Core3H Commands – DDC_V (v124)

19.11.2019

Author: Armin Felke, felke@mpifr-bonn.mpg.de, Sven Dornbusch dornbusch@mpifr-bonn.mpg.de

Legend of notation:

- [A] → expression A is optional
- (A|B) → either expression A or expression B
- <x> → a placeholder for a value named “x”

version

Displays the version of the firmware (distinguishing between hardware and software version)

Arguments: –

sysstat

Displays information about the current status of the system and gives an overview of the state of the most important user settings.

Arguments: –

sysstat_fs

Displays information about the current status of the system (machine readable).

Arguments: –

mode_fs

Displays mode information (machine readable).

Arguments: –

status_fs

Displays time sync information (machine readable).

Arguments: –

devices

Lists all available devices in the system.

Arguments: –

Each device is displayed by its name and memory address.

Note: the device's absolute memory address is implementation dependent and may be subject to change in a future version.

Available devices:

- **eth0**, **eth1**: the two ethernet cores
- **regbank0**: this is the legacy register bank from the previous FiLa10G design.
  The device is available for backwards compatibility only. Supported control and status registers are transparently mapped to the registers of device "fila10g" (see FiLa10G register documentation for more information)
- **fila10g**: this is the FiLa10G core device.
  Direct access to the registers of the device is forbidden.
- **core3**: this is the Core3 core device. (Core3 Adapter)
- **mk5b_header**: this register bank represents the Mark5B header which is used to configure the Mark5B Framer. Direct access is allowed but only recommended if functionality is used that is not accessible through predefined commands. The header register layout is compliant to the Mark5B standard.
- **vdif_header**: this register bank represents the VDIF header which is used to configure the VDIF Framer. Direct access is allowed but only recommended if functionality is used that is not accessible through predefined commands. The header register layout is compliant to the VDIF standard.
- **chan_perm**: This memory space configures the channel permutations on the input data bits. It represents an array of 128 bytes, each of them containing an index. The index determines which bit index is sourced by which input bit.
  The device can be accessed both by register and memory commands.
  Example 1: regwrite chan_perm 0 0x01010101
  This maps the stream bit indexes 3,2,1,0 to input bit index 1
  Example 2: regwrite chan_perm 0 0x03020100
  This restores the default mapping for the bits 3,2,1,0
  (i.e. stream bit index = input bit index)
  Note: the same channel permutation functionality is internally used to implement the bitmask feature. Using both a bitmask and a custom permutation may lead to unexpected results.
- **fila10g_output0**: a device that allows checking of outgoing packets at output 0
  (see "output" command)
- **fila10g_output1**: a device that allows checking of outgoing packets at output 1
  (see "output" command)
- **fila10g_dest0**: the device that configures the IP/Port destinations of outgoing packets on eth0
  (see "destination" command)
- **fila10g_dest1**: the device that configures the IP/Port destinations of outgoing packets on eth1
  (see "destination" command)
- **debug**: this is the debugging view of the FiLa10G core device. Direct access to the registers is allowed.

Warning: the device should only be available during testing and be disabled in the final release version!

regread

Reads the value of a device register.

Arguments: <device name> <register index>
<device name>: the name of the device as given by the "devices" command
<register index>: the index of the device register to be read

Note: not all devices have readable registers. Some are forbidden and are not directly accessible by the user. See command "devices" for more information.

regread_dec

Reads the value of a device register (machine readable).

Arguments: <device name> <register index>
<device name>: the name of the device as given by the "devices" command
<register index>: the index of the device register to be read

Note: not all devices have readable registers. Some are forbidden and are not directly accessible by the user. See command "devices" for more information.

regwrite

Writes the value of a device register.

Arguments: <device name> <register index> <register value>
<device name>: the name of the device as given by the "devices" command
<register index>: the index of the device register to be written
<register value>: the numeric value to be written.
Note: not all devices have writable registers. Some are read-only or forbidden and are not directly accessible by the user. See command "devices" for more information.

regupdate

Updates only certain bits of the value of a device register.

Arguments: <device name> <register index> <register value> <bit mask value>
<device name>: the name of the device as given by the "devices" command
<register index>: the index of the device register to be written
<register value>: the numeric value to be written.
<bit mask value>: the bitmask specifying which bits are overwritten and which remain unchanged

The bits to be updated are given by the bitmask. Bit positions in the bitmask that are set to '0' will not be modified, Bits positions that are set to '1' will be overwritten with the bit at same position in the new value.

read

Reads a data block from a memory location and displays it on the console.

