## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="App" /></td>
<td><strong>App</strong> Utilities for executing the Liquid application.</td>
</tr>
<tr>
<td><img src="image" alt="Bit" /></td>
<td><strong>Bit</strong> Represents the measured value, in the computational basis, of a qubit.</td>
</tr>
<tr>
<td><img src="image" alt="Circuit" /></td>
<td><strong>Circuit</strong> The circuit representation of an operation in a quantum algorithm. Circuits are generally created using <code>Circuit.Compile</code>.</td>
</tr>
<tr>
<td><img src="image" alt="CMat" /></td>
<td><strong>CMat</strong> A dense matrix of complex numbers.</td>
</tr>
<tr>
<td><img src="image" alt="CSMat" /></td>
<td><strong>CSMat</strong> A sparse matrix of complex numbers.</td>
</tr>
<tr>
<td><img src="image" alt="CVec" /></td>
<td><strong>CVec</strong> A block-sparse vector of complex numbers.</td>
</tr>
<tr>
<td><img src="image" alt="Fermion" /></td>
<td><strong>Fermion</strong> Hamiltonian simulation for fermionic systems.</td>
</tr>
<tr>
<td><strong>Gate</strong></td>
<td>A quantum gate.</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td><strong>GateOp</strong></td>
<td>Gate operation type. This is used in <strong>Gate</strong> definitions.</td>
</tr>
<tr>
<td><strong>Hamiltonian</strong></td>
<td>Base class for Hamiltonian dynamics simulators.</td>
</tr>
<tr>
<td><strong>HamiltonianGates</strong></td>
<td>A collection of gates that are useful for Hamiltonian simulation and annealing.</td>
</tr>
<tr>
<td><strong>Ket</strong></td>
<td>Represents a state vector.</td>
</tr>
<tr>
<td><strong>KrausOp</strong></td>
<td>Entries for Kraus operators in Channel Gate type</td>
</tr>
<tr>
<td><strong>Noise</strong></td>
<td>A complete noise model for a specific circuit.</td>
</tr>
<tr>
<td><strong>NoiseEvents</strong></td>
<td>Noise statistics that are tracked for normal and error-correcting gates.</td>
</tr>
<tr>
<td><strong>NoiseModel</strong></td>
<td>A noise model for a particular type of gate (or set of gates).</td>
</tr>
<tr>
<td><strong>NoiseStat</strong></td>
<td>Statistics tracked for each time that noise is applied.</td>
</tr>
<tr>
<td><strong>NoisyMats</strong></td>
<td>Utility class for computing a Pauli rotation matrix. This is used to run quantum</td>
</tr>
</tbody>
</table>
chemistry circuits with noise injected.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>💣</td>
<td>Operations</td>
</tr>
<tr>
<td>💣</td>
<td>The Operations module provides definitions of basic gates. It also includes some handy operators for manipulating qubit lists, and some operations for building gates from existing gates.</td>
</tr>
<tr>
<td>💣</td>
<td>QECC</td>
</tr>
<tr>
<td>💣</td>
<td>Base class for quantum error correcting codes.</td>
</tr>
<tr>
<td>💣</td>
<td>Qubit</td>
</tr>
<tr>
<td>💣</td>
<td>Represents a quantum bit. New Qubits are created using the Ket Add methods.</td>
</tr>
<tr>
<td>💣</td>
<td>RunMode</td>
</tr>
<tr>
<td>💣</td>
<td>Trotterization types.</td>
</tr>
<tr>
<td>💣</td>
<td>Spin</td>
</tr>
<tr>
<td>💣</td>
<td>Hamiltonian for spin systems, such as the Ising model or a spin glass.</td>
</tr>
<tr>
<td>💣</td>
<td>SpinTerm</td>
</tr>
<tr>
<td>💣</td>
<td>A single term in a Spin Hamiltonian.</td>
</tr>
<tr>
<td>💣</td>
<td>Stabilizer</td>
</tr>
<tr>
<td>💣</td>
<td>Steane7</td>
</tr>
<tr>
<td>💣</td>
<td>Implementation of a Steane 7-bit quantum error correcting code, [[7,1,3]], based on the QECC class.</td>
</tr>
</tbody>
</table>
## Tests
A collection of sample Liquid simulations and tests, plus some utility routines to make it easier to write new samples.

## Util
General utilities used by the rest of the system

## UtilLQDAAttribute
Allows a function to be visible from a LIQUiD script or the command line

## UtilprocStatsT
Current process memory usage statistics. Returned by the procStates function.

### Structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex</td>
<td>Data type for complex numbers.</td>
</tr>
</tbody>
</table>
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

App Class

Utilities for executing the Liquid application.

Inheritance Hierarchy

```
System
Object
Microsoft.Research.Liquid.App
```

Namespace: Microsoft.Research.Liquid  
Assembly: Liquid1 (in Liquid1.dll)  
Version: 1.0.5981.24943 (1.0.*)

Syntax

```
[<AbstractClassAttribute>]
[<SealedAttribute>]
type App = class end
```

The `App` type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![RunLiquid]</td>
<td>The main routine for running Liquid. This function gets the command-line</td>
</tr>
<tr>
<td></td>
<td>parameters from the environment, parses them, and executes the requested</td>
</tr>
<tr>
<td></td>
<td>Liquid method.</td>
</tr>
</tbody>
</table>

See Also
App Methods

The App type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunLiquid</td>
<td>The main routine for running Liquid. This function gets the command-line parameters from the environment, parses them, and executes the requested Liquid method.</td>
</tr>
</tbody>
</table>

See Also

Reference
App Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
AppRunLiquid Method

The main routine for running Liquid. This function gets the command-line parameters from the environment, parses them, and executes the requested Liquid method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4*)

### Syntax

```
static member RunLiquid : unit -> int
```

#### Return Value

Type: Int32

0 if execution succeeded, or 1 if an error occurred.

### See Also

Reference  
**App Class**  
Microsoft.Research.Liquid Namespace
Bit Class

Represents the measured value, in the computational basis, of a qubit.

Inheritance Hierarchy

- System
- Object
- Microsoft.Research.Liquid.Bit

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
[<SealedAttribute>]
[<SerializableAttribute>]
type Bit =
    class
        interface IEquatable<Bit>
        interface IStructuralEquatable
        interface IComparable<Bit>
        interface IComparable
        interface IStructuralComparable
    end
```

The Bit type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>The integer value of a measured qubit, either 0 or 1. Note that this property will throw an exception if the value is</td>
</tr>
</tbody>
</table>
unknown.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dump</td>
<td>Dumps the measured value.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Gets a string representation of this value. The string will be &quot;Zero&quot;, &quot;One&quot;, or &quot;?&quot;. (Overrides ObjectToString.)</td>
</tr>
</tbody>
</table>

Remarks

Possible values are:

- **Unknown**: The qubit has not been measured since it was initialized or since the last time it was reanimated.
- **Zero**: The qubit was last measured as $|0\rangle$ in the computational basis and has not been reanimated since.
- **One**: The qubit was last measured as $|1\rangle$ in the computational
basis and has not been reanimated since.

See Also

Reference
Microsoft.Research.Liquid Namespace
Bit Properties

The Bit type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>The integer value of a measured qubit, either 0 or 1. Note that this property will throw an exception if the value is unknown.</td>
</tr>
</tbody>
</table>

See Also

Reference
Bit Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Bitv Property

The integer value of a measured qubit, either 0 or 1. Note that this property will throw an exception if the value is unknown.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

```fsharp
member v : int with get
```

Property Value
Type: Int32

See Also

Reference
Bit Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Bit Methods

The `Bit` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dump</td>
<td>Dumps the measured value.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Gets a string representation of this value. The string will be &quot;Zero&quot;, &quot;One&quot;, or &quot;?&quot;.</td>
</tr>
<tr>
<td></td>
<td>(Overrides <code>Object.ToString</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

Bit Class
BitDump Method

Dumps the measured value.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

## Syntax

```fsharp
member Dump :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<String, Unit>>, Int32>
    level : FSharpOption<Int32> -&gt; unit
```

### Parameters

**f**

Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<String, Unit>>, Int32>`

The optional output function to use. The default is `showLogInd`.

**level**

Type: `Microsoft.FSharp.Core.FSharpOption<Int32>`

The optional indentation level. The default is 0.

## See Also

Reference

- **Bit Class**
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
BitToString Method

Gets a string representation of this value. The string will be "Zero", "One", or "?".

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
abstract ToString : unit -> string
override ToString : unit -> string
```

Return Value

Type: **String**
The string

### See Also

**Reference**

- Bit Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Circuit Class

The circuit representation of an operation in a quantum algorithm. Circuits are generally created using Circuit.Compile.

Inheritance Hierarchy

- System
  - Object

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
F# Copy

[<SerializableAttribute>]
type Circuit = class end
```

The Circuit type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>![ NotEmpty ]</td>
</tr>
<tr>
<td>![ ]</td>
<td>![ Is this Circuit element not the &quot;Empty&quot; circuit. ]</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>![ Compile ]</td>
</tr>
<tr>
<td>![ ]</td>
<td>![ Compiles a function ]</td>
</tr>
</tbody>
</table>
implemented as a sequence of gate function calls into a Circuit.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dump</td>
<td>Dumps this circuit recursively to the console and/or log.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>FindIds</td>
<td>Find Ids below this point and the time needed to execute the circuit (internal)</td>
</tr>
<tr>
<td>Fold</td>
<td>Rewrites this circuit by identifying opportunities for parallelism and turning Seq elements into Par elements where possible. This makes for a better rendering.</td>
</tr>
<tr>
<td>GateCount</td>
<td>Gets the count of gates in circuit. Note that pure label gates -- that is, gates of type String -- are never counted.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Grow</td>
<td>Creates an optimized &quot;grown&quot; version of this Circuit. The type and limits of the conversion are</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GrowGates</td>
<td>Creates an equivalent circuit to this circuit by aggregating existing unitary gates into larger unitary gates. The new circuit will execute faster because it has fewer matrix applications required.</td>
</tr>
<tr>
<td>GrowSingle</td>
<td>Converts this circuit into a single unitary gate, implemented by a single matrix. This allows maximum optimization of circuit execution. This circuit must be completely unitary to be converted into a single matrix. If it is not, an exception will be raised by this method.</td>
</tr>
<tr>
<td>RemoveRedund</td>
<td>Creates a new circuit logically equivalent to this circuit, but with redundant gates removed. For instance, if this routine found a sequence of two (X) gates in succession on the same qubit, it would remove both of them from the result.</td>
</tr>
<tr>
<td>Render</td>
<td>Renders this circuit to a file.</td>
</tr>
<tr>
<td>RenderHT</td>
<td>Renders a circuit to both svg, for HTML, and tikz, for TeX.</td>
</tr>
<tr>
<td>Reverse</td>
<td>Reverses this circuit, which must contain only unitary gates. In addition, bottom-level gates are replaced by their adjoints.</td>
</tr>
</tbody>
</table>
This method will raise an exception if this circuit contains a non-unitary gate.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>Runs this Circuit. The state of the Ket containing the passed-in qubits will be modified.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string representing the current circuit element. (Overrides <code>Object.ToString</code>.)</td>
</tr>
<tr>
<td>Wires</td>
<td>Gets the list of wires (qubit IDs) touched by this circuit.</td>
</tr>
</tbody>
</table>

### Remarks

The possible types of circuits are:

- **Seq**: A list of sub-circuits executed sequentially.
- **Par**: A list of sub-circuits executed in parallel.
- **Apply**: Application of a single gate.
- **Ext**: Application of a single gate that extends a parent, such as an adjoint or control. This type of circuit is created from a Modify gate.
- **BitCon**: Application of a classically-controlled (BCOp) gate.
- **Wrap**: A sub-circuit wrapped into a single gate for more concise rendering.
- **Empty**: A circuit that does nothing.

### See Also

Reference

Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Circuit Properties

The Circuit type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotEmpty</td>
<td>Is this Circuit element not the &quot;Empty&quot; circuit.</td>
</tr>
</tbody>
</table>

See Also

Reference

Circuit Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CircuitNotEmpty Property

Is this Circuit element not the "Empty" circuit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member NotEmpty : bool with get
```

