**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. |
|  |  |  |  |



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.

# 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); |

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**