error (1002) in case the play pointer
is not incrementing. This is not the same as playing halted (end of scan or
end
of data), which is not considered to be an error. Check for either case with a
'status?' query.
2003 May 16 (day 136); This tar file contains a new release from Conduant,
6.0beta, dated May 15, which adds some new features and fixes some errors as
noted below.
Some of the following changes were made to be in accord with Revision 2.5 of
the
writeup.
Bank mode is always on, and the 'bank_mode' command and query and the
'reset=dismount' and 'reset=mount' commands are no longer supported (except in
the old Anova chassis).
ROT rate read from ROT broadcasts is now used (instead of a fixed 32e6) to
allow
for speed_up. (This applies to correlators only and to delayed play only.)
Play_rate is not affected.
Bug fix: net2disk and file2disk do not generate incorrect error 1001.
Bug fix: we think that EndM5 now works on all versions of Linux.
Bug fix: disk2file now correctly does default file name.
Bug fix: reset=erase now properly clears the directories (but also leaves a
copy
of the (empty) directory in line on disk, so the first scan starts at byte
81952.)
Bug fix: record=off at end of medium (full disks) now works properly.
Bug fix: the fourth argument of mode=... now correctly defaults.
Most errors are now remembered (even if printed with debuggery) and printed
(and
cleared) by either a 'status?' or an 'error?' query. Thus errors may be
remembered even after they have been corrected. (This may be regarded as
either
an improvement or a bug fix.)
Following Linux tradition, the Mark5A command line now has a help option, -h,
which lists the various startup parameters.
The 'protect' command and query (write protection) now work except that the
'protect?' query does not work in non-bank mode (Anova chassis), and
protection
in non-bank mode has not been tested.
The 'play=on' command and several other commands now check more carefully for
other data transfers in progress and return errors if so.
The extra debuggery from a 'bank_mode?' query was moved to 'bank_set?'
(because
bank_mode command and query are no longer supported).

We have augmented the 'status?' query as follows:

bit 12	disk2file	0x00001000
bit 13	file2disk	0x00002000
bit 14	disk2net	0x00004000
bit 15	net2disk	0x00008000
bit 16	in2net	0x00010000
bit 17	net2out	0x00020000

(Some of these are still partly ambiguous in this version.) When one of these bits is set, the corresponding command is active (transferring data) or waiting to become active. In this case, other data-transfer commands will return errors.

We have further augmented the 'status?' query as follows:

bit 20	0x00100000	Bank A selected
bit 21	0x00200000	Bank A ready
bit 22	0x00400000	Bank A full
bit 23	0x00800000	Bank A write protected
bit 24	0x01000000	Bank B selected
bit 25	0x02000000	Bank B ready
bit 26	0x04000000	Bank B full
bit 27	0x08000000	Bank B write protected

These add, of course, so a typical response might be:

```
  !status? 0 : 0x02300001 ;
```

which means that bank A is selected and ready and bank B is ready. This bank-status augmentation of 'status?' works only when data transfers (e.g., recording and playing) are not in progress.

Also 'status?' bit 0x000008, delayed completion, is now set whenever any data-transfer activity, such as recording, playing, or transfers to or from disk or net, is active or waiting.

The 'rtime?' query now also returns the percentage remaining unrecorded disk space as its third parameter.

A new reset command, 'reset=abort', aborts data transfers disk2net, disk2file, and file2disk. (We may expanded this list in the future to abort other operations also.) This command returns immediately, but there may be a delay of up to two seconds before the data transfer stops. During this delay, 'status?' queries will show what's happening.

This version of Mark5A was upgraded to set and use the fill-pattern detection in the output section of the I/O board. On playing, fill pattern inserted by the SS card in lieu of unreadable data is converted to an equally long wrong-parity time sequence, which should enable the correlator to maintain synchronization through this unreadable-data segment. This requires version 0x18 or later of the "Output design revision" (ODR, which is the last parameter returned by 'DTS_id?').

A number of small changes in this version make it work better with the old Anova chassis (in non-bank mode, of course). These changes were only partly successful; this version in an Anova chassis generates some spurious error messages. Even though most things work, this is likely to be annoying.

2003 April 10 (day 100); The tar file with this date, Mark5A.100.tar.gz, remains available on web for downgrading.

Important: This version of Mark5A implements bank mode and contains other major changes. This version of Mark5A can read and play SS disks recorded with older versions, however, older versions of Mark5A can not read or play SS disks recorded with this version. Thus correlators should be upgraded to this version

before attempting to read or play SS disks recorded with this version.
This version works with the older Anova chassis, but, of course, without bank mode.
Important: This version has an upgrade from Conduant, sslinux452beta.tgz, which
contains an updated Jungo driver; see notes above. The sspxf.bib file in $SS/bib
and the library in $SS/lib are also special later updates.
Important: The command-line options for Mark5A have been changed to be in accord
with Linux conventions. There are three setable parameters: -m sets the msglev (range -1 to 3, default 1), -f sets formal parsing (0 for FALSE, 1 for TRUE, default TRUE), and -s sets the maximum number of simultaneous control socket connections (range 1 to 7, default 7). At field stations with old versions of the Field System, try:

```
  Mark5A  -m 0  -f 0  &
```

to give debug printing and informal parsing. At Haystack-designed correlators, try:

```
  Mark5A  -m 0  &
```

to give debug printing (and formal parsing). Note that parts of page 5 of Mark5A
Software are thus out of date.
Many of the following changes were made to be in accord with Revision 2.4 of the
writeup.
Track_check now has track data rate (as in scan_check and data_check) and decoded track number converted to 2--33 or 102--133 notation and followed by a 'D' to indicate a duplicated track (i.e., the track you asked for is correctly a
duplicate of this track in this mode), or '?' to indicate an error (i.e., the wrong track was found). The '-' in place of missing bytes was deleted.
DTS_id now also shows the command-set revision and the input- and output-design
revisions from the I/O board. The input- and output-design revisions will both be 0 in case there is no I/O board (i.e., Mark5P).
DTS-id now reads a file /etc/hardware_id for the serial number and uses the system name only if this file is not readable or not the correct length.
Bug fix: Corrected error checking in mode command.
Bug fix: Rtime now reloads the record pointer before calculating. This allows rtime to give an (approximate) answer during recording in progress.
This new Conduant version contains working bank-mode functions, and the bank_select and bank_mode commands and queries have been rewritten to correspond. Also status polling now checks periodically on which bank is selected. This allows the operator to switch banks using just the key switches and disk insertions. (But automatic bank switching during record or play is not
yet implemented.)
This version from Conduant also contains label commands, which were used to implement VSN. (But the VSN serial-number cross-checking is not yet implemented.) To support VSN, tstMark5A now accepts key words as short as 3 chars.
Bank_mode accepts just one parameter, which now defaults to "on", and bank_mode
is "on" at startup if in a TK200 chassis. Bank mode can't work in the old Anova
chassis.
Bug fix: Bank_set now re-reads status and disk information whenever the bank changes.
Previous errors newly reinserted in informal-parsing mode: '=' in response to disc_check and disc_serial queries.
Bug fix: Dplay() (delayed play, correlator only) now writes an error message and

starts playing immediately if the requested start ROT is in the past or less than 0.1 second in the future.
Reset has another option for its parameter: "erase_last_scan".
Bug fix: Reset=... tries to re-read device status only if disks are mounted.
Reset=dismount is illegal in bank mode.
Bug fix: Bank_mode=on now redoes the initialization.
Bank_set has another option for its parameter: "inc" changes to the other bank.
Scan_check, data_check, and track_check now calculate the frame period in bytes
using an algorithm that is more likely to show any errors. Scan_check, in particular, is now more persnickety. The point is, after all, to show up errors
and problems.
Scan_set now also accepts "s+" as its second argument to set the play pointer to
65536 bytes past the start of the scan.
Various improvements were made in in2net, which should work now and should report overflow (i.e., can't keep up) to either an in2net or status query.
Bug fix: The status query now works properly during recording or playing.
During recording, polling now sets an error (1001) in case the record pointer is
not incrementing. This is not the same as recording halted for end of medium, which is not considered to be an error. Check for either case with a 'status?' query.
Various minor improvements were made especially in error messages and in minor housekeeping tasks associated with bank switching.
2003 March 5; Conduant has fixed a bug in their on-board software dealing with LBL for disks greater than 137 Gbytes. If you are using 200-Gbyte SS disks
in a Mark 5, then you should upgrade from the February-20 version of Mark5A. The
upgrade is in this sspxf.bib file. After upgrading to the February-20 tar file,
then put this sspxf.bib file into $SS/bib overwriting the older file with the same name. Then SSReset and restart Mark5A. (Do not apply this sspxf.bib upgrade
to later versions.)
2003 February 20; This version will show "2003y044d14h" in the response to "DTS_id?" (because the latest Mark-5 update therein is February 13, but the latest Conduant update and the date of the tar file is February 20). The tar file with this date, Mark5A.pre.tar.gz, remains available on web for downgrading.
This tar file includes a new sspxf.bib file from Conduant, which should make the
in2net command work again. This same bib file also contains a work around for the bug in LBA in Western Digital disk drives with "JB" in the model number.
Bug Fix: The mode command now generates an error return on an attempt to go into
64-track mode when the resulting play_rate clockgen would be above 45.36 MHz. The play_rate command also checks this. The actual limit is for the track data rate not to exceed 40 MHz for up through 32 tracks or 20 MHz for 64 tracks.
Many of the following changes were made to be in accord with Revision 2.3 of the
writeup.
On record=off, the scan number is now set to the just-recorded scan, and the play pointer is set to the beginning of this scan.
Dir_info is a query only and no longer changes the scan number.
The scan_dir query is no longer supported, and, instead, the scan's starting and
ending byte numbers were added onto the return from the scan_set query in place

of the number of scans. (The number of scans is available from dir_info.)
A scan_check query no longer increments the scan pointer.
Name changes for this new writeup: Changed track_select to track_set and changed
bank_select to bank_set. (But the old names also still work.)
Other changes: One of the errors previously fixed was put back in order to work
with the old version of the Field System, namely an '=' after the '?' in the return from the position query. This incorrect '=' occurs only in INformal parsing mode (described below), which is necessary for this version of the Field
System.
The scan_set command now allows "inc" as its first argument, which increments the scan number or, after the last scan, loops back to the first.
The track_select command now allows "inc" in place of either or both of the track numbers to mean increment to the next higher-numbered track or loop back to 2 at the end of the list.
The second argument of scan_set now defaults to "S", so the play pointer is always set by this command (by default to the start of the scan).
Bug fix: Scan_play is now more nearly correct, and play handles end-of-medium in
a consistent way. Playing stops at the end of the scan (scan_play) or end-of-medium (either play or scan_play) and updates the play pointer, but the corresponding queries then return "halted" to show what happened.
Bug fixes: More incorrect "=" deleted from query replies.
Note that many of the comments below for the previous release apply also to this
one.
2003 January 28; Scan_play now is also a query.
Disk_serial, disk_model, and disk_size are now faster; they each should now come
in under the one-second limit even with 16 disks. (The delay was moved to the reset=mount sequence, which now also has more debuggery.)
Data_check, track_check, and scan_check in case SS or TVG now return three additional numbers in order: starting and ending word numbers (32 bits each) where the pattern was found, and the length of the buffer. In the ideal case, the first number would be 0, and the last two numbers would be equal.
Bug fix: Device info is needed on reset=mount.
Note that many of the comments below for the previous release apply also to this
one.
2003 January 24; Dozens of small changes were made for this update. Most were
made to be compatible with Revision 2.2 of the writeup. Note particularly that the arguments of the mode command and query were revised, and the returns from data_check, track_check, and scan_check were correspondingly revised. Verify that you have a compatible suman to use this version at a Haystack-designed correlator. A tar file with the previous version (Mark5A.old.tar.gz, 2002 December 27) remains available on web.
Mark5A now has formal and informal parsing modes. In formal parsing mode, most of the '=' and ':' prescribed in the writeup are required. This allows, for example, the sequence ": :" to mean to take the default or the previous value (depending on the specs for the command) for the missing parameter that would be
between the two ':'s. In informal parsing mode (the only parsing mode in previous versions of Mark5A), many of these parsing rules are relaxed. In particular, just a space can be used in place of "=" and ':'. A '-' can now be used as a placeholder in informal parsing mode. Also a successful record=on returns 1 (delayed completion) in formal parsing mode but returns 0 in informal
parsing mode.
Informal parsing mode might be helpful when you are typing commands and

queries
by hand. At the field stations, you might have to use informal parsing mode: Check your version of the Field System to determine which is wanted. Haystack-designed correlator software expects Mark5A to be in formal parsing mode.
This version of Mark5A normally comes up in formal parsing mode. To select informal parsing mode, add a second parameter of 0 on the command line. For example:

```
  Mark5A  0  0  &
```

where the first 0 selects debuggery, and the second 0 selects formal parsing mode off, that is informal parsing mode.
This version of Mark5A allows functions that don't need to talk with the SS board to work even when disks are dismounted or are being mounted or dismounted.
This allows, for example, configuration and interrogation of the I/O board while
disks are being changed.
There is a new query, rtime, for recording time remaining to end of medium, and
the corresponding parameter was deleted from dir_info.
Bank_mode and bank_select are present but not yet functional (NYI).
Data_check returns blank as its last parameter in case skipped cannot be determined, and this is probably not an error. But scan_check returns '?' in the
same situation because this probably is an error. We now try to use '?' and blank consistently in this way.
This tar file contains a slightly revised SDK version (sslinux541.tgz, January 16) from Conduant but with the same Jungo driver.
Bug fixes: Many changes were made to the handling of play_rate for various modes
to make these consistent. These changes are intended to be helpful, but, if in doubt, set play_rate after changing modes.
Bug fix: Ignore formatter serial number in VLBA mode.
Bug fix: Delete '=' sign in the return from various queries.
Bug fix: The spurious error message during reset=mount was programmed around (we
still don't know where it's coming from).
Bug fix: Mark5A now does a CardReset before an error halt.
2002 December 27; This version incorporates almost all the updates and changes in Revision 2.1 of the writeup.
Bug fix: Data_check, track_check, and scan_check now do not back over the start
of the buffer and now correctly skip a tape-frame header that is truncated by the prescribed starting byte number.
Bug fix: A copy of the directory is written to file /var/dir/Mark5A (for DirList
to read) at record=off even if the SS disks are full (end of medium).
2002 December 18; This release back tracks on a recent change that is incompatible with current versions of the Field System, namely the return code from record=on. This was changed back to 0 pending a release of the FS that understands return code 1 for this case.
Bug fix: Input_mode (obsolescent) now handles an odd-numbered formatter error correctly.
2002 December 16; Bug fix: A floating-point round-off error in scan_check, data_check, and track_check was fixed by forcing the time offsets between nearby
tape frames to be exact integer multiples of 1.25 milliseconds and the time offsets between distant tape frames to be exact integer multiples of this (previously determined) tape-frame spacing. (The infamous quarter-millisecond offset was already being corrected.)
Bug fix: At play=off after scan_play, PlaybackLength is set back to 0 to avoid affecting a subsequent play=on;