**Property Value**  
Type: Boolean

### See Also

**Reference**  
Circuit Class  
Microsoft.Research.Liquid Namespace
Circuit Methods

The Circuit type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compile</td>
<td>Compiles a function implemented as a sequence of gate function calls into a Circuit.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dumps this circuit recursively to the console and/or log.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>FindIds</td>
<td>Find Ids below this point and the time needed to execute the circuit (internal)</td>
</tr>
<tr>
<td>Fold</td>
<td>Rewrites this circuit by identifying opportunities for parallelism and turning Seq elements into Par elements where possible. This makes for a better rendering.</td>
</tr>
<tr>
<td>GateCount</td>
<td>Gets the count of gates in circuit. Note that pure label gates -- that is, gates of type String -- are never counted.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Grow</td>
<td>Creates an optimized &quot;grown&quot; version of this Circuit. The type and limits of the conversion are specified in the grow parameter.</td>
</tr>
<tr>
<td>GrowGates</td>
<td>Creates an equivalent circuit to this circuit by aggregating existing unitary gates into larger unitary gates. The new circuit will execute faster because it has fewer matrix applications required.</td>
</tr>
<tr>
<td>GrowSingle</td>
<td>Converts this circuit into a single unitary gate, implemented by a single matrix. This allows maximum optimization of circuit execution. This circuit must be completely unitary to be converted into a single matrix. If it is not, an exception will be raised by this method.</td>
</tr>
</tbody>
</table>
| RemoveRedund | Creates a new circuit logically equivalent to this circuit, but with redundant gates removed. For instance, if this routine found a sequence of two X gates in succession on the
same qubit, it would remove both of them from the result.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Render</td>
<td>Renders this circuit to a file.</td>
</tr>
<tr>
<td>RenderHT</td>
<td>Renders a circuit to both svg, for HTML, and tikz, for TeX.</td>
</tr>
<tr>
<td>Reverse</td>
<td>Reverses this circuit, which must contain only unitary gates. In addition, bottom-level gates are replaced by their adjoints. This method will raise an exception if this circuit contains a non-unitary gate.</td>
</tr>
<tr>
<td>Run</td>
<td>Runs this Circuit. The state of the Ket containing the passed-in qubits will be modified.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string representing the current circuit element. (Overrides ObjectToString.)</td>
</tr>
<tr>
<td>Wires</td>
<td>Gets the list of wires (qubit IDs) touched by this circuit.</td>
</tr>
</tbody>
</table>

See Also

Reference
Circuit Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CircuitCompile Method

Compiles a function implemented as a sequence of gate function calls into a Circuit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member Compile :
    f : FSharpFunc<FSharpList<Qubit>, Unit> 
    qs : FSharpList<Qubit> -> Circuit
```

### Parameters

**f**
- Type: `Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>`
- The gate function to compile.

**qs**
- Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
- The qubits the new Circuit will operate on.

### Return Value

Type: `Circuit`  
A new Circuit that represents the function calls

### See Also

Reference  
Circuit Class
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CircuitDump Method

Dumps this circuit recursively to the console and/or log.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```fsharp
member Dump :
  f : FSharpOption<FSharpFunc<int, FSharpFunc<String, Unit>>, Int32>,
level : FSharpOption<int> -> unit
```

Parameters

- `f`
  Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<String, Unit>>, Int32>`
  The optional output function to use. The default is `showLogInd`.

- `level`
  Type: `Microsoft.FSharp.Core.FSharpOption<int>`
  The optional indentation level. The default is 0.

See Also

- Reference
  Circuit Class
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CircuitFindIds Method

Find Ids below this point and the time needed to execute the circuit (internal)

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
member FindIds : 
    detail : int * 
    cntEmpty : FSharpOption<bool> -> Tuple<FSharpSet, int>
```

### Parameters

* **detail**  
  Type: System.Int32  
  What level of detail for when we hit a "Wrap" gate

* **cntEmpty**  
  Type: Microsoft.FSharp.Core.FSharpOption<bool>  
  An option allowing Empty gates to be counted like Native. The default is false, which is to not count Empty gates as Native.

### Return Value

Type: Tuple<FSharpSet<int>, int>  
Set of ids, total time to execute

### See Also

Reference  
Circuit Class
CircuitFold Method

Rewrites this circuit by identifying opportunities for parallelism and turning Seq elements into Par elements where possible. This makes for a better rendering.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.∗)

### Syntax

```fsharp
member Fold :  
    aggressive : FSharpOption<bool>  ->  Circuit
```

#### Parameters

**aggressive**  
Type: Microsoft.FSharp.Core.FSharpOption< bool >  
An option to fold as much as possible by decomposing this circuit into basic gates before folding. The default is to not decompose Wrap gates and other aggregates before folding.

#### Return Value

Type: Circuit  
The new folded circuit

### See Also

**Reference**  
[Microsoft.FSharp.Core.FSharpOption](https://fsharp.org/)  
[Microsoft.Research.Liquid Namespace](https://fsharp.org/liquid/)

CircuitGateCount Method

Gets the count of gates in circuit. Note that pure label gates -- that is, gates of type String -- are never counted.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member GateCount :
  doParallel : FSharpOption<bool> *
  gMatch : FSharpOption<FSharpFunc<Gate, bool>>
```

### Parameters

**doParallel**  
Type: `Microsoft.FSharp.Core.FSharpOption`\`bool\`  
An option to only count the longest sub-circuit of a parallel component, rather than adding all gates in all components. This is useful when computing gate depth. The default is false, which counts all gates, summing over parallel sub-circuits.

**gMatch**  
Type: `Microsoft.FSharp.Core.FSharpOption`\`FSharpFunc<Gate, bool>`  
An optional function to filter which gates should be counted. If a match function is provided, only gates that return true are included in the count. The default is to count all non-String gates.

### Return Value

Type: `Int32`
The count of low-level gates in the circuit

See Also

Reference

Circuit Class
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CircuitGrow Method

Creates an optimized "grown" version of this Circuit. The type and limits of the conversion are specified in the grow parameter.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
member Grow : 
    k : Microsoft.Research.Liquid.Ket * 
```

Parameters

- **k**
  - Type: Microsoft.Research.Liquid.Ket
  - A Ket this Circuit could be run with.

- **gp**
  - Type: Microsoft.Research.Liquid.GrowPars
  - The grow parameters to use. See the GrowPars type for details.

Return Value

Type: Circuit

A new, optimized Circuit

See Also

Reference
Circuit Class
Microsoft.Research.Liquid Namespace
CircuitGrowGates Method

Creates an equivalent circuit to this circuit by aggregating existing unitary gates into larger unitary gates. The new circuit will execute faster because it has fewer matrix applications required.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

Syntax

```fsharp
member GrowGates :
    k : Ket *
    gp : FSharpOption<GrowPars> -> Circuit
```

Parameters

- **k**
  - Type: Microsoft.Research.LiquidKet
  - A Ket this Circuit could be run with.

- **gp**
  - Type: Microsoft.FSharp.Core.FSharpOption<GrowPars>
  - Optional grow parameters. The default is to use all of the GrowPars defaults. See the GrowPars type for details.

Return Value

- Type: Circuit
  - New optimized Circuit

See Also

Reference

Circuit Class
CircuitGrowSingle Method

Converts this circuit into a single unitary gate, implemented by a single matrix. This allows maximum optimization of circuit execution. This circuit must be completely unitary to be converted into a single matrix. If it is not, an exception will be raised by this method.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```fsharp
member GrowSingle : gp : GrowPars -> Circuit
```

Parameters

\( gp \)
- Type: Microsoft.Research.Liquid.GrowPars
  Optional grow parameters. The default is to use all of the GrowPars defaults. See the \texttt{GrowPars} type for details.

Return Value
- Type: \texttt{Circuit}
  The new single-gate Circuit

See Also

Reference
- Circuit Class
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CircuitRemoveRedund Method

Creates a new circuit logically equivalent to this circuit, but with redundant gates removed. For instance, if this routine found a sequence of two X gates in succession on the same qubit, it would remove both of them from the result.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
member RemoveRedund : unit -> Circuit
```

**Return Value**

Type: Circuit

The new, trimmed circuit

### See Also

**Reference**

Circuit Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CircuitRender Method

Renders this circuit to a file.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5981.24943)

## Syntax

```fsharp
member Render :
    file : string *
    typ : FSharpOption<string> *
    detail : FSharpOption<int> *
    split : FSharpOption<float> *
    scale : FSharpOption<float> -> unit
```

## Parameters

**file**

Type: System.String  
The full name of the file to create, including the extension.

**typ**

Type: Microsoft.FSharp.Core.FSharpOption<String>  
The optional format for the rendered graphics. Possible values are:
- "qc": QCircuit Liquid format  
- "tikz": TikZ Liquid format for TeX and LaTeX  
- "svg": Vector graphics format for HTML

The default is "svg".

**detail**

Type: Microsoft.FSharp.Core.FSharpOption<Int32>  
An option specifying how many levels of Wrap to unwrap. The default is 999.
split
Type: Microsoft.FSharp.Core.FSharpOption<Double>
An option specifying what percentage of the total gates should go into each figure, if the circuit won't fit into a single figure. The default value varies with the figure size.

scale
Type: Microsoft.FSharp.Core.FSharpOption<Double>
An option specifying a scaling percentage for the rendering, with 100.0 being full size. The default value varies with the figure size.

Remarks
Detail is 0 for least, increasing wrap levels by each increment. That is, at level 0 only the top-level gate of a Wrap is displayed; at level 1, the top-level circuit that implements the Wrap is displayed; at level 2, the top-level circuit is displayed with its Wrap gates also unwrapped once; etc.

See Also
Reference
Circuit Class
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CircuitRenderHT Method

Renders a circuit to both svg, for HTML, and tikz, for TeX.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

▲ Syntax

```fsharp
member RenderHT : 
    file : string  *
    detail : FSharpOption<int>  *
    split : FSharpOption<float>  *
    scale : FSharpOption<float> -> unit
```

**Parameters**

- **file**
  
  Type: **System.String**
  
  The base name of the files to create. Appropriate extensions will be added for the created files: ".htm" for the SVG rendering, and ".tex" for the TikZ rendering.

- **detail**
  
  Type: **Microsoft.FSharp.Core.FSharpOption<int>**
  
  An option specifying how many levels of Wrap to unwrap. The default is 999.

- **split**
  
  Type: **Microsoft.FSharp.Core.FSharpOption<float>**
  
  An option specifying what percentage of the total gates should go into each figure, if the circuit won't fit into a single figure. The default value varies with the figure size.

- **scale**
  
  Type: **Microsoft.FSharp.Core.FSharpOption<float>**
An option specifying a scaling percentage for the rendering, with 100.0 being full size. The default value varies with the figure size.

Remarks

Detail is 0 for least, increasing wrap levels by each increment. That is, at level 0 only the top-level gate of a Wrap is displayed; at level 1, the top-level circuit that implements the Wrap is displayed; at level 2, the top-level circuit is displayed with its Wrap gates also unwrapped once; etc.

See Also

Reference
Circuit Class
Microsoft.Research.Liquid Namespace
CircuitRender(String, FSharpOption<String>, FSharpOption<int32>, FSharpOption<double>, FSharpOption<double>)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CircuitReverse Method

Reverses this circuit, which must contain only unitary gates. In addition, bottom-level gates are replaced by their adjoints. This method will raise an exception if this circuit contains a non-unitary gate.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4*)

Syntax

```
F# Copy

member Reverse : unit -> Circuit
```

Return Value

Type: Circuit
The resulting reversed circuit

See Also

Reference
Circuit Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CircuitRun Method

Runs this Circuit. The state of the Ket containing the passed-in qubits will be modified.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

F#