Arguments: <start address> [<end address>]
<start address>: the start address of the block to be read
<end_address> (optional): specifies end of block to be read, otherwise the end address of the device at this position is taken by default.

Note: valid data blocks are memory regions that are mapped to a device. The user does only have access permissions to register type and memory type devices.

write

Writes a value (of size 1,2, or 4 byte) to a memory location.

Arguments: (b|s|l) <address> <value>
first argument: 'b' for byte value, 's' for 16 bit value, 'l' for 32 bit value
<address>: address of the write operation
<value>: value to be written. Will be interpreted according to first argument.

Note: valid data blocks are memory regions that are mapped to a device. The user does only have access permissions to register type and memory type devices.

setbase

Sets base address for relative memory access.

Arguments: <base address>
<base address>: new base address

readbase

Same as read with the only difference that the start address and end address is not absolute but is computed relative to the base address (see also "setbase" command).

Arguments: <start address> [<end address>]

writebase

Same as write with the only difference that the address is not absolute but is computed relative to the base address (see also "setbase" command).

Arguments: (b|s|l) <address> <value>

vsi_samplerate

Gets or sets the VSI input sample rate (samples per second).

Arguments: [<#samples per second> [<decimation divisor>]]
<#samples per second> (optional): lets the system know the sample rate of the VSI input (samples per second)
<decimation divisor> (optional): a divisor d in the range 1..255.
d=1 is the default, i.e. no decimation. d>1 decimates the input such that the resulting sample rate is 1/d of the original rate.

All arguments are optional. If the command is called without arguments the current VSI input sample rate is displayed.

vsi_bitmask

Gets or sets the VSI input bitmask in order to select the channels to be processed

Arguments: [[[[<bitmask-4>] <bitmask-3>] <bitmask-2>] <bitmask-1>|reset]
<bitmask-X> (optional): a 32 bit bitmask each, that specifies the active bits of VSI-X.
Omitted bitmasks are assumed to be zero (0x00000000).
The only rule for
the values is that in total at least one bit must be active.

All arguments are optional. If the command is called without arguments the current VSI bitmask is displayed.

The eight - up to 32 bit wide - bitmasks specify which bits of a VSI input stream are active and being processed, the others are discarded. This effectively reduces the data amount.

If the command is called with the "reset" argument, all of the bitmasks are reset to their default value (0xffffffff = all active).

In general, the number of active bits in the bitmask should be a power of two (1,2,4,8,16,32,64,128). If the number of active bits is not a power of two, the highest active bit is replicated in the resulting data word until the the next power of two is reached. This special case is displayed as a warning by the command.

Note: The bitmask is displayed as "(invalid)" if some custom settings (e.g. a custom channel permutation) would lead to an impossible bitmask setup.

vsi_inputwidth

Gets or sets the effective bit width of the VSI input, that is to be processed by FiLa10G.

Arguments: [<input width in bits>]
<input width in bits> (optional): allowed input widths: 1,2,4,8,16,32,64,128

If the command is called without arguments the current input width is displayed. Otherwise the input width is set to the new value.

FiLa10G has a 128 bit wide input width, such that up to 128 parallel input bit streams are supported. Only the <input width in bits> bit streams with the lowest index are fed to the processing pipeline.

Note: The internal data path width of FiLa10G is 64 bit. Input widths lower than 64 result in the input data being packed to the internal data path width. An input width of 128 will result in the input data being serialized to two consecutive 64 bit words.

Note: In general, the input width should be equal to or lower than the width of the actual VSI input source. Higher input widths will likely waste memory on the recorder.

Note: When a VSI input is selected ("inputselect"), the input width is automatically set to the actual size of the selected input.

Note: the "VSI input width" is also modified by the command "vsi_bitmask". Thus, using a custom bitmask and a custom input width at the same time may lead to unexpected results.

vsi_swap

Swaps two given VSI inputs

Arguments: [((vsi<X> vsi<Y>)|reset)]
vsi<X>: one of the two logical vsi inputs to be swapped
vsi<Y>: the other of the two logical vsi inputs to be swapped

If the command is called without arguments the current mapping of physical to local VSI inputs is displayed.

If the command is called with the “reset” as an argument, the mapping is reset to its default, i.e. no swapping is applied at all.

If the two vsi inputs are specified, the logical vsi inputs are swapped with the effect that their association to physical VSI input ports will be the other way round. Applying the same swap a second time undoes the swapping.

