

SBI VVC – Quick Reference

For general information see UVVM VVC Framework Essential Mechanisms located in `uvvm_vvc_framework/doc`. **CAUTION:** shaded `code/description` is preliminary.

sbi_write (VVCT, vvc_instance_idx, addr, data | { num_words, randomisation}, msg, [scope])

Example: `sbi_write(SBI_VVCT, 1, x"1000", x"40", "Set baud rate to 9600");`
`sbi_write(SBI_VVCT, 1, x"1001", 7, RANDOM, "Write 7 random bytes to UART TX");`

sbi_read (VVCT, vvc_instance_idx, addr, [TO_SB,] msg, [scope])

Example: `sbi_read(SBI_VVCT, 1, x"1000", "Read baud rate");`
`sbi_read(SBI_VVCT, 1, x"1002", TO_SB, "Read UART RX and send to Scoreboard", C_SCOPE);`

sbi_check (VVCT, vvc_instance_idx, addr, data, msg, [alert_level, [scope]])

Example: `sbi_check(SBI_VVCT, 1, x"1155, x"3B", "Check data from UART RX");`

sbi_poll_until (VVCT, vvc_instance_idx, addr, data, msg, [max_polls, [timeout, [alert_level, [scope]]]])

Example: `sbi_poll_until(SBI_VVCT, 1, x"1155", x"0D", "Read UART RX until CR is found");`

SBI VVC Configuration record '**vvc_config**' -- accessible via **shared_sbi_vvc_config**

Record element	Type	C_SBI_VVC_CONFIG_DEFAULT
inter_bfm_delay	t_inter_bfm_delay	C_SBI_INTER_BFM_DELAY_DEFAULT
cmd_queue_count_max	natural	C_CMD_QUEUE_COUNT_MAX
cmd_queue_count_threshold	natural	C_CMD_QUEUE_COUNT_THRESHOLD
cmd_queue_count_threshold_severity	t_alert_level	C_CMD_QUEUE_COUNT_THRESHOLD_SEVERITY
result_queue_count_max	natural	C_RESULT_QUEUE_COUNT_MAX
result_queue_count_threshold	natural	C_RESULT_QUEUE_COUNT_THRESHOLD
result_queue_count_threshold_severity	t_alert_level	C_RESULT_QUEUE_COUNT_THRESHOLD_SEVERITY
bfm_config	t_sbi_bfm_config	C_SBI_BFM_CONFIG_DEFAULT
msg_id_panel	t_msg_id_panel	C_VVC_MSG_ID_PANEL_DEFAULT

SBI VVC Status record signal '**vvc_status**' -- accessible via **shared_sbi_vvc_status**

Record element	Type
current_cmd_idx	natural
previous_cmd_idx	natural
pending_cmd_cnt	natural

Common VVC procedures applicable for this VVC

- See UVVM Methods QuickRef for details.

await_completion()

enable_log_msg()

disable_log_msg()

fetch_result()

flush_command_queue()

terminate_current_command()

terminate_all_commands()

insert_delay()

get_last_received_cmd_idx()



sbi_vvc.vhd



UVVM™

VHDL 2008 only

VVC target parameters

Name	Type	Example(s)	Description
VVCT	t_vvc_target_record	SBI_VVCT	VVC target type compiled into each VVC in order to differentiate between VVCs.
vvc_instance_idx	integer	1	Instance number of the VVC

VVC functional parameters

Name	Type	Example(s)	Description
addr	unsigned	x"5A"	The address of a SW accessible register. Could be offset or full address depending on the DUT
data	std_logic_vector	x"D3"	The data to be written (in sbi_write) or the expected data (in sbi_check/sbi_poll_until).
msg	string	"Read from DUT"	A custom message to be appended in the log/alert
timeout	time	100 ns	Timeout to be used in the sbi_poll_until BFM procedure. 0 ns means no timeout.
max_polls	integer	1	Maximum number of polls allowed in the sbi_poll_until procedure. 0 means no limit.
alert_level	t_alert_level	ERROR or TB_WARNING	Set the severity for the alert that may be asserted by the procedure.
scope	string	"SBI VVC"	A string describing the scope from which the log/alert originates. In a simple single sequencer typically "SBI BFM". In a verification component typically "SBI VVC".

