Common VVC Methods (Command Distribution Methods) – Quick Reference

await_completion (vvc_target, vvc_instance_idx, [vvc_channel,] [wanted_idx,] [timeout, [msg, [scope]]])
await_completion (ANY_OF, vvc_info_list, timeout, [list_action, [msg, [scope]]])

Example: await_completion(SBI_VVCT, 1, 100 ns, “Waiting for all SBI commands to complete”);
Example: await_completion(UART_VVCT, 1, RX, v_idx, 100 ns, “Waiting for UART receive to complete”, C_SCOPE);
Example: await_completion(ANY_OF, my_vvc_info_list, 1 ms, KEEP_LIST, "Waiting for any VVC in the list to complete", C_SCOPE);

await_any_completion ()

Note: this procedure will be deprecated in future releases, see page 4 for syntax and more info.

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

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

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

Example: disable_log_msg(SBI_VVCT, 1, ID_BFM);

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

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, [scope]]])

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

insert_delay (vvc_target, vvc_instance_idx, [vvc_channel,] delay, [msg, [scope]])

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, [scope]]])

Example: terminate_current_command(SBI_VVCT, 1);

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

Example: terminate_all_commands(UART_VVCT, 1, RX);

get_last_received_cmd_idx (vvc_target, vvc_instance_idx, [vvc_channel, [scope]])

Example: v_cmd_idx := get_last_received_cmd_idx (SBI_VVCT, 1);
Example: v_cmd_idx := get_last_received_cmd_idx (UART_VVCT, 1, RX);

UVVM™

VHDL 2008 only

UVVM methods - 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
vvc_info_list	t_vvc_info_list	v_vvc_info_list	A list of protected type containing one or several VVC IDs (name, instance, channel) & command index. VVC IDs and corresponding command index can be added to the list by using the procedure **add(name, instance, [channel,] [cmd_idx])**. The name is a string that should match the C_VVC_NAME in the VVC's vvc_methods_pkg.vhd.
void	t_void	VOID	An empty input parameter for procedure waiting for UVVM to be initialized.

UVVM methods - functional parameters

Name	Type	Example(s)	Description
wanted_idx	natural	50	The index to be fetched or awaited
list_action	t_list_action	KEEP_LIST, CLEAR_LIST	An enumerated type to either keep the VVC IDs or remove them from the list after await_completion() has finished.
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
quietness	t_quietness	QUIET	The logging of the command can be turned off by setting quietness=QUIET.
scope	string	"Sequencer 1"	A string describing the scope from which the log/alert originates.

UVVM VVC Framework command broadcasting and multicasting

Commands in UVVM can be distributed to all instances of a VVC or to all VVCs using dedicated parameters.

Command Parameter	Description

VVC_BROADCAST	The VVC_BROADCAST command parameter can be used when a command is to target all VVCs within the test environment, reducing the number of command instructions needed in the testbench.
	
	Example:
	`enable_log_msg(VVC_BROADCAST, ALL_MESSAGES); -- enable logging for all VVCs`
	`await_completion(VVC_BROADCAST, 10 us); -- wait for all VVCs to complete`

ALL_INSTANCES	The ALL_INSTANCES command parameter can be used when a command is targeting all instances of a VVC within the test environment, reducing the number of command instructions needed in the testbench.
	
	Example:
	`enable_log_msg(SBI_VVCT, ALL_INSTANCES, ALL_MESSAGES); -- enable logging for all instances of SBI_VVCT`
	`await_completion(SBI_VVCT, ALL_INSTANCES, 100 ns); -- wait for all instances of SBI_VVCT to complete`

ALL_CHANNELS	The ALL_CHANNELS command parameter can be used when a command is targeting all channels of a VVC within the test environment, reducing the number of command instructions needed in the testbench.
	
	Example:
	`enable_log_msg(UART_VVCT, 1, ALL_CHANNELS, ALL_MESSAGES); -- enable logging for all channels of SBI_VVCT instance 1`
	`await_completion(UART_VVCT, ALL_INSTANCES, ALL_CHANNELS, 100 ns); -- wait for all instances and channels of UART_VVCT to complete`

C_VVCT_ALL_INSTANCES	See description above. C_VVCT_ALL_INSTANCES = ALL_INSTANCES. Warning! This command parameter might be removed in a future release and we encourage the use of ALL_INSTANCES.

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_completion()	**await_completion(vvc_target, vvc_instance_idx, timeout, msg, scope)**
	await_completion(vvc_target, vvc_instance_idx, wanted_idx, timeout, msg, scope)
	await_completion(vvc_target, vvc_instance_idx, vvc_channel, timeout, msg, scope)
	await_completion(vvc_target, vvc_instance_idx, vvc_channel, wanted_idx, timeout, msg, scope)
	await_completion(ANY_OF, vvc_info_list, timeout, list_action, msg, scope)
	
	Tells the VVC to await the completion of either all pending commands or a specified command index.
	A message will be logged before and 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 possible multicast to ALL_INSTANCES or ALL_CHANNELS of a VVC.
	