Note: When a VSI input is selected ("inputselect"), the swapping is automatically reset.

inputselect

Selects one of the avaiable input data sources.

Arguments: <input source>
<input source>: valid values are: (tvg|vsi1|vsi2|vsi1-2|vsi1-2-3-4|gps)

FiLa10G's physical input width is 128 bit wide. The selections listed above result in the following input data being fed to the system (most significant bit on the left side, least significant bit on the right side):

tvg:	0xAAAAAAAA 0xAAAAAAAA 0xAAAAAAAA <32-bit TVG input data>
vsi1:	0xAAAAAAAA 0xAAAAAAAA 0xAAAAAAAA <32-bit vsi1 input data>
vsi2:	0xAAAAAAAA 0xAAAAAAAA 0xAAAAAAAA <32-bit vsi2 input data>
vsi1-2:	0xAAAAAAAA 0xAAAAAAAA <32-bit vsi2 input data> <32-bit vsi1 input data>
vsi1-2-3-4:	<32-bit vsi4 input data> <32-bit vsi3 input data> <32-bit vsi2 input data> <32-bit vsi1 input data>
gps:	0xAAAAAAAA 0xAAAAAAAA 0xAAAAAAAA 0xAAAAAAAA *)

*) “gps” is a constant data pattern equipped with the Clock and 1PPS signal of the GPS device.

Recommendation: execute command "reset" or "reset keepsync" after changing the input source.

This command implicitly resets the VSI bit mask, input width and, VSI input swap settings to their respective defaults. The input width is reset to 32 bit for “tvg”, “vsi1”, and “vsi2”. It is reset to 64 bit for “vsi1-2” and it is reset to 128 bit (full input width) for “vsi1-2-3-4” and “gps”.

splitmode

Enables (on) or disables (off) split mode

Arguments: [on|off]

Enable the split mode to have the two outputs (output0 and output1) produce different data.

When split mode is enabled, the selected and bitmasked input data is split into two exaclty in the middle. The two halves will be processed as if they were independent streams. The lower half (regarding its bit position) is outputted at output0 and the higher half is outputted at output1.

Split mode requires an input width of at least 2 bit.

Note: Raw, Mark5B or VDIF format can be applied independently to each of the split streams by the help of the “start X+Y” command syntax.

Note: In order to specify a correct VDIF frame setup you have to take into account that the effective input width is halved when spit mode is enabled.

Note: a restriction of split mode is that only output0 is capable of producing multi-threaded VDIF data. At output1 the data will always be forced to be single-threaded, regardless of the chosen frame setup.

tvg_samplerate

Gets or sets the TVG sample rate (samples per second).

Arguments: [<#samples per second>]
<#samples per second> (optional): sets the sample rate (samples per second) of the TVG input.
Valid range is: 1...64000000

Note: the TVG is operated with a 64MHz clock.

tvg_mode

Gets or sets the test vector generator mode.

Arguments: [<tvg mode>]
<tvg mode> (optional): valid values are: (all-0|all-1|vsi-h|cnt)

If the command is called without arguments the current TVG mode is displayed.

tvr_mode

Gets or sets the test vector receiver mode.

Arguments: [<tvr mode>]
<tvr mode> (optional): valid values are: (all-0|all-1|vsi-h|cnt)

If the command is called without arguments, the current TVR mode is displayed.

time

Displays the current time of the active 1PPS source.

Arguments: –

The displayed time is the (synchronized) VDIF time in UTC format.

timesync

Performs time synchronization to the active 1PPS source.

Arguments: [<YYYY>-<MM>-<DD>T<hh>:<mm>:<ss>[(+|-)<ZZ:ZZ>]]
<YYYY>: the current year in four digit representation (range: 2000..2130).
<MM>: the current month (range: 1..12)
<DD>: the current day (range: 1..31)
<hh>: the current hour (range: 0..23)
<mm>: the current minute (range: 1..59)
<ss>: the current second (range: 1..59)

(+|-)<ZZ:ZZ>: the time zone offset (only full hours are supported)
the offset is optional. if omitted offset 00:00 (=GMT) is the default.
(Example time: 2013-07-09T15:41:33+01:00)

The current UTC can be passed as an argument. If it is omitted, the GPS time will be taken for time synchronisation. For this to work a GPS device must be installed in the system.

A valid 1PPS signal is required for the time synchronization to succeed. If successful, both VDIF time and Mark5B time were synchronized by this command.

