Generic Queue - Quick Reference

UVVM Support Component

The Generic Queue operate as a FIFO that can hold generic elements, which enable large FIFO and e.g. record elements possibilities.

Methods
add([instance], element)
fetch([instance], [identifier_option, identifier])
get([instance])
flush([instance])
reset([instance])
v_empty := **is_empty**([instance])
set_scope([instance], scope)
v_scope := **get_scope**([instance])
v_count := **get_count**([instance])
insert([instance], identifier_option, identifier, element)
delete([instance], element)
delete([instance], identifier_option, identifier, range_option)
delete([instance], identifier_option, identifier_min, identifier_max)

peek([instance], [identifier_option, identifier])
v_position := **find_position**([instance], element)
v_entry_num := **find_entry_num**([instance], element)
v_entry_num := **get_entry_num**([instance], position_val)
v_exists := **exists**([instance], element)
print_queue([instance])
set_queue_count_threshold([instance], queue_count_alert_level)
v_queue_count_threshold := **get_queue_count_threshold**([instance])
set_queue_count_threshold_severity([instance], alert_level)
v_queue_count_threshold_severity := **get_queue_count_threshold_severity**([instance])
set_queue_count_max([instance], queue_count_max)
v_queue_count_max := **get_queue_count_max**([instance])

Note: All methods used without parameters must be called with dummy parameter VOID.

Generic Queue – Functional parameters

Name	Type	Example(s)	Description
instance	integer	2	One generic queue variable can have multiple independent queues referred to as instances.
identifier_option	t_identifier_option	POSITION, ENTRY_NUM	A queue element can be identified by POSITION or ENTRY_NUM.
identifier	integer	2	The position or entry number.
identifier_min	integer	3	The minimum position or entry number of a range.
identifier_max	integer	10	The maximum position or entry number of a range.
range_option	t_range_option	SINGLE, AND_LOWER, AND_HIGHER	The range that is affected.
scope	String	C_QUEUE_SCOPE	The scope for the generic queue. Has to be set prior to usage of several procedure and functions.
name	String	“rx_packet_queue”	The name of the generic queue.
element	t_generic_elment	v_rx_data	The element that shall be pushed to the queue.
void	t_void	VOID	Unused, empty input parameter.
queue_count_max	natural	1000	The maximum number of elements the queue shall hold.
queue_count_alert_level	natural	950	The number of elements the queue can hold before an alert is raised.
alert_level	t_alert_level	TB_WARNING	The alert level is raised when the number of elements in the queue exceeds queue_count_alert_level.

Note: default queue size is 2048 bits and the size can be adjusted with the C_NUMBER_OF_BITS_IN_DATA_BUFFER located in the adaptations package.

UVVM™

VHDL 2008 only

Generic Queue – Generics

Generic element	Type	DEFAULT
<generic_element>	t_generic_element	<none>
GC_QUEUE_COUNT_MAX	natural	1000
GC_QUEUE_COUNT_MAX_THRESHOLD	natural	950

Package declaration example:

```
package td_queue_pkg is new uvvm_vvc_framework.ti_generic_queue_pkg
       generic map(   t_generic_element => integer,
                      GC_QUEUE_COUNT_MAX => 1000,
                      GC_QUEUE_COUNT_THRESHOLD => 950);
```

Queue declaration example:

```
use work.td_queue_pkg.all;
shared variable generic_queue : t_generic_queue;
```

Generic Queue details

All Generic Queue functions and procedures are defined in the UVVM Generic Queue package, ti_generic_queue_pkg.vhd. The generic queue can be used with a single instance or with multiple instances. An instance is a separate queue. When multiple instances are used the methods are called with the instance parameter. When only using a single generic queue instance, the instance parameter may be omitted if that instance is instance 1. Multiple instances are typically used when multiple queues with the same data types are needed. All parameters in brackets are optional.

1 Generic Queue details and examples

Method	Description

add()	**add([instance], element)**
	
	This procedure adds an element at the back of a generic queue.
	The queue element is generic, meaning that it can be of any type, specified by the package declaration (see page 2 for example).
	Note that if no scope is set for the queue, a TB_WARNING will be raised. An alert, set by set_queue_count_threshold_severity(), will be raised when the queue reach a level, set by set_queue_count_threshold(). Also note that trying to add() to a full queue will raise a TB_ERROR.
	
	Example:
	`generic_queue.add(v_data_packet);`
	`generic_queue.add(2, v_data_packet);`

fetch()	**fetch([instance], [identifier_option, identifier])**
	
	This function returns an element from the generic queue and removes it from the queue.
	Note that the oldest element in the queue is returned first if no identifier is specified. Attempting to fetch() from the queue without setting queue scope first (see set_scope()), will trigger a TB_WARNING, and attempting to fetch() from an empty queue will trigger a TB_ERROR.
	
	Example:
	`v_data_packet := generic_queue.fetch(VOID);`
	`v_data_packet := generic_queue.fetch(2, POSITION, 5);`
	`v_data_packet := generic_queue.fetch(2, ENTRY_NUM, 14);`

get()	**get([instance])** This function is deprecated. Use fetch.

flush()

flush([instance])

This procedure empties the queue. A TB_WARNING will be raised if no scope is set for the queue prior to calling flush().

Example:

```
generic_queue.flush(VOID);
generic_queue.flush(2);
```

reset()

reset([instance])

This procedure empties the queue and resets the entry number. A TB_WARNING will be raised if no scope is set for the queue prior to calling reset().