```fsharp
member Run : qubits : FSharpList<Qubit> -> unit
```

Parameters

`qubits`

Type: Microsoft.FSharp.Collections.FSharpList Qubit
The list of Qubits to operate on. These must correspond to the qubits the circuit was compiled with; that is, they must have the same qubit IDs as those.

See Also

Reference
Circuit Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CircuitToString Method

Returns a string representing the current circuit element.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```fsharp
abstract ToString : unit -> string
override ToString : unit -> string
```

**Return Value**

**Type:** String  
The string representation

### See Also

**Reference**

Circuit Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CircuitWires Method

Gets the list of wires (qubit IDs) touched by this circuit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member Wires : unit -> FSharpList<int>
```

### Return Value

**Type:** FSharpList<Int32>

A list of the wires used

### See Also

Reference

- Circuit Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CMat Class

A dense matrix of complex numbers.

Inheritance Hierarchy

SystemObject  Microsoft.Research.LiquidCMat

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0.*

Syntax

F#

[<SerializableAttribute>]
type CMat = class end

The CMat type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMat(Double, Double)</td>
<td>Creates a new matrix from arrays of real and imaginary parts.</td>
</tr>
<tr>
<td>CMat(Int32, FSharpListTupleInt32, Int32, Double, Double)</td>
<td>Creates a square matrix from a sparse list of elements.</td>
</tr>
<tr>
<td>CMat(Int32, FSharpOptionBoolean)</td>
<td>Creates a square identity or zero matrix.</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Gets an individual element of the matrix, as a Complex number.</td>
</tr>
<tr>
<td>Length</td>
<td>The number of rows or columns in the matrix, for square matrices.</td>
</tr>
<tr>
<td>LengthC</td>
<td>The number of columns in the matrix.</td>
</tr>
<tr>
<td>LengthR</td>
<td>The number of rows in the matrix.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds another matrix to this matrix. The two matrices must have the same number of rows and columns.</td>
</tr>
<tr>
<td>Adj</td>
<td>Computes the adjoint (complex conjugate transpose) of this matrix.</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears this matrix, setting it to a 0x0 matrix.</td>
</tr>
<tr>
<td>Copy</td>
<td>Makes a new copy, independent copy of this matrix.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>CreMat</td>
<td>Creates an array that can be filled in and passed to the array-based CMat constructor.</td>
</tr>
<tr>
<td>CreZer</td>
<td>Creates an array filled with zeros that can be filled in and passed to the array-based CMat constructor.</td>
</tr>
<tr>
<td>Div</td>
<td>Scales this matrix by a constant divisor. This matrix is updated in place, rather than a new matrix being created.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dumps this matrix. Note that there is no limit on the size of the result; every row and column entry is included.</td>
</tr>
<tr>
<td>DumpML</td>
<td>Dumps this matrix in Matlab format for debugging.</td>
</tr>
<tr>
<td>DumpNarrow</td>
<td>Dumps this matrix. This method will produce a relatively compact representation of the matrix.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>FixUnitary</strong></td>
<td>Makes this matrix closer to being unitary by applying an iterative correction.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object.</strong>)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object.</strong>)</td>
</tr>
<tr>
<td><strong>Kron(CMat)</strong></td>
<td>Computes the Kronecker Product of this matrix with another matrix.</td>
</tr>
<tr>
<td><strong>Kron(Int32, FSharpOptionBoolean)</strong></td>
<td>Computes the Kronecker Product of this matrix with an identity matrix</td>
</tr>
<tr>
<td><strong>Mul(Double)</strong></td>
<td>Scales this matrix by a constant multiplier. This matrix is updated in place, rather than a new matrix being created.</td>
</tr>
<tr>
<td><strong>Mul(CMat)</strong></td>
<td>Computes the product of this matrix with another matrix. This matrix is on the left-hand side of the product, and the argument matrix is on the right-hand side. This matrix must have the same number of columns as the other matrix has</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Mul(CVec)</td>
<td>Calculates the product of this matrix and a vector. The vector's length must match the number of columns in this matrix.</td>
</tr>
<tr>
<td>Power</td>
<td>Computes an integer power of this matrix. NOTE: This destroys the contents of the current matrix.</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a matrix from a stream.</td>
</tr>
<tr>
<td>ToString</td>
<td>Gets a string representation of this matrix. Note that there is no limit on the size of the result; every row and column entry is included. (Overrides <code>Object.ToString</code>.)</td>
</tr>
<tr>
<td>UnitaryError</td>
<td>Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality</td>
</tr>
</tbody>
</table>
of different rows is not checked.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>Writes this matrix to a stream for serialization.</td>
</tr>
</tbody>
</table>

## Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdditionAssignment</td>
<td>Adds one matrix to another.</td>
</tr>
<tr>
<td>DivisionAssignment</td>
<td>Scales a matrix by a constant divisor. The matrix is updated in place, rather than a new matrix being created.</td>
</tr>
<tr>
<td>Multiply(CMat, CMat)</td>
<td>Computes the product of two matrices. The first matrix must have the same number of columns as the second matrix has rows.</td>
</tr>
<tr>
<td>Multiply(CMat, CVec)</td>
<td>Computes the product of a matrix and a vector. The vector’s length must match the number of columns in the matrix.</td>
</tr>
<tr>
<td>MultiplyAssignment</td>
<td>Scales a matrix by a constant multiplier. The matrix is updated in place, rather than a new matrix being created.</td>
</tr>
</tbody>
</table>
MultiplyBang

Computes the Kronecker product of two matrices

See Also

Reference
Microsoft.Research.Liquid Namespace
[T::Microsoft.Research.Liquid.CSMat]

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CMat Constructor

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMat(Double, Double)</td>
<td>Creates a new matrix from arrays of real and imaginary parts.</td>
</tr>
<tr>
<td>CMat(Int32, FSharpListTupleInt32, Int32, Double, Double)</td>
<td>Creates a square matrix from a sparse list of elements.</td>
</tr>
<tr>
<td>CMat(Int32, FSharpOptionBoolean)</td>
<td>Creates a square identity or zero matrix.</td>
</tr>
</tbody>
</table>

See Also

Reference
CMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMat Constructor (Double, Double)

Creates a new matrix from arrays of real and imaginary parts.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
new :  
    rs : float[][]  *  
    is : float[][]  ->  CMat
```

### Parameters

**rs**

Type: SystemDouble  
The real part of each matrix element.

**is**

Type: SystemDouble  
The real part of each matrix element.

### See Also

- Reference
- CMat Class
- CMat Overload
- Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CMat Constructor (Int32, FSharpList<Tuple<Int32, Int32, Double, Double>)

Creates a square matrix from a sparse list of elements.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
new :  
    len : int *  
    xyris : FSharpList<Tuple<int, int, float, float>
```

### Parameters

**len**  
Type: System.Int32  
The row (or column) length of the matrix.

**xyris**  
Type: Microsoft.FSharp.Collections.FSharpList<Tuple<Int32, Int32, Double, Double>  
A list of element location and value tuples, with elements in the order (row,col,real,imag).

### See Also

Reference  
CMat Class
CMat Constructor (Int32, FSharpOptionBoolean)

Creates a square identity or zero matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
new :  
    n : int *  
    zero : FSharpOption<bool> -> CMat
```

### Parameters

**n**
- Type: `System.Int32`  
  The row (or column) length of the matrix.

**zero**
- Type: `Microsoft.FSharp.Core.FSharpOption<bool>`  
  Option to initialize to a zero matrix rather than to the identity matrix. The default is to initialize to the identity.

### See Also

**Reference**
- CMat Class
- CMat Overload
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMat Properties

The `CMat` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Gets an individual element of the matrix, as a Complex number.</td>
</tr>
<tr>
<td>Length</td>
<td>The number of rows or columns in the matrix, for square matrices.</td>
</tr>
<tr>
<td>LengthC</td>
<td>The number of columns in the matrix.</td>
</tr>
<tr>
<td>LengthR</td>
<td>The number of rows in the matrix.</td>
</tr>
</tbody>
</table>

## See Also

Reference

`CMat Class`

`Microsoft.Research.Liquid Namespace`

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMat.Item Property

Gets an individual element of the matrix, as a Complex number.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5981.24943)

### Syntax

```fsharp
member Item : Complex with get, set
```

### Parameters

- **i**
  - Type: System.Int32
  - The row index.

- **j**
  - Type: System.Int32
  - The column index.

### Return Value

- **Type:** Complex
  - The complex value of the matrix element.

### See Also

**Reference**
- CMat Class
- Microsoft.Research.Liquid Namespace
CMatLength Property

The number of rows or columns in the matrix, for square matrices.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5)

Syntax

F#

```fsharp
member Length : int with get
```

Property Value

Type: Int32

See Also

Reference

CMat Class
Microsoft.Research.Liquid Namespace
CMatLengthC Property

The number of columns in the matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)&

### Syntax

```
F#

member LengthC : int with get
```

### Property Value

**Type:** Int32

### See Also

**Reference**
- CMat Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatLengthR Property

The number of rows in the matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

⚠ **Syntax**

```fsharp
member LengthR : int with get
```

**Property Value**

Type: Int32

⚠ **See Also**

Reference

CMat Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMat Methods

The **CMat** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds another matrix to this matrix. The two matrices must have the same number of rows and columns.</td>
</tr>
<tr>
<td><strong>Adj</strong></td>
<td>Computes the adjoint (complex conjugate transpose) of this matrix.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Clears this matrix, setting it to a 0x0 matrix.</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Makes a new copy, independent copy of this matrix.</td>
</tr>
<tr>
<td><strong>CreMat</strong></td>
<td>Creates an array that can be filled in and passed to the array-based CMat constructor.</td>
</tr>
<tr>
<td><strong>CreZer</strong></td>
<td>Creates an array filled with zeros that can be filled in and passed to the array-based CMat constructor.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Div</strong></td>
<td>Scales this matrix by a constant divisor. This matrix is updated in place, rather than a new matrix being created.</td>
</tr>
<tr>
<td><strong>Dump</strong></td>
<td>Dumps this matrix. Note that there is no limit on the size of the result; every row and column entry is included.</td>
</tr>
<tr>
<td><strong>DumpML</strong></td>
<td>Dumps this matrix in Matlab format for debugging.</td>
</tr>
<tr>
<td><strong>DumpNarrow</strong></td>
<td>Dumps this matrix. This method will produce a relatively compact representation of the matrix.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>FixUnitary</strong></td>
<td>Makes this matrix closer to being unitary by applying an iterative correction.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the</td>
</tr>
</tbody>
</table>
### Kron(CMat)
Computes the Kronecker Product of this matrix with another matrix.

### Kron(Int32, FSharpOptionBoolean)
Computes the Kronecker Product of this matrix with an identity matrix.

### Mul(Double)
Scales this matrix by a constant multiplier. This matrix is updated in place, rather than a new matrix being created.

### Mul(CMat)
Computes the product of this matrix with another matrix. This matrix is on the left-hand side of the product, and the argument matrix is on the right-hand side. This matrix must have the same number of columns as the other matrix has rows.

### Mul(CVec)
Calculates the product of this matrix and a vector. The vector's length must match the number of columns in this matrix.