If a reset=mount or reset=dismount fails, it can now be retried. This allows recovery from certain disk-mounting errors without exiting Mark5A.
Bug fix: An extra record=off when not recording no longer gives a bogus scan containing only a copy of the directory.
2002 December 9; Mark5A functions disc2file and file2disc were updated to Large File Support (LFS), which allows reading and writing files larger than 3 Gbytes. A minor error in file2disc was also corrected.
Mark5A on startup with msglev 0 now contains a desperation attempt to read the directory. Try this if you're sure that your disk(s) have a directory but Mark5A
can't otherwise read it.
This tar file also contains some minor upgrades from Conduant, namely sstest, ssopen, and sspxf.bib. The sstest and ssopen upgrades affect only users with eight-pack modules using both together. The sspxf.bib upgrade fixes the return to bypass after either recording or playing. At the field stations, if your decoder fails between scans (i.e., when you're not recording), then you need this upgrade.
The return codes have been updated to agree with the latest draft revision of "VLBI Standard Software Interface Specification---VSI-S" (see VSI-S). Most notably, code 3, formerly "syntax or parameter error," was split into code 3, "syntax error," and code 8, "parameter error." Code 7 now means "no such keyword," for example, mispeld. And code 1, delayed completion, is now used in all appropriate cases including record and play. This might require changes in cooperating software such as the Field System.
2002 December 4; This tar file contains a slightly revised SDK version (sslinux54.tgz) from Conduant but with the same Jungo driver.
The stand-alone programs Net2file and File2net now support files larger than 2 Gbytes using Large File Support (LFS) under Linux, HP-UX, and perhaps other operating systems. With blocks of data larger than 2 Gbytes, use Net2file with the Mark5A command disc2net to transfer data from SS disks to ordinary files, and use the Mark5A command net2disc with File2net to transfer data from ordinary
files to SS disks. Use these schemes even on the same machine (localhost) because the Mark5A commands disc2file and file2disc do not (yet) support large files (LFS).
Beware: There are some bugs in manipulating files larger than 2 Gbytes. Under Linux, for example, using '*' and '?' for file-name expansion fails under csh and tcsh but works under sh and bash.
Track numbers for the 8- and 16-track cases were revised yet again. The track_check command now returns "-" as its last argument (in place of skipped) if the track number read from the tape-frame header does not agree with the number in track_select. (And this generates a Mark5A warning message if debuggery is on.) Tracks with data that are duplicated (fanned out) on play according to Mark 5A Track Mapping, tables 1 and 2, are now also checked by track_check, but, if you track_select a duplicated track, then "-" instead of skipped will show that the track number in the tape-frame header is not the same
as the track number that you requested.
For VLBA mode, track numbers are checked against table 18, page 24 in Mark IIIA/IV/VLBA Tape Formats, Recording Modes and Compatibility. If incorrect, the
warning message will show the expected and found VLBA track numbers from this table.
2002 November 27; Commands to the I/O board are now sent only when the corresponding parameters have changed. This allows, in particular, changing track_select during play without losing sync. Scan_play is partly functional: It
starts playing at the start of the scan, plays to the end, and stops. But it does not go into bypass mode at the end of playing, and it does not update the play pointer, unless or until a play=off. Readback of VLBA mode in the mode query was fixed. Scan_set will now accept either b (for beginning) or s (start)

for the starting byte position of a scan. The test program, tstMark5A, now uses
readline() instead of fgets() to read what you type. This allows line editing and history recalls of previous commands and queries. See the man or info pages
on readline. But this readline() version of tstMark5A.c no longer compiles on HP-UX and other platforms, alas. The April-30 version, which works on HP-UX, is
available as tstMark5A.c.bak alongside the tar file. Don't forget to re-install
the Jungo driver if you're upgrading from a much older version.
2002 November 25; This tarball contains yet another new release from Conduant
with yet another new driver from Jungo. Important: A Jungo driver
re-installation is required after installing this tarball. Other changes include: Revision and SMART information were added to the debuggery that prints
during initialization. A check for odd-numbered formatter serial numbers was added, and mode=... returns an error if an attempt is made to configure an odd-numbered formatter for a parity-striped mode. The track numbers in 8- and 16-track modes were changed again. Scan-set returns an error if it can't interpret its second argument. Data_check, scan_check, and track_check now all work correctly in 64-track mode provided that the paired tracks (i.e., the tracks with 100 added to or subtracted from the track numbers) are configured and have valid sync. If an incorrect track number is found by track_check and debuggery is on, then a warning message is printed. There are also several minor
bug fixes.
2002 November 18; Important: The check for even-numbered formatter serial numbers (noted below) was removed from this tarball because some in-use formatters (notably at Westford) still have odd-numbered serial numbers. Most of
the changes in Revision 1.9 of the writeup have been made in this tarball, but some of the obsolete commands and queries, such as input_mode and play_mode, have also been retained. There are several minor bug fixes.
2002 November 15; Bug fixes: data_check, track_check, and scan_check should now work correctly with any even-numbered formatter. Clock frequency should now
be correct for 64-track mode. A messy business with a play=off (from suman on the correlator) interfering with a reset=dismount or reset=mount seems to have been fixed. Also some, but not all, of the changes in Revision 1.9 of the writeup have been incorporated.
2002 November 14; Bug fix: The track_check query was flawed in the previous version. The interactions between the results returned by data_check, track_check, and scan_check and the commands to the I/O board (input_mode and play_mode) have been removed. This update does not yet incorporate the changes in Revision 1.9 of the writeup.
2002 November 12; This tarball contains many updates summarized as follows: Conduant has now incorporated the new Jungo driver with scatter/gather. This improves the speed of eVLBI operations. A Jungo driver re-installation is required after installing this tarball. To be compatible with this new Jungo driver, mmap(), as used to write the directory file, was replaced by ordinary write()s. The directory reading at startup is now more robust. The play_rate query now returns two frequencies, the effective rate as specified, and a new third (last) parameter: the actual rate including parity bits (*9/8) and, with VLBA format, also the header bits (*126/125). (Is this correct for the VLBA correlator?) The dir_info query now returns the estimated recording time remaining (in seconds) as its fifth (last) parameter. This parameter is accurate
provided that the input_mode, play_mode, and play_rate parameters are correctly
set. The track numbers are now correct for the 16- and 8-track cases. The

efficiency of scan_check, data_check, and track_check was improved. A test that
the bit preceding sync must be 0 was added to reduce false sync detections. (The
new I/O board also needs this.) Another stand-alone program, DirList, for
directory listing, was added to this tarball; see the writeup on page 7 of
Mark5A Software.
2002 October 10; This tarball contains many upgrades summarized as follows:
Mark5A now talks with the new (under development) combined I/O board. On
machines with the old separate input and output boards, Mark5A detects this and
falls back to the old stub behavior. Many small changes are associated with
these new capabilities. The new outBoard() and inBoard() are in a separate file,
IOBoard.c. Other fixes include: Revised wait-for-start-play (used at the
Haystack correlator) and improved interrupt of this wait on abort. TVG clock was
deleted from input_mode (there is no such clock). Mark5A now checks for and
rejects negative play pointer. The status query now works almost always.
Get_stats query now increments the drive number even if error or no such drive.
Day number in data_check, track_check, and scan_check was corrected (+1) in case
VLBA format. But this release does NOT contain the new faster Jungo driver.
2002 September 23; Fixed track_select to accept and reject the correct track
number ranges (accept 2 to 33 and 102 to 133).
2002 September 23; Several minor fixes, which should avoid "Segmentation
violation" on scan_check, data_check, or track_check.
2002 September 19; Fixed status query so that it is mostly correct even when
recording or playing.
2002 September 18; Fixed the spurious WARNING message on startup in case no
slave drives.
2002 September 18; Added an unauthorized additional parameter to set_scan or
scan_set: "start" or "s" sets the play pointer to the start of the scan, "c" or
"center" sets the play pointer to the center of the scan, and "e" or "end" sets
the play pointer to 1000000 bytes before the end of the scan, which allows
data_check or track_check to start there.
2002 September 18; Increased read-buffer size to accomodate 64-track mode (in
data_check, track_check, and scan_check). This takes some extra milliseconds in
case a check fails.
2002 September 17; Scan_play partly works---it starts playing the proper scan
but doesn't stop at the end of the scan. Use play=off to stop.
2002 September 16; Scan_check now increments the scan counter even if it
can't decode VLBI data (but not on errors).
2002 September 16; Set_scan or scan_set does NOT update play pointer to that
scan. (Reverse change below.)
2002 September 16; Record query now returns "halted" in lieu of "off" if end
of medium. (Play query already had a "halted" if end of recording.)
2002 September 16; Record query now returns scan number and scan name.
2002 September 16; Set_scan or scan_set command now updates play pointer to
that scan.
2002 September 16; Fixed short-file error on disc2file.

2002 September 16; Added time and speed debuggery for disc2net and disc2file.

2002 September 13; Fixed (actually a work-around for) the missing-directory-after-reboot problem.
2002 September 10; Fixed date and time "Error" in data_check, track_check, and scan_check with VLBA-format data. Other minor changes especially to debug prints.
2002 September 4; Scan_set and set_scan both work, are equivalent, and are now also queries.
2002 September 3; Scan_check now seems to be working. This query is recommended for Field System use after each scan in lieu of data_check.
2002 August 30; Not-erased disk hangs up trying to read directory: No fix, but now message says what's wrong. Try SSErase.
2002 August 30; Revised sspxf.bib (was faulty in 5.3.beta) from Conduant. Skip query results for negative skip are now correct.
2002 August 29; New API and bib files from Conduant: 5.3.beta. As a result, skip as query now works. Fixes also to skip command.
2002 August 28; Two fixes to play: Dplay now rejects waits greater than one minute, and play=off cancels a wait in Dplay, if any.
2002 August 28; Added buffer-size parameters as unauthorized additional parameters to in2net and disc2net or disk2net (sndbuf (default 16384) and socbuf
(default SOCBUF = 131072) bytes) and to net2out and net2disc or net2disk (rcvbuf
(default 87380) bytes). See setsockopt(). These are used for speed "tuning" of eVLBI.
2002 August 28; Scan_set and scan_dir are implemented, and next_scan is retained as a synonym for scan_dir (with added scan number).
2002 August 23; Minor fixes to net2disc and disc2net.
2002 August 22; Two new Mark-5-associated stand-alone programs, Net2file and
File2net, are ready for beta testing and are included in this tarball. Instructions will be ready soon; meanwhile read the headers of the source files,
Net2file.c and File2net.c.
2002 August 19; Added "skipped" as the last parameter on data_check and track_check. This shows the number of bytes skipped from the last previous data_check or track_check and should be zero if both are within the same scan.
2002 August 16; Changed ROT timing for Dplay() (used only at correlator).
2002 August 15; Fixed track_check to check just the prescribed track. (Formerly it wandered around on other tracks.)
2002 August 13; Net2disc now accepts and shows scan name.
2002 August 12; Track_check seems to be working. Use track_select to set which track to check (but with no effect on hardware).
2002 August 8; Modifications to date and time printed from data_check? We now
attempt to calculate and print the year and day of the year as a best guess, but
these will be wrong if data were taken more than about 10 years ago for Mark-4 format or 1000 days (about 2.74 years) ago for VLBA format.
2002 August 7; Another fix to VLBA time in data_check? (Day number will be 0.)
2002 August 6; Disc2file and file2disc seem to be working.
2002 August 5; Start_stats and get_stats now conform to the published specs except for default values.
2002 August 2; New libssapi.a from Conduant. Start_stats and get_stats now work but with counts instead of seconds for bin definitions.
2002 August 2; fixed VLBA time in data_check? New sspxf.bib from Conduant.
2002 August 1; fixed disc_size?, disc_serial?, and disc_model? for case multiple masters and no slaves.
2002 July 31; fixed disc_model? to include the whole string.

2002 July 31; fixed status check on data_check? Revised skip.
2002 July 30; fixed glitch at end of response to DTS_id?
2002 July 25; new functions: SS_rev1, SS_rev2, OS_rev1, OS_rev2
2002 July 24; new libssapi.a from Conduant
2002 July 24; new API (5.3.alpha_forMIT.tgz) from Conduant
2002 July 12; new ssatap3.bib from Conduant
Revised: 2003 October 09, JAB

Four Mark-5A Tutorials

You should have a copy of the Mark-5A software writeup, "Mark 5A command set," available at http://web.haystack.edu/mark5/command5a.pdf.

(1) Record short scans from the test vector generator (TVG) and check them

On the Mark-5 machine, put a disk pack (not write protected) into bank A, and turn on the bank-A keyswitch. We will erase and write on this disk pack, so be sure that there is *no data that you want to save* on this disk pack in bank A. Put another disk pack (not write protected) into bank B and turn on its keyswitch. We will write on (but not erase) this disk pack in bank B. Then start up the Mark5A program:

```
Mark5A -m 0 &
```

That -m 0 gives debuggery, which is helpful if something goes wrong, but which we'll ignore in this tutorial, and the & puts Mark5A in the background to allow this terminal to be used also for other things. After a minute or so, when Mark5A is ready, see (the last line):

```
Mark5A Ready.  End with EndM5, please
```

Start the tstMark5A program optionally, and for this tutorial preferably, in a different terminal:

```
tstMark5A
tstMark5A Ready (end with ^C)
>
```

If tstMark5A is run on a different machine, then the tstMark5A command line needs to have the name of the Mark-5 machine (it defaults to localhost). Lines typed by the operator into tstMark5A are preceded by the prompt > and are sent through a socket to Mark5A. The Field System can also do this; use a 'mk5=' prefix.

Check status:

```
> status?
!status? 0 : 0x02300001 ;
```

A ? ends a query. The line beginning with ! is the response to the status query from Mark5A. The first 0 in the response signifies that this query completed OK. The 1 in the far right means ready. The embedded 2 means that disk bank B is ready, and the 3 (that's 2 + 1) means that disk bank A is ready and selected. (This happens automatically, if possible, on Mark5A startup.) If there had been a pending error, then the '!status?' query response would have printed an error message and cleared the error, if possible, for next time. More information on the '!status?' response is in **Appendix A**, and detailed information on all the Mark-5A commands and queries is in "Mark 5A command set" cited above.

Which version of the software and hardware do we have?

```
> DTS_id?
!DTS_id? 0 : Mark5A : 2003y202d10h : 1 : mark5-21 : 1 : 1 : 2.5x : 0xb4 : 0x17 ;
```

This says that the Mark-5A software was last diddled on year 2003, day 202, at about 10 o'clock. This machine's serial number is mark5-21. The corresponding writeup is version 2.5 but with extensions (x). The I/O board has input design revision 0xb4 and output design revision 0x17. Certain features depend on these software and hardware versions.