Note: if GPS time is used and “tvg” is the selected input, the 1PPS of the test vector generator will automatically be synchronizied to the 1PPS of the GPS device.

(Note: this command has a legacy mode, which allows it to be called with two arguments. If called with two arguments, its function is identical to former legacy command "mk5b_timesync". This should only be used with legacy batch scripts.)

mk5b_timesync

Performs Mark5B time synchronization to the active 1PPS source.

Arguments: <years since 2000> <modified julian day> <seconds>
<years since 2000>: the number of years since 2000 (=currentYear-2000)
<modified julian day>: the current modified julian day (only the last three digits are required)
<seconds>: the past seconds within the current modified julian day

A valid 1PPS signal is required for the time synchronization to succeed. If successful, the Mark5B time is synchronized with this command.

vdif_timesync

Performs VDIF time synchronization to the active 1PPS source.

Arguments: <half years since 2000> <seconds>
<half years since 2000>: the number of past half years since 2000 (only complete half years are counted)
<seconds>: the number of seconds since the beginning of the current half year

A valid 1PPS signal is required for the time synchronization to succeed. If successful, the VDIF time is synchronized with this command.

vdif_leapsecs

Gets/sets the number of UTC leap seconds since VDIF reference epoch.

Arguments: [<leap seconds>]
<leap seconds>: the number of additional leap seconds since VDIF reference epoch (optional)

Generally, the number of leap seconds since the start of the VDIF reference epoch is zero, since UTC leap seconds are inserted at the end of a half-year and the reference epoch is set to the beginning of the current half-year.

This command exists for the case in which a UTC leap second needs to be inserted between the beginning of the reference epoch and the current time. With this command the user can manually insert any number of missing leap seconds. Furthermore, a negative leap second number allows the removal of seconds relative to the start of epoch.

vdif_station

Gets/sets the VDIF station ID.

Arguments: [<VDIF station ID>]
<VDIF station ID> (optional): a two character VDIF station ID (cmp. VDIF standard)

Note: setting this value directly affects the header data of the VDIF data format.

vdif_legacy

Enables (on) or disables (off) VDIF legacy headers.

Arguments: (on|off)

Note: setting this value directly affects the header data of the VDIF data format.

vdif_frame

Gets or sets the properties of VDIF frames.

Arguments: [<channel bit width> <#channels per frame> [<payload size in bytes> [ct=(on|off)]]
<channel bit width> (optional): the size of each channel in bits (allowed values: 1,2,4,8,16,32,64)
<#channels per frame> (optional): number of channels per VDIF frame (allowed values: 1,2,4,8,16,32,64,128)
<payload size in bytes> (optional): the total payload size (= frame size without header data) of the VDIF frames
ct=(on|off) (optional): enables (ct=on) or disables (ct=off) corner tunring feature (default: ct=on)

All arguments are optional. If the command is called without arguments, the properties are not modified but only displayed.

If successful, the command displays the resulting number of frames per second and the number of data threads, according to the currently selected input. A warning is displayed instead if the VDIF frame properties do not match the currently selected input. The command fails if the desired frame setup is not supported (the frame setup is not changed in this case).

Note: after having set the VDIF frame properties and when processing has already been started, a reset of the datapath (i.e. "reset" or "reset keepsync") is mandatory.

Following rules apply to the properties:

1. The width of a data sample in a frame (<sample bit width>) equals to <#channels per frame> * <channel bit width> bits.
2. <sample bit width> must be 128 or less
3. If one of the properties is set and <payload size in bytes> is omitted, the optimal payload size is computed for the currently selected input.
4. Setting an "illegal" payload size is allowed but with the following restrictions:
   a) If a <payload size in bytes> is given that exceeds the maximum supported payload size for this configuration, the payload size is automatically capped to the maximum.
   b) The payload size is ensured to be a multiple of 8 bytes by truncation of the remaining bytes.
   c) If <payload size in bytes> will result in the last frame of a second interval to be incomplete, this frame is silently discarded.

See document “Fila10G VDIF Frame Setup Tables ct=on/off” for a list of supported frame setups.

For more information about VDIF frame formats refer to the VDIF standard.

Note: a channel bit width of 64 bit simple data is not directly supported by the VDIF standard, such that in this case complex data is assumed and the complex flag is asserted in the header.

Note: this command exists for user convenience. As an alternative to this command, all of the VDIF frame properties can be directly set by editing FiLa10G's VDIF header registers.

vdif_ct

Enables (on) or disables (off) the VDIF corner turning feature.