### Power
Computes an integer power of this matrix. NOTE: This destroys the current instance. (Inherited from Object.)
contents of the current matrix.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Read</strong></td>
<td>Reads a matrix from a stream.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Gets a string representation of this matrix. Note that there is no limit on the size of the result; every row and column entry is included. (Overrides ObjectToString.)</td>
</tr>
<tr>
<td><strong>UnitaryError</strong></td>
<td>Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes this matrix to a stream for serialization.</td>
</tr>
</tbody>
</table>

See Also
CMatAdd Method

Adds another matrix to this matrix. The two matrices must have the same number of rows and columns.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
member Add : 
  m2 : CMat -> unit
```

### Parameters

- **m2**
  
  **Type:** Microsoft.Research.Liquid.CMat  
  The matrix to add to this one.

### See Also

**Reference**
- CMat Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatAdj Method

Computes the adjoint (complex conjugate transpose) of this matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
**Version:** 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member Adj : unit -> CMat
```

#### Return Value

Type: **CMat**  
The adjoint matrix

### See Also

Reference  
**CMat Class**  
**Microsoft.Research.Liquid Namespace**

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatClear Method

Clears this matrix, setting it to a 0x0 matrix.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

⚠️ **Syntax**

```fsharp
member Clear : unit -> unit
```

⚠️ **See Also**

Reference
- CMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatCopy Method

Makes a new copy, independent copy of this matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0*)

### Syntax

```fsharp
member Copy : unit -> CMat
```

**Return Value**

**Type:** CMat  
The new matrix

### See Also

- **Reference**
  - CMat Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatCreMat Method

Creates an array that can be filled in and passed to the array-based CMat constructor.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member CreMat :
    lenR : int *
    lenC : int *
    f : FSharpFunc<int, FSharpFunc<int, float>
```

### Parameters

**lenR**
- Type: System.Int32
- The number of rows

**lenC**
- Type: System.Int32
- The number of columns

**f**
- Type: Microsoft.FSharp.Core.FSharpFunc<int, Microsoft.FSharp.Core.FSharpFunc<int, float>
- The function to call to compute the elements of the array. It will get called with the row first and then the column (curried).

### Return Value

**Type:** Double
**The new array**
CMatCreZer Method

Creates an array filled with zeros that can be filled in and passed to the array-based CMat constructor.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member CreZer :
    lenR : int *
    lenC : int -> float[][]
```

### Parameters

**lenR**
Type: System.Int32
The number of rows

**lenC**
Type: System.Int32
The number of columns

### Return Value
Type: Double
The new array

### See Also

Reference
CMat Class
Microsoft.Research.Liquid Namespace
CMatDiv Method

Scales this matrix by a constant divisor. This matrix is updated in place, rather than a new matrix being created.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```f#
member Div :
  div : float -> CMat
```

**Parameters**

*div*

Type: System.Double
The scale divisor. The matrix is multiplied by 1/div.

**Return Value**

Type: CMat
This matrix, as updated after scaling

▲ See Also

**Reference**
CMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatDump Method

Dumps this matrix. Note that there is no limit on the size of the result; every row and column entry is included.

**Namespace**: Microsoft.Research.Liquid  
**Assembly**: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Dump :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<string, Unit>>
    level : FSharpOption<int> -> unit
```

### Parameters

- **f**
  - Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<string, Unit>>`
  - The optional output function to use. The default is `showLogInd`.

- **level**
  - Type: `Microsoft.FSharp.Core.FSharpOption<int>`
  - The optional indentation level. The default is 0.

### See Also

- Reference
  - CMat Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatDumpML Method

Dumps this matrix in Matlab format for debugging.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member DumpML : 
  f : FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<String, Unit>> *
  level : FSharpOption<int> *
  prec : FSharpOption<int> -> unit
```

### Parameters

- **f**
  Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<String, Unit>>`
  The optional output function to use. The default is `showLogInd`.

- **level**
  Type: `Microsoft.FSharp.Core.FSharpOption<int>`
  The optional indentation level. The default is 0.

- **prec**
  Type: `Microsoft.FSharp.Core.FSharpOption<int>`
  precision 0=low 1=normal 2=high 3=full (optional=1)

### See Also

- **Reference**  
  CMat Class  
  Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CMatDumpNarrow Method

Dumps this matrix. This method will produce a relatively compact representation of the matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member DumpNarrow :  
  f : FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<string, Unit>>  
  level : FSharpOption<int> -> unit
```

### Parameters

- **f**
  Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<string, Unit>>`  
The optional output function to use. The default is `showLogInd`.

- **level**
  Type: `Microsoft.FSharp.Core.FSharpOption<int>`  
The optional indentation level. The default is 0.

### See Also

- **Reference**
  - CMat Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatFixUnitary Method

Makes this matrix closer to being unitary by applying an iterative correction.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

```
Member FixUnitary :
    toI : FSharpOption<float> *
    maxIter : FSharpOption<int> -> Tuple<int, double>
```

Parameters

tol
Type: Microsoft.FSharp.Core.FSharpOption<Double>
An optional desired tolerance for the deviation from unitarity, as measured by UnitaryError. The default is 1.0e-13.

maxIter
Type: Microsoft.FSharp.Core.FSharpOption<Int32>
An optional maximum number of iterations to perform. The default is 5.

Return Value
Type: Tuple<Int32, Double>
A tuple of the number of iterations performed, for cost estimates, and the final deviation from unitarity.

See Also
Reference

CMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatKron Method

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kron(CMat)</strong></td>
<td>Computes the Kronecker Product of this matrix with another matrix.</td>
</tr>
<tr>
<td><strong>Kron(Int32, FSharpOptionBoolean)</strong></td>
<td>Computes the Kronecker Product of this matrix with an identity matrix</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- CMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatKron Method (CMat)

Computes the Kronecker Product of this matrix with another matrix.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
member Kron : m2 : CMat -> CMat
```

### Parameters

\( m2 \)

Type: Microsoft.Research.Liquid.CMat

The right-hand side matrix in the Kronecker product.

### Return Value

Type: CMat

The result of the Kronecker product

### See Also

- **Reference**
  - CMat Class
  - Kron Overload
  - Microsoft.Research.Liquid.Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatKron Method (Int32, FSharpOption<Boolean>)

Computes the Kronecker Product of this matrix with an identity matrix

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```
F# Copy

member Kron :
    len : int *
    left : FSharpOption<bool> -> CMat
```

### Parameters

**len**

- **Type:** System.Int32
- The dimension of the identity matrix (N of NxN).

**left**

- **Type:** Microsoft.FSharp.Core.FSharpOption<Boolean>
- An option that, if true, specifies that the identity matrix should be on the left side of the Kronecker product. The default is false, which means that the identity is on the right.

### Return Value

- **Type:** CMat
- The result of the Kronecker product

### See Also

Reference
CMatMul Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mul(Double)</td>
<td>Scales this matrix by a constant multiplier. This matrix is updated in place, rather than a new matrix being created.</td>
</tr>
<tr>
<td>Mul(CMat)</td>
<td>Computes the product of this matrix with another matrix. This matrix is on the left-hand side of the product, and the argument matrix is on the right-hand side. This matrix must have the same number of columns as the other matrix has rows.</td>
</tr>
<tr>
<td>Mul(CVec)</td>
<td>Calculates the product of this matrix and a vector. The vector’s length must match the number of columns in this matrix.</td>
</tr>
</tbody>
</table>

See Also

- Reference
- CMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CMatMul Method (Double)

Scales this matrix by a constant multiplier. This matrix is updated in place, rather than a new matrix being created.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.24943.0)

Syntax

```fsharp
member Mul : float -> CMat
```

Parameters

`mul`

Type: System.Double
The scale factor to multiply by.

Return Value

Type: CMat
This matrix, as updated after scaling

See Also

- Reference
  - CMat Class
  - Mul Overload
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatMul Method (CMat)

Computes the product of this matrix with another matrix. This matrix is on the left-hand side of the product, and the argument matrix is on the right-hand side. This matrix must have the same number of columns as the other matrix has rows.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Mul : m2 : CMat -> CMat
```

### Parameters

**m2**

Type: Microsoft.Research.Liquid.CMat

The second matrix.

### Return Value

Type: CMat

The resulting product matrix.

### See Also

**Reference**

CMat Class
Mul Overload
Microsoft.Research.Liquid Namespace
CMatMul Method (CVec)

Calculates the product of this matrix and a vector. The vector's length must match the number of columns in this matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Mul : CVec -> CVec
```

### Parameters

- **v**
  - Type: Microsoft.Research.Liquid.CVec
  - The vector to multiply

### Return Value

- Type: CVec
  - The resulting product vector

### See Also

- Reference
  - CMat Class
  - Mul Overload
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatPower Method

Computes an integer power of this matrix. NOTE: This destroys the contents of the current matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member Power : n : int -> CMat
```

**Parameters**

\( n \)

Type: System.Int32  
The power to raise this matrix to; must be greater than 0.

**Return Value**

Type: CMat  
The resulting matrix

### See Also

Reference  
CMat Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatRead Method

Reads a matrix from a stream.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
static member Read :  
br : BinaryReader  ->  CMat
```

### Parameters

*br*

Type: System.IO.BinaryReader  
A BinaryReader to deserialize a matrix from.

### Return Value

Type: CMat  
The new matrix

### See Also

Reference  
CMat Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatToString Method

Gets a string representation of this matrix. Note that there is no limit on the size of the result; every row and column entry is included.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```fsharp
abstract ToString : unit -> string
override ToString : unit -> string
```

**Return Value**

Type: **String**
The string representation

### See Also

**Reference**

CMat Class  
Microsoft.Research.Liquid Namespace
CMatUnitaryError Method

Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0

Syntax

```fsharp
member UnitaryError : unit -> float
```

Return Value
Type: **Double**
The worst deviation from 1.

See Also

Reference
CMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatWrite Method

Writes this matrix to a stream for serialization.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

F# Copy

member Write :
  bw : BinaryWriter -> unit

Parameters

bw
  Type: System.IO.BinaryWriter
  A BinaryWriter to serialize this matrix to.

See Also

Reference
CMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
# CMat Operators

The `CMat` type exposes the following members.

## Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AdditionAssignment</code></td>
<td>Adds one matrix to another.</td>
</tr>
<tr>
<td><code>DivisionAssignment</code></td>
<td>Scales a matrix by a constant divisor. The matrix is updated in place, rather than a new matrix being created.</td>
</tr>
<tr>
<td><code>Multiply(CMat, CMat)</code></td>
<td>Computes the product of two matrices. The first matrix must have the same number of columns as the second matrix has rows.</td>
</tr>
<tr>
<td><code>Multiply(CMat, CVec)</code></td>
<td>Computes the product of a matrix and a vector. The vector's length must match the number of columns in the matrix.</td>
</tr>
<tr>
<td><code>MultiplyAssignment</code></td>
<td>Scales a matrix by a constant multiplier. The matrix is updated in place, rather than a new matrix being created.</td>
</tr>
<tr>
<td><code>MultiplyBang</code></td>
<td>Computes the Kronecker</td>
</tr>
</tbody>
</table>
product of two matrices

See Also

Reference
CMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CMatAdditionAssignment Operator

Adds one matrix to another.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```fsharp
F# does not support this operator.
```

### Parameters

- **m1**
  - Type: Microsoft.Research.Liquid.CMat
  - The left matrix. This matrix is modified.

