VLBA Sensitivity Upgrade MEMO 18

MARK 5 MEMO 61

Mark5C Software Interface Specification

National Radio Astronomy Observatory & MIT Haystack Observatory

January 7, 2008

1 Introduction

This document describes the requirements of the program to be used as the primary software interface on Mark5C VLBI recorders. This program will be the analog of `Mark5A` used in Mark5A systems. In this document the assumed name of this interface program will be `mark5c`.

2 Starting `mark5c`

The following command line options should be supported by `mark5c`:

arg.	parameter	default	description
`-e`	*command*		execute *command* immediately after starting
`-f`	*file*		execute the `mark5c` commands contained in *file* upon start
`-h`			display program version number, print help information. and quit
`-m`	*level*	0	set output message level to *level* (between 0 and 3)[1]
`-s`	*number*	7	set maximum number of simultaneous connections to *number*
`-b`	*path*		set `STREAMSTOR_BIB_PATH` to *path*
`-p`	*path*		set the path where `system` programs can live (see § 8.4.2)

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

⟶ `mark5c -s 3 -m 0 -e "disk_state_mask=1:0:1;MAC_list=00.50.8D.E9.36.F4" &`

⟶ `mark5c -s 7 -m 1 -b /usr/local/share/streamstor/bib &`

The behavior of `mark5c` should not depend on the filename of the executable (or links to it). Command line switches should not change the functionality of the software (i.e., there should be no FIFO mode).

3 Stopping `mark5c`

`mark5c` should be cleanly stoppable upon receipt of a SIGINT signal (i.e., as if control-C were used). A signal handler that closes any open files and puts any hardware back into a safe state should be implemented. If

recording is in process, it should be stopped so the directory contents can be properly updated. Any standard wrapper program (such as a logger) should similarly respond properly to a SIGINT signal and shall cascade the SIGINT to `mark5c` if necessary.

4 Connectivity

Communication with the Mark5C program shall be via a TCP connection to port *TBD* on the Mark5C unit. Multiple connections, up to a maximum supplied by a command line argument, shall be simultaneously allowed, with commands and queries being executed in the order received, regardless of their origin. I note that this system has worked remarkably well in Mark5A and I see no reason to change this aspect.

5 Command, Query and Response Syntax

VSI-S shall define the syntax of communication into and out of `mark5c`. For more detail, see Mark5A command set section 3 (Haystack Mark5 Memo # 39).

6 Bank and Non-bank Modes

Support of non-bank mode (writing on two disk modules simultaneously) is highly desirable as a means to double the record rate over the rate supported by a single disk module. When the highest data rates are not required, it is expected that bank mode would be preferred to allow for increased operational flexibility. Mixed bank mode and non-bank mode on individual modules will not be supported. In non-bank mode, means to determine the Volume Serial Numbers (VSNs) of both mounted modules shall exist.

7 Access to Data

The biggest fundamental difference between Mark5C and its predecessors is that Mark5C is a write-only device. Correlation playback is done with a generic Conduant *Big River* device or any Mark5 unit (A, B, or C) across its PCI or PCIe bus. In the earlier systems, functions that peek at the recorded data (i.e. `data_check`) fit naturally into `Mark5A` due to its dual write/read functionality. In the Mark5C system, which is write-only and sports a much enhanced directory structure, it is questionable whether or not such functionality is needed or desirable. In order to keep `mark5c` minimal I suggest that all but the simplest diagnostic software be provided as separate programs that could be run with the `system` command (see § 8.4.2). Commands `disk2file` and `file2disk` are expected to be useful for diagnostics and should be included.

8 Command Set

The Mark5C system is a much simpler device than its predecessors; its specified command set should reflect this simplicity by including only those commands necessary for recording data and verifying proper operation. The following commands and queries form the language for communication with mark5c. Most of the commands are familiar from `Mark5A/B`. Only about half of the `Mark5A` commands, or variants there of, are required.

8.1 Unchanged commands

The following commands require little or no change from their Mark5A implementation. Note that 'disk' is spelled without a 'c'. Support for the 'disc' spelling will not be supported in `mark5c`. All commands remain case insensitive though upper case is suggestively used to identify acronyms.

command/query	description
bank_info?	Get bank information (query only)
bank_set	Set/get active bank for recording or reading
dir_info?	Get directory information (query only)
disk_model?	Get disk model numbers (query only)
disk_serial?	Get disk serial numbers (query only)
disk_size?	Get disk sizes in bytes (query only)
disk_state	Set/get Disc Module Status (DMS)
disk_state_mask	Set mask to enable changes in DMS
DTS_id?[1]	Get system information (query only)
error?	Get error number/message (query only)
get_stats?	Get disk performance statistics (query only)
protect	Set/get module write protect flag
recover	Recover data lost due to abnormal recording condition
ss_rev?[2]	Get StreamStor software/firmware versions info
start_stats	Start gathering disk performance statistics
VSN	Set/get module extended Volume Serial Number

Notes:

1. mark5c should return a proper version number instead of a compilation date.
2. mark5c should determine the information needed for ss_rev at startup time, print it to stdout (depending on the message level) and save it so that ss_rev can be run even when the StreamStor card is busy.