Arguments: (on|off)

Note: the “ct=on/off” flag is implicitly set by command “vdif_frame”. Only use “vdif_ct” if VDIF frame header registers are written directly by the user.

vdif_enc

Enables (on) or disables (off) the standard VDIF encoding of the input data.

Arguments: (on|off)

The standard setting is “on” , i.e. the VSI input data is encoded as mandated by the VDIF standard.
If set to “off” no VDIF encoding is applied to the VSI input data and it remains unmodified.

vdif_interval

Gets/sets the active frame ID interval within a 1PPS period.

Arguments: [<minimum frame ID> <maximum frame ID>]
<minimum frame ID> (optional): the minimum frame ID that is transmitted to the output.
allowed range: 0...4294967294
<maximum frame ID> (optional): the maximum frame ID that is transmitted to the output.
allowed range: 0...4294967295
"*" is a shortcut for maximal possible frame ID

Frames are filtered by their ID, such that only frame with an ID in the range <minimum frame ID> to <maximum frame ID> appear in the output stream. The others are completely discarded from the stream.

The frame index starts with 0 for the first VDIF frame in the second interval. The frame index increases with each following VDIF frame until it restarts again with 0 for the first frame of the next second interval.

Execute "vdif_interval 0 *" to reset the interval setting to its standard value.

Note: the frame ID interval setting applies to the VDIF data format only!

vdif_userdata

Gets or sets the current user data fields that are embedded in the VDIF frame header.

Arguments: [<d0> [<d1> [<d2> [<d3>]]]]
<d0> (optional): the value of the first user data field (unsigned integer in hex or decimal format)
<d1> (optional): the value of the second user data field (unsigned integer in hex or decimal format)
<d2> (optional): the value of the third user data field (unsigned integer in hex or decimal format)
<d3> (optional): the value of the fourth user data field (unsigned integer in hex or decimal format)

All arguments are optional. If the command is called without arguments, the values of the user data fields are not modified but only displayed.

For each argument that is specified, the given value will be written to the respective user data field.

Note: setting a user data field by the help ot this command is equivalent to writing directly into the respective vdif_header registers, but more comfortable.

destination

Gets/sets the output destination.

Arguments: <output index> [(<IPv4 address>[:<port>]|none) [<thread ID>]]
<output index>: the index of the output for which the destination is to be get or set.
allowed values: 0, or 1
<IPv4 address>(optional): IP address, format x.x.x.x
<port>(optional): IP port number
<thread ID>(optional): specifies the ID of the data thread for which the destination is to be set

Exceute the command in the form "destination <output index>" to get a list of all current destination settings for the respective output.

Executing the command in the form "destination <output index> <IPv4 address>:<port>" sets the destination for the respective output. All frames coming from this output are sent to the specified destination. This overwrites all other destination settings previously made for this output.

Executing the command in the form "destination <output index> none" effectively disables the respective output and no frames will be sent at all. This overwrites all other destination settings previously made for this output.

Executing the command in the form "destination <output index> <IPv4 address>:<port> <thread ID>" sets the destination only for the given thread ID at the repective output. Only frames from this thread are addressed to the given destination, other destination settings are not affected. This is useful for multi-threaded VDIF formats.
A thread can be disabled by setting its destination to "none" and no frames of this thread will be sent in this case.

tengbinfo

Retrieves the current parameters of a 10Gb Ethernet device.
Arguments: <device name>
<device name>: "eth0", "eth1", “eth2” or “eth3”

tengbcfg

Sets the parameters of a 10Gb Ethernet device.

Arguments: <device name> <tengbcfg parameters>
<device name>: "eth0", "eth1", “eth2” or “eth3”
<tengbcfg parameters>: the paramters to be set, example would be:
tengbcfg eth0 ip=192.168.1.28 gateway=192.168.1.1 nm=24
for setting ip-address, gateway and netmask for eth0
See TenGBE core documentation for further information.

tengbarp

Sets one ARP entry in a 10Gb Ethernet device.

Arguments: <device name> <ARP table index> <MAC address>
<device name>: "eth0", "eth1", “eth2” or “eth3”
<ARP table index>: index of ARP table entry to be modified
<MAC address>: MAC address to be set

arp

Enables (on) or disables (off) ARP queries on both Ethernet cores (eth0 and eth1).