- **m2**
  - Type: Microsoft.Research.Liquid.CMat
  - The right matrix

### See Also

- Reference
  - CMat Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
**CMatDivisionAssignment Operator**

Scales a matrix by a constant divisor. The matrix is updated in place, rather than a new matrix being created.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
F# does not support this operator.
```

### Parameters

- **m1**
  - Type: Microsoft.Research.Liquid.CMat
  - The matrix to scale.

- **div**
  - Type: System.Double
  - The scale divisor. The matrix is multiplied by 1/div.

### See Also

- Reference  
  - CMat Class  
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiply(CMat, CMat)</td>
<td>Computes the product of two matrices. The first matrix must have the same number of columns as the second matrix has rows.</td>
</tr>
<tr>
<td>Multiply(CMat, CVec)</td>
<td>Computes the product of a matrix and a vector. The vector's length must match the number of columns in the matrix.</td>
</tr>
</tbody>
</table>

### See Also

- **Reference**
  - CMat Class
  - Microsoft.Research.Liquid Namespace
CMatMultiply Operator (CMat, CMat)

Computes the product of two matrices. The first matrix must have the same number of columns as the second matrix has rows.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4*)

### Syntax

```
static let inline (*)
    m1 : CMat *
    m2 : CMat : CMat
```

### Parameters

- **m1**
  - Type: Microsoft.Research.LiquidCMat  
  - The left (first) matrix

- **m2**
  - Type: Microsoft.Research.LiquidCMat  
  - The right (second) matrix

### Return Value

- Type: CMat  
  - The resulting product matrix

### See Also

- Reference  
  - CMat Class
CMatMultiply Operator (CMat, CVec)

Computes the product of a matrix and a vector. The vector's length must match the number of columns in the matrix.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```fsharp
static let inline (*)
    m : CMat *
    v : CVec  : CVec
```

### Parameters

- **m**
  
  Type: Microsoft.Research.LiquidCMat
  
  The matrix

- **v**
  
  Type: Microsoft.Research.LiquidCVec
  
  The vector

### Return Value

Type: CVec

The resulting product vector

### See Also

- Reference
  - CMat Class
CMatMultiplyAssignment Operator

Scales a matrix by a constant multiplier. The matrix is updated in place, rather than a new matrix being created.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```
F# does not support this operator.
```

### Parameters

**m1**
- Type: Microsoft.Research.Liquid.CMat
  - The matrix to scale.

**mul**
- Type: System.Double
  - The constant to multiply by.

### See Also

- **Reference**  
  - CMat Class  
  - Microsoft.Research.Liquid.Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CMatMultiplyBang Operator

Computes the Kronecker product of two matrices

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

F# does not support this operator.

Parameters

$m1$
Type: Microsoft.Research.Liquid.CMat
The left matrix

$m2$
Type: Microsoft.Research.Liquid.CMat
The right matrix

Return Value
Type: CMat
The result of the Kronecker product

See Also

Reference
CMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Complex Structure

Data type for complex numbers.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
[<SealedAttribute>]  
[<SerializableAttribute>]  
type Complex =  
    struct  
        interface IEquatable<Complex>  
        interface IStructuralEquatable  
        interface IComparable<Complex>  
        interface IComparable  
        interface IStructuralComparable  
    end
```

The `Complex` type exposes the following members.

### Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>Complex</td>
</tr>
</tbody>
</table>

### Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Gets the imaginary part of this Complex number.</td>
</tr>
<tr>
<td>I</td>
<td>The square root of negative one, as a Complex number.</td>
</tr>
<tr>
<td>MCC</td>
<td>Gets the squared magnitude of this Complex number.</td>
</tr>
<tr>
<td>One</td>
<td>One, as a Complex number</td>
</tr>
<tr>
<td>r</td>
<td>Gets the real part of this Complex number.</td>
</tr>
<tr>
<td>Tol</td>
<td>Tolerance for comparing two Complex numbers. This is used in various places in the system.</td>
</tr>
<tr>
<td>Zero</td>
<td>Zero, as a Complex number</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conj</td>
<td>Computes the complex conjugate of this Complex number.</td>
</tr>
<tr>
<td>Copy</td>
<td>Duplicates this Complex number in a new instance.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dumps this Complex number to the console and log with an optional indentation.</td>
</tr>
<tr>
<td>Equals</td>
<td>Indicates whether this instance</td>
</tr>
</tbody>
</table>
and a specified object are equal. (Inherited from `ValueType`.)

- **GetHashCode**: Returns the hash code for this instance. (Inherited from `ValueType`.)
- **GetType**: Gets the `Type` of the current instance. (Inherited from `Object`.)
- **Narrow**: Converts this Complex number to a short human-readable string.
- **ToString**: Converts this Complex number to a human-readable string. (Overrides `ValueType.ToString`.)
- **Wide**: Converts this Complex number to a string suitable for input to other programs.

### Top

#### Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Addition</strong></td>
<td>Adds two Complex numbers.</td>
</tr>
<tr>
<td><strong>Multiply(Double, Complex)</strong></td>
<td>Multiplies a real and a Complex number.</td>
</tr>
<tr>
<td><strong>Multiply(Complex, Complex)</strong></td>
<td>Multiplies two Complex numbers.</td>
</tr>
<tr>
<td><strong>Multiply(Complex, Double)</strong></td>
<td>Multiplies a Complex and a real number.</td>
</tr>
<tr>
<td>Operation</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>Subtraction</strong></td>
<td>Subtracts one Complex number from another.</td>
</tr>
<tr>
<td><strong>TwiddleTwiddle</strong></td>
<td>Gets the complex conjugate of a Complex number.</td>
</tr>
<tr>
<td><strong>UnaryNegation</strong></td>
<td>Gets the negation of a Complex number.</td>
</tr>
</tbody>
</table>

**See Also**

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Complex Constructor

Constructs a complex number from its real and imaginary parts.

**Namespace**: Microsoft.Research.Liquid  
**Assembly**: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
new :  
    r : float *  
    i : float -> Complex
```

### Parameters

**r**  
Type: **SystemDouble**  
Real value

**i**  
Type: **SystemDouble**  
Imaginary value

### See Also

**Reference**
- Complex Structure
- Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Complex Properties

The `Complex` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Gets the imaginary part of this <code>Complex</code> number.</td>
</tr>
<tr>
<td>I</td>
<td>The square root of negative one, as a <code>Complex</code> number.</td>
</tr>
<tr>
<td>MCC</td>
<td>Gets the squared magnitude of this <code>Complex</code> number.</td>
</tr>
<tr>
<td>One</td>
<td>One, as a <code>Complex</code> number</td>
</tr>
<tr>
<td>r</td>
<td>Gets the real part of this <code>Complex</code> number.</td>
</tr>
<tr>
<td>Tol</td>
<td>Tolerance for comparing two <code>Complex</code> numbers. This is used in various places in the system.</td>
</tr>
<tr>
<td>Zero</td>
<td>Zero, as a <code>Complex</code> number</td>
</tr>
</tbody>
</table>

See Also

Reference

- `Complex Structure`
- `Microsoft.Research.Liquid Namespace`
Complexi Property

Gets the imaginary part of this Complex number.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```f#
member i : float with get
```

Property Value

Type: Double

The imaginary part

See Also

Reference

Complex Structure
Microsoft.Research.Liquid Namespace
ComplexI Property

The square root of negative one, as a Complex number.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

#### Syntax

```fsharp
static member I : Complex with get
```

**Property Value**  
**Type:** Complex

#### See Also

**Reference**  
[Complex Structure](#)  
[Microsoft.Research.Liquid Namespace](#)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
ComplexMCC Property

Gets the squared magnitude of this Complex number.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member MCC : float with get
```

**Return Value**

**Type:** Double  
The squared magnitude (r*r+i*i)

### See Also

**Reference**
- Complex Structure
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
ComplexOne Property

One, as a Complex number

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member One : Complex with get
```

Property Value

Type: Complex

### See Also

Reference

Complex Structure  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Complex r Property

Gets the real part of this Complex number.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```
F#

member r : float with get
```

### Property Value

**Type:** Double

The real part

### See Also

Reference

Complex Structure

Microsoft.Research.Liquid Namespace
ComplexTol Property

Tolerance for comparing two Complex numbers. This is used in various places in the system.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```
static member Tol : float with get, set
```

### Property Value

Type: **Double**

### See Also

- Reference
  - Complex Structure
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
ComplexZero Property

Zero, as a Complex number

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member Zero : Complex with get
```

**Property Value**

Type: Complex

### See Also

**Reference**

Complex Structure  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Complex Methods

The **Complex** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conj</td>
<td>Computes the complex conjugate of this Complex number.</td>
</tr>
<tr>
<td>Copy</td>
<td>Duplicates this Complex number in a new instance.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dumps this Complex number to the console and log with an optional indentation.</td>
</tr>
<tr>
<td>Equals</td>
<td>Indicates whether this instance and a specified object are equal. (Inherited from <strong>ValueType.</strong>)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns the hash code for this instance. (Inherited from <strong>ValueType.</strong>)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object.</strong>)</td>
</tr>
<tr>
<td>Narrow</td>
<td>Converts this Complex number to a short human-readable string.</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts this Complex number to a human-readable string.</td>
</tr>
<tr>
<td>Wide</td>
<td>Converts this Complex number to a string suitable for input to other programs.</td>
</tr>
</tbody>
</table>

See Also

Reference

Complex Structure  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
ComplexConj Method

Computes the complex conjugate of this Complex number.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```
F# Copy

member Conj : unit -> Complex
```

Return Value
Type: Complex
The conjugate

See Also

Reference
Complex Structure
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
ComplexCopy Method

Duplicates this Complex number in a new instance.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```f#
member Copy : unit -> Complex
```

**Return Value**

Type: **Complex**

The new Complex

### See Also

Reference

Complex Structure

Microsoft.Research.Liquid Namespace
ComplexDump Method

Dumps this Complex number to the console and log with an optional indentation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

## Syntax

```fsharp
member Dump :
  f : FSharpOption<FSharpFunc<int, FSharpFunc<String, Unit>>, Int32>,
  level : FSharpOption<int> -> unit
```

### Parameters

- **f**  
  The optional output function to use. The default is `showLogInd`.

- **level**  
  Type: `Microsoft.FSharp.Core.FSharpOption<Microsoft.FSharp.Core.Int32>`
  The optional indentation level. The default is 0.

## See Also

- Reference  
  Complex Structure  
  Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

ComplexNarrow Method

Converts this Complex number to a short human-readable string.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

 mái Syntax

F#

```fsharp
member Narrow : unit -> string
```

Return Value  
Type: **String**  
The formatted string representation

See Also

**Reference**  
Complex Structure  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

ComplexToString Method

Converts this Complex number to a human-readable string.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```
abstract ToString : unit -> string
override ToString : unit -> string
```

### Return Value

**Type:** String  
The formatted string representation

### See Also

**Reference**
- Complex Structure
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
ComplexWide Method

Converts this Complex number to a string suitable for input to other programs.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
member Wide : unit -> string
```

### Return Value

Type: **String**  
The formatted string representation

### See Also

**Reference**

Complex Structure  
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Complex Operators

The Complex type exposes the following members.

Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>Adds two Complex numbers.</td>
</tr>
<tr>
<td>Multiply(Double, Complex)</td>
<td>Multiplies a real and a Complex number.</td>
</tr>
<tr>
<td>Multiply(Complex, Complex)</td>
<td>Multiplies two Complex numbers.</td>
</tr>
<tr>
<td>Multiply(Complex, Double)</td>
<td>Multiplies a Complex and a real number.</td>
</tr>
<tr>
<td>Subtraction</td>
<td>Subtracts one Complex number from another.</td>
</tr>
<tr>
<td>TwiddleTwiddle</td>
<td>Gets the complex conjugate of a Complex number.</td>
</tr>
<tr>
<td>UnaryNegation</td>
<td>Gets the negation of a Complex number.</td>
</tr>
</tbody>
</table>

See Also

Reference
Complex Structure
Microsoft.Research.Liquid Namespace
ComplexAddition Operator

Adds two Complex numbers.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

### Syntax

```fsharp
static let inline (+)
    a : Complex *
    b : Complex : Complex
```

### Parameters

- **a**  
  Type: Microsoft.Research.LiquidComplex  
  The left-hand operand

- **b**  
  Type: Microsoft.Research.LiquidComplex  
  The right-hand operand

### Return Value

- **Type:** Complex  
  The sum of the two operands

### See Also

- **Reference**  
  Complex Structure  
  Microsoft.Research.Liquid Namespace
# Complex Multiply Operator

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiply(Double, Complex)</td>
<td>Multiplies a real and a Complex number.</td>
</tr>
<tr>
<td>Multiply(Complex, Complex)</td>
<td>Multiplies two Complex numbers.</td>
</tr>
<tr>
<td>Multiply(Complex, Double)</td>
<td>Multiplies a Complex and a real number.</td>
</tr>
</tbody>
</table>

## See Also

Reference
- `Complex Structure`
- `Microsoft.Research.Liquid Namespace`

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

ComplexMultiply Operator (Double, Complex)

Multiplies a real and a Complex number.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4*)

### Syntax

