

GPIO VVC – Quick Reference

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

gpio_set (VVCT, vvc_instance_idx, data, msg, [scope])

Example: `gpio_set(GPIO_VVCT, 1, C_BAUDRATE_9600, "Set baudrate to 9600");`

gpio_get (VVCT, vvc_instance_idx, [TO_SB,] msg, [scope])

Example: `gpio_get(GPIO_VVCT, 1, "Read GPIO baudrate, and store result in VVC. To be retrieved using fetch_result()");`
`gpio_get(GPIO_VVCT, 1, TO_SB, "Read GPIO baudrate and send result to scoreboard for checking");`

gpio_check (VVCT, vvc_instance_idx, data_exp, msg, [alert_level, [scope]])

Example: `gpio_check(GPIO_VVCT, 1, x"3B", "Check data from UART RX", ERROR);`

gpi_expect (VVCT, vvc_instance_idx, data_exp, timeout, msg, [alert_level, [scope]])

Example: `gpi_expect(GPIO_VVCT, 1, x"0D", 2 ms, "Read UART RX until CR is found or timeout", ERROR);`

VVC



gpio_vvc.vhd

Common VVC procedures applicable for this VVC

- See UVVM Methods QuickRef for details.

Name

`await_completion()`
`await_any_completion()`
`enable_log_msg()`
`disable_log_msg()`
`flush_command_queue()`
`terminate_current_command()`
`fetch_result()`
`insert_delay()`
`get_last_received_cmd_idx()`
`terminate_current_command()`

GPIO VVC Configuration record 't_vvc_config'

- Accessible via `shared_gpio_vvc_config` – se section 2.

Record element

`inter_bfm_delay`
`[cmd/result]_queue_count_max`
`[cmd/result]_queue_count_threshold`
`[cmd/result]_queue_count_threshold_severity`
`result_queue_count_max`
`result_queue_count_threshold_severity`
`result_queue_count_threshold`
`bfm_config`
`msg_id_panel`

GPIO VVC Status record signal 't_vvc_status'

- Accessible via `shared_gpio_vvc_status` – se section 3.

Record element

`current_cmd_idx`
`previous_cmd_idx`
`pending_cmd_idx`



UVVM™

VVC target parameters

Name	Type	Example(s)	Description
VVCT	t_vvc_target_record	GPIO_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
data	std_logic_vector	x"FF"	The data to be written (in gpio_set).
data_exp	std_logic_vector	x"FF"	The expected data to be read (in gpio_check or gpio_expect).
msg	string	"Set baudrate"	A custom message to be appended in the log/alert
alert_level	t_alert_level	ERROR or TB_WARNING	Set the severity for the alert that may be asserted by the method.
scope	string	"GPIO VVC"	A string describing the scope from which the log/alert originates. In a simple single sequencer typically "GPIO BFM". In a verification component typically "GPIO VVC".

VVC entity signals

Name	Type	Direction	Description
gpio_vvc_if	t_gpio_if	Inout	See GPIO BFM documentation

