**GPIO VVC** –Quick Reference

**VVC**

For general information see UVVM Essential Mechanisms located in uvvm\_vvc\_framework/doc.

|  |
| --- |
| gpio\_set (VVCT, vvc\_instance\_idx, data, msg, [scope]) |
| Example: gpio\_set(GPIO\_VVCT, 1, C\_BAUDRATE\_9600, “Set baudrate to 9600”); |

*gpio\_vvc.vhd*

|  |
| --- |
| gpio\_get (VVCT, vvc\_instance\_idx, msg, [scope]) |
| Example: gpio\_get(GPIO\_VVCT, 1, “Read baudrate”); |

|  |
| --- |
| 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: gpio\_expect(GPIO\_VVCT, 1, x"0D", 2 ms, “Read UART RX until CR is found or timeout”, ERROR); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Common VVC procedures applicable for this VVC** - See UVVM Methods QuickRef for details. |  | **GPIO VVC Configuration record ‘t\_vvc\_config’** - Accessible via **shared\_gpio\_vvc\_config** – se section 2. |  | **GPIO VVC Status record signal ‘t\_vvc\_status’** - Accessible via **shared\_gpio\_vvc\_status** – se section 3. |
| **Name** |  | **Record element** |  | **Record element** |
| await\_completion() |  | inter\_bfm\_delay |  | current\_cmd\_idx |
| await\_any\_completion() |  | [cmd/result]\_queue\_count\_max |  | previous\_cmd\_idx |
| enable\_log\_msg() |  | [cmd/result]\_queue\_count\_threshold |  | pending\_cmd\_idx |
| disable\_log\_msg() |  | [cmd/result]\_queue\_count\_threshold\_severity |  |  |
| flush\_command\_queue() |  | result\_queue\_count\_max |  |  |
| terminate\_current\_command() |  | result\_queue\_count\_threshold\_severity |  |  |
| fetch\_result() |  | result\_queue\_count\_threshold |  |  |
| insert\_delay() |  | bfm\_config |  |  |
| get\_last\_received\_cmd\_idx() |  | msg\_id\_panel |  |  |
| terminate\_current\_command() |  |  |  |  |



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 [ ].*

# VVC procedure details and examples

|  |  |
| --- | --- |
| **Procedure** | **Description** |
| **gpio\_set()** | **gpio\_set (VVCT, vvc\_instance\_idx, data, msg, [scope])**  The gpio\_set() 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 gpio\_set() procedure, described in the GPIO BFM QuickRef.  Example:  gpio\_set (GPIO\_VVCT, 1, C\_BAUDRATE\_9600, “Set baudrate to 9600”, C\_SCOPE); |
| **gpio\_get()** | **gpio\_get (VVCT, vvc\_instance\_idx, msg, [scope])**  The gpio\_get() 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 gpio\_get() 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 fetch\_result below).  Example:  gpio\_get (GPI\_VVCT, 1, “Read baudrate”, C\_SCOPE);  **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 : 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"); |
| **gpio\_check()** | **gpio\_check (VVCT, vvc\_instance\_idx, data\_exp, msg, [alert\_level, [scope]])**  The gpio\_check() 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 gpio\_check() procedure, described in the GPIO BFM QuickRef.  Example:  gpio\_check (GPIO\_VVCT, 1, x”F5”, “Check data from UART RX”, ERROR, C\_SCOPE); |
| **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); |

# 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 |

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;

# 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** |
| current\_cmd\_idx | natural | Command index currently running |
| previous\_cmd\_idx | natural | Previous command index to run |
| pending\_cmd\_cnt | natural | Pending number of commands in the command queue |

# 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\_testcase\_inactivity\_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.

# Additional Documentation

Additional documentation about UVVM and its features can be found under “/uvvm\_vvc\_framework/doc/”.

# Compilation

The GPOI VVC must be compiled with VHDL 2008.   
It is dependent on the following libraries

* ***UVVM Utility Library (UVVM-Util), version 2.10.0 and up***
* ***UVVM VVC Framework, version 2.6.0 and up***
* ***GPIO BFM***
* ***Bitvis VIP Scoreboard***

Before compiling the GPIO VVC, make sure that uvvm\_vvc\_framework and uvvm\_util 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** |
| bitvis\_vip\_gpio | gpio\_bfm\_pkg.vhd | GPIO BFM |
| bitvis\_vip\_gpio | transaction\_pkg.vhd | GPIO transaction package with DTT types, constants etc. |
| bitvis\_vip\_gpio | vvc\_cmd\_pkg.vhd | GPIO VVC command types and operations |
| bitvis\_vip\_gpio | ../uvvm\_vvc\_framework/src\_target\_dependent/td\_target\_support\_pkg.vhd | UVVM VVC target support package, compiled into the GPIO VVC library. |
| bitvis\_vip\_gpio | ../uvvm\_vvc\_framework/src\_target\_dependent/td\_vvc\_framework\_common\_methods\_pkg.vhd | UVVM framework common methods compiled into the GPIO VVC library |
| bitvis\_vip\_gpio | vvc\_methods\_pkg.vhd | GPIO VVC methods |
| bitvis\_vip\_gpio | ../uvvm\_vvc\_framework/src\_target\_dependent/td\_queue\_pkg.vhd | UVVM queue package for the VVC |
| bitvis\_vip\_gpio | ../uvvm\_vvc\_framework/src\_target\_dependent/td\_vvc\_entity\_support\_pkg.vhd | UVVM VVC entity methods compiled into the GPIO VVC library |
| bitvis\_vip\_gpio | gpio\_vvc.vhd | GPIO VVC |

# Simulator compatibility and setup

See README.md for a list of supported simulators.

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

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.

**INTELLECTUAL**

**PROPERTY**