UVVM Common Methods – Quick Reference

await_uvvm_initialization (void)

Example: await_uvvm_initialization(VOID);

await_completion (vvc_target, vvc_instance_idx, [vvc_channel,] [wanted_idx,] [timeout, [msg]])

Example: await_completion(SBI_VVCT, 1, 100 ns, "Waiting for all SBI commands to complete");

await_any_completion (vvc_target, vvc_instance_idx, [vvc_channel,] [wanted_idx,] lastness, [timeout, [msg, [await_completion_idx]]])

Example: await_any_completion(SBI_VVCT, 1, NOT_LAST, 100 ns, "Add SBI_VVC#1 to the await_any_completion group");
await_any_completion(SBI_VVCT, 2, LAST, 100 ns, "Add SBI_VVC#2 as the last member of the group: Waiting until the first in the group completes their commands");

disable_log_msg (vvc_target, vvc_instance_idx, [vvc_channel,] msg_id, [msg])

Example: disable_log_msg(SBI_VVCT, 1, ID_BFM);

enable_log_msg (vvc_target, vvc_instance_idx, [vvc_channel,] msg_id, [msg])

Example: enable_log_msg(UART_VVCT, 1, RX, ID_BFM);

flush_command_queue (vvc_target, vvc_instance_idx, [vvc_channel,] [msg])

Example: flush_command_queue(AXILITE_VVCT, 1);

fetch_result (vvc_target, vvc_instance_idx, [vvc_channel,] wanted_idx, result, [fetch_is_accepted,] [msg, [alert_level]])

Example: fetch_result(SBI_VVCT, 1, v_idx, v_result, v_fetch_is_accepted);

insert_delay (vvc_target, vvc_instance, [vvc_channel,] delay, [msg])

Example: insert_delay(SBI_VVCT, 1,100 ns);
Example: insert_delay(UART_VVCT, 1, TX, 10); -- 10 Clock cycles delay using the VVC clk

terminate_current_command (vvc_target, vvc_instance_idx, [vvc_channel, [msg]])

Example: terminate_current_command(SBI_VVCT, 1);

terminate_all_commands (vvc_target, vvc_instance_idx, [vvc_channel, [msg]])

Example: terminate_all_commands(UART_VVCT, 1, RX);

get_last_received_cmd_index (vvc_target, vvc_instance, [vvc_channel,], [msg])

Example: get_last_received_cmd_index (SBI_VVCT, 1);
Example: get_last_received_cmd_index (UART_VVCT, 1, RX);

UVVM™

VHDL 2008 only

UVVM methods package - target parameters

Name	Type	Example(s)	Description
vvc_target	t_vvc_target_record	UART_VVCT	VVC target type compiled into each VVC in order to differentiate between VVCs.
vvc_instance_idx	Integer	1	Instance number of the VVC used in this method
vvc_channel	t_channel	TX, RX or ALL_CHANNELS	The VVC channel of the VVC instance used in this method
void	t_void	VOID	An empty input parameter for procedure waiting for UVVM to be initialized.

UVVM methods package - functional parameters

Name	Type	Example(s)	Description
wanted_idx	natural	50	The index to be fetched or awaited
timeout	time	100 ns	The maximum time to await completion of a specified command, or all pending commands. An alert of severity ERROR will be triggered if the awaited time is equal to the specified timeout.
msg	string	"Awaiting CR from UART"	A message parameter to be appended to the log when the method is executed.
msg_id	t_msg_id	ID_SEQUENCER	The ID to enable/disable with enable/disable_log_msg(). For more info, see the UVVM-Util documentation.
result	t_vvc_result	v_result	The output where the fetched data is to be placed with fetch_result()
fetch_is_accepted	boolean	v_fetch_is_accepted	Output containing a Boolean that states if the fetch command was accepted or not. Will be false if the specified command index has not been stored.
alert_level	t_alert_level	TB_WARNING	The alert level used for the alert which occurs when a fetch_result() command is not accepted
delay	time or natural	100 ns or 10	Delay to be inserted in the insert_delay() procedure, either as time or number of clock cycles

UVVM VVC Framework Common Methods details

All VVC procedures are defined in the UVVM VVC framework common methods package, td_vvc_framework_common_methods_pkg.vhd

1   UVVM VVC Framework Common Methods details and examples

Method	Description
await_uvvm_initialization()	**await_uvvm_initialization(VOID)** This procedure is a blocking procedure that must be called from the TB sequencer prior to any VVC calls to ensure that the UVVM engine has been initialized and is ready. This procedure will check the shared_uvvm_state on each delta cycle until the UVVM engine has been initialized. Example: `await_uvvm_initialization(VOID);`

await_completion()	**await_completion(vvc_target, vvc_instance, timeout, msg)**
	await_completion(vvc_target, vvc_instance, wanted_idx, timeout, msg)
	await_completion(vvc_target, vvc_instance, vvc_channel, timeout, msg)
	await_completion(vvc_target, vvc_instance, vvc_channel, wanted_idx, timeout, msg)
	