The status query above shows that bank A is selected, but we can also check with:

```
> bank_set?
!bank_set? 0 : A ;
```

Next let's erase whatever is now on the disk pack in bank A. Be *sure* that you don't want to save any data now on this disk pack. Then:

```
> reset = erase
!reset = 0 ;
```

That = sign (instead of a ?) after reset means a command. Spaces in command and query lines are optional; I've put spaces to be more readable. This erase operation takes only a fraction of a second, and the 0 return means that it has completed successfully.

Set the mode to TVG, test-vector generator, to prepare to make a TVG recording:

```
> mode = TVG
!mode = 0 ;
> mode?
!mode? 0 : tvg :  : st :  : - : 0 ;
```

Many keywords, such as mode, can be either queries (with ?) or a commands (with =). Mark5A is not case sensitive (except for file names), so TVG and tvg are equivalent. In the TVG case, only the first and second parameters returned by the mode query are significant.

The internal TVG is on the Mark-5 I/O board, and it's clock is set by the play_rate command (even though we'll be recording rather than playing). The maximum clock rate is determined by the number of disks and the mode; here let's try 8 MHz (8 Msamples/sec or 8 Mbaud per track):

```
> play_rate = data : 8.0
!play_rate = 0 ;
> play_rate?
!play_rate? 0 : 8.000 : 8.000 : 8.000 ;
```

In the TVG case, the three numbers from play_rate are the same.

Start recording (the scan name, TVG-1, is arbitrary):

```
> record = on : TVG-1
!record = 1 ;
```

That 1 in the response means delayed completion, in this case that is recording in progress, and Mark5A will continue recording this scan until recording is commanded off (or until space runs out on the disks). Some or all of the lights on the front of the disk pack should now be on, but, since we're recording at only a modest rate, maybe not fully bright.

Some queries work even while recording is in progress:

```
> status?
!status? 0 : 0x00000049 ;
```

That 49 is 40 (recording in progress) plus 8 (delayed completion action in progress) plus 1 (ready) and no known errors. (We lose the bank-status responses while a data transfer is in progress.)

The position query can be used to track the progress of recording:

```
> position?
!position? 0 : 320024576 : 0 ;
```

A little later:

```
> position?
!position? 0 : 672067584 : 0 ;
```

That second (big) number is the record pointer; if it does not increase with time during recording, then something is wrong with recording.

Let it record for a minute or so, then end this scan:

```
> record = off
!record = 0 ;
> record?
!record? 0 : off : 1 : TVG-1 ;
```

So recording is now off, and we've just made scan number 1, whose name is TVG-1.

At record=off, the play pointer is set to the start of the scan just recorded, so we can check a snippet of this scan with:

```
> data_check?
!data_check? 0 : tvg : 0 : 249856 : 249856 ;
```

A data_check, track_check, or scan_check on a TVG scan all give the same result. The three numbers after tvg are the starting word number where Mark5A first found TVG pattern, the word number where TVG pattern ended, and the size of the buffer, all in units of 32-bit words. So the correct answers for these three numbers are 0 for the first, and the second and third should be equal. Although we've tested only a small snippet of it, this looks to be a good TVG scan. We could play this scan, but without a test-vector receiver (TVR), we can't test playback.

Now let's change to bank B:

```
> bank_set = B
!bank_set = 1 ;
```

Bank_set=inc (meaning increment or, in this case, change) would have done the same. The 1 return means delayed completion; expect up to a 3-second delay while the bank is switching. During this time, many commands and queries return errors. You can also change banks by turning *on* the keyswitch on the bank you want and turning *off* the keyswitch on the bank you don't want. Then check this changeover:

```
> bank_set?
!bank_set? 0 : B ;
```

The status query also shows that bank B is now active:

```
> status?
!status? 0 : 0x03200001 ;
```

More information about the status-query response is in **Appendix A**.

The mode and play_rate settings have not changed, so let's make another short TVG recording, this one onto the disks in bank B:

```
> record = on : TVG-2
!record = 1 ;
```

And after recording for at least a few seconds, then end this scan:

```
> record = off
!record = 0 ;
> record?
!record? 0 : off : 35 : TVG-2 ;
```

So we've just recorded scan number 35, whose name is TVG-2. On record=off, the scan number (scan pointer) defaults to the scan just completed, so:

```
> scan_check?
!scan_check? 0 : 35 : TVG-2 : tvg : 0 : 249856 : 249856 ;
```

This also looks to be a good TVG scan. More about scan_check is in the following section.

If this is a test scan that we do not want to keep, then we can `reset=erase_last_scan` to delete just this one scan, or we can `reset=erase`, as above, to delete all scans on the disks in this disk pack.

(2) Record a VLBI scan and copy a prescribed one minute of it to another machine

Mark5A and tstMark5A should be running from the previous tutorial. We can use the disks in either bank. Now we want to make a VLBI recording from a formatter, so the formatter needs to be set up, checked, and connected to this Mark-5's I/O board. We assume that this has been done, but we can also check some parts of this setup and connection. Let's set Mark5A to record 16 even-numbered tracks from a Mark-4 formatter:

```
> mode = mark4 : 16
!mode = 0 ;
> mode?
!mode? 0 : mark4 : 16 : mark4 : 16 : S : 1 ;
```

In the response to the mode query, the first `mark4:16` is for the input section, the second for the output section of the Mark-5A I/O board. That S means that the output section sees the correct format of VLBI data and has synchronized to it; this is an *important check* for recording VLBI data.

Start recording (the scan name, TestMark4-16, is arbitrary):

```
> record = on : TestMark4-16
!record = 1 ;
```

Use the position query to track the progress of the record pointer:

```
> position?
!position? 0 : 43909526022 : 0 ;
```

A little later:

```
> position?
!position? 0 : 44012206180 : 0 ;
```

If the record pointer stops increasing with time, then something is wrong with recording. And a status query will show certain errors:

```
> status?
!status? 0 : 0x00000049 ;
```

This all seems to be OK.

Let it record for a minute or two, then end the recording:

```
> record = off
!record = 0 ;
```

Check the scan that we've just recorded:

```
> scan_check?
!scan_check? 0 : 36 : TestMark4-16 : mark4 : 16 : 2003y199d14h35m26.590s : 159.6s : 4.000 : 0 ;
```

At record=off, the scan number (scan pointer) defaults to the scan just recorded. From left to right, the 0 means that scan_check completed OK, this is scan number 36, whose name is TestMark4-16, it's mode is mark4, 16 tracks, and it started at year 2003, day of the year 199, 14 hours, 35 minutes, 26.590 seconds UT, it contains 159.6 seconds of data, at 4.000 Msamples/sec (nominal track data rate), and that last 0 means that no data were lost or added during this scan.

We can also do a data_check starting at the current play pointer, which, at record=off, defaults to the start of the scan just completed:

```
> data_check?
!data_check? 0 : mark4 : 16 : 2003y199d14h35m26.590s : 19632 : 0.00500s : 40000 : 0 ;
```

This data_check response repeats some of the information above. Also the first frame header was found 19632 bytes into the scan, and there are 0.00500 seconds of time and 40000 bytes (on the SS disks) from one frame header to the next.

Since this was a 16-track recording, only even-numbered tracks from 2 to 32 were recorded. If a decoder is available, then we can play this scan and check tracks on the decoder, two at a time. Or we can check tracks on the disks using track_check. Let's first point to track 2:

```
> track_set = 2
!track_set = 0 ;
```

Now a track_check will check this track:

```
> track_check?
!track_check? 0 : mark4 : 16 : 2003y199d14h35m26.590s : 19632 : 0.00500s : 4.000 : 2 : 0 ;
```

This track_check response repeats some of the information above, but the 2 in the next-to-last position shows that track 2 was found correctly. Let's try track 3, which we can get by an inc (for increment), which adds one to the previous value:

```
> track_set = inc
!track_set = 0 ;
> track_check?
!track_check? 0 : mark4 : 16 : 2003y199d14h35m26.590s : 19632 : 0.00500s : 4.000 : 2 D : 0 ;
```

Instead of track 3, track_check reports, correctly, a duplicate of track 2 (2 D). If the wrong track had been found, then that D would have been a ?.

Let's try track 32:

```
> track_set = 32
!track_set = 0 ;
> track_check?
!track_check? 0 : ? ;
```

Oops! That ? means that track 32 is faulty or was not correctly recorded. In general, a returned blank means unknown; a ? alone means unknown and also probably an error. Further checking shows that tracks numbered above 16 were not correctly recorded. Something must be done about this!

To use rtime (below), we need to set the play_rate to agree with the formatter's sample rate (clock rate) divided by the fanout to get a track data rate. (As a check, the formatter's track data rate is displayed by the decoder, and the scan_check above has also calculated it.) In this case, the formatter was set to 4 MHz with no fanout:

```
> play_rate = data : 4.0
!play_rate = 0 ;
```

Now how much disk space is left for future recordings?

```
> rtime?
!rtime? 0 : 54415.0 : 435.321480064 : 90.7 : mark4 : 16 : 4.000 : 64.0 ;
```

This means that we can record approximately 54415 more seconds of data onto the 435.3 Gbytes (which is 90.7% of the total) remaining on these disks provided that we don't change mode or clock rate. The mode used in these calculations is mark4:16, the track data rate is 4.000 MHz (Msamples/second or Mbaud), and this corresponds to 64.0 Mbaud total recording rate.

Next let's copy part of this scan to a file on another machine (which need not be a Mark 5). First we need to set up to receive a file on this other machine (fritz, in this case) using the stand-alone program Net2file. To a Linux shell, try:

```
Net2file
```

Here we've taken all the defaults; the data will be saved in a file named save.data in the current directory. Note that we set up the *receiving machine first*.

Then, back in the terminal with tstMark5A:

```
> scan_set = : 36m : +1m
!scan_set = 0 ;
```

After the = sign in this command, we could abbreviate the scan name, provided that there is no preceding scan with the same abbreviation in its name, or, in this case, leave out the scan name because it defaults to the last scan recorded. The second parameter in this command sets the start byte to the first (and in this case only) occurrence of 36m0s in this scan. (Otherwise we might have to calculate and type those eleven- or twelve-digit numbers.) We can verify this start-byte calculation by another data_check, which reads data starting at the play pointer (same as start byte in this case):

```
> data_check?
!data_check? 0 : mark4 : 16 : 2003y199d14h36m00.000s : 0 : 0.00500s : 40000 : 0 ;
```

The third parameter in the scan_set command above (+1m) also sets the end byte to one minute later in time than the start byte. Thus we're ready to write one minute of data from 14h36m to 14h37m to, in this example, a file on the other machine. To send this off, we first connect to this other machine, whose name is fritz:

```
> disk2net = connect : fritz
!disk2net = 0 ;
```

And verify that we're ready to send:

```
> disk2net?
!disk2net? 0 : waiting : 43619737088 : 0 : 44099737088 ;
```

Those two big numbers are the starting and ending byte numbers as calculated and set by scan_set above. Start the transfer:

```
> disk2net = on
!disk2net = 1 ;
```

The 1 response means delayed completion; in this case, data transfer is in progress. As time passes, we can check how we're doing:

```
> disk2net?
!disk2net? 0 : active : 43619737088 : 43907571199 : 44099737088 ;
```

That middle number, the now byte, increases and will equal the end byte when the transfer is complete:

```
> disk2net?
!disk2net? 0 : waiting : 43619737088 : 44099737088 : 44099737088 ;
```

Then, since this is all that we want to send just now, end the transfer:

```
> disk2net = disconnect
!disk2net = 0 ;
```

As a result of this disconnect, on the target machine (fritz), you should now see:

```
Net2file:  The End
```

On this machine, try:

```
ls -l save.data
```

And see:

```
-rw-rw-r--    1 jball    users    480000000 Jul 18 12:01 save.data
```

File save.data contains the prescribed one minute of data. As an exercise for the reader, transfer this file to the SS disk(s) on a Mark-5 machine using File2net on fritz and net2disk in Mark5A; then use scan_check to verify good data and the prescribed times.

(3) Check a recorded disk pack

Here we check a disk pack already recorded and either ready to be sent to a correlator or already at a correlator. Start up Mark5A and tstMark5A and check status as above. The mode doesn't matter, and we don't need a formatter or decoder because we don't intend to either record or play. DirList is a stand-alone program to be run from a Linux shell prompt on the Mark-5 machine. It prints the contents of the directory as read by Mark5A. Here is an example:

```
   DirList
  nscans 227, n 226, recpnt 657003363992, plapnt 0, playRate 8.000 MHz
   n' scan name                                  start byte       end byte
 ---- ---------                                  ----------       --------
    1 188-1700a                                       81952     1603782808
    2 188-1704a                                  1603864760     3529321832
    3 188-1706b                                  3529403784     7021613440
    4 188-1709a                                  7021695392     8979248240
    5 188-1711                                   8979330192    10584384520
    6 188-1713a                                 10584466472    15581564320
    7 188-1716                                  15581646272    17186440760
   . . .
  219 189-0425                                 631003995856   637408867376
  220 189-0430                                 637408949328   639558271360
  221 189-0434                                 639558353312   641163785328
  222 189-0436a                                641163867280   647569356496
  223 189-0452                                 647569438448   649174512968
  224 189-0454                                 649174594920   650779792528
  225 189-0456                                 650779874480   652385314896
  226 189-0502                                 652385396848   653990159864
  227 189-0504                                 653990241816   657003363992
```

Note that we ran DirList after starting Mark5A and after the correct disk pack is active; otherwise we might have seen the directory from a different disk pack previously in this machine. This disk pack has 227 scans, numbered from 1, with names and byte positions as listed. This tabulation can be compared with the schedule (or ovex) to check for missing scans, and the starting and ending byte numbers can be checked against the log (or lvex) or even used to replace incorrect numbers.

Let's check the first scan:

```
> scan_set = 1
!scan_set = 0 ;
> scan_check?
!scan_check? 0 : 1 : 188-1700a : mark4 : 32 : 2003y188d16h59m50.097s : 50.1s : 8.000 : 0 ;
```

As in the tutorial above, reading from left to right, this scan check is OK, on scan number 1, whose name is 188-1700a, the mode is mark4:32, and this scan started at year 2003, day of the year 188, 16 hours, 59 minutes, and 50.097 seconds UT, the scan lasted 50.1 seconds, the nominal track data rate was 8.000 Msamples/second, and 0 data were lost or added. Since this scan's name is 188-1700a, we expect it to start on day 188 at about 17:00 UT, and it actually started about 10 seconds early. You might want to verify these parameters; for example, was this scan really intended to last only 50 seconds?