VVC entity signals

Name	Type	Direction	Description
clk	std_logic	Input	VVC Clock signal
sbi_vvd_master_if	t_sbi_if	Inout	See SBI BFM documentation

VVC entity generic constants

Name	Type	Default	Description
GC_ADDR_WIDTH	integer	8	Width of the SBI address bus
GC_DATA_WIDTH	integer	32	Width of the SBI data bus
GC_INSTANCE_IDX	natural	1	Instance number to assign the VVC
GC_SBI_CONFIG	t_sbi_bfm_config	C_SBI_BFM_CONFIG_DEFAULT	Configuration for the SBI BFM, see SBI BFM documentation.
GC_CMD_QUEUE_COUNT_MAX	natural	1000	Absolute maximum number of commands in the VVC command queue
GC_CMD_QUEUE_COUNT_THRESHOLD	natural	950	An alert will be generated when reaching this threshold to indicate that the command queue is almost full. The queue will still accept new commands until it reaches C_CMD_QUEUE_COUNT_MAX.
GC_CMD_QUEUE_COUNT_THRESHOLD_SEVERITY	t_alert_level	WARNING	Alert severity which will be used when command queue reaches GC_CMD_QUEUE_COUNT_THRESHOLD.
GC_RESULT_QUEUE_COUNT_MAX	natural	1000	Maximum number of unfetched results before result_queue is full.
GC_RESULT_QUEUE_COUNT_THRESHOLD	natural	950	An alert with severity 'result_queue_count_threshold_severity' will be issued if result queue exceeds this count. Used for early warning if result queue is almost full. Will be ignored if set to 0.
GC_RESULT_QUEUE_COUNT_THRESHOLD_SEVERITY	t_alert_level	WARNING	Severity of alert to be initiated if exceeding result_queue_count_threshold

VVC details

All VVC procedures are defined in `vvc_methods_pkg` (dedicated this VVC), and `uvvm_vvc_framework.td_vvc_framework_common_methods_pkg` (common VVC procedures)

It is also possible to send a multicast to all instances of a VVC with `ALL_INSTANCES` as parameter for `vvc_instance_idx`.

Note: Every procedure here can be called without the optional parameters enclosed in [].

1 VVC procedure details and examples

Procedure	Description
sbi_write()	<p>sbi_write(VVCT, vvc_instance_idx, addr, data { num_words, randomisation }, msg, [scope])</p> <p>The <code>sbi_write()</code> VVC procedure adds a write command to the SBI VVC executor queue, which will run as soon as all preceding commands have completed. The <code>sbi_write()</code> command has two variants using either just data for a basic single transaction, or <code>num_words</code> + <code>randomisation</code> for a more advanced version. When the basic write command is scheduled to run, the executor calls the SBI BFM <code>sbi_write()</code> procedure, described in the SBI BFM QuickRef.</p> <p>When the more advanced randomisation command is applied the basic BFM <code>sbi_write()</code> transaction is executed <code>num_words</code> times with new random data each time – according to the given randomisation profile.</p> <p>Current defined randomisation profiles are: <code>RANDOM</code>: Standard uniform random. This is provided as an example.</p> <p>Example:</p> <pre>sbi_write(SBI_VVCT, 1, x"1000", x"40", "Set UART baud rate to 9600", C_SCOPE); sbi_write(SBI_VVCT, 1, x"1001", 7, RANDOM, "Write 7 random bytes to UART TX");</pre> <p>It is recommended to use constants to improve the readability of the code, e.g.:</p> <pre>sbi_write(SBI_VVCT, 1, C_ADDR_UART_BAUDRATE, C_BAUDRATE_9600, "Set UART baud rate to 9600");</pre>

sbi_read()

sbi_read (VVCT, vvc_instance_idx, addr, [TO_SB,] msg, [scope])

The `sbi_read()` VVC procedure adds a read command to the SBI VVC executor queue, which will run as soon as all preceding commands have completed. When the read command is scheduled to run, the executor calls the SBI BFM `sbi_read()` procedure, described in the SBI BFM QuickRef.

The value read from DUT will not be returned in this procedure call since it is non-blocking for the sequencer/caller, but the read data will be stored in the VVC for a potential future fetch (see example with *fetch_result* below).

If the option `TO_SB` is applied the read data will be sent to the SBI_VVC dedicated scoreboard where it will be checked against the expected value (provided by the testbench)

Example:

```
sbi_read(SBI_VVCT, 1, x"1000", "Read UART baud rate", C_SCOPE);
sbi_read(SBI_VVCT, 1, x"1002", TO_SB, "Read UART RX and send to Scoreboard", C_SCOPE);
```

It is recommended to use constants to improve the readability of the code, e.g.:

```
sbi_read(SBI_VVCT, 1, C_ADDR_UART_BAUDRATE, "Read UART baud rate");
```

Example with `fetch_result()` call: Result is placed in `v_data`

```
variable v_cmd_idx      : natural;      -- Command index for the last read
variable v_data          : work.vvc_cmd_pkg.t_vvc_result; -- Result from read.
(...)
sbi_read(SBI_VVCT, 1, C_ADDR_UART_BAUDRATE, "Read from Peripheral 1");
v_cmd_idx := get_last_received_cmd_idx(SBI_VVCT, 1);
await_completion(SBI_VVCT, 1, v_cmd_idx, 1 us, "Wait for read to finish");
fetch_result(SBI_VVCT, 1, v_cmd_idx, v_data, "Fetching result from read operation");
```

sbi_check()

sbi_check (VVCT, vvc_instance_idx, addr, data, msg, [alert_level, [scope]])

The `sbi_check()` VVC procedure adds a check command to the SBI VVC executor queue, which will run as soon as all preceding commands have completed. When the check command is scheduled to run, the executor calls the SBI BFM `sbi_check()` procedure, described in the SBI BFM QuickRef. The `sbi_check()` procedure will perform a read operation, then check if the read data is equal to the expected data in the 'data' parameter. If the read data is not equal to the expected 'data' parameter, an alert with severity 'alert_level' will be issued. The read data will not be stored in this procedure.

Examples:

```
sbi_check(SBI_VVCT, 1, x"1155, x"3B", "Check data from UART RX");
sbi_check(SBI_VVCT, 1, x"1155, x"3B", "Check data from UART RX", TB_ERROR, C_SCOPE);
```

It is recommended to use constants to improve the readability of the code, e.g.:

```
sbi_check(SBI_VVCT, 1, C_ADDR_UART_RX, C_UART_START_BYTE, "Check data from UART RX");
```

sbi_poll_until()

sbi_poll_until (VVCT, vvc_instance_idx, addr, data, msg, [max_polls, [timeout, [alert_level, [scope]]]])

The `sbi_poll_until()` VVC procedure adds a `poll_until` command to the SBI VVC executor queue, which will run as soon as all preceding commands have completed. When the write command is scheduled to run, the executor calls the SBI BFM `sbi_poll_until()` procedure, described in the SBI BFM QuickRef. The `sbi_poll_until()` procedure will perform a read operation, then check if the read data is equal to the data in the 'data' parameter. If the read data is not equal to the expected 'data' parameter, the process will be repeated until the read data is equal to the expected data, or the procedure is terminated by either a terminate command, a timeout or the poll limit set in `max_polls`. The read data will not be stored by this procedure.

Examples:

```
sbi_poll_until(SBI_VVCT, 1, x"1155", x"0D", "Read UART RX until CR is found");
sbi_poll_until(SBI_VVCT, 1, x"1155", x"0D", "Read UART RX until CR is found", 5, 0 ns, TB_WARNING, C_SCOPE);
```

It is recommended to use constants to improve the readability of the code, e.g.:

```
sbi_poll_until(SBI_VVCT, 1, C_ADDR_UART_RX, C_CR_BYTE, "Read UART RX until CR is found");
```

2 VVC Configuration

Record element	Type	C_SBI_BFM_CONFIG_DEFAULT	Description
inter_bfm_delay	t_inter_bfm_delay	C_SBI_INTER_BFM_DELAY_DEFAULT	Delay between any requested BFM accesses towards the DUT. - TIME_START2START: Time from a BFM start to the next BFM start (A TB_WARNING will be issued if access takes longer than TIME_START2START). - TIME_FINISH2START: Time from a BFM end to the next BFM start. Any insert_delay() command will add to the above minimum delays, giving for instance the ability to skew the BFM starting time.
cmd_queue_count_max	natural	C_CMD_QUEUE_COUNT_MAX	Maximum pending number in command queue before queue is full. Adding additional commands will result in an ERROR.
cmd_queue_count_threshold	natural	C_CMD_QUEUE_COUNT_THRESHOLD	An alert with severity "cmd_queue_count_threshold_severity" will be issued if command queue exceeds this count. Used for early warning if command queue is almost full. Will be ignored if set to 0.
cmd_queue_count_threshold_severity	t_alert_level	C_CMD_QUEUE_COUNT_THRESHOLD_SEVERITY	Severity of alert to be initiated if exceeding cmd_queue_count_threshold
result_queue_count_max	natural	C_RESULT_QUEUE_COUNT_MAX	Maximum number of unfetched results before result_queue is full.
result_queue_count_threshold	natural	C_RESULT_QUEUE_COUNT_THRESHOLD	An alert with severity 'result_queue_count_threshold_severity' will be issued if result queue exceeds this count. Used for early warning if result queue is almost full. Will be ignored if set to 0.
result_queue_count_threshold_severity	t_alert_level	C_RESULT_QUEUE_COUNT_THRESHOLD_SEVERITY	Severity of alert to be initiated if exceeding result_queue_count_threshold
bfm_config	t_sbi_bfm_config	C_SBI_BFM_CONFIG_DEFAULT	Configuration for SBI BFM. See quick reference for SBI BFM
msg_id_panel	t_msg_id_panel	C_VVC_MSG_ID_PANEL_DEFAULT	VVC dedicated message ID panel

The configuration record can be accessed from the Central Testbench Sequencer through the shared variable array, e.g.:

```
shared_sbi_vvc_config(1).inter_bfm_delay.delay_in_time := 50 ns;
shared_sbi_vvc_config(1).bfm_config.id_for_bfm         := ID_BFM;
```

3 VVC Status

The current status of the VVC can be retrieved during simulation. This is achieved by reading from the shared variable `shared_sbi_vvc_status` record from the test sequencer. The record contents can be seen below:

Record element	Type	Description
<code>current_cmd_idx</code>	natural	Command index currently running
<code>previous_cmd_idx</code>	natural	Previous command index to run
<code>pending_cmd_cnt</code>	natural	Pending number of commands in the command queue

4 Activity watchdog

The VVCs support an activity watchdog which monitors VVC activity and will alert if no VVC activity is registered within a selected timeout value. The VVCs will register their presence to the activity watchdog at start-up, and report when busy and not, using dedicated activity watchdog methods and triggering the `global_trigger_activity_watchdog` signal, during simulations.

Include `activity_watchdog(num_exp_vvc, timeout, alert_level, msg)` in the testbench to start using the activity watchdog. More information can be found in UVVM Essential Mechanisms PDF in the UVVM VVC Framework doc folder.

5 VVC Interface

In this VVC, the interface has been encapsulated in a signal record of type `t_sbi_if` in order to improve readability of the code. Since the SBI interface busses can be of arbitrary size, the interface vectors have been left unconstrained. These unconstrained vectors need to be constrained when the interface signals are instantiated. For this interface, it could look like:

```
signal sbi_if_1 : t_sbi_if( addr (C_ADDR_WIDTH-1 downto 0),
                           wdata(C_DATA_WIDTH-1 downto 0),
                           rdata(C_DATA_WIDTH-1 downto 0) );
```

6 Additional Documentation

Additional documentation about UVVM and its features can be found under `“/uvvm_vvc_framework/doc/”`. For additional documentation on the SBI protocol, please see the SBI BFM QuickRef.

7 Compilation

The SBI VVC must be compiled with VHDL 2008.

It is dependent on the following libraries

- **UVVM Utility Library (UVVM-Util), version 2.11.0 and up**
- **UVVM VVC Framework, version 2.7.1 and up**
- **SBI BFM**
- **Bitvis VIP Scoreboard**

Before compiling the SBI VVC, assure that `uvvm_vvc_framework`, `uvvm_util` and `bitvis_vip_scoreboard` have been compiled.

See UVVM Essential Mechanisms located in `uvvm_vvc_framework/doc` for information about compile scripts.

Compile order for the SBI VVC:

Compile to library	File	Comment
bitvis_vip_sbi	sbi_bfm_pkg.vhd	SBI BFM
bitvis_vip_sbi	transaction_pkg.vhd	SBI transaction package with DTT types, constants etc.
bitvis_vip_sbi	vvc_cmd_pkg.vhd	SBI VVC command types and operations
bitvis_vip_sbi	../uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd	UVVM VVC target support package, compiled into the SBI VVC library.
bitvis_vip_sbi	../uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd	Common UVVM framework methods compiled into the SBI VVC library
bitvis_vip_sbi	vvc_methods_pkg.vhd	SBI VVC methods
bitvis_vip_sbi	../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd	UVVM queue package for the VVC
bitvis_vip_sbi	../uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd	UVVM VVC entity support compiled into the SBI VVC library
bitvis_vip_sbi	sbi_vvc.vhd	SBI VVC

8 Simulator compatibility and setup

See README.md for a list of supported simulators.

For required simulator setup see **UVVM-Util** Quick reference.

IMPORTANT

This is a simplified Verification IP (VIP) for SBI.

The given VIP complies with the basic SBI protocol and thus allows a normal access towards a SBI interface. This VIP is not a SBI protocol checker.

For a more advanced VIP please contact Bitvis AS at support@bitvis.no

INTELLECTUAL PROPERTY

Disclaimer: This IP and any part thereof are provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with this IP.