```
static let inline (*)&  
  a : float *  
  b : Complex : Complex
```

### Parameters

- **a**
  - Type: System.Double
  - The real number

- **b**
  - Type: Microsoft.Research.Liquid.Complex
  - The Complex number

### Return Value

Type: Complex  
The product of the two numbers

### See Also

- **Reference**  
  - Complex Structure  
  - Multiply Overload
ComplexMultiply Operator (Complex, Complex)

Multiplies two Complex numbers.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
static let inline (*)
  a : Complex *
  b : Complex , Complex
```

### Parameters

**a**  
Type: Microsoft.Research.LiquidComplex  
Left hand operand.

**b**  
Type: Microsoft.Research.LiquidComplex  
Right hand operand.

### Return Value

Type: Complex  
The product of the two numbers

### See Also

Reference  
Complex Structure  
Multiply Overload
Complex Multiply Operator (Complex, Double)

Multiplies a Complex and a real number.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```
static let inline (*)
    a : Complex *
    b : float  : Complex
```

### Parameters

**a**  
Type: Microsoft.Research.LiquidComplex  
The Complex number

**b**  
Type: SystemDouble  
The real number

### Return Value

Type: Complex  
The product of the two numbers

### See Also

Reference  
Complex Structure  
Multiply Overload
ComplexSubtraction Operator

Subtracts one Complex number from another.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static let inline (-)  
  a : Complex *  
  b : Complex : Complex
```

#### Parameters

- **a**
  - Type: Microsoft.Research.LiquidComplex  
    The left-hand operand

- **b**
  - Type: Microsoft.Research.LiquidComplex  
    The right-hand operand

#### Return Value

Type: Complex  
The difference of the two operands

### See Also

- Reference  
  Complex Structure  
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
ComplexTwiddleTwiddleOperator

Gets the complex conjugate of a Complex number.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

F# does not support this operator.

Parameters

a

Type: Microsoft.Research.LiquidComplex
The Complex number to conjugate.

Return Value

Type: Complex
The conjugated Complex number

See Also

Reference
Complex Structure
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
ComplexUnaryNegation Operator

Gets the negation of a Complex number.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```fsharp
static let inline (-)  
  a : Complex  : Complex
```

### Parameters

- **a**  
  Type: Microsoft.Research.Liquid.Complex  
  The Complex number to negate

### Return Value

- **Type:** Complex  
  The negated Complex number, (-r,-i)

### See Also

- **Reference**  
  Complex Structure  
  Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMat Class

A sparse matrix of complex numbers.

Inheritance Hierarchy

```
System.Object  Microsoft.Research.LiquidCSMat
```

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.8)

Syntax

```
[<SerializableAttribute>]
type CSMat = class end
```

The CSMat type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CSMat(Int32, FSharpList&lt;Tuple&lt;Int32, Int32, Double, Double)&gt;)</code></td>
<td>Creates a matrix from a list of elements.</td>
</tr>
<tr>
<td><code>CSMat(Int32, FSharpOptionBoolean)</code></td>
<td>Creates a new square identity or zero matrix.</td>
</tr>
<tr>
<td><code>CSMat(CMat, FSharpOptionDouble)</code></td>
<td>Creates a sparse matrix from a dense matrix.</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Gets an element of the matrix.</td>
</tr>
<tr>
<td>Length</td>
<td>The dimension of this matrix; that is, the number of rows or columns.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adj</td>
<td>Computes the adjoint (complex conjugate transpose) of this matrix.</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears out this matrix, setting all elements to zero.</td>
</tr>
<tr>
<td>Copy</td>
<td>Makes a copy of this matrix.</td>
</tr>
<tr>
<td>Dense</td>
<td>Creates a dense matrix from this sparse matrix.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dumps this matrix as a list of row-column indices and element complex values. Note that all non-zero elements are listed, so the output may be quite long.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DumpDense</strong></td>
<td>Dumps this sparse matrix in the same format as a dense matrix.</td>
</tr>
<tr>
<td><strong>DumpMCC</strong></td>
<td>Dumps this matrix as a list of row-column indices and element squared magnitudes. Note that all non-zero elements are listed, so the output may be quite long.</td>
</tr>
<tr>
<td><strong>DumpML</strong></td>
<td>Dumps this matrix in MatLab format.</td>
</tr>
<tr>
<td><strong>Equals(Object)</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Equals(CSMat, FSharpOptionDouble)</strong></td>
<td>Determines whether this matrix is equal to another.</td>
</tr>
<tr>
<td><strong>Filled</strong></td>
<td>Return indicies of filled entries in the sparse matrix (may include zeros)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Kron(Int32)</strong></td>
<td>Computes the Kronecker Product of this matrix with an identity matrix. The</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Kron(CSMat)</strong></td>
<td>Computes the Kronecker Product of this matrix with another matrix.</td>
</tr>
<tr>
<td><strong>Mul(CSMat)</strong></td>
<td>Computes the product of this matrix and another matrix. Highly efficient.</td>
</tr>
<tr>
<td><strong>Mul(CVec)</strong></td>
<td>Computes the product of this matric and a vector.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Reads a matrix from a stream. The matrix must originally have been written using the <strong>Write</strong> method.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Gets a string representation of this matrix. Note that all non-zero elements are listed, so this string may be quite long. (Overrides <strong>ObjectToString</strong>.)</td>
</tr>
<tr>
<td><strong>UnitaryError</strong></td>
<td>Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of ((this , \text{adj}) \times this). In other words, it calculates the largest identity matrix is on the right-hand side of the product.</td>
</tr>
</tbody>
</table>
deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.

Write

W"rites this matrix to a stream. The matrix may be recreated by using the Read method.

 Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiply(CSMat, CSMat)</td>
<td>Computes the product of two matrices. Highly efficient.</td>
</tr>
<tr>
<td>Multiply(CSMat, CVec)</td>
<td>Computes the product of a matrix and a vector.</td>
</tr>
<tr>
<td>MultiplyBang</td>
<td>Computes the Kronecker product of two matrices</td>
</tr>
</tbody>
</table>

See Also

Reference
Microsoft.Research.Liquid Namespace
[T::Microsoft.Research.Liquid.CMat]
### CSMat Constructor

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSMat(Int32, FSharpList&lt;Tuple&lt;Int32, Int32, Double, Double&gt;&gt;)</td>
<td>Creates a matrix from a list of elements.</td>
</tr>
<tr>
<td>CSMat(Int32, FSharpOption&lt;Boolean&gt;)</td>
<td>Creates a new square identity or zero matrix.</td>
</tr>
<tr>
<td>CSMat(CMat, FSharpOption&lt;Double&gt;)</td>
<td>Creates a sparse matrix from a dense matrix.</td>
</tr>
</tbody>
</table>

See Also

Reference

CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMat Constructor (Int32, FSharpList<TupleInt32, Int32, Double, Double>)

Creates a matrix from a list of elements.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

## Syntax

```fsharp
new :
  len : int *
  xyris : FSharpList<Tuple<int, int, float, float>>
```

### Parameters

**len**
- **Type:** System.Int32  
The dimension of the matrix; that is, the row or column count.

**xyris**
- **Type:** Microsoft.FSharp.Collections.FSharpList<TupleInt32, Int32, Double, Double>  
A list of elements. Each element should be a tuple in the form (row,col,real,imag).

## See Also

Reference  
CSMat Class
CSMat Constructor (Int32, FSharpOptionBoolean)

Creates a new square identity or zero matrix.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

Syntax

```
F#

new : n : int * zero : FSharpOption<bool> -> CSMat
```

Parameters

\( n \)

Type: System.Int32
The dimension of the matrix; that is, the row or column count.

\( zero \)

Type: Microsoft.FSharp.Core.FSharpOption<bool>
An option to create a zero matrix rather than an identity matrix, if true. The default is false, which creates an identity matrix.

See Also

Reference
CSMat Class
CSMat Overload
Microsoft.Research.Liquid Namespace
CSMat Constructor (CMat, FSharpOptionDouble)

Creates a sparse matrix from a dense matrix.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
new : 
    m : CMat *
    tol : FSharpOption<float> -> CSMat
```

Parameters

\textit{m}

Type: Microsoft.Research.Liquid\text{CMat}
The source matrix.

\textit{tol}

Type: Microsoft.FSharp.Core\text{FSharpOption\text{Double}}
An optional tolerance for identifying zero elements. The default is to use Complex.tol as the maximum magnitude to consider 0.

See Also

Reference
CSMat Class
CSMat Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CSMat Properties

The CSMat type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Gets an element of the matrix.</td>
</tr>
<tr>
<td>Length</td>
<td>The dimension of this matrix; that is, the number of rows or columns.</td>
</tr>
</tbody>
</table>

See Also

Reference

CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatItem Property

Gets an element of the matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
member Item : Complex with get, set
```

### Parameters

- **x**
  - Type: `SystemInt32`
  - The row index of the desired element.

- **y**
  - Type: `SystemInt32`
  - The column index of the desired element.

### Return Value

- **Type:** `Complex`
- The matrix element, as a Complex number

### See Also

**Reference**
- `CSMat Class`
- `Microsoft.Research.Liquid Namespace`

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatLength Property

The dimension of this matrix; that is, the number of rows or columns.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

**Syntax**

```fsharp
member Length : int with get
```

**Property Value**

Type: Int32

**See Also**

Reference  
CSMat Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMat Methods

The `CSMat` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adj</strong></td>
<td>Computes the adjoint (complex conjugate transpose) of this matrix.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Clears out this matrix, setting all elements to zero.</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Makes a copy of this matrix.</td>
</tr>
<tr>
<td><strong>Dense</strong></td>
<td>Creates a dense matrix from this sparse matrix.</td>
</tr>
<tr>
<td><strong>Dump</strong></td>
<td>Dumps this matrix as a list of row-column indices and element complex values. Note that all non-zero elements are listed, so the output may be quite long.</td>
</tr>
<tr>
<td><strong>DumpDense</strong></td>
<td>Dumps this sparse matrix in the same format as a dense matrix.</td>
</tr>
<tr>
<td><strong>DumpMCC</strong></td>
<td>Dumps this matrix as a list of row-column indices and</td>
</tr>
</tbody>
</table>
element squared magnitudes. Note that all non-zero elements are listed, so the output may be quite long.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DumpML</strong></td>
<td>Dumps this matrix in MatLab format.</td>
</tr>
<tr>
<td><strong>Equals(Object)</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Equals(CSMat,</strong></td>
<td>Determines whether this matrix is equal to another.</td>
</tr>
<tr>
<td><strong>FSharpOptionDouble)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Filled</strong></td>
<td>Return indicies of filled entries in the sparse matrix (may include zeros)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Kron(Int32)</strong></td>
<td>Computes the Kronecker Product of this matrix with an identity matrix. The</td>
</tr>
<tr>
<td></td>
<td>identity matrix is on the right-hand side of the product.</td>
</tr>
<tr>
<td><strong>Kron(CSMat)</strong></td>
<td>Computes the Kronecker Product of this matrix with an identity matrix. The</td>
</tr>
<tr>
<td></td>
<td>identity matrix is on the right-hand side of the product.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Mul(CSMat)</td>
<td>Computes the product of this matrix and another matrix. Highly efficient.</td>
</tr>
<tr>
<td>Mul(CVec)</td>
<td>Computes the product of this matrix and a vector.</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a matrix from a stream. The matrix must originally have been written using the Write method.</td>
</tr>
<tr>
<td>ToString</td>
<td>Gets a string representation of this matrix. Note that all non-zero elements are listed, so this string may be quite long. (Overrides ObjectToString.)</td>
</tr>
<tr>
<td>UnitaryError</td>
<td>Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.</td>
</tr>
</tbody>
</table>
Write

Writes this matrix to a stream. The matrix may be recreated by using the Read method.

See Also

Reference
CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatAdj Method

Computes the adjoint (complex conjugate transpose) of this matrix.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

<table>
<thead>
<tr>
<th>F#</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>member Adj : unit -&gt; CSMat</td>
<td></td>
</tr>
</tbody>
</table>

Return Value
Type: CSMat
A new matrix that is the adjoint of this matrix

See Also

Reference
CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CSMatClear Method

Clears out this matrix, setting all elements to zero.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Clear : unit -> unit
```

### See Also

Reference  
CSMat Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
**CSMatCopy Method**

Makes a copy of this matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5)

### Syntax

```
member Copy : unit -> CSMat
```

**Return Value**

Type: CSMat  
The new matrix

### See Also

Reference

- CSMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatDense Method

Creates a dense matrix from this sparse matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Dense : 
    zeros : FSharpOption<bool>  ->  CMat
```

### Parameters

**zeros**  
Type: `Microsoft.FSharp.Core.FSharpOption<bool>`  
Option to create a dense matrix the same size as this matrix but filled with zeroes, rather than copying this matrix. The default is to copy the entries of this matrix to the new dense matrix.

### Return Value

Type: `CMat`  
The new dense matrix

### See Also

- Reference
- CSMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatDump Method

Dumps this matrix as a list of row-column indices and element complex values. Note that all non-zero elements are listed, so the output may be quite long.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member Dump :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<int, FSharpOption<String, unit>>, unit, unit>
    level : FSharpOption<int> -> unit
```

### Parameters

- **f**
  - Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int, FSharpOption<String, unit>, unit>, unit>, unit>`
  - The optional output function to use. The default is `showLogInd`.