8.2 Commands requiring change

The following commands are to be retained, but changed to reflect the new requirements of the Mark5C system.

8.2.1 mode – Set data recording mode

Command: ⟶ mode= <data mode> : <data submode1> : [<data submode2>] ;
⟵ !mode= <return code> ;

Query: ⟶ mode?;
⟵ !mode? <return code> : <data mode> : <data submode1> : [<data submode2>] ;

Purpose: Select among Mark5C native mode or Mark5B compatibility mode.

Settable parameters:

parameter	type	allowed values	comments
<data mode>	char	mark5c |	record in mark5c format (see Note 1)
		mark5b |	record in mark5b format (data source=ext)
		tvg5c |	record TVG data in mark5c format
		tvg5b |	record TVG data in mark5b format (data source=tvg)
<data submode1>	int	> 0	(mark5c) number of channels
	hex	$\neq$ 0x0	(mark5b) bit-stream mask
<data submode2>	int	1 | 2 | 4 | 8 | 16	(mark5b) decimation ratio
			(mark5c) not used

Notes:

1. For Mark5C formats the parameters required to completely specify the setup of a scan are split between commands `mode` and `packet`. The parameters set by `packet` are lower-level and are used by the StreamStor card to determine the acceptance of packets. The parameters set by `mode` are used in the generation of the scan directory listing to aid in the unpacking of data and are not used on the StreamStor card.

8.2.2 record – Record data from 10Ge input to module(s)

Command: ⟶ `record=` <record on/off> : <scan label> ;
⟵ `!record=` <return code> ;

Query: ⟶ `record?;`
⟵ `!record?` <return code> : <status> : <scan #> : <scan label> :
[<# packets recv'd>] : [<# FCS errors>] : [<# length errors>] :
[<# PSN errors>] ;

Purpose: Turn recording on/off; assign scan label

Settable parameters:

parameter	type	allowed values	comments
<record on/off>	char	`on` | `off`	
<scan label>	ASCII		only for `on` (see Note 1)

Monitor-only parameters:

parameter	type	values	comments
<status>	char	`on` | `off` | `halted` | `overflow` | `waiting`	normal conditions end of media encountered FIFO error on SS card no packet received in last **TBD** ms
<scan #>	int	≥ 1	scan number on the module (or module pair)
<# packets recv'd>	int	≥ 0	number of packets received during this scan
<# FCS errors>	int	≥ 0	number of packets rejected due to CRC errors
<# length errors>	int	≥ 0	number of packets rejected due to length
<# PSN errors>	int	≥ 0	number of packets received with non-sequential PSN

Notes:

1. The scan name should follow the standardized format including experiment name, station code, and perhaps a scan name, each separated by underscores, e.g. `bw088n_la_02`. Whitespace and the characters `'/.:;=_+'"\` may not be contained in the three sub-fields.

8.2.3 reset – Reset Mark5 unit (command only)

Command: ⟶ `reset=` <control> : [<bank>] ;
⟵ `!reset=` <return code> ;

Purpose: Reset system; erase/mount/dismount modules

Settable parameters:

parameter	type	allowed values	comments
<control>	char	`erase` |	causes record pointer to be set to zero, forces bank mode.
		`erase_last_scan` |	resets record pointer to beginning of last scan.
		`nberase` |	causes modules to be erased and starts non-bank mode.
		`abort` |	causes certain operations to stop.
		`mount` |	causes module in bank <bank> to be mounted.
		`dismount`	causes module in bank <bank> to be dismounted.
<bank>	char	`A` | `B`	used only for mount and dismount options.

8.2.4 status – Get system status (query only)

Query: ⟶ `status?;`
⟵ `!status?` <return code> `:` <status word> `;`

Purpose: Get general system status

Monitor-only parameters:

parameter	type	bit value	comments
<status word>	hex	0x0001	Bit 0 – system 'ready'
		0x0002	Bit 1 – error message(s) pending
		0x0004	Bit 2 – system command has control of StreamStor
		0x0008	Bit 3 – 'delayed-completion' commands pending
		0x0010	Bit 4 – 'delayed-completion' queries pending
		0x0020	Bit 5 – disk-FIFO mode
		0x0040	Bit 6 – record 'on'
		0x0080	Bit 7 – media full
		0x0100	Bit 8 – not used
		0x0200	Bit 9 – not used
		0x0400	Bit 10 – recording can't keep up; some data lost
		0x0800	Bit 11 – not used
		0x1000	Bit 12 – disk2file active
		0x2000	Bit 13 – file2disk active
		0x4000	Bit 14 – not used
		0x8000	Bit 15 – not used
		0x10000	Bit 16 – not used
		0x20000	Bit 17 – not used
		0x40000	Bit 18 – DIM ready to record
		0x80000	Bit 19 – not used
		0x100000	Bit 20 – bank A selected
		0x200000	Bit 21 – bank A ready
		0x400000	Bit 22 – bank A media full or faulty
		0x800000	Bit 23 – bank A write protected
		0x1000000	Bit 24 – bank B selected
		0x2000000	Bit 25 – bank B ready
		0x4000000	Bit 26 – bank B media full or faulty
		0x8000000	Bit 27 – bank B write protected

8.3 New commands

Below are listed some proposed command/query names with proposed functionality. Without knowing the full streamstor API it is not clear if this represents a natural factoring of the required functionality.

8.3.1 fill_pattern – Set/get StreamStor 32-bit fill pattern

Command: ⟶ `fill_pattern=` <pattern> ;
⟵ `!fill_pattern=` <return code> ;

Query: ⟶ `fill_pattern?;`
⟵ `!fill_pattern?` <return code> : <pattern> ;

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

Settable parameters:

parameter	type	values	comments
<pattern>	hex	0x00000000 - 0xFFFFFFFF	The fill pattern

Notes:

1. This fill pattern is used both at record time to replace invalid data when recording in PSN monitor mode 1 and at read time (i.e., `disk2file`) in cases where data cannot be recovered from one or more disks in the module.
2. If failures occur at both record time (bad packets) and read time (disk went bad), then it is possible that two different fill patterns need to be detected, and the duration of the bad-packet fill pattern could be less than the length of a packet data frame. The use of two different fill patterns has the potential advantage of distinguishing between record and playback problems.
3. If the same fill pattern is used at record time and read time, the 64-bit data granularity will ensure that the fill patterns are always *in phase* with each other.

8.3.2 MAC_list – Set/get list of MAC addresses to accept

Command: ⟶ `MAC_list=` <MAC #1> : [<MAC #2>] : ⋯ ;
⟵ `!MAC_list=` <return code> ;

Query: ⟶ `MAC_list?;`
⟵ `!MAC_list?` <return code> : <MAC #1> : [<MAC #2>] : ⋯ ;

Purpose: Set/get list of MAC addresses (up to 16) to accept

Settable parameters:

parameter	type	allowed values	comments
<MAC #n>	ASCII		See note 1.

Notes:

1. All MAC addresses should be period separated 8-bit hextets, e.g. `00.50.8D.E9.36.F4`. Traditionally colons are used to separate the digits, but this is incompatible with VSI-S.
2. A MAC address of 00.00.00.00.00.00 can be specified to accept valid data from any source.

8.3.3 packet – Set/get packet acceptance criteria

Command: ⟶ `packet=` <DPOFST> : <DFOFST> : <length> : <PSN mode> : <PSNOFST> ;
⟵ `!packet=` <return code> ;

Query: ⟶ `packet?;`
⟵ `!packet?` <return code> : <DPOFST> : <DFOFST> : <length> : <PSN mode> : <PSNOFST> ;

Purpose: Set/get packet acceptance criteria

Settable parameters:

parameter	type	allowed values	comments
<DPOFST>	int	≥ 0	payload byte offset from beginning of payload to first recorded data
<DFOFST>	int	≥ 0	payload byte offset to beginning of recording
<length>	int	> 0	number of bytes to record per packet (see Note 1)
<PSN mode>	int	0 | 1 | 2	Packet Serial Number (PSN) monitor mode (see Note 2)
<PSNOFST>	int	≥ 0	payload byte offset from beginning of payload to PSN (for PSN-monitor modes 1 or 2)

Notes:

1. The length of data to be recorded must be a multiple of 64 bits.
2. PSN-monitor mode 0 will disable packet serial number checking. PSN-monitor mode 1 will replace invalid packets with the specified fill pattern. PSN-monitor mode 2 will prevent invalid packets from being written to disk.

8.4 personality – Set/get personality

Command: ⟶ `personality=` <type> : [<root>] ;
⟵ `!personality=` <return code> ;

Query: ⟶ `personality?;`
⟵ `!personality?` <return code> : <type> : [<root>] ;

Purpose: Set/get personality (i.e., emulation mode)

Settable parameters:

parameter	type	allowed values	comments
<type>	char	mark5c |	Mark5C Normal operating mode (default)
		mark5c- |	Mark5C− emulation mode (see Note 3)
		file	Write data to filesystem, not StreamStor (see Note 4)
<root>	char		(file only) A filesystem path (see Note 4)

Notes:

1. A *personality* is here defined as a set of functions bound to the commands and queries described in this document. A possible implementation of this is for each personality to have a `setup()` function and a `shutdown()` function that are called to initialize and clean up from a personality change, and a set of functions that are mapped to the commands and queries. Nothing is to preclude the various personalities from sharing a subset of functionality. The implementation of `mark5c` should make it easy to add new personalities to the program.
2. This command cannot be issued while a delayed completion operation is in effect or while data is being recorded on any type of medium.