	Tells the VVC to await the completion of either all pending commands or a specified command index.
	A message with log ID ID_IMMEDIATE_CMD_WAIT will be logged before waiting, and a message with log ID ID_IMMEDIATE_CMD will be logged at the end of the wait.
	The procedure will report an alert if not all commands have completed within the specified time, *timeout*. The severity of this alert will be TB_ERROR.
	It is also available as a broadcast to all VVCs.
	
	Examples:
	`await_completion(SBI_VVCT, 1, 16 ns, "Await execution. For single entry queue");`
	`await_completion(SBI_VVCT, 1, v_cmd_idx, 100 ns, "Wait for sbi_read to finish");`
	Broadcast:
	`await_completion(VVC_BROADCAST, 100 ns, "Wait for all VVCs to finish");`

await_any_completion()

await_any_completion(vvc_target, vvc_instance, lastness, timeout, msg, await_completion_idx)
await_any_completion(vvc_target, vvc_instance, wanted_idx, lastness, timeout, msg, await_completion_idx)
await_any_completion(vvc_target, vvc_instance, vvc_channel, lastness, timeout, msg, await_completion_idx)
await_any_completion(vvc_target, vvc_instance, vvc_channel, wanted_idx, lastness, timeout, msg, await_completion_idx)

Adds a VVC to the await_any_completion group, so that the sequencer can wait until **any** VVC in the group completes.
In the same way as await_completion, each await_any_completion call can specify that the VVC in question shall wait for either all pending commands (default) or a specified command index (wanted_idx parameter).

When the sequencer calls await_any_completion with 'lastness' = NOT_LAST, it is not blocked so that it can continue adding members to the await_any_compleiton group by calling await_any_completion for each VVC.
When the sequencer calls await_any_completion with 'lastness' = LAST, the sequencer is blocked until **any** of the VVCs in the group are done waiting for their command(s) to complete.

The optional parameter await_completion_idx is useful for separating the groups when calling await_any_completion from multiple sequencers simultaneously:

Each VVC in the group will log a message with ID ID_IMMEDIATE_CMD_WAIT before waiting, and a message with log ID ID_IMMEDIATE_CMD at the end of the wait.
The procedure will report an alert if not all commands have completed within the specified time, *timeout*. The severity of this alert will be TB_ERROR.

The following example is a sequence of calls that results in waiting until the **first** of the 3 VVCs completes:

```
await_any_completion(SBI_VVCT, 1,NOT_LAST, 1 ms, "Adding SBI VVC to group: waits until all commands are complete");
await_any_completion(AXISTREAM_VVCT, 3, v_cmd_idx, NOT_LAST, 1 ms,
                "Adding AXI VVC#3 to group: this VVC will wait until v_cmd_idx is complete");
await_any_completion(AXISTREAM_VVCT, 4, LAST, 1 ms,"Adding AXI VVC#4 and concluding group. Will now wait for first VVC in group");
```

Limitations:
- While forming a group using await_any_comletion(..NOT_LAST) calls followed by (…LAST) call, do not send other commands to the affected VVCs in between these calls.
- Multiple sequencers cannot call await_any_completion() on the same VVC instance simultaneously.

disable_log_msg()

disable_log_msg(vvc_target, vvc_instance, msg_id, msg)
disable_log_msg(vvc_target, vvc_instance, vvc_channel, msg_id, msg)

Instruct the VVC to disable a given log ID. This call will be forwarded to the UVVM Utility Library disable_log_msg function. For more information about the disable_log_msg() method, please refer to the UVVM-Util QuickRef.
It is also available as a broadcast to all VVCs.

Examples:

```
disable_log_msg(SBI_VVCT, 1, ID_LOG_BFM, "Disabling SBI BFM logging");
disable_log_msg(UART_VVCT, 1, TX, ID_LOG_BFM, "Disabling UART TX BFM logging");
```
Broadcast:
```
disable_log_msg (VVC_BROADCAST, ALL_MESSAGES, "Disables all messages in all VVCs");
```

enable_log_msg()

enable_log_msg(vvc_target, vvc_instance, msg_id, msg)
enable_log_msg(vvc_target, vvc_instance, vvc_channel, msg_id, msg)

Instruct the VVC to enable a given log ID. This call will be forwarded to the UVVM Utility Library enable_log_msg function. For more information about the enable_log_msg() method, please refer to the UVVM-Util QuickRef.
It is also available as a broadcast to all VVCs.