- **level**
  - Type: `Microsoft.FSharp.Core.FSharpOption<int>`
  - The optional indentation level. The default is 0.

### See Also

- Reference
- CSMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CSMatDumpDense Method

Dumps this sparse matrix in the same format as a dense matrix.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

```fsharp
member DumpDense :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<String, Unit>>>
    level : FSharpOption<int> -> unit
```

Parameters

- **f**
  Type: `Microsoft.FSharp.Core.FSharpOption` of `FSharpFunc<Int32, FSharpFunc<String, Unit>>`
  The optional output function to use. The default is `showLogInd`.

- **level**
  Type: `Microsoft.FSharp.Core.FSharpOption` of `Int32`
  The optional indentation level. The default is 0.

See Also

Reference
CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatDumpMCC Method

Dumps this matrix as a list of row-column indices and element squared magnitudes. Note that all non-zero elements are listed, so the output may be quite long.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
member DumpMCC :  
  f : FSharpOption<FSharpFunc<int, FSharpFunc<int32, FSharpFunc<string, unit>>>  
  level : FSharpOption<int> -> unit
```

### Parameters

- **f**
  - Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int32, FSharpFunc<string, unit>>>`
  - The optional output function to use. The default is `showLogInd`.

- **level**
  - Type: `Microsoft.FSharp.Core.FSharpOption<int32>`
  - The optional indentation level. The default is 0.

### See Also

**Reference**
- CSMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CSMatDumpML Method

Dumps this matrix in MatLab format.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```fsharp
member DumpML :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<String, Unit>> *
    level : FSharpOption<int> *
    nam : FSharpOption<String> *
    idxOffset : FSharpOption<int> -> unit
```

Parameters

`f`
Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<String, Unit>>`
The optional output function to use. The default is `showLogInd`.

`level`
Type: `Microsoft.FSharp.Core.FSharpOption<int>`
The optional indentation level. The default is 0.

`nam`
Type: `Microsoft.FSharp.Core.FSharpOption<String>`
An optional name for the matrix. The default is A.

`idxOffset`
Type: `Microsoft.FSharp.Core.FSharpOption<int>`
An optional number of rows and columns to skip. If this is not zero, then the square submatrix starting at this offset is dumped instead of the full matrix. The default is 0.
See Also

Reference
CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
### CSMatEquals Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(CSMat, FSharpOptionDouble)</td>
<td>Determines whether this matrix is equal to another.</td>
</tr>
</tbody>
</table>

#### See Also

Reference
- CSMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatEquals Method (CSMat, FSharpOptionDouble)

Determines whether this matrix is equal to another.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
member Equals : CSMat * FSharpOption<Double> -> bool
```

### Parameters

- **m2**
  - Type: `Microsoft.Research.LiquidCSMat`
  - The sparse matrix to compare to.

- **tol**
  - Type: `Microsoft.FSharp.CoreFSharpOption<Double>`
  - An optional tolerance for considering real or imaginary parts equal. The default is Complex.Tol.

### Return Value

- Type: `Boolean`
  - true if the matrices are equal, within the tolerance limit, or false otherwise.

### See Also

- Reference
CSMatFilled Method

Return indicies of filled entries in the sparse matrix (may include zeros)

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```
member Filled : unit -> IEnumerable<Tuple<int, int>>
```

Return Value

Type: IEnumerable<Tuple<int, int>>
Sequence of x,y tuples to access matrix with

See Also

Reference
CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatKron Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kron(Int32)</td>
<td>Computes the Kronecker Product of this matrix with an identity matrix. The identity matrix is on the right-hand side of the product.</td>
</tr>
<tr>
<td>Kron(CSMat)</td>
<td>Computes the Kronecker Product of this matrix with another matrix.</td>
</tr>
</tbody>
</table>

See Also

Reference
CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatKron Method (Int32)

Computes the Kronecker Product of this matrix with an identity matrix. The identity matrix is on the right-hand side of the product.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.43)

### Syntax

```f#
member Kron :  
    lenI : int -> CSMat
```

### Parameters

- **lenI**
  - Type: SystemInt32
  - The dimension of the identity matrix (N of NxN).

### Return Value

- Type: CSMat
- The result of the Kronecker product

### See Also

- Reference
  - CSMat Class
  - Kron Overload
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatKron Method (CSMat)

Computes the Kronecker Product of this matrix with another matrix.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

### Syntax

```fsharp
member Kron :
  m2 : CSMat -> CSMat
```

### Parameters

- **m2**
  
  Type: Microsoft.Research.LiquidCSMat
  
  The right-hand side matrix in the Kronecker product.

### Return Value

Type: CSMat

The result of the Kronecker product

### See Also

- **Reference**
  
  CSMat Class
  
  Kron Overload
  
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
C$\text{SMat}$Mul Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Mul(CSMat)}$</td>
<td>Computes the product of this matrix and another matrix. Highly efficient.</td>
</tr>
<tr>
<td>$\text{Mul(CVec)}$</td>
<td>Computes the product of this matrix and a vector.</td>
</tr>
</tbody>
</table>

See Also

Reference
C$\text{SMat}$ Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatMul Method (CSMat)

Computes the product of this matrix and another matrix. Highly efficient.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Mul :  
    m2 : CSMat -> CSMat
```

### Parameters

**m2**
- Type: Microsoft.Research.LiquidCSMat
  - The right-hand matrix to be multiplied by this matrix.

### Return Value

**Type:** CSMat  
New resulting sparse matrix

### See Also

**Reference**
- CSMat Class  
- Mul Overload  
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatMul Method (CVec)

Computes the product of this matrix and a vector.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)&

## Syntax

```f#
member Mul : 
    v : CVec -> CVec
```

### Parameters

- **v**
  - Type: Microsoft.Research.Liquid.CVec  
    The vector to multiply.

### Return Value

- Type: CVec  
  The resulting vector

## See Also

- Reference:
  - CSMat Class
  - Mul Overload
  - Microsoft.Research.Liquid Namespace
CSMatRead Method

Reads a matrix from a stream. The matrix must originally have been written using the Write method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.∗)

### Syntax

```fsharp
static member Read :  
    br : BinaryReader  ->  CSMat
```

### Parameters

*br*

- **Type:** System.IO.BinaryReader
  - The stream to read the data from.

### Return Value

- **Type:** CSMat
  - The read-in matrix

### See Also

- Reference
  - CSMat Class  
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatToString Method

Gets a string representation of this matrix. Note that all non-zero elements are listed, so this string may be quite long.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

**Syntax**

```fsharp
abstract ToString : unit -> string
override ToString : unit -> string
```

**Return Value**  
Type: String  
The string representation

**See Also**

**Reference**  
CSMat Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CSMatUnitaryError Method

Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.7)

### Syntax

```fsharp
member UnitaryError : unit -> float
```

### Return Value

**Type:** Double

The worst deviation from 1.

### See Also

**Reference**

CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CSMatWrite Method

Write this matrix to a stream. The matrix may be recreated by using the Read method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

**Syntax**

```fsharp
member Write :  
  bw : BinaryWriter -> unit
```

**Parameters**

`bw`

Type: System.IO.BinaryWriter  
The stream to serialize this matrix to

**See Also**

Reference

- CSMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CSMat Operators

The CSMat type exposes the following members.

Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiply(CSMat, CSMat)</td>
<td>Computes the product of two matrices. Highly efficient.</td>
</tr>
<tr>
<td>Multiply(CSMat, CVec)</td>
<td>Computes the product of a matrix and a vector.</td>
</tr>
<tr>
<td>MultiplyBang</td>
<td>Computes the Kronecker product of two matrices</td>
</tr>
</tbody>
</table>

See Also

Reference
CSMat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
**CSMatMultiply Operator**

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiply(CSMat, CSMat)</td>
<td>Computes the product of two matrices. Highly efficient.</td>
</tr>
<tr>
<td>Multiply(CSMat, CVec)</td>
<td>Computes the product of a matrix and a vector.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

- CSMat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CSMatMultiply Operator (CSMat, CSMat)

Computes the product of two matrices. Highly efficient.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```fsharp
static let inline (*)
    m1 : CSMat *
    m2 : CSMat : CSMat
```

### Parameters

**m1**
Type: Microsoft.Research.LiquidCSMat  
The left-hand matrix.

**m2**
Type: Microsoft.Research.LiquidCSMat  
The right-hand matrix.

### Return Value
Type: CSMat  
The resulting product matrix

### See Also

Reference
CSMat Class  
Multiply Overload
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CSMatMultiply Operator (CSMat, CVec)

Computes the product of a matrix and a vector.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
static let inline (*)
  m : CSMat *
  v : CVec : CVec
```

### Parameters

- **m**
  - Type: Microsoft.Research.LiquidCSMat
  - The matrix

- **v**
  - Type: Microsoft.Research.LiquidCVec
  - The vector

### Return Value

- Type: CVec
  - The resulting vector

### See Also

- Reference
  - CSMat Class
  - Multiply Overload
CSMatMultiplyBang Operator

Computes the Kronecker product of two matrices

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
F# does not support this operator.
```

### Parameters

- **m1**
  - Type: Microsoft.Research.Liquid.CSMat
  - The left matrix

- **m2**
  - Type: Microsoft.Research.Liquid.CSMat
  - The right matrix

### Return Value

- Type: CSMat
  - The result of the Kronecker product

### See Also

- **Reference**
  - CSMat Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVec Class

A block-sparse vector of complex numbers.