	To await the completion of one out of several VVCs in a group use the overload with the vvc_info_list.
	The vvc_info_list of type t_vvc_info_list (protected type) is a local variable that needs to be declared in the sequencer. The list_action default is to clear the list.
	This overload will block the sequencer while waiting, but not the VVCs, so they can continue to receive commands from other sequencers.
	Important: to use the vvc_info_list, the package uvvm_vvc_framework.ti_protected_types_pkg.all must be included in the testbench. Note that the command with the vvc_info_list requires VVCs supporting the VVC activity register introduced in UVVM release v2020.05.19
	
	Examples:
	`await_completion(SBI_VVCT, 1, 16 ns, "Wait for SBI instance 1 to finish", C_SCOPE);`
	`await_completion(SBI_VVCT, 1, v_cmd_idx, 100 ns, "Wait for sbi_read to finish", C_SCOPE);`
	Multicast:
	`await_completion(SBI_VVCT, ALL_INSTANCES, 100 ns, "Wait for all SBI instances to finish", C_SCOPE);`
	`await_completion(UART_VVCT, 1, ALL_CHANNELS, 100 ns, "Wait for all UART channels from instance 1 to finish", C_SCOPE);`
	Using vvc_info_list:
	`variable my_vvc_info_list : t_vvc_info_list;`
	
	`my_vvc_info_list.add("SBI_VVC", 1);`
	`my_vvc_info_list.add("AXISTREAM_VVC", 3, v_cmd_idx);`
	`my_vvc_info_list.add("UART_VVC", ALL_INSTANCES, ALL_CHANNELS);`
	`await_completion(ANY_OF, my_vvc_info_list, 1 ms, KEEP_LIST, "Wait for any VVC in the list to finish", C_SCOPE);`

await_any_completion()

Replaced by await_completion(ANY_OF, vvc_info_list, timeout, list_action, msg, scope) above to allow VVCs to accept commands while waiting for completion.
This command still works as previously, but with less functionality than the new await_completion(ANY_OF, ...)
Warning! This procedure will soon be deprecated and removed.
For details and examples for using this call see UVVM release v2020.05.12 or any earlier releases.

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

disable_log_msg()

disable_log_msg(vvc_target, vvc_instance_idx, msg_id, msg, quietness, scope)
disable_log_msg(vvc_target, vvc_instance_idx, vvc_channel, msg_id, msg, quietness, scope)

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", NON_QUIET, C_SCOPE);
```
Broadcast:
```
    disable_log_msg(VVC_BROADCAST, ALL_MESSAGES, "Disables all messages in all VVCs", NON_QUIET, C_SCOPE);
```

enable_log_msg()

enable_log_msg(vvc_target, vvc_instance_idx, msg_id, msg, quietness, scope)
enable_log_msg(vvc_target, vvc_instance_idx, vvc_channel, msg_id, msg, quietness, scope)

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", NON_QUIET, C_SCOPE);
```
Broadcast:
```
    enable_log_msg(VVC_BROADCAST, ID_LOG_BFM, "Enabling BFM logging for all VVCs", NON_QUIET, C_SCOPE);
```

flush_command_queue()

flush_command_queue(vvc_target, vvc_instance_idx, msg, scope)
flush_command_queue(vvc_target, vvc_instance_idx, vvc_channel, msg, scope)

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", C_SCOPE);
```
Broadcast:
```
    flush_command_queue(VVC_BROADCAST, "Flushing command queues", C_SCOPE);
```

fetch_result()

fetch_result(vvc_target, vvc_instance_idx, wanted_id, result, msg, alert_level, scope)
fetch_result(vvc_target, vvc_instance_idx, vvc_channel, wanted_id, result, msg, alert_level, scope)
fetch_result(vvc_target, vvc_instance_idx, wanted_id, result, fetch_is_accepted, msg, alert_level, scope)
fetch_result(vvc_target, vvc_instance_idx, vvc_channel, wanted_id, result, fetch_is_accepted, msg, alert_level, scope)

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", C_SCOPE);
```
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_idx, delay, msg, scope)
insert_delay(vvc_target, vvc_instance_idx, vvc_channel, delay, msg, scope)

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

terminate_current_command()

terminate_current_command(vvc_target, vvc_instance_idx, msg, scope)
terminate_current_command(vvc_target, vvc_instance_idx, vvc_channel, msg, scope)

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

terminate_all_commands()

terminate_all_commands(vvc_target, vvc_instance_idx, msg, scope)
terminate_all_commands(vvc_target, vvc_instance_idx, vvc_channel, msg, scope)

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", C_SCOPE);
```
Broadcast:
```
terminate_all_commands(VVC_BROADCAST,"Terminating all commands in all VVCs", C_SCOPE);
```

get_last_received_cmd_idx()

get_last_received_cmd_idx(vvc_target, vvc_instance_idx, scope)
get_last_received_cmd_idx(vvc_target, vvc_instance_idx, vvc_channel, scope)

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:
```
v_cmd_idx := get_last_received_cmd_idx(SBI_VVCT, 1, C_SCOPE);
```