We can check a sequence of these scans as follows:

```
> scan_set = inc ; scan_check?
!scan_set = 0 ; !scan_check? 0 : 2 : 188-1704a : mark4 : 32 : 2003y188d17h03m56.062s : 60.2s : 8.000 : 0 ;
> scan_set = inc ; scan_check?
!scan_set = 0 ; !scan_check? 0 : 3 : 188-1706b : mark4 : 32 : 2003y188d17h06m21.080s : 109.1s : 8.000 : 0 ;
> scan_set = inc ; scan_check?
!scan_set = 0 ; !scan_check? 0 : 4 : 188-1709a : mark4 : 32 : 2003y188d17h08m50.057s : 61.2s : 8.000 : 0 ;
> scan_set = inc ; scan_check?
!scan_set = 0 ; !scan_check? 0 : 5 : 188-1711 : mark4 : 32 : 2003y188d17h10m57.062s : 50.2s : 8.000 : 0 ;
```

What's done here is to type a command, scan_set=inc, which increments the scan number, and a query, scan_check?, on the same line with a semicolon as separator. Then this dual-purpose line can be repeated, as shown, by typing just an up-arrow and Enter for each scan to be checked: An up-arrow copies the previous line and Enter sends it again through tstMark5A to Mark5A. Thus we have quickly checked scans 2 through 5 with only two keystrokes per scan.

We can spot check also some scans from near the end of this disk pack:

```
> scan_set = 224 ; scan_check?
!scan_set = 0 ; !scan_check? 0 : 224 : 189-0454 : mark4 : 32 : 2003y189d04h54m42.095s : 50.2s : 8.000 : 0 ;
> scan_set = inc ; scan_check?
!scan_set = 0 ; !scan_check? 0 : 225 : 189-0456 : mark4 : 32 : 2003y189d04h56m46.097s : 50.2s : 8.000 : 0 ;
> scan_set = inc ; scan_check?
!scan_set = 0 ; !scan_check? 0 : 226 : 189-0502 : mark4 : 32 : 2003y189d05h02m17.105s : 50.1s : 8.000 : 0 ;
> scan_set = inc ; scan_check?
!scan_set = 0 ; !scan_check? 0 : 227 : 189-0504 : mark4 : 32 : 2003y189d05h04m05.098s : 94.2s : 8.000 : 0 ;
```

Note that we first put scan_set=224 to point to somewhere near the end, then repeat the dual-purpose line as above. These all seem to be OK.

Let's also do at least spot checks of the tracks in a scan (the last scan in this case) using a similar scheme. Tracks are numbered from 2, and all tracks from 2 to 33 should have been recorded. So try:

```
> track_set = 2 ; track_check?
!track_set = 0 ; !track_check? 0 : mark4 : 32 : 2003y189d05h04m05.098s : 41904 : 0.00250s : 8.000 : 2 : 0 ;
> track_set = inc ; track_check?
!track_set = 0 ; !track_check? 0 : mark4 : 32 : 2003y189d05h04m05.098s : 41904 : 0.00250s : 8.000 : 3 : 0 ;
> track_set = inc ; track_check?
!track_set = 0 ; !track_check? 0 : mark4 : 32 : 2003y189d05h04m05.098s : 41904 : 0.00250s : 8.000 : 4 : 0 ;
> track_set = inc ; track_check?
!track_set = 0 ; !track_check? 0 : mark4 : 32 : 2003y189d05h04m05.098s : 41904 : 0.00250s : 8.000 : 5 : 0 ;
. . .
!track_set = 0 ; !track_check? 0 : mark4 : 32 : 2003y189d05h04m05.098s : 41904 : 0.00250s : 8.000 : 30 : 0 ;
> track_set = inc ; track_check?
!track_set = 0 ; !track_check? 0 : mark4 : 32 : 2003y189d05h04m05.098s : 41904 : 0.00250s : 8.000 : 31 : 0 ;
> track_set = inc ; track_check?
!track_set = 0 ; !track_check? 0 : mark4 : 32 : 2003y189d05h04m05.098s : 41904 : 0.00250s : 8.000 : 32 : 0 ;
> track_set = inc ; track_check?
!track_set = 0 ; !track_check? 0 : mark4 : 32 : 2003y189d05h04m05.098s : 41904 : 0.00250s : 8.000 : 33 : 0 ;
```

The track number, as before, is the next-to-last parameter returned by track_check. Rechecking scans and tracks is desirable after any change in mode or in formatter configuration. Here the tracks seem to be OK, and we have not found any problems with the data on this disk pack.

(4) Condition and check the disks in a disk pack

New disks should be conditioned before their first use especially if they are to be used anywhere near their maximum data rates. Conditioning can be done by a three-pass procedure: Write, read, and write, each through the full length of the disk. Or a special procedure inside SSErase can be used to condition up to sixteen disks at a time using a two-pass procedure. This saves lots of time, but it's still slow.

Important: Note that any operation with SSErase, as the name implies, erases any data that may be on the SS disk or disks.

For the following procedure, Mark5A should *not* be running. Shut it down, if need be, using `EndM5`. Mount the disk pack to be erased and conditioned in bank A and turn on the keyswitch. In the following example, we have a two-pack of 120-Gbyte disks. Start SSErase:

```
SSErase -m 0 -c 1 &
```

The optional -m 0 turns on lots of debug printing, the -c 1 turns on conditioning (otherwise it just erases), and the optional & puts SSErase into the background and so allows the same terminal to be used for other things. Then see:

```
SSErase DEBUG:  msglev set to 0
SSErase DEBUG:  cond set to 1
SSErase DEBUG:  XLRDeviceFind() OK
SSErase DEBUG:  Trying to XLROpen() ...
SSErase DEBUG:  XLROpen() OK
SSErase DEBUG:  Bank A is not write protected
SSErase DEBUG:  GetDeviceInfo() OK
SSErase DEBUG:  BoardType PCI-816V100, SerialNum 7140, NumDrives 2,
  NumBuses 2, TotalCapacity 58577740 * 4096 bytes
  MaxBandwidth 0  PciBus 0x1  PciSlot 0x3
SSErase DEBUG:  XLRSetOption() drive stats OK
SSErase DEBUG:  XLRSetDriveStats() OK
SSErase DEBUG:  Erase start OK
SSErase NOTE:  This will run a long time
SSErase DEBUG:  In progress, 117295349760
SSErase DEBUG:  In progress, 114364121088
SSErase DEBUG:  In progress, 111450783744
SSErase DEBUG:  In progress, 108538363904
SSErase DEBUG:  In progress, 105528819712
SSErase DEBUG:  In progress, 102638551040
SSErase DEBUG:  In progress, 99755556864
SSErase DEBUG:  In progress, 96894779392
SSErase DEBUG:  In progress, 94038392832
SSErase DEBUG:  In progress, 91216281600
. . .
SSErase DEBUG:  In progress, 13076398080
```

```
SSErase DEBUG:  In progress, 11179917312
SSErase DEBUG:  In progress, 9380888576
SSErase DEBUG:  In progress, 7606108160
SSErase DEBUG:  In progress, 5839257600
SSErase DEBUG:  In progress, 4182507520
SSErase DEBUG:  In progress, 2527723520
SSErase DEBUG:  In progress, 885260288
SSErase DEBUG:  Erase finished
SSErase DEBUG:  That took 6110.3 seconds
SSErase DEBUG:  XLRSetUserDir() OK
SSErase DISK:  0, WD-WMA8C2588610
SSErase STATS:  0 : 1211339 : 1830784 : 605143 : 3388 : 11908 : 216 : 17 : 11 ;
SSErase DISK:  2, WD-WMA8C2588979
SSErase STATS:  2 : 1219792 : 1825763 : 599746 : 3410 : 14071 : 19 : 3 : 2 ;
SSErase DEBUG:  XLRClose() OK
```

With -m 0, the "In progress" debug prints occur about once a minute so that you'll know that the program hasn't gone comatose. Here we've skipped most of these prints. Conditioning these two disks took 102 minutes. The eight "STATS" numbers are stats (see get_stats) for these two disks and are within the normal range although the first disk is not as good as the second as indicated by larger numbers in the right-most (longest-time) three bins. The stats after conditioning usually look worse than the stats for the same disks after normal recording.

If, after conditioning a set of disks, you break them up into a different configuration, then an additional erase step is required. In particular, if you condition two eight-packs at a time (sixteen disks), then do an additional SSErase (without the -c 1 conditioning) on each pack in bank A separately. A similar SSErase is needed for each part of an eight-pack that is split up into individual disks or two-packs. This extra SSErase (without -c 1 conditioning) will take only a few seconds in part because SSErase has no reset at its end. When you're all done with SSErase, then we recommend an SSReset.

Appendix A: Summary of return from 'status?' query

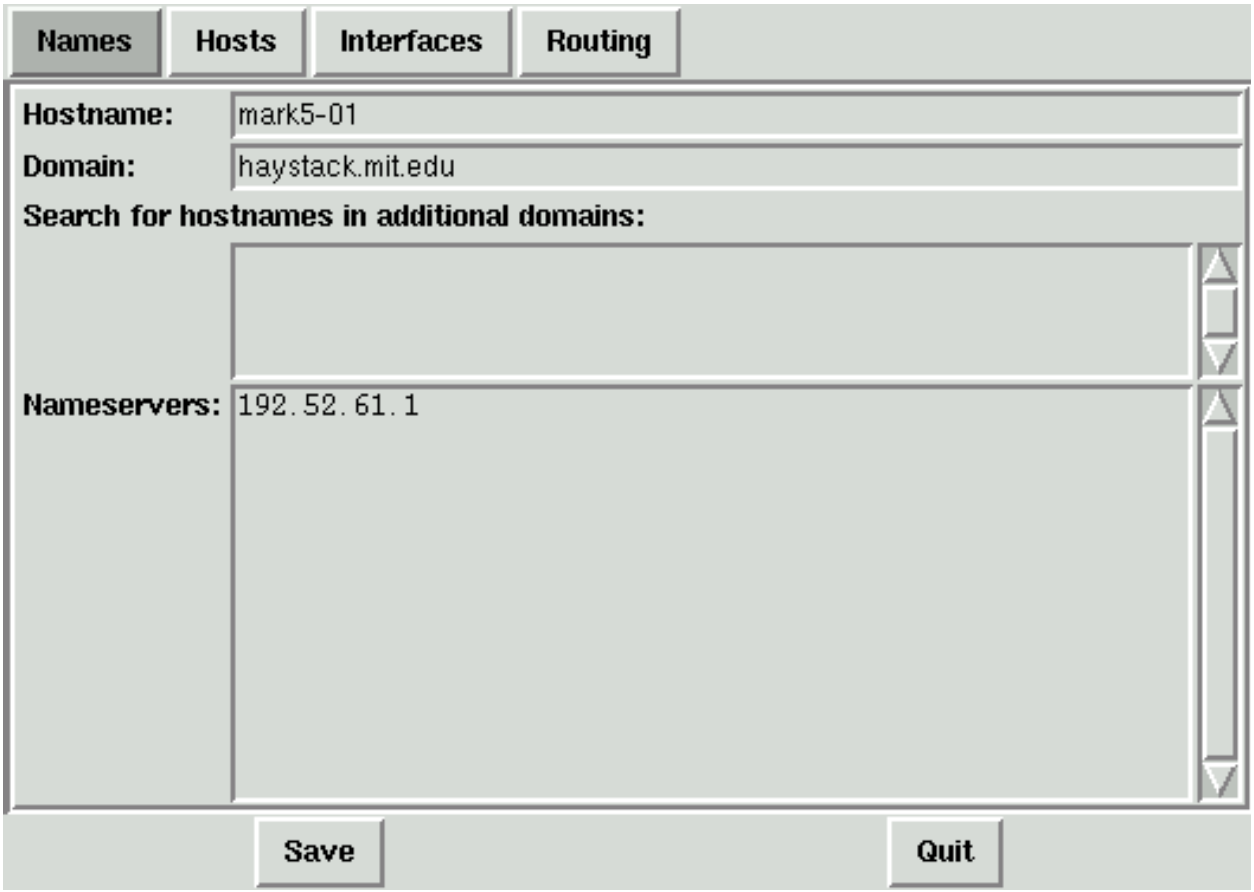
0x00000001	Ready
0x00000002	Error (message may be appended)
0x00000004	(Not used)
0x00000008	Delayed completion action in progress
0x00000010	Delayed completion request in progress
0x00000020	(Not used)
0x00000040	Recording in progress
0x00000080	Media full (recording halted)
0x00000100	Playing in progress
0x00000200	End of scan (scan_play) or end of data (playing halted)
0x00000400	Recording throttled (can't keep up, some data loss)
0x00000800	(Not used)
0x00001000	Disk2file active
0x00002000	File2disk active
0x00004000	Disk2net active
0x00008000	Net2disk active or waiting
0x00010000	In2net sending (on)
0x00020000	Net2out active or waiting
0x00040000	(Not used)
0x00080000	(Not used)
Following are set only if in bank mode and no data transfers are in progress:	
0x00100000	Bank A selected
0x00200000	Bank A ready
0x00400000	Bank A media full
0x00800000	Bank A write protected
0x01000000	Bank B selected
0x02000000	Bank B ready
0x04000000	Bank B media full
0x08000000	Bank B write protected

Revised: 2003 September 14, JAB

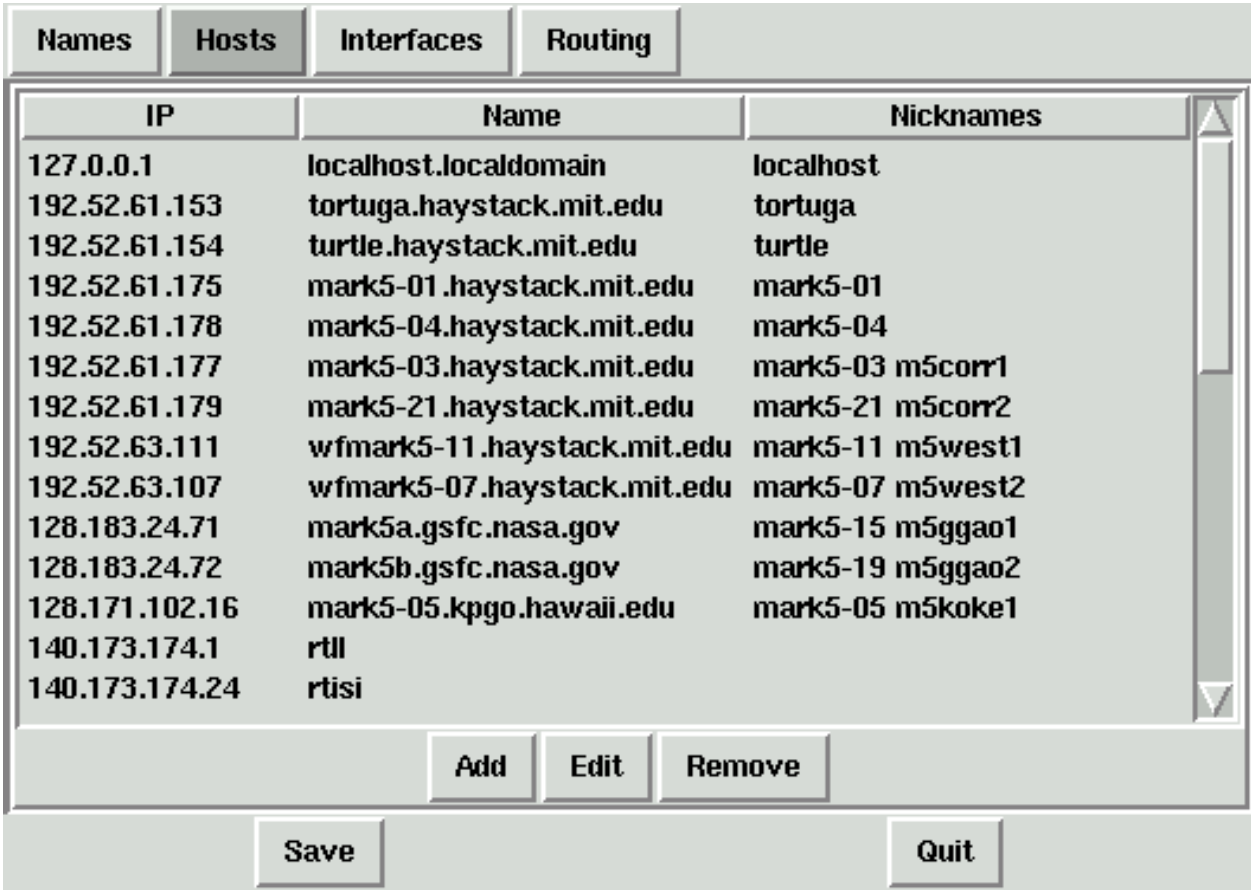
RedHat Linux Network Configuration

To do the network configuration of your RedHat Linux Mark-5 computer, you'll need to have the following information, perhaps from your system administrator (sysadmin): official hostname, (local) domain name, internet protocol (IP) address of this computer, the netmask for this network, the IP address of at least one domain name server (DNS), and the IP address of a gateway.