Example:

```
generic_queue.reset(VOID);
generic_queue.reset(2);
```

is_empty()

is_empty([instance])

This function returns true if the queue is empty and false otherwise.

Example:

```
If generic_queue.is_empty(VOID) then ...
```

set_scope()

set_scope([instance],scope)

This procedure will set the scope of the queue.
Note that most of the procedures and functions in the generic queue will raise a TB_WARNING if no scope has been set for the queue.

Example:

```
generic_queue.set_scope(C_QUEUE_SCOPE);
generic_queue.set_scope(2, C_QUEUE_SCOPE);
```

get_scope()

get_scope([instance])

This function returns the scope of the queue as a string.

Example:

```
v_queue_scope := generic_queue.get_scope(VOID);
v_queue_scope := generic_queue.get_scope(2);
```

get_count()

get_count([instance])

This function returns the number of elements currently in the queue.

Example:
```
v_num_elements := generic_queue.get_count(VOID);
v_num_elements := generic_queue.get_count(2);
```

insert()

insert([instance], identifier_option, identifier, element)

This procedure inserts an element at the specified position in the generic queue. If identifier_option is POSITION, the element is inserted at that position. If identifier_option is ENTRY_NUM, the element is inserted after that entry number.

Example:
```
generic_queue.insert(POSITION, 5, v_data_packet);
generic_queue.insert(2, ENTRY_NUM, 8, v_data_packet);
```

delete()

delete([instance], [element])
delete([instance], identifier_option, identifier, range_option)
delete([instance], identifier_option, identifier_min, identifier_max)

This procedure deletes the specified element from the generic queue. Element to be deleted can be specified by a matching element or by identifier and range.

Example:
```
generic_queue.delete(v_data_packet);
generic_queue.delete(2, ENTRY_NUM, 8, SINGLE);
generic_queue.delete(2, POSITION, 12, AND_HIGHER);
generic_queue.delete(2, ENTRY_NUM, 3, 6);
```

peek()

peek([instance], [identifier_option, identifier])

This function returns an element from the generic queue without deleting it. Element can be specified by POSITION or ENTRY_NUM.

Example:
```
v_data_packet := generic_queue.peek(VOID); --first element in queue, instance 1
v_data_packet := generic_queue.peek(2, ENTRY_NUM, 8);
v_data_packet := generic_queue.peek(2, POSITION, 2);
```

find_position()

find_position([instance], element)

This function returns the position of the matching element. Returns -1 if no matching element is found.

Example:
```
v_position := generic_queue.find_position(v_data_packet);
v_position := generic_queue.find_position(2, v_data_packet);
```

find_entry_num()

find_entry_num([instance], element)

This function returns the entry number of the matching element. Returns -1 if no matching element is found.

Example:
```
v_entry_num := generic_queue.find_entry_num(v_data_packet);
v_entry_num := generic_queue.find_entry_num(2, v_data_packet);
```

get_entry_num()

get_entry_num([instance], position_val)

This function returns the entry number of the element in the specified position.

Example:
```
v_entry_num := generic_queue.get_entry_num(6);
v_entry_num := generic_queue.get_entry_num(2, 8);
```

exists()

exists([instance], element)

This function returns true if a matching element is found in the generic queue and false otherwise.

Example:
```
if generic_queue.exists(element) then ...
if generic_queue.exists(2, element) then ...
```

print_queue()

print_queue([instance])

This procedure prints the position and entry number for all elements in the generic queue.

Example:
```
generic_queue.print_queue(VOID);
generic_queue.print_queue(2);
```

set_queue_count_max()

set_queue_count_max([instance], queue_count_max)

This procedure sets the maximum number of elements the queue can hold.
Note that a TB_ERROR is raised if parameter queue_count_max is less than the number of elements currently in the queue.

Example:
```
generic_queue.set_queue_count_max(1000);
generic_queue.set_queue_count_max(2, 1000);
```

get_queue_count_max()

get_queue_count_max([instance])

This function returns the maximum number of elements the queue can hold.

Example:
```
v_queue_max_elements := generic_queue.get_queue_count_max(VOID);
v_queue_max_elements := generic_queue.get_queue_count_max(2);
```

set_queue_count_threshold()

set_queue_count_threshold([instance], queue_count_alert_level)

This procedure sets the threshold value that will raise an alert, set by set_queue_count_threshold_severity(), if the number of queue elements exceeds the queue_count_alert_level.

Example:
```
generic_queue.set_queue_count_threshold(950);
generic_queue.set_queue_count_threshold(2, 950);
```

get_queue_count_threshold()

get_queue_count_threshold([instance])

This function returns the threshold value that will raise an alert, set by set_queue_count_threshold_severity(), if the number of queue elements exceeds the queue_count_alert_level.

Example:
```
v_queue_threshold := generic_queue.get_queue_count_threshold(VOID);
v_queue_threshold := generic_queue.get_queue_count_threshold(2);
```

set_queue_count_threshold_severity()

set_queue_count_threshold_severity(severity_level)

This procedure sets the severity level for the alert that is raised when the number of queue elements exceeds the value set by set_queue_count_threshold().

Example:
```
generic_queue.set_queue_count_threshold_severity(TB_WARNING);
```

get_queue_count_threshold_severity()

get_queue_count_threshold_severity(void)

This function return the severity level for the alert that is raised when the number of queue elements exceeds the value set by set_queue_count_threshold().

Example:
```
v_queue_level_severity := generic_queue.get_queue_count_threshold_severity(VOID);
```