Examples:
```
enable_log_msg(SBI_VVCT, 1, ID_LOG_BFM, "Enabling SBI BFM logging");
enable_log_msg(UART_VVCT, 1, TX, ID_LOG_BFM, "Enabling UART TX BFM logging");
```
Broadcast:
```
enable_log_msg (VVC_BROADCAST, ID_LOG_BFM, " Enabling BFM logging for all VVCs");
```

flush_command_queue()

flush_command_queue(vvc_target, vvc_instance, msg)
flush_command_queue(vvc_target, vvc_instance, vvc_channel,msg)

Flushes the VVC command queue for the specified VVC target/channel. The procedure will log information with log ID ID_IMMEDIATE_CMD.
It is also available as a broadcast to all VVCs.

Example:
```
flush_command_queue(SBI_VVCT, 1, "Flushing command queue");
```
Broadcast:
```
flush_command_queue (VVC_BROADCAST, " Flushing command queues");
```

fetch_result()

fetch_result(vvc_target, vvc_instance,   wanted_id, result,   msg, alert_level)
fetch_result(vvc_target, vvc_instance, vvc_channel, wanted_id, result,   msg, alert_level)
fetch_result(vvc_target, vvc_instance,   wanted_id, result, fetch_is_accepted, msg, alert_level)
fetch_result(vvc_target, vvc_instance, vvc_channel, wanted_id, result, fetch_is_accepted, msg, alert_level)

Fetches a stored result using the command index. A result is stored when using e.g. the read or receive commands in a VVC. The fetched result is available on the 'result' output. The Boolean output 'fetch_is_accepted' is used to indicate if the fetch was successful or not. A fetch can fail if e.g. the wanted_id did not have a result to store, or the wanted_id read has not yet been executed. Omitting the 'fetch_is_accepted' parameter causes the parameters to be checked automatically in the procedure. On successful fetch, a message with log ID ID_UVVM_CMD_RESULT is logged.

Example:
```
fetch_result(SBI_VVCT,1, v_cmd_idx, v_data, v_is_ok, "Fetching read-result");
```
Full example:
```
sbi_read(SBI_VVCT, 1, C_ADDR_FIFO_GET, "Read from FIFO");
v_cmd_idx := get_last_received_cmd_idx(SBI_VVCT,1); -- Retrieve the command index
await_completion(SBI_VVCT, 1, v_cmd_idx, 100 ns, "Wait for sbi_read to finish");
fetch_result(SBI_VVCT, 1, v_cmd_idx, v_data, v_is_ok, "Fetching read-result");
check_value(v_is_ok, ERROR, "Readback OK via fetch_result()");
```

insert_delay()

insert_delay(vvc_target, vvc_instance, delay, msg)
insert_delay(vvc_target, vvc_instance, vvc_channel, delay, msg)

This method inserts a delay of 'delay' clock cycles or 'delay' seconds in the VVC.
It is also available as a broadcast to all VVCs.

Examples:
```
insert_delay(SBI_VVCT,1, 100, "100T delay");
insert_delay(SBI_VVCT,1, 50 ns, "50 ns delay");
```
Broadcast:
```
insert_delay (VVC_BROADCAST, 50 ns, "Insert 50 ns delay to all VVCs");
```

---

terminate_current_command()

terminate_current_command(vvc_target, vvc_instance, msg)
terminate_current_command(vvc_target, vvc_instance, vvc_channel, msg)

This method terminates the current command in the VVC, if the currently running BFM command supports the terminate signal.
It is also available as a broadcast to all VVCs.

Example:
```
terminate_current_command(SBI_VVCT, 1, "Terminating current command");
```
Broadcast:
```
terminate_current_command (VVC_BROADCAST, "Terminating current command in all VVCs");
```

---

terminate_all_commands()

terminate_all_commands(vvc_target, vvc_instance, msg)
terminate_all_commands(vvc_target, vvc_instance, vvc_channel, msg)

This method terminates the current command in the VVC, if the currently running BFM command supports the terminate signal. The terminate_all_commands() procedure also flushes the VVC command queue, removing all pending commands.
It is also available as a broadcast to all VVCs.

Example:
```
terminate_all_commands(SBI_VVCT, 1, "Terminating all commands");
```
Broadcast:
```
terminate_all_commands (VVC_BROADCAST,"Terminating all commands in all VVCs");
```

---

get_last_received_cmd_idx()

get_last_received_cmd_idx (vvc_target, vvc_instance, msg)
get_last_received_cmd_idx (vvc_target, vvc_instance, vvc_channel, msg)

This method is used to get the command index of the last command received by the VVC interpreter. Necessary for getting the command index of a read for fetch_result.

Example:
```
get_last_received_cmd_idx(SBI_VVCT, 1);
```