VVC entity generic constants

Name	Type	Default	Description
GC_DATA_WIDTH	natural	32	Bits in the GPIO data word
GC_INSTANCE_IDX	natural	1	Instance number to assign the VVC
GC_DEFAULT_LINE_VALUE	boolean	TRUE	Default value of input or output GPIO.
GC_GPIO_BFM_CONFIG	t_gpio_bfm_config	C_GPIO_BFM_CONFIG_DEFAULT	Configuration for the GPIO BFM, see GPIO 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
gpio_set()	<p>gpio_set (VVCT, vvc_instance_idx, data, msg, [scope])</p> <p>The <code>gpio_set()</code> VVC procedure adds a SET command to the GPIO VVC executor queue, that will run as soon as all preceding commands have completed. When the SET command is scheduled to run, the executor calls the GPIO BFM <code>gpio_set()</code> procedure, described in the GPIO BFM QuickRef.</p> <p>Example:</p> <pre>gpio_set (GPIO_VVCT, 1, C_BAUDRATE_9600, "Set baudrate to 9600", C_SCOPE);</pre>
gpio_get()	<p>gpio_get (VVCT, vvc_instance_idx, [TO_SB,] msg, [scope])</p> <p>The <code>gpio_get()</code> VVC procedure adds a GET command to the GPIO VVC executor queue, that will run as soon as all preceding commands have completed. When the transmit command is scheduled to run, the executor calls the GPIO BFM <code>gpio_get()</code> procedure, described in the GPIO BFM QuickRef. The received data from DUT will not be returned in this procedure call since it is non-blocking for the sequencer/caller, but the received data will be stored in the VVC for a potential future fetch (see example with <code>fetch_result</code> below).</p> <p>If the option <code>TO_SB</code> is applied, the received data will be sent to the GPIO VVC dedicated scoreboard. There, it is checked against the expected value (provided by the testbench).</p> <p>Example:</p> <pre>gpio_get (GPIO_VVCT, 1, "Read baudrate", C_SCOPE);</pre> <p>Example with <code>fetch_result()</code> call: Result is placed in <code>v_data</code></p> <pre>variable v_cmd_idx : natural; -- Command index for the last read variable v_data : std_logic_vector(31 downto 0); -- Result from read (...) gpio_get(GPIO_VVCT, 1, "Read baudrate"); v_cmd_idx := get_last_received_cmd_idx(GPIO_VVCT, 1); await_completion(GPIO_VVCT,1, v_cmd_idx, 1 us, "Wait for receive to finish"); fetch_result(GPIO_VVCT,1, v_cmd_idx, v_data, "Fetching result from receive operation");</pre>
gpio_check()	<p>gpio_check (VVCT, vvc_instance_idx, data_exp, msg, [alert_level, [scope]])</p> <p>The <code>gpio_check()</code> VVC procedure adds a CHECK command to the GPIO VVC executor queue, that will run as soon as all preceding commands have completed. When the CHECK command is scheduled to run, the executor calls the GPIO BFM <code>gpio_check()</code> procedure, described in the GPIO BFM QuickRef.</p> <p>Example:</p> <pre>gpio_check (GPIO_VVCT, 1, x"F5", "Check data from UART RX", ERROR, C_SCOPE);</pre>

gpio_expect()

gpio_expect (VVCT, vvc_instance_idx, data_exp, timeout, msg, [alert_level, [scope]])

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

Example:

```
gpio_expect(GPIO_VVCT, 1, x"0D", 2 ms, "Read UART RX until CR is found or timeout", ERROR, C_SCOPE);
```

2 VVC Configuration

Record element	Type	C_SPI_VVC_CONFIG_DEFAULT	Description
inter_bfm_delay	t_inter_bfm_delay	C_GPIO_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 triggered if command count 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_gpio_bfm_config	C_GPIO_BFM_CONFIG_DEFAULT	Configuration for GPIO BFM. See QuickRef for GPIO BFM
msg_id_panel	t_msg_id_panel	C_VVC_MSG_ID_PANEL_DEFAULT	VVC dedicated message ID panel. See section 16 of uvvm_vvc_framework/doc/UVVM_VVC_Framework_Essential_Mechanisms.pdf for how to use verbosity control.

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

```
shared_gpio_vvc_config(C_VVC_IDX).inter_bfm_delay.delay_in_time := 10 ms;
```

3 VVC Status

The current status of the VVC can be retrieved during simulation. This is done by reading from the shared variable `shared_spi_vvc_status` record from the test sequencer. The record contains status for both channels, specified with the channel axis of the shared `spi_vvc_status` array. 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 Transaction Info

This VVC supports transaction info, a UVVM concept for distributing transaction information in a controlled manner within the complete testbench environment. The transaction info may be used in many different ways, but the main purpose is to share information directly from the VVC to a DUT model.

Table 4.1 GPIO transaction info record fields. Transaction type: `t_base_transaction` (BT).