Inheritance Hierarchy

System Object Microsoft.Research.LiquidCVec

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

Syntax

F#

[<SerializableAttribute>]
type CVec = class end

The CVec type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVec(UInt64, Boolean)</td>
<td>Creates a new vector filled with zeroes.</td>
</tr>
<tr>
<td>CVec(Double, FSharpOptionDouble, FSharpOptionBoolean)</td>
<td>Creates a new vector from initial value vectors, real and imaginary. Note that the length of the new vector must be no more than 2^20.</td>
</tr>
<tr>
<td>CVec(UInt64,</td>
<td>Creates a new vector. By</td>
</tr>
</tbody>
</table>
FSharpOptionDouble, FSharpOptionDouble, FSharpOptionBoolean) default, the vector is all zero. Optionally, the real and impaginary initial values may be provided; in this case, the vector must be no more than 2^20 in length.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSafe</td>
<td>Gets the imaginary part of an element of this vector by index.</td>
</tr>
<tr>
<td>Length</td>
<td>The length of this vector</td>
</tr>
<tr>
<td>rSafe</td>
<td>Gets the real part of an element of this vector by index.</td>
</tr>
<tr>
<td>Safe</td>
<td>Gets an element of this vector by index.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddMCC</td>
<td>Calculates the norm of this vector. The norm is the square root of the sum of the complex magnitudes of the vector elements.</td>
</tr>
<tr>
<td>Copy</td>
<td>Creates an exact copy of this vector.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Dump</strong></td>
<td>Dumps this vector using the provided function.</td>
</tr>
<tr>
<td><strong>DumpMCC</strong></td>
<td>Dump vector MCC with provided function</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GlobalPhase</strong></td>
<td>Estimates the global phase of a vector that represents a quantum state.</td>
</tr>
<tr>
<td></td>
<td>The estimate is computed as the complex phase of the vector element with the</td>
</tr>
<tr>
<td></td>
<td>greatest amplitude.</td>
</tr>
<tr>
<td><strong>Kron</strong></td>
<td>Computes the Kronecker product of this vector and another.</td>
</tr>
<tr>
<td><strong>NonZeros</strong></td>
<td>Gets non-zero indices upto a max count</td>
</tr>
<tr>
<td><strong>Normalize</strong></td>
<td>Normalizes this vector to length 1.0.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Reads a new vector from a binary file. Note that this routine is only</td>
</tr>
<tr>
<td></td>
<td>intended to read vectors written.</td>
</tr>
</tbody>
</table>
with the **Write** method.

<table>
<thead>
<tr>
<th><strong>Operator</strong></th>
<th><strong>Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Scale" /></td>
<td>Scale</td>
<td>Scale all elements of the vector (unsafe)</td>
</tr>
<tr>
<td><img src="image" alt="ToString" /></td>
<td>ToString</td>
<td>Converts this vector to a string. At most the first 512 entries are displayed. (Overrides <code>ObjectToString</code>.)</td>
</tr>
<tr>
<td><img src="image" alt="Write" /></td>
<td>Write</td>
<td>Writes this vector to a binary file.</td>
</tr>
<tr>
<td><img src="image" alt="Zero" /></td>
<td>Zero</td>
<td>Zeroes this vector. Note that this also densely fills in the vector.</td>
</tr>
</tbody>
</table>

### Operators

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="MultiplyBang" /></td>
<td>Computes the Kronecker product of two vectors.</td>
</tr>
</tbody>
</table>

### See Also

**Reference**

*Microsoft.Research.Liquid Namespace*

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVec Constructor

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVec(UInt64, Boolean)</td>
<td>Creates a new vector filled with zeroes.</td>
</tr>
<tr>
<td>CVec(Double, FSharpOptionDouble, FSharpOptionBoolean)</td>
<td>Creates a new vector from initial value vectors, real and imaginary. Note that the length of the new vector must be no more than 2^20.</td>
</tr>
<tr>
<td>CVec(UInt64, FSharpOptionDouble, FSharpOptionDouble, FSharpOptionBoolean)</td>
<td>Creates a new vector. By default, the vector is all zero. Optionally, the real and imaginary initial values may be provided; in this case, the vector must be no more than 2^20 in length.</td>
</tr>
</tbody>
</table>

See Also

Reference
CVec Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVec Constructor (UInt64, Boolean)

Creates a new vector filled with zeroes.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

**Syntax**

```fsharp
new :  
    _len : uint64 *  
    force : bool -> CVec
```

**Parameters**

_\_len_

Type: System.UInt64  
The length of the vector

\_force_

Type: System.Boolean  
Option to force complete (non-sparse) allocation of the vector.  
The default to false, which leaves the vector sparse.

**See Also**

Reference  
CVec Class  
CVec Overload  
Microsoft.Research.Liquid Namespace
CVec Constructor (Double, FSharpOptionDouble, FSharpOptionBoolean)

Creates a new vector from initial value vectors, real and imaginary. Note that the length of the new vector must be no more than 2^20.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

**Syntax**

```fsharp
new :  
    rs : float[] *  
    is : FSharpOption<float[]> *  
    force : FSharpOption<bool> -> CVec
```

**Parameters**

- **rs**
  - Type: System.Double
  - The vector of real initial values

- **is**
  - Type: Microsoft.FSharp.Core.FSharpOption<Double>
  - An optional vector of imaginary initial values; default is zero

- **force**
  - Type: Microsoft.FSharp.Core.FSharpOption<Bool>
  - Option to force complete (non-sparse) allocation of the vector; defaults to false
See Also

Reference
CVec Class
CVec Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVec Constructor (UInt64, FSharpOptionDouble, FSharpOptionDouble, FSharpOptionBoolean)

Creates a new vector. By default, the vector is all zero. Optionally, the real and imaginary initial values may be provided; in this case, the vector must be no more than $2^{20}$ in length.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```fsharp
new : _len : uint64 *
    rsInit : FSharpOption<float[]> *
    isInit : FSharpOption<float[]> *
    forceAlloc : FSharpOption<bool> -> CVec
```

Parameters

_\_len_
Type: System.UInt64
The length of the vector

\textit{rsInit}
Type: \texttt{Microsoft.FSharp.Core.FSharpOption<Double>}
Optional real parts of the initial values; if provided, there must be an entry for each element in the vector. Default is zero.

\textit{isInit}
Optional imaginary of the initial values; if provided, there must be an entry for each element in the vector. Default is zero.

forceAlloc

Option to force complete (non-sparse) allocation of the vector; defaults to false

See Also

Reference
CVec Class
CVec Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The CVec type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSafe</td>
<td>Gets the imaginary part of an element of this vector by index.</td>
</tr>
<tr>
<td>Length</td>
<td>The length of this vector</td>
</tr>
<tr>
<td>rSafe</td>
<td>Gets the real part of an element of this vector by index.</td>
</tr>
<tr>
<td>Safe</td>
<td>Gets an element of this vector by index.</td>
</tr>
</tbody>
</table>

See Also

Reference

CVec Class

Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVeciSafe Property

Gets the imaginary part of an element of this vector by index.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0*)

### Syntax

```fshARP
member iSafe : float with get, set
```

### Parameters

\( i \)

Type: **System.UInt64**  
The index of the element to get

### Return Value

Type: **Double**  
The imaginary part of the Complex value of the element

### See Also

**Reference**
- CVec Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecLength Property

The length of this vector

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

**Syntax**

```
F#

member Length : uint64 with get
```

**Property Value**

Type: UInt64

**See Also**

Reference

CVec Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecrSafe Property

Gets the real part of an element of this vector by index.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.2)

### Syntax

```fsharp
member rSafe : float with get, set
```

**Parameters**

- **i**
  - Type: System.UInt64
  - The index of the element to get

**Return Value**

- Type: Double
  - The real part of the Complex value of the element

### See Also

**Reference**

CVec Class  
Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecSafe Property

Gets an element of this vector by index.

**Namespace**: Microsoft.Research.Liquid  
**Assembly**: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```fsharp
member Safe : Complex with get, set
```

**Parameters**

\( i \)

Type: `System.UInt64`

The index of the element to get

**Return Value**

Type: `Complex`

The Complex value of the element

### See Also

**Reference**

CVec Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The **CVec** type exposes the following members.

## CVec Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddMCC</strong></td>
<td>Calculates the norm of this vector. The norm is the square root of the sum of the complex magnitudes of the vector elements.</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Creates an exact copy of this vector.</td>
</tr>
<tr>
<td><strong>Dump</strong></td>
<td>Dumps this vector using the provided function.</td>
</tr>
<tr>
<td><strong>DumpMCC</strong></td>
<td>Dump vector MCC with provided function</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GlobalPhase</td>
<td>Estimates the global phase of a vector that represents a quantum state. The estimate is computed as the complex phase of the vector element with the greatest amplitude.</td>
</tr>
<tr>
<td>Kron</td>
<td>Computes the Kronecker product of this vector and another.</td>
</tr>
<tr>
<td>NonZeros</td>
<td>Gets non-zero indices up to a max count.</td>
</tr>
<tr>
<td>Normalize</td>
<td>Normalizes this vector to length 1.0.</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a new vector from a binary file. Note that this routine is only intended to read vectors written with the Write method.</td>
</tr>
<tr>
<td>Scale</td>
<td>Scale all elements of the vector (unsafe)</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts this vector to a string. At most the first 512 entries are displayed. (overrides ObjectToString.)</td>
</tr>
<tr>
<td>Write</td>
<td>Writes this vector to a binary file.</td>
</tr>
<tr>
<td>Zero</td>
<td>Zeroes this vector. Note that this also densely fills in the vector.</td>
</tr>
</tbody>
</table>

See Also

Top
CVecAddMCC Method

Calculates the norm of this vector. The norm is the square root of the sum of the complex magnitudes of the vector elements.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
member AddMCC : unit -> float
```

**Return Value**  
Type: **Double**  
This vector's norm

### See Also

**Reference**  
CVec Class  
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CVec Copy Method

Creates an exact copy of this vector.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
F# Copy

member Copy : unit -> CVec
```

**Return Value**  
**Type:** CVec

The new vector

### See Also

**Reference**  
CVec Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecDump Method

Dumps this vector using the provided function.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.6)

⚠️ Syntax

```fsharp
member Dump :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<int, FSharpFunc<int, string>>, FSharpFunc<int, string>> *
    level : FSharpOption<int> *
    maxNZ : FSharpOption<int> *
    -> unit
```

**Parameters**

- **f**
  - **Type:** Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<int, FSharpFunc<int, string>>, FSharpFunc<int, string>>
  - The optional output function to use. The default is `showLogInd`.

- **level**
  - **Type:** Microsoft.FSharp.Core.FSharpOption<int>
  - The optional indentation level. The default is 0.

- **maxNZ**
  - **Type:** Microsoft.FSharp.Core.FSharpOption<int>
  - The optional maximum number of non-zeros to dump. The default is 256

⚠️ See Also

- Reference
  - CVec Class
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CVecDumpMCC Method

Dump vector MCC with provided function

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```fsharp
member DumpMCC :
    f : FSharpOption<FSharpFunc<int>, FSharpFunc<int, FSharpFunc<String, Unit>>, Level, Thresh, Both, MaxNZ> -> unit
```

Parameters

- **f**
  Type: Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int, FSharpFunc<String, Unit>>>, Level, Thresh, Both, MaxNZ>
  The optional output function to use. The default is `showLogInd`.

- **level**
  Type: Microsoft.FSharp.Core.FSharpOption<int>
  The optional indentation level. The default is 0.

- **thresh**
  Type: Microsoft.FSharp.Core.FSharpOption<float>
  Output threshold (optional=tol*1.e+2)

- **both**
  Type: Microsoft.FSharp.Core.FSharpOption<bool>
  Output complex value as well (optional=false)

- **maxNZ**
Type: `Microsoft.FSharp.Core.FSharpOption`<br>Max non-zeros to dump (optional=256)

**See Also**

Reference
- CVec Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecGlobalPhase Method

Estimates the global phase of a vector that represents a quantum state. The estimate is computed as the complex phase of the vector element with the greatest amplitude.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

**Syntax**

```fsharp
member GlobalPhase : unit -> Tuple<float, Complex>
```

**Return Value**

Type: Tuple<Double, Complex>
A tuple whose first element is the estimated phase angle, in radians, and whose second element is the unit-magnitude Complex number with the opposite phase.

**See Also**

Reference
CVec Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecKron Method

Computes the Kronecker product of this vector and another.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
member Kron : v2 : CVec -> CVec
```

Parameters

v2
Type: Microsoft.Research.Liquid.CVec
Vector to multiply this one by

Return Value
Type: CVec
The result vector

See Also

Reference
CVec Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecNonZeros Method

Gets non-zero indices upto a max count

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
member NonZeros :  
  mx : int *  
  thresh : FSharpOption<float>  ->  FSharpList<

Parameters

mx
  Type: System.Int32
  Max length