3. The *emulation* personality causes data received through the system NIC to written to the StreamStor media (see Mark5 Memo # 62/VLBA Sensitivity Upgrade Memo # 19 for a description of the Mark5C− functionality). This is similar to net2disk from previous Mark5 generations.

4. The *file* personality causes incoming data received through the system NIC to be written to a filesystem (perhaps a RAID array). The optional parameter should be a directory specifying the root of the filesystem to write. Files written to this directory will have systematically determined filenames bearing close resemblence to scan names on Mark5 modules. Additionally, a file containing the equivalent of a Mark5 scan list will be created in the specified directory.

5. Implementation of this command is optional – its absense will not imply non-conformance with the Mark5C software specification.

8.4.1 ss_ifconfig – Set/get StreamStor 10G NIC configuration

Command: ⟶ `ss_ifconfig=` <state> : <MTU> : <mode> : <MAC address> ;
Query: ⟶ `ss_ifconfig?;`
⟵ `!ss_ifconfig?` <return code> : <state> : <MTU> : <mode> : <MAC address> ;

Purpose: Set or get the parameters of the StreamStor 10G NIC daugher board.

Settable parameters:

parameter	type	values	comments
<state>	char	active | inactive |	enable 10G interface disable 10G interface
<MTU>	int	$\geq 64, \leq 9000$	Maximum packet size to accept default: 9000
<mode>	char	normal | all	apply packet filtering (default) accept all packets
<MAC address>	ASCII		See note 2.

Notes:

1. All parameters specified by this command apply only to the 10G ethernet interface on the StreamStor daughter board.

2. The format of the MAC address shall be a period separated 8-bit hextet, e.g. `00.50.8D.E9.36.F4`.

8.4.2 system – Start and monitor an external program

Command: ⟶ `system=` <SS flag> : <program> : [<arg 1>] : [<arg 2>] : ⋯ ;
⟵ `!system=` <return code> ;

Query: ⟶ `system?;`
⟵ `!system?` <return code> : <status> : <SS flag> : <program> : [<out 1>] : ⋯ ;

Purpose: Run a program, possibly using the StreamStor card, from within `mark5c`

Settable parameters:

parameter	type	allowed values	comments
<SS flag>	char	SS | noSS	set to SS if <program> talks to SS card (see Note 2) set to noSS otherwise
<program>	char		program to execute (see Note 3)
<arg #>	char		command line arguments for <program>

Monitor only parameters:

parameter	type	values	comments
<status>	char	error |	an error occured starting <program>
		active |	the program is still running
		completed	the program finished
<out #>	char		list of most recent stdout values from <program> (see Note 5)

Notes:

1. A new thread will be started that will monitor stdout from <program>.
2. If SS is set, XLRClose() will be called before starting <program> and XLROpen() will be called once <program> completes, preventing the use of any StreamStor functionality from within `mark5c`.
3. The program to execute must live in a special directory specified with a command line argument. If this command line argument is not supplied, an error will be returned when calling `system`.
4. To stop <program> prematurely, use `reset=abort` which will send a SIGINT signal to the process.
5. If stdout of <program> contains characters that cannot be returned using the VSI-S format (such as colons or semicolons) they will be replaced with whitespace. Each continuous non-whitespace substring will be returned as a separate <out> argument.

Example:

⟶ `system=SS : SSErase : -m : 0 ;`
⟵ `!system=1;`

8.5 Optional commands

Below are listed some Mark5A/B commands/queries that might still be relevant to `mark5c` but are of lesser importance.

command/query	description
`os_rev?`	Get details of operating system (query only)
`bank_switch`	Enable/disable automatic bank switching
`disk2file`	Transfer data from module to file
`file2disk`	Transfer a file to module

9 Software environment

Software should be managed using modern software engineering methods. A Makefile (preferably generated by gnu automake) should drive compilation and installation. A division of the code into multiple portions (for example, one portion could be just the streamstor libraries, drivers and firmware and another could be the source for Mark5C) is encouraged. The software should compile without warnings and should work when standard compiler optimizations are enabled (i.e., use of `-O2` for gcc). Code should be managed using revision control (e.g. subversion). Version numbering and revision history should be maintained with the source code. If written in C or C++, a `.h` file containing constant definitons and relevant data structure definitions should be usable by other programs that interface to `mark5c` without modification. This/these files should be installed in an appropriate *include* directory upon installation. Debian and Redhat packages should be Makefile targets to ease installation and maintenance.