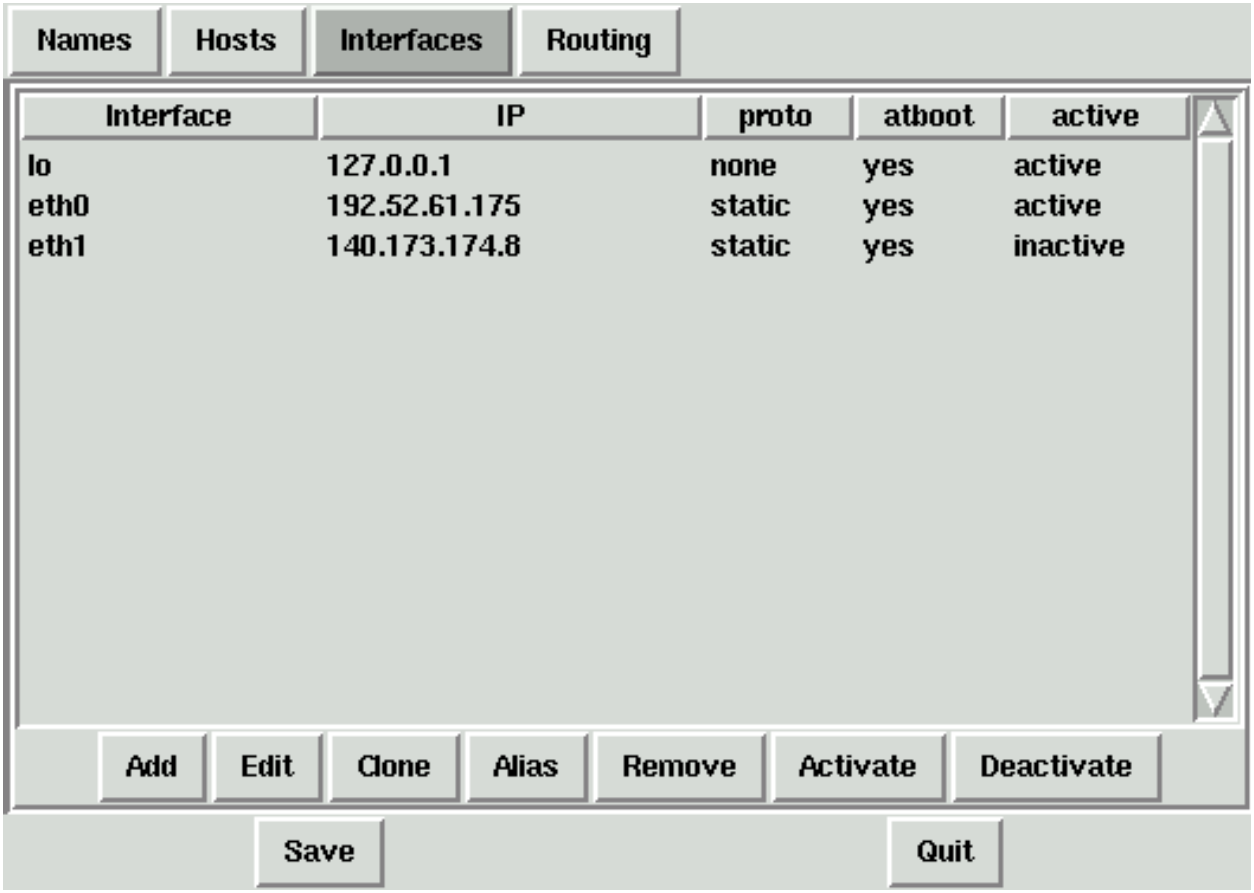
You'll need to be in X mode (try `startx`), then bring up the network-configuration box by typing `netcfg` to a Linux shell prompt. If you're not already logged in as root, then you'll be asked for the root password.



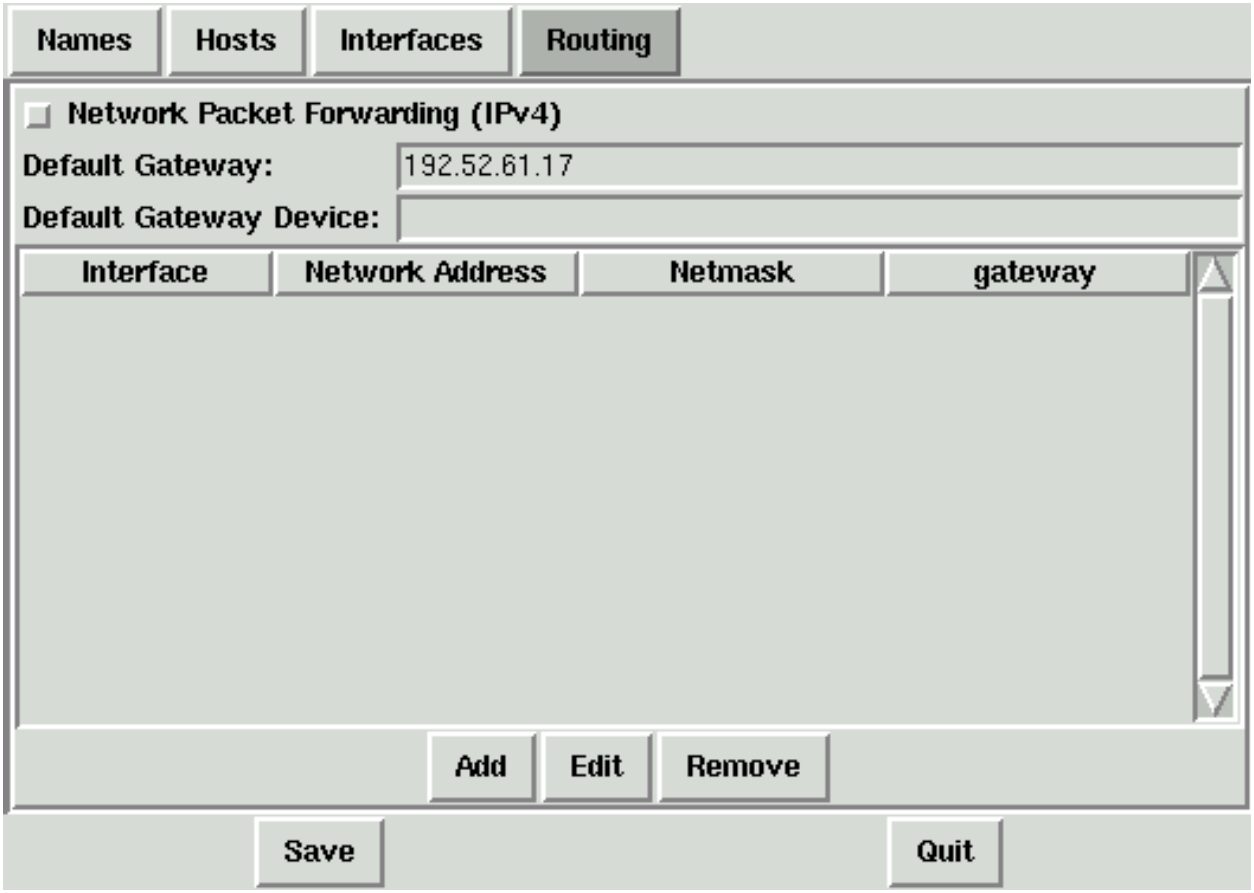
Show this view by clicking on the 'Names' tab. 'Hostname' is the official name of this computer as listed in your domain name server (DNS), 'Domain' is the official domain name also as in your DNS. In the 'Nameservers' box, you need to put the IP address of at least one local DNS, which will be used to look up IP numbers of other computers. Add or correct this information by clicking on the appropriate box and typing or backspacing and retyping. Do not 'Save' or 'Quit' until you've done the other three tabs.



Show this view by clicking on the 'Hosts' tab. Two lines are required in this box; one should already be there, namely the first line with localhost. You'll need to 'Add' or click on and then 'Edit' the line that has the official name of your computer, mark5-01 in this example. You'll need to type the official IP number, the official name with domain, and any nicknames that you want, typically the name without the domain as in this example. If there are other lines with other computers, as in this example, you can either 'Remove' them, if they are incorrect or unneeded, or otherwise leave them.



Show this view by clicking on the 'Interfaces' tab. Two lines are required in this box; one should already be there, namely the first line with 'lo' and '127.0.0.1'. You'll need to 'Add' or click on and 'Edit' a line with 'eth0'. You'll need the official IP address of this computer, '192.52.61.175' in this example, and the netmask, '255.255.255.0' for a class-C network in this example. 'Activate interface at boot time' should be pushed (red), and '...protocol' should be 'static'. You may need to click 'Activate'. Your computer may not have an 'eth1'.



Show this view by clicking on the 'Routing' tab. You'll need to 'Edit' the 'Default Gateway' to the IP number of your gateway to the rest of the world. Nothing more is needed.

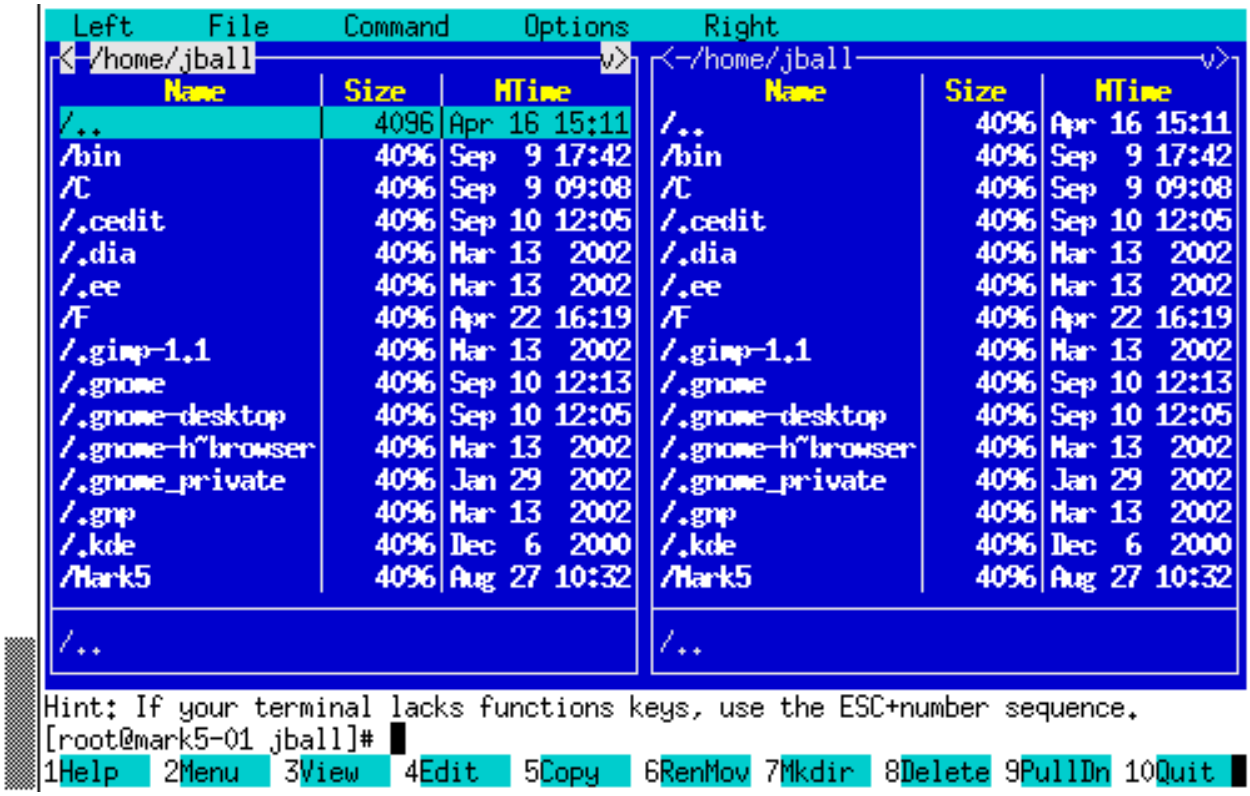
Go back and review all the names and numbers under all four tabs before clicking 'Save' and then 'Quit'.

Revised: 2003 September 14, JAB

Additional Mark-5 Linux Configurations

These notes are intended to help with the several topics that need to be done when you receive a new Mark-5 machine from either Haystack or Conduant. Network configuration is covered in a separate document, "RedHat Linux Network Configuration."

Some of these operations involve editing a text file. If you are not already familiar with either of the standard Linux editors, vi or emacs, then you might want to look into Midnight Commander (mc), which is a Linux imitation of the old Norton Commander from DOS. mc is run mostly with function keys and arrow keys, in text mode, so it can be used even without X, and it contains a simple and easy-to-use text editor. Try it first just to look around at what's on the disk.



Note that most of the following operations must be done as root.

Time Zone

The local time zone is determined by the file /etc/sysconfig/clock, which will contain a line such as

```
ZONE="America/New_York"
```

You may change this to any of the similar arguments found in /usr/share/zoneinfo and its many subdirectories. Changes in this file take effect on the next boot.

Update Time from NTP

Look at the file /etc/rc.d/rc.local. Near the end, you should find a line such as

```
/usr/bin/ntpdate -b -p 8 -u gauss
```

or

```
/usr/jball/bin/ntpdate -b -p 8 -u gauss
```

where *gauss* is the name or IP address of a Network Time Protocol (NTP) server machine. Use your editor to change *gauss* to the name or IP address of a nearby and accessible NTP server. See http://www.eecis.udel.edu/~mills/ntp/ if in doubt. Check that you can reliably ping the chosen NTP server. This rc.local file is executed on boot.