Arguments: [(on|off)]

If arguments are omitted, the current arp status is displayed on the console.

start

Starts/restarts sending of (formatted) output data.

Arguments [(vdif|mk5b|raw|<X>+<Y>+...) [force]]
<X>+<Y>+...: applies format X to output0 and format Y to output1 and so on for all outputs
(the formats (X,Y,...) can be “vdif”, “mk5b” or “raw”)

The used output data format is either VDIF (“vdif”), Mark5B (“mk5b”) or Raw (=unformatted, “raw”) format. The format can be set the same for both outputs or, alternatively, to independent formats by using the <X>+<Y>+... notation (e.g. “start vdif+mk5b” which will make output0 produce VDIF and output1 produce Mark5B data).

Raw format requires no time synchronization.
Both VDIF and Mark5B format require the respective timer to be synchronized (see "timesync" / "mk5b_timesync" / "vdif_timesync" commands).

For fast testing: append argument "force" to the command to automatically synchronize timer to “zero” time (=“2000-01-01T00:00:00”). Provided that a valid 1PPS signal is available, this will always be succeessful.

stop

Stops sending of output data (the opposite of "start").

Arguments: –

tick

Enables continuous 1PPS tick display on the console.

Arguments: –

The VDIF time must be synchronized before using this command.
When enabled the current time is displayed every second (synchronized with 1PPS source).
Press <Space> or <Enter> key (or almost any other key) to leave the tick mode.

By pressing “+” or “-” the time can be adjusted by +1s or -1s, respectively. The total adjustment range is -60 to +60 seconds.

test

Enables continuous TVR test result display on the console.

Arguments: –

Only the lower 32 bit (→ one VSI channel) of the input stream are tested.

When enabled the test report for the last second interval is displayed every second (synchronized with 1PPS source). The current time displayed, too, every second.
Press <Space> or <Enter> key (or almost any other key) to leave the test mode.

output

Displays output debug information.

Arguments: [<output index> [<frame ID> [<duration>]]]
<output index>: 0, 1, 2, 3 (default: 0)
<frame ID>: Index of frame within current second starting with 0. (default: 0)
    The index restarts with 0 again at 1PPS.
    All frames are indexed irrespectively of the output
<duration>: Duration of debug sequence (in seconds). (default: 1)

All arguments are optional.

The command displays parts of FiLa10g's output stream on the console.

The command displays the first 16 words of the first frame that was sent within the current second interval. The frame is recorded at the output with the given index.
1PPS/frame-header/end-of-frame/frame-drop bits attached to the stream are displayed together with the data.

If duration is given, a new frame is displayed every second. The debug sequence can be cancelled by pressing <Space> or <Enter> key (or almost any other key).

reset

Resets FiLa10G's datapath and erases synchronized time.

Arguments: [keepsync]

If "reset" is called without arguments, the complete datapath is reset and time synchronization is lost.

If "reset" is called with argument "keepsync", FiLa10G tries to maintain the current time synchronization. For this to work the input stage of the data path and the timers are not reset. Time synchronization will not be correct anymore in the rare but possible case that a data sample with 1PPS flag is lost during the reset process.

sysclk

[hidden command]
Gets or sets the system clock (frequency in MHz).

Arguments: [(64|66|133)]

Note: initially, after boot process, the system clock is 133 MHz.

The internal processing pipeline is clocked by the system clock. Adjusting the system clock is required in some cases.

Note: the system clock is automatically switched to 133 MHz when vsi1-2-3-4 is selected as current input to allow for a data rate of 8Gbps.

Warning: if not used correctly, over-/underclocking can lead to data loss!

reboot

Reboots the system.

Arguments: –

FiLa10G's hardware and software is reset to its initial state, i.e. as it was directly after the programming of the FPGA, and lets the FiLa10G system boot again.

Warning: all previously configured settings and states are lost when rebooting!

core3_init

Initializes FiLa10G for the CORE3 system.

Arguments: [(independent|[half_]merged|pfb)]

core3_mode

gets/sets the CORE3 mode

Arguments: [(independent|[half_]merged|pfb)]

core3_bitselect

Note: deprecated! - replaced by threshold selection

selects the second IF input bit

Arguments: (6|7|8)

core3_bstat

Displays the CORE3 input bit statistics for the selected Sampler.

Arguments: (0|1|2|3)

core3_power

Displays CORE3 power statistics of the samplers

Arguments: –

core3_corr

Displays CORE3 correlation results between samplers 0-1, 1-2, 2-3

Arguments: --