Info field	Type	Default	Description
<code>operation</code>	<code>t_operation</code>	<code>NO_OPERATION</code>	Current VVC operation, e.g. <code>INSERT_DELAY</code> , <code>POLL_UNTIL</code> , <code>READ</code> , <code>WRITE</code> .
<code>data</code>	<code>slv(31 downto 0)</code>	<code>0x0</code>	The data to be written (in <code>gpio_set</code>).
<code>data_exp</code>	<code>slv(31 downto 0)</code>	<code>0x0</code>	The expected data to be read (in <code>gpio_check</code> or <code>gpio_expect</code>).
<code>vvc_meta</code>	<code>t_vvc_meta</code>	<code>C_VVC_META_DEFAULT</code>	VVC meta data of the executing VVC command.
→ <code>msg</code>	string	" "	Message of executing VVC command.
→ <code>cmd_idx</code>	integer	-1	Command index of executing VVC command.
<code>transaction_status</code>	<code>t_transaction_status</code>	<code>C_TRANSACTION_STATUS_DEFAULT</code>	Set to <code>INACTIVE</code> , <code>IN_PROGRESS</code> , <code>FAILED</code> or <code>SUCCEEDED</code> during a transaction.

See UVVM VVC Framework Essential Mechanisms PDF, section 6, for additional information about transaction types and transaction info usage.

5 Scoreboard

This VVC has built in Scoreboard functionality where data can be routed by setting the `T0_SB` parameter in supported method calls, i.e. `gpio_get()`. Note that the data is only stored in the scoreboard and not accessible with the `fetch_result()` method when the `T0_SB` parameter is applied.

The GPIO scoreboard is per default a 128 bits wide standard logic vector. When sending expected data to the scoreboard, where the data width is smaller than the default scoreboard width, we recommend zero-padding the data with the `pad_sb_slv()` function. I.e. `GPIO_VVC_SB.add_expected(<GPIO VVC instance number>, pad_sb_slv(<exp data>))`;

See the Generic Scoreboard Quick Reference PDF in the Bitvis VIP Scoreboard document folder for a complete list of available commands and additional information. The GPIO VVC scoreboard is accessible from the testbench as a shared variable `GPIO_VVC_SB`, located in the `vvc_methods_pkg.vhd`. All of the listed Generic Scoreboard commands are available for the GPIO VVC scoreboard using this shared variable.

6 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.

7 Additional Documentation

Additional documentation about UVVM and its features can be found under `"/uvvm_vvc_framework/doc/".`

8 Compilation

The GPIO VVC must be compiled with VHDL 2008.

It is dependent on the following libraries

- **UVVM Utility Library (UVVM-Util), version 2.13.0 and up**
- **UVVM VVC Framework, version 2.8.0 and up**
- **GPIO BFM**
- **Bitvis VIP Scoreboard**

Before compiling the GPIO VVC, make sure 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 GPIO VVC:

Compile to library	File	Comment
<code>bitvis_vip_gpio</code>	<code>gpio_bfm_pkg.vhd</code>	GPIO BFM
<code>bitvis_vip_gpio</code>	<code>transaction_pkg.vhd</code>	GPIO transaction package with DTT types, constants etc.
<code>bitvis_vip_gpio</code>	<code>vvc_cmd_pkg.vhd</code>	GPIO VVC command types and operations
<code>bitvis_vip_gpio</code>	<code>../uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd</code>	UVVM VVC target support package, compiled into the GPIO VVC library.
<code>bitvis_vip_gpio</code>	<code>../uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd</code>	UVVM framework common methods compiled into the GPIO VVC library
<code>bitvis_vip_gpio</code>	<code>vvc_methods_pkg.vhd</code>	GPIO VVC methods
<code>bitvis_vip_gpio</code>	<code>../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd</code>	UVVM queue package for the VVC
<code>bitvis_vip_gpio</code>	<code>../uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd</code>	UVVM VVC entity methods compiled into the GPIO VVC library
<code>bitvis_vip_gpio</code>	<code>gpio_vvc.vhd</code>	GPIO VVC

9 Simulator compatibility and setup

See README.md for a list of supported simulators.

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

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.