Then look in the directory /etc/cron.daily for a file named ntpdate or timeupdate. If you have such a file (the name doesn't matter), and it contains an ntpdate line such as above, then edit it to replace *gauss* just as you did above. If such a file does not exist, then make a file ntpdate and type a line:

```
#!/bin/bash
```

(with that # in column 1) followed by an ntpdate line as above. Save this file and change the permissions on this new file:

```
chmod 0755 ntpdate
```

and make sure that it is owned by root. It will be executed once a day.

Alternatives to this scheme are described in "Mark 5 Linux Configuration Instructions."

Revised: 2003 September 14, JAB

Mark-5 Auxiliary Programs

Several sometimes-useful auxiliary programs accompany the Mark-5 system. Following are a brief descriptions of these. Two programs from Conduant:

```
ssopen
```

which opens and does a quick test of the SS card to verify readability, and

```
sstest
```

which *erases* and *writes* a scan of 33554432 bytes of SS pattern onto the SS disks. *Important*: Do *not* use sstest on disks that have data to be saved.

The following programs are Mark-5 specific.

```
tstMark5A [ machine ]
```

where *machine* defaults to localhost. Mark5A should already be running on *machine*, but `tstMark5A` can run on some other machine including most Linux and HP-UX machines and probably others. This program accepts commands and queries as defined in the Mark5A software documentation, sends them to Mark5A on *machine*, and prints the responses.

```
SSErase [ -m m ] [ -c c ] [ -h ]
```

where *m* (msglev) is the debug message level, range -1 to 3, default 1, *c* sets conditioning (0 for FALSE, 1 TRUE, default FALSE), and `-h` (alone) prints a help message. Mark5A must *not* be running. *This program erases all data on SS disks.* SSErase is useful to test the write-ability of disks and to prepare disks to be recorded. If a disk pack is write protected, then SSErase asks permission to remove this protection as is necessary to erase.

If *c* is 1 (TRUE), then SSErase also goes through the long conditioning process on whatever disks it can access, up to 16 at a time in both banks. If *m* is set to 0, then debug prints about once a minute during conditioning to show what's happening.

Conditioning disks is recommended before recording especially if they are to be recorded at or near their maximum data rate. Conditioning amounts to a write-read-write cycle through the whole set of disks, but SSErase does this in two rather than three passes. Printed with debuggery and counting down twice is the number of bytes per bus. With an 8-disk pack, for example, this count starts at a quarter of the total capacity, which is twice the capacity of a single disk.

Conditioning one 120-Gbyte disk takes about 101 minutes; two 120-Gbyte disks (as two masters), 103 minutes; four 120-Gbyte disks (as four masters), 111 minutes; eight 120-Gbyte disks (an eight pack), 157 minutes; and sixteen 120-Gbyte disks (two eight packs), 278 minutes. 200-Gbyte disks take about $^{5}/_{3}^{rds}$ as long—no surprise. Conditioning one 200-Gbyte disk takes about 160 minutes; eight 200-Gbyte disks, 286 minutes; and sixteen 200-Gbyte disks, 465 minutes. These times are approximate and for the ideal case, but, if any of the disks has a problem, then times can be much longer.

After conditioning two disk packs at a time (i.e., more than eight disks), or in any case where you change the disks into a different configuration, then you should also SSErase (without -c 1) each disk pack separately before recording. This takes only a few extra seconds.

`SSReset`

This program performs an XLRCardReset(), which often helps extricate the system from a no-fair state. Use *before starting* `Mark5A`.

`DirList [ -m` ***`m`*** `] [ -f` ***`filename`*** `] [ -h ]`

where ***`m`*** (msglev) is the debug message level, range -1 to 3, default 1, ***`filename`*** sets the name of the Mark-5 directory file, default /var/dir/Mark5A, and -h (alone) prints a help message. DirList reads the Mark-5 directory and lists the contents including the starting and ending byte numbers of all completed scans. DirList can run simultaneously with Mark5A, in which case the listing will be up to date except for any scan being recorded at the time. If Mark5A is dismounted (i.e., no disk pack active), then the listing will be appropriate for the SS disk(s) that were in place before the dismount. Or if Mark5A is not running, then the listing will be appropriate for whatever was happening last time Mark5A was running. This is equivalent to saying that DirList does not read SS disks, instead it reads whatever Mark5A has written into /var/dir/Mark5A or ***`filename`***.

Following are notes about two stand-alone programs intended to exchange data with Mark-5 machines and primarily intended to run on computers other than Mark-5 machines. The source code contains hints for compiling on Linux and HP-UX.

`Net2file [` ***`filename`*** `]`

accepts a connection from a Mark-5 machine and writes the received data to ***`filename`*** or to save.data if ***`filename`*** is blank. This file will be created if necessary or appended. Start Net2file *first*, then command disc2net or in2net in the Mark-5 machine. Monitor progress with

`ls -l` ***`filename`***

but this will lag the actual progress because of buffering. Net2file will end when the Mark-5 machine disconnects or with <Ctrl>C, after which ***`filename`*** will be ready to use.

`File2net` ***`machine`*** `[` ***`filename`*** `[` ***`startbyte`*** `[` ***`endbyte`*** `] ] ]`

sends a file or part of a file to a Mark-5 machine. Command net2disc or net2out in the Mark-5 machine *first*, then start File2net. ***`Filename`*** defaults to save.data, ***`startbyte`*** defaults to `0`, and ***`endbyte`*** defaults to the end of the file. File2net ends when the prescribed transfer is done.

Revised: 2003 September 14, JAB

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

HAYSTACK OBSERVATORY

WESTFORD, MASSACHUSETTS 01886

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

4 September 2003

TO: Distribution
FROM: A. R. Whitney and J. A. Ball
SUBJECT: Mark 5A command set (Revision 2.6)

1. Mark5A program

The commands detailed in this memo are implemented by a program named `Mark5A` running under Red Hat Linux on the Mark 5P or Mark 5A system. The details concerning the operation of `Mark5A` are available in documents at http://web.haystack.mit.edu/mark5/Mark5.htm (see 'Mark 5P test procedures' for instructions for using Mark 5A in a simple interactive mode; see 'Mark 5A control program and utilities' for much more detail).

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

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

where

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

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

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

h – help on startup parameters

When run in a TK200 chassis (with '8-pack' disk modules), Mark5A will operate only in so-called 'bank mode'. When run in an obsolete Anova chassis, Mark5A will operate only in the old Mark5P 'non-bank mode'.

2. Notes on command set

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

1. All commands/queries are implemented using the VSI-S communications protocol and command/response syntax.
2. Commands/queries are case *insensitive*.

3. Versions of program 'Mark5A' with a revision date earlier than the date on this memo may not implement all commands indicated in this memo or, in some cases, may implement them in a different way (use 'DTS_id' query to get revision date of current system software – see 'System Queries and Responses').

3. VSI-S Command, Query and Response Syntax

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

3.1 *Command Syntax*

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

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

where <keyword> is a VSI-S command keyword. The number of fields may either be fixed or indefinite; fields are separated by colons and terminated with a semi-colon. A field may be of type decimal integer, decimal real, integer hex, character, literal ASCII or a special 'time' code (see Section 7.2). White space between tokens in the command line is ignored, however most character fields disallow embedded white space. VSI-S keywords are listed in Section 9.

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 playback 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

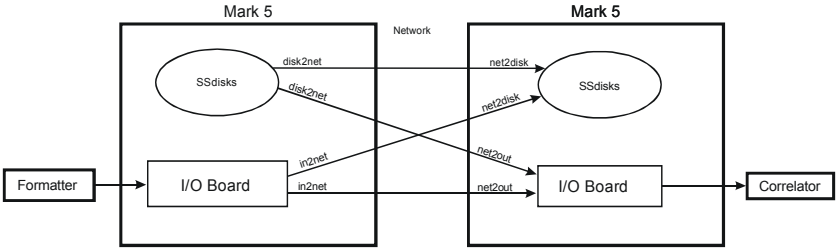


Figure 1: Mark 5 to Mark 5 transfer through network

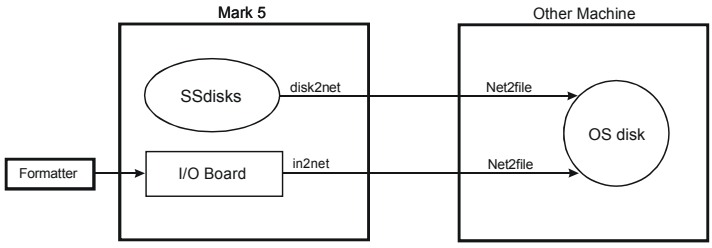


Figure 2: Mark 5 to file transfer through network

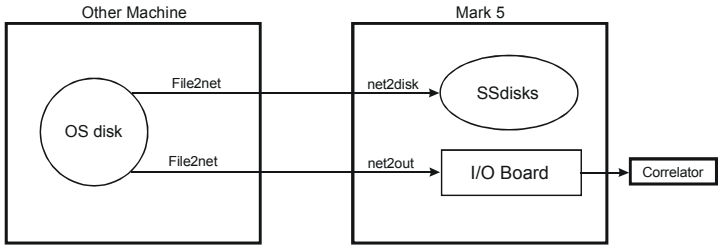


Figure 3: File to Mark 5 transfer through network

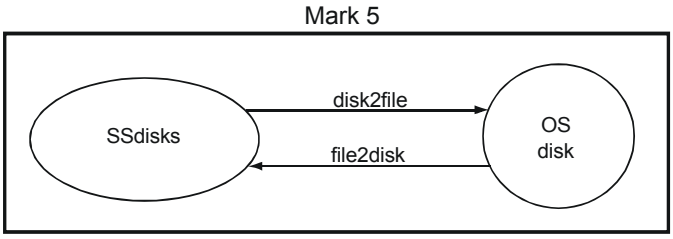


Figure 4: Internal Mark 5 to file transfer

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

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

5.1 Record Pointer

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

The record pointer can be modified in the following ways:

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

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

5.2 Play Pointer

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

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

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

5.3 Scan Pointer

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

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

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

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

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

5.3 Directory management

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

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

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

6.1 General

DTS_id	p. 21	Get system information (query only)
error	p. 22	Get error number/message (query only)
OS_rev1	p. 30	Get details of operating system (query only)
OS_rev2	p. 31	Get more details of operating system (query only)
reset	p. 41	Reset Mark 5 unit (command only)
SS_rev1	p. 49	Get StreamStor firmware/software revision levels, part 1 (query only)
SS_rev2	p. 50	Get StreamStor firmware/software revision levels, part 2 (query only)
status	p. 52	Get system status (query only)
task_ID	p. 53	Set task ID (primarily for correlator use)

6.2 Record/Play

mode	p. 26	Set data recording/playback mode
play	p. 32	Play data from current play pointer position
play_rate	p. 34	Set playback data rate; set tvg rate
position	p. 36	Get current record and play pointers (query only)
record	p. 38	Record data from Mark 5 input to disks
rtime	p. 42	Get remaining record time on current disk set (query only)
scan_play	p. 45	Play scan specified by current value of scan_set parameters
scan_set	p. 46	Set scan playback parameters for scan_play, disk2file and disk2net commands.
skip	p. 48	Skip forward|backward specified # of bytes while playing

6.3 Data Checking

data_check	p. 12	Check data starting at position of current play pointer (query only)
scan_check	p. 43	Get scan parameters (query only)
track_check	p. 54	Check data on selected track (query only)
track_set	p. 56	Select tracks for monitoring with DQA or ‘track_check’

6.4 Data Transfer

disk2file	p. 18	Transfer data from Mark 5 to file
disk2net	p. 19	Transfer data from Mark 5 to network
file2disk	p. 23	Transfer data from file to Mark 5
in2net	p. 25	Transfer data directly from Mark 5 input to network
net2disk	p. 28	Transfer data from network to Mark 5
net2out	p. 29	Transfer data directly from network to Mark 5 output

6.5 Bank Management

bank_set	p. 10	Select active bank for recording or playback
bank_switch	p. 11	Enable|disable automatic bank-switching (not yet implemented)

6.6 Data Directory

dir_info	p. 14	Get directory information (query only)

6.7 Disk Info

disk_model	p. 15	Get disk model numbers (query only)
disk_serial	p. 16	Get disk serial numbers (query only)
disk_size	p. 17	Get disk sizes (query only)
get_stats	p. 24	Get disk-performance statistics (query only)
replaced_blks	p. 40	Get number of replaced blocks during playback
start_stats	p. 51	Start gathering disk-performance statistics.
VSN	p. 57	Write module VSN to permanent area

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

8. Mark 5A Command Set Details

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

bank_set – Select active bank for recording or playback

[command list]

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

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

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

Settable parameters:

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

Monitor-only parameters:

Parameter	Type	Values	Comments
<active bank>	char	A | B	Currently active module

Notes:

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

bank_set

bank_switch – Enable/disable automatic bank-switching (Not Yet Implemented)

[command list]

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

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

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

Settable parameters:

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

Notes:

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

bank_switch

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

[command list]

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

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

Monitor-only parameters:

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

Notes:

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

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

data_check

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

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

data_check

dir_info – Get directory information (query only)

[command list]

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

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

Monitor-only parameters:

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

Notes:
1. The scan directory is automatically appended each time data are recorded to the disks (added after end of recording; length 81952 bytes).

dir_info

disk_model – Get disk model numbers (query only)

[command list]

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

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

Monitor-only parameters:

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

disk_model

disk_serial – Get disk serial numbers (query only)

[command list]

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

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

Monitor-only parameters:

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

disk_serial

disk_size – Get disk sizes (query only)

[command list]

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

Purpose: Returns a list capacities of currently mounted disks.

Monitor-only parameters:

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

disk_size

disk2file – Transfer data from Mark 5 to file

[command list]

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

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

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

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<dest filename>	literal ASCII	no spaces allowed	See Comments	Default is name of scan in directory. Filename must include path if path is not default.
<start byte#>	int | null		See Comments	Absolute byte#; if null, defaults to <start_play> position as set and/or reported by scan_set
<end byte#>	int | null		See Comments>	Absolute end byte#; if preceded by '+', increment from <start byte#> by specified value; if null, defaults to <end play> position as set and/or reported by scan_set.
<option>	char	w | a	a	w – create file if necessary, or erase (truncate at 0) existing file. a – create file if necessary, or append to existing file (cf., fopen()).

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	active | inactive	Current status of transfer
<start byte#>	int		
<current byte#>	int		Current byte number being transferred
<end byte#>	int		
<option>	char		

disk2file

disk2net – Transfer data from Mark 5 to network

[command list]

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

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

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

Settable parameters:

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

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	waiting | active | inactive	Current status of transfer
<start byte#>	int		
<current byte#>	int		Current byte number being transferred
<end byte#>	int		

Notes:

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

disk2net

5. Only one data transfer activity may be active at any given time. That is, among ‘record=on’, ‘play=on’, ‘in2net=..’, ‘disk2net=..’, ‘net2disk=..’, ‘net2out=..’, ‘disk2file’=..’, ‘file2disk=..’, ‘data_check’ and ‘track_check’ ‘scan_check’, only one may be active at any given time.

disk2net

DTS_id – Get system information (query only)

[command list]

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

Purpose: Get Mark 5 system information

Monitor-only parameters:

Parameter	Type	Values	Comments
<system type>	char	mark5P | mark5A	
<software revision date>	time		Date stamp on current version of Mark 5 software
<media type>	int	1	Per VSI-S spec: 1 – magnetic disk [0 – magnetic tape; 2 – real-time (non-recording)]
<serial number>	ASCII		System serial number; generally is in the form 'mark5-xx' where xx is the system serial number
<#DIM ports>	int	1	Number of DIM ports in this DTS
<#DOM ports>	int	1	Number of DOM ports in this DTS
<command set revision>	char		Mark 5 command set revision level corresponding to this software release (e.g. '2.3a')
<Input design revision>	int		Revision level of Input section of Mark 5A I/O board
<Output design revision>	int		Revision level of Output section of Mark 5A I/O board

DTS_id

error – Get error number/message (query only)

[command list]

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

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

Monitor-only parameters:

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

Notes:

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

error

file2disk – Transfer data from file to Mark 5

[command list]

Command syntax: file2disk = <source filename> : <start byte#> : <end byte#> : <scan name>;
Command response: !file2disk = <return code>;

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

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

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<source filename>	literal ASCII	no spaces allowed	save.data	Filename must include path if not default.
<start byte#>	int		0	Absolute byte number; if unspecified, assumed to be zero
<end byte#>	int		0	If =0, will copy to end of file
<scan name>	literal ASCII	max 16 chars; no spaces allowed	<source filename>	Scan name to be saved to scan directory

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	active | inactive	Current status of transfer
<start byte#>	int		
<current byte#>	int		Current source byte# being transferred
<end byte#>	int		
<scan name>	literal ASCII		

file2disk

get_stats – Get disk performance statistics (query only)

[command list]

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

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

Monitor-only parameters:

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

Notes:

1. Each subsequent 'get_stats' query returns current performance statistics for the next mounted drive; recycles through mounted drives. Bin counts are not cleared. See details in Notes on 'start_stats' command.
2. The 'get_stats' query may not be issued during active recording or playback.
3. Drive statistics and replaced-block counts are cleared and re-started whenever a new disk module is mounted or a 'start_stats' command is issued.

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> : <#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 'on' – start data transfer 'off' – end data transfer 'disconnect' – disconnect socket See Notes with 'disk2net'
<remote hostname>	char			Required only first 'connect'; optional thereafter

Monitor-only parameters:

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

Notes:

1. Important: Due to current software problem, a scratch disk is required in Bank A for in2net operation; will be fixed in future update.
2. See Notes with 'disk2net' command for usage rules and restrictions.
3. If the data rate is too fast for the network to handle, the FIFO will eventually overflow; this will be reported by either a 'status?' query or an 'in2net?' query with an error message.
4. After 'in2net=off', but before 'in2net=disconnect', <#bytes received> shows the approximate total #bytes transferred from the input source; the #bytes currently sent through 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.

in2net

mode – Set data recording/playback mode

[command list]

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

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

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

Settable parameters:

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

Monitor-only parameters:

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

Notes:

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

mode

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

mode:submode	Recorded formatter track#'s	FPDP bit streams
'mark4:8' or 'vlba:8' - 8 tks	2-17 even (headstack 1)	Trk 2 to FPDP streams 0,8,16,24; trk 4 to 1,9,17,25; etc.
'mark4:16' or 'vlba:16' - 16 tks	2-33 even (headstack 1)	Trk 2 to FPDP streams 0,16; trk 4 to 1,17; etc.
'mark4:32' or 'vlba:32' - 32 tks	2-33 all (headstack 1)	Correspond to FPDP bit streams 0-31, respectively
'mark4:64' or 'vlba:64' - 64 tks	2-33 (headstacks 1 and 2)	Trks 2 from both hdstks mux'ed to FPDP bit stream 0, etc.
'st' (any submode)	2-33 all (headstack 1)	Correspond to FPDP bit streams 0-31, respectively; only mode for Mark 5P

8. In all modes except 'tvg' mode, the data clock is provided by the external data source. In 'tvg' mode, the clock-rate is set by 'play_rate' command.
9. The 'sync status' parameter is relevant only in output mode 'mark4' or 'vlba' where parity must be restored. If 'sync'ed', the I/O board has properly synchronized to the data frames and is properly de-multiplexing and restoring parity.
10. The '# of sync attempts' returned value in the 'mode=' command counts the number of sync attempts the Mark 5A I/O board output section had to make before parity-stripped data ('mark4' or 'vlba') was re-sync'ed, as necessary for parity re-insertion. A large number indicates a problem, perhaps in the output clock or the data itself. The counter is reset to zero on a subsequent 'mode=' command.
11. If in 'tvg' mode, TVG is operated at clock-rate set by 'play_rate' command.

mode

net2disk – Transfer data from network to disks

[command list]

Command syntax: net2disk = <control> : <scan name>;
Command response: !net2disk = <return code> ;

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

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 name>	literal ASCII			Optional scan name to be assigned to this data; defaults to previously specified name or, if none, to 'net2disk'

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	active | inactive | waiting	Current status of transfer
<scan name>	char		Returned only if <status> is 'active' or 'waiting'

Notes:
1. See Notes with 'disk2net' command for usage rules and restrictions.

net2disk

net2out – Transfer data directly from network to Mark 5 output

[command list]

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

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

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

Settable parameters:

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

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	active | inactive | waiting	Current status of transfer

Notes:

1. See Notes with 'disk2net' command for usage rules and restrictions.

net2out

OS_rev1 – Get details of operating system (query only)

[command list]

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

Purpose: Get detailed information about operating system.

Monitor-only parameters:

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

OS_rev1

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

[command list]

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

Purpose: Get more detailed information about operating system.

Monitor-only parameters:

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

OS_rev2

play – Play data from from current play pointer position

[command list]

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

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

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

Settable parameters:

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

Monitor-only parameters:

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

Notes:

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

play

without a preceding 'play=arm' will begin play, but after an indeterminate delay. Any change in play pointer position after a 'play=arm' will invalidate the arming and cause the 'play?' status to be returned as 'off'.

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

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

5. Note that record/play pointers may have values as large as ~2×10^{13} (~44 bits), so pointer arithmetic must be handled appropriately.
6. Playback clock rate is set by the 'play_rate' command
7. When playing, the playback pointer will update to show the approximate position. If the playback pointer is noted not to be incrementing, an error flag is set in the 'status?' query which can be used as a first order check of proper playback.

play_rate – Set playback data rate; set tvg rate

[command list]

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

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

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

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<play rate reference>	char	data | clock | clockgen | ext	clock	'data' – set output track data rate (not including parity) to specified value. 'clock' – set output track clock rate (including parity) to specified value. 'clockgen' – set clock generator chip to specified frequency; max is 40 MHz.. 'ext' – external clock select See also Notes below.
<rate>	real	MHz	9	>0 – set rate to specified value; freq resolution of clock generator chip is ~20 mHz. If in 'tvg' mode, sets on-board clock generator rate regardless of value of <play rate reference>; see Notes.

Monitor-only parameters:

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

Notes:

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

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

play_rate

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

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

play_rate

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

[command list]

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

Purpose: Get current value of record and play pointers.

Monitor-only parameters:

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

Notes:

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

position

protect – Set write-protection on/off for active module

[command list]

Command syntax: protect = <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
<protect on/off>	char	on | off	-	

Notes:
1. When write-protection is ‘on’, ‘record=on’, ‘net2disk=open’ and ‘file2disk=…’ commands will be rejected.
2. Command may be issued only when not recording or playing.

protect

record – Record data from Mark 5 input to disks

[command list]

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

Query syntax: record? ;
Query response: !record ? <return code> : <status>: <scan name> : [<source>] : [<experiment name>] ;

Purpose: Turn recording on|off; assign scan name and experiment name.

Settable parameters:

Parameter	Field	Type	Allowed values	Default	Comments
<record on/off>	1	char	on | off		'on' automatically appends to the end of the existing recording; 'bypass' is active while recording is 'on'. 'off' stops recording and leaves system in 'bypass' mode.
<scan name>	2	ASCII	max 16 chars (no spaces)		Optional: relevant only if record is 'on'. '+' character not allowed in scan name. No checking is done for duplicate scan names.
<source>	4	ASCII	max 16 chars (no spaces)		Optional: relevant only if record is 'on'. Recorded permanently into directory.
<experiment name>	3	ASCII	max 16 chars (no spaces)		Optional: relevant only if record is 'on'. Recorded permanently into directory.

Monitor-only parameters:

Parameter	Type	Values	Comments
<status>	char	on | off | halted	'halted' indicates end-of-media was encountered while recording.

Notes:

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

mode:submode	Recorded track#'s	FPDP bit streams
'mark4:8' or 'vlba:8' - 8 tks	2-16 even (headstack 1)	Trk 2 to FPDP streams 0,8,16,24; trk 4 to 1,9,17,25; etc.
'mark4:16' or 'vlba:16' - 16 tks	2-33 even (headstack 1)	Trk 2 to FPDP streams 0,16; trk 4 to 1,17; etc.
'mark4:32' or 'vlba:32' - 32 tks	2-33 all (headstack 1)	Correspond to FPDP bit streams 0-31, respectively
'mark4:64' or 'vlba:64' - 64 tks	2-33 (headstacks 1 and 2)	Trks 2 from both hdstks mux'ed to FPDP bit stream 0, etc.
'st' (any submode) – 32 tks	2-33 all (headstack 1)	Correspond to FPDP bit streams 0-31, respectively; only mode for Mark 5P
tvg	equivalent to tracks 2-33	TVG data; correspond to FPDP bit streams 0-31, respectively,

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

record

3. After record is turned ‘on’, the user should periodically query ‘status’ for details; if recording stops on its own accord (due to end-of-media, etc.), this will be reflected in the response to the ‘status’ query as ‘recording stopped’, and a ‘record’ query will show the status as ‘halted’; a subsequent command to turn record ‘off’ or ‘on’ will reset the relevant bits (5-4) in the ‘status’ response.
4. When playing, the playback pointer will update to show the approximate position. If the playback pointer is noted not to be incrementing, an error flag is set in the ‘status?’ query which can be used as a first order check of proper playback.

record

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

[command list]

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

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

Monitor-only parameters:

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

Notes:

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

replaced_blks

reset – Reset Mark 5 unit (command only)

[command list]

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

Purpose: Reset system; mount/dismount disks

Settable parameters:

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

Notes:

1. The former 'reset=mount' and 'reset=dismount' commands are no longer supported; the keyswitches associated with the disk modules are used for all mount and dismount operations.
2. 'reset=abort' returns immediately, but there may be a delay of up to two seconds before the data transfer stops. During this delay, a 'status?' query will show what is happening.

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> : <mode> : <submode> : <track data rate> : <total recording rate> ;

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

Monitor-only parameters:

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

Notes:

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

rtime

scan_check – Get scan parameters (query only)

[command list]

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

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

Monitor-only parameters:

Parameter	Type	Values	Comments
<scan number>	int		Start at 1 for first recorded scan
<scan name>	literal ASCII		
<data mode>	char	st | mark4 | vlba | tvg | SS	See 'mode' command for explanation of data modes; 'tvg' corresponds to VSI test pattern;' SS' corresponds to StreamStor test pattern
<data submode>	int	8 | 16 | 32 | 64 | mark4 | vlba	'8|16|32|64' if <data mode> is 'mark4' or 'vlba'; 'mark4|vlba' if <data mode> is 'st'
<start time>	time		Time tag at first frame header in scan. See Note 5 below.
<scan length>	time		Scan length in seconds
<track data rate>	real	Mbps	Excludes parity bits; will always be 0.125, 0.25, 0.5, 1, 2, 4, 8 or 16 (Mbps)
<#missing bytes>	int	See Note 6	Should always be =0 for normally recorded data. >0 indicates #bytes that have been dropped somewhere within scan <0 indicates #bytes that have been added somewhere within scan

Notes:

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

scan_check

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

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

scan_check

scan_play – Play scan specified by current value of scan_set parameters

[command list]

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

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

Purpose: Play scan specified by current value of scan_set parameters

Settable parameters:

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

Monitor-only parameters:

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

Notes:

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

scan_play

scan_set – Set scan playback parameters

[command list]

Command syntax: scan_set = <scan name|scan number> : [<start play>] : [<end play>] ;
Command response: !scan_set = <return code> ;

Query syntax: scan_set? ;
Query response: !scan_set? <return code> : <scan number> : <scan name> : <start play byte#> : <end play byte#> ;

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

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<scan name|number>	int or ASCII	scan number | scan name | 'inc'	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 name, case insensitive (see Note 1). If 'inc', increments to next scan; cycles back to first scan at end. If null field, defaults to last fully recorded scan (e.g. if recording is in progress, defaults to previous scan).
<start play>	char | time | int	s | c | e | s+ | <time> | +<time> | -<time> | +<bytes> | -<bytes>	s	s|c|e|s+: Set start_play position to 'start', 'center', 'end' (actually ~1MB before end) of scan, or specified <time> within scan; this is convenient if you want to do a subsequent 'data_check' or 'track_check' at a prescribed position. 's+' sets play pointer to 65536 bytes past the start of the scan. <time>: time within scan: see Notes 2 & 3 +<time>: offset time from beginning of scan (i.e. '+30s' will start 30 seconds from beginning of scan) -<time>: offset time from end of scan (i.e. '-30s' will start 30 seconds before end of scan) +<bytes>: offset number of bytes from beginning of scan. -<bytes>: offset number of bytes from end of scan
<end play>	time | int	<time> | +<time> | -<time> | +<bytes> | -<bytes>	end-of scan	<time>: Time at which to end playback; see Notes 2 & 3. If preceded by '+', indicates duration of data (in record-clock time) from <start_play> time. +<time>: offset time from <start play> position. -<time>: offset time from end-of-scan +<bytes>: offset bytes from <start play> position -<bytes>: offset bytes from end of scan

Monitor-only parameters:

Parameter	Type	Values	Comments
<scan number>	int		Returns current 'scan pointer', which relates only to the 'scan_play' command and to the 'scan_check" and 'scan_dir' queries. Followed by '+' if scan automatically switched to another disk module (not yet implemented). Prefaced by '+' if scan is continuation from another disk module (not yet implemented).
<scan name>	ASCII		Scan name assigned in 'record=on' command, if any.
<start play byte#>	int	bytes	Absolute byte position to start playback.
<end play byte#>	int	bytes	Absolute byte position to stop playback.

scan_set

Notes:

1. The ‘scan_set’ command can use ‘abbreviated’ scan names: If the first parameter is all numeric, scan_set will first try to interpret this as a scan number. If it is not numeric or the scan number does not exist, scan_set will find the first scan whose scan name, experiment name or source name contains the string in the first parameter (case insensitive).
2. When ‘record=off’ is issued or end-of-media (following a ‘record=on’) is encountered, the default scan playback parameters are set to playback the entire just-recorded scan.
3. If the <start play> or <end play> parameter is a <time> value, the specified time must be specified with sufficient significance to resolve any ambiguity. For example, ‘30s’ would set the play pointer to start at the first ‘30s’ mark in the scan (regardless of the value of the minute); similarly, ‘12m30s’ would have a one-hour ambiguity. If the specified times are outside the bounds of the recorded scan, an error code ‘6’ will be returned.

scan_set

skip – Skip forward/backwards specified # of bytes while playing

[command list]

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

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

Purpose: Skip forward/backwards specified # of bytes while playing

Settable parameters:

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

Monitor-only parameters:

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

Notes:

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

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

[command list]

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

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

Monitor-only parameters:

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

SS_rev1

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

[command list]

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

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

Monitor-only parameters:

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

SS_rev2

start_stats – Start gathering disk-performance statistics

[command list]

Command syntax: start_stats = [<bin 0 bound> : <bin 1 bound> :….: <bin 7 bound>] ;
Command response: !start_stats = <return code> ;

Query syntax: start_stats? ;
Query response: !start_stats ? <return code> : <bin 0 bound> : <bin 1 bound> :….: <bin 7 bound> ;

Purpose: Start gather disk performance statistics

Settable parameters:

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

Notes:

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

start_stats

status – Get system status (query only)

[command list]

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

Purpose: Get general system status

Monitor-only parameters:

Parameter	Type	Values	Comments
<status word>	hex	-	Bit 0 – (0x0001) system 'ready' Bit 1 – (0x0002) error message(s) pending; (message may be appended); messages may be queued; error is cleared by this command. See also 'error?' query Bit 2 – 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) not used Bit 6 - (0x0040) record 'on' Bit 7 - (0x0080) media full (recording halted) --- Bit 8 - (0x0100) playback 'on' Bit 9 - (0x0200) end-of-scan or end-of-media (playback halted) Bit 10 – (0x0400) recording can't keep up; some lost data Bit 11 – 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 19-18 – not used --- Note: Bits 20-27 are active only when in bank mode and no data transfers are in progress. Bit 20 – (0x100000) Bank A selected Bit 21 – (0x200000) Bank A ready Bit 22 – (0x400000) Bank A media full 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 Bit 27 – (0x8000000) Bank B write protected

status

task_ID – Set task ID (primarily for correlator use)

[command list]

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

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

Purpose: Set task ID (primarily for correlator use)

Settable parameters:

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

Notes:

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

task_ID

track_check – Check data on selected track (query only)

[command list]

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

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

Monitor-only parameters:

Parameter	Type	Values	Comments
<data mode>	char	st | mark4 | vlba | tvg | SS	See 'mode' command for explanation of data modes; 'tvg' corresponds to VSI test pattern;' SS' corresponds to StreamStor test pattern '?' indicates unknown format.
<data submode>	int	8 | 16 | 32 | 64 | mark4 | vlba	'8|16|32|64' if <data mode> is 'mark4' or 'vlba'; 'mark4|vlba' if <data mode> is 'st'
<data time>	time		Time tag from next 'track' frame header beyond current play pointer. See Note 5 of 'scan_check'.
<byte offset>	int	bytes-	Byte offset from current play pointer to beginning of next 'track' frame header of target track
<track frame period>	time		Time tag difference between adjacent track frames; allows original track data rate to be determined.
<track data rate>	real	MHz	Track data rate of source data from formatter.
<decoded track#>	int	2-33, 102-133	Track# decoded from auxiliary data field of target track; followed by 'D' if track is a 'duplicated' track; followed by'?' if not decodable or unallowed track#. See Note 3.
<#missing bytes>	int	bytes	Number of missing bytes between last and current 'track_check'; Should be =0 if immediately previous 'track_check' was within same scan Meaningless if immediately previous 'track_check' was in a different scan. See Note 4. See also Note 6 in 'scan_check'

Notes:

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

track_check

5. Regarding the ‘data time’ value returned by the ‘data_check?’, ‘scan_check?’ and ‘track_check?’ queries: The Mark 4 time-tags contain the day-of-year (DOY) but only the final digit of the year; the VLBA time-tags contain, instead, the last 3 digits of the Julian day number (misnamed MJD). To show the year and DOY in the returned values of ‘data time’ requires some assumptions. For Mark 4, we assume the most recent year consistent with the unit-year and DOY written in the Mark 4 time-tag; this algorithm reports the proper year provided the data were taken no more than 10 years ago. For VLBA, we assume the most recent Julian Day Number (JDN) consistent with the last 3 digits available in the VLBA time-tag; this algorithm reports the proper year provided the data were taken no more than 1000 days ago.

track_check

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

[command list]

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

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

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

Settable parameters:

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

Notes:

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

track_set

VSN – Write module VSN to permanent area

[command list]

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

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

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

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
VSN	char			Permanent 'extended-VSN' analogous to tape VSN, which will survive 'reset=erase' command; min 8 chars, max 255 chars (typical VSN might be 'MPI-0153/960/1024')

Monitor-only parameters:

Parameter	Type	Allowed values	Comments
<disk#>	int	0-7	First disk# in module in which there is a serial-number discrepancy
<original S/N>	char		Serial number of disk in position <disk#> when VSN was written
<new S/N>	char		Serial number of disk now in position <disk#>
'Disk serial-number mismatch'	char		Warning message

Notes:

1. The 'VSN=..' command is normally issued only when the module is first assembled or when the disk configuration is changed. The serial numbers of the resident disks are noted.
2. The 'VSN?' query compares the serial numbers of the original disks to the serial numbers of the currently-resident disks and reports only the first discrepancy. Issuing a 'VSN=…' command or a 'reset=erase' command will update the disk-serial# list to the currently-resident disks.

VSN

Mark 5 Linux Configuration Instructions

MIT Haystack Observatory

14 July 2003

1. Introduction

This memo contains step-by-step instructions for local setup of a Mark 5 system under Red Hat Linux. The following areas are addressed:

1. Network configuration
2. Time-zone configuration
3. ntp configuration

A monitor, keyboard and mouse must be connected to the Mark 5 system.

If you do not know the root password, contact Richard Crowley at rcrowley@haystack.mit.edu or phone him at 978-692-4764 or 781-981-5503.

2. Information You Will Need

You will need the following information to configure the Mark 5 system:

<IP address> (e.g., '192.52.61.57')

<local hostname> (e.g., 'Mark5-15')

<alias> (a nickname for your system; usually same as <local hostname>)

<local domain> (e.g., 'haystack.mit.edu

<network mask> (e.g., typically '255.255.255.0' for class C network, '255.255.0.0' for class B, or '255.0.0.0' for class A; check with your sysadmin)

<nameserver IP> (e.g., '192.52.61.1') – local DNS name server IP address

<gateway IP> (e.g., '192.52.61.17') – local gateway IP address

<ntp server> pick from list at http://www.eecis.udel.edu/~mills/ntp/clock1.html; IP address preferred, but can use name (e.g., '164.122.7.123' or 'tick.mhpcc.edu'). Choose one with which you have good connectivity.

3. Determine Red Hat Release Number

Enter `cat /etc/redhat-release` to determine you Red Hat Linux release. It will probably be either 7.1 or 7.2, though we are currently working to include later releases as well.

Follow the appropriate instructions below.

4. Network Configuration (Red Hat Release 7.1)

1. Connect your system to your local network with an RJ-45 network cable.
2. Log in as *root*
3. *startx* to start X-Windows GUI interface[1]
4. Click on Gnome (footprint icon) Programs System Network Configurationto start the GUI Network Configurator.

[1] Since X-windows is set up to work with a specific monitor, there is a possibility that it might not work with a different monitory. Please contact us if that is the case.

5. You will see 4 tabs, with information to be entered under each:
    a. Names
        Hostname – set to <local hostname>
        Domain – set to <local domain>
        Nameservers – set to <nameserver IP> (can add others if they exist)
    b. Hosts
        IP – set to <IP address>
        Name – set to <local hostname>.<local domain>
            (i.e. 'Mark5-15.haystack.mit.edu') Leave the '127.0.0.1' entry.
        Nicknames – set to <alias>
    c. Interfaces
        Click on 'eth0'
        If 'eth0' is 'inactive', click on 'Activate'
        Click 'Edit'
            IP – set to <IP address>
            Netmask – set to <network mask>
            Network Broadcast:
                Check 'Activate interface at boot time'
                Uncheck 'Allow any user to (de)activate interface'
        Click 'Done'
    d. Routing
        Default Gateway – set to <gateway IP>
6. Click '*Save'*, then '*Quit'*
7. Log out of X-Windows: *Gnome→Log Out*
8. Mark sure '*Action*' is '*Logout'* – click '*Yes'*.
9. `/sbin/shutdown -r now` to reboot and activate changes.

5. Network Configuration (Red Hat Release 7.2)

1. Connect your system to your local network with an RJ-45 network cable.
2. Log in as `root`
3. `startx` to start X-Windows GUI interface
4. Click on *Gnome*(footprint icon)→*Programs→System→Network Configuration* to start the GUI Network Configurator; enter root password as requested
5. You will see 4 tabs, with information to be entered under each:
    a. Hardware
        Should show your Ethernet interface as '*eth0*'.
        No changes are normally necessary.
    b. Devices
        Select '*eth0*', click '*Edit*'
        i. General
            Select '*Activate device when computer starts'*

De-select '*Allow all users to enable and disable this device'*

ii. Protocols

Click on 'TCP/IP', then 'Edit'

a. TCP/IP

De-select '*Automatically obtain IP address ...*'

Address – set to <IP address>

Subnet Mask – set to <network mask>

Default Gateway Address – set to <gateway IP>

b. Hostname

Hostname – empty

Deselect 'Automatically obtain DNS information from provider'

Click 'OK'

c. Routing

Should be empty

Click 'OK'

iii. Hardware Device

Should be OK as is. Make sure '*Enable Device Alias support'* and '*Use Hardware Address'* are both de-selected.

Click '*OK'*

c. Hosts

If there is already an entry with <IP address>, etc., for this system click 'Delete' or 'Edit', as appropriate. If new entry to be created, click 'Add'.

Address – set to <IP address>

Hostname – set to <local hostname>

Alias – set to <alias>

Click '*OK'*

d. DNS

Hostname – set to <local hostname>

Domain – set to <local domain>

Primary DNS – set to <nameserver IP>

Secondary DNS – as applicable

Tertiary DNS – as applicable

DNS Search Path – should be empty

Search Domain – should be empty

Click *'Close',* then *'Yes'*

6. Log out of X-Windows: *Gnome→Log Out*
7. Mark sure '*Action*' is '*Logout'* – click '*Yes'*.

8. */sbin/shutdown -r now* to reboot and activate changes[2].

6. Check Network Configuration

Check the results of the network configuration as follows.

1. Login as `root`
2. Check local hostname and domain

   Enter *cat /etc/hosts* go view contents of /etc/hosts file:

   There should be a line of the form

   `<IP address>  <local hostname>.<local domain>  <local hostname>`

   Alternately,

   /sbin/ifconfig -a will show the IP address assigned to your machine.

   hostname will show the hostname of your machine
3. Check name server

   Enter *cat /etc/resolv.conf* to view contents of resolv.conf file

   You should see the following lines, in any order:

   ```
   search <local domain>
   nameserver <nameserver>
   domain <local domain>   (may be present; not necessary)
   ```
4. Check subnet mask

   Enter */sbin/ifconfig -a*

   Find the value of 'Mask' in the listing; should be <network mask>
5. Check routing table

   /sbin/route -n will list the routing table

   It should have at least 3 entries, as follows:

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
<routing mask>	0.0.0.0	255.255.255.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	<gateway IP>	0.0.0.0	UG	0	0	0	eth0

   where

   <routing mask> is same as <gateway IP> except '0' in 4th subfield (i.e. if <gateway IP> is '192.52.61.xx', then <routing mask> is '192.52.61.0'

7. Check Network Operation

When you think everything is all set, ping an address inside your local network and one outside of your local network to test network connectivity. The ping target may be either named or an IP address. Following is an example of a successful ping:

```
(Initiate ping)    ping planck.haystack.edu
(Response)         Pinging planck.haystack.edu [192.52.61.1] with 32 bytes of data:
                   Reply from 192.52.61.1: bytes=32 time time<1ms
```

8. Time-zone Configuration

The local time zone is set in file `/etc/sysconfig/clock`, which will contain a line similar to

```
ZONE="America/New_York"
```

[2] For efficiency, you may want to also set the time-zone configuration before re-booting (see Section 8).

The directory `/usr/share/zoneinfo` contains all possible timezones, including sub-directories with additional possibilities. In this example, `America/New_York` is the filename (within the `/usr/share/zoneinfo` directory) that resolves the New York time zone. Starting at `/usr/share/zoneinfo`, drill down as needed to find the file that resolves your particular time zone. Then, as root, run your favorite editor on `/etc/sysconfig/clock` to edit the ZONE=… line with that filename. Examples:

```
ZONE="Japan"
ZONE="Europe/Berlin"
ZONE="America/Indiana/Indianapolis"
```

Reboot for the new time zone to take effect.

9. ntp Configuration

9.1 Set time at boot

Time is updated at boot time by adding (or editing) the following line in `/etc/rc.d/rc.local`

```
/usr/sbin/ntpdate -b -p 8 -u <ntp server>
```

9.2 Daily time update

There are two methods to ensuring that your system properly keeps time over the long term:

9.2.1. Establish a cron job to update time daily

There are 2 ways to do this (must be root):

a. Enter `crontab -e` to open up the default editor and enter (or edit) the following line:

```
30 2 * * * /usr/sbin/ntpdate -b -p 8 -u <ntp server>
```

Save the file[3]. This will cause the time to be updated everyday at 2:30 am (2:30 am was chosen arbitrarily; could be any time).

b. Create or edit file `timeupdate` or `ntpdate` (one of these files will probably already exist) in directory `/etc/cron.daily` containing the line

```
/usr/sbin/ntpdate -b -p 8 -u <ntp server>
```

Change the file permissions to 755 (`chmod 755 filename`). The script in any file in directory `/etc/cron.daily` is executed once per day at about 4:00am (by default), so technically this file may have any name you wish.

9.2.2 Install NTP

Instructions for installing NTP are in `/usr/share/doc/ntp-*/index.htm`.

If you choose this option, you should remove the ntp file in `/etc/cron.daily`

[3] The file is `/var/spool/cron/root`, which should not be directly edited. Entering `crontab -e` as root opens/creates this file and brings up the default editor (usually vi). Entering `crontab -e` as a user does the same thing, except the corresponding file is `/var/spool/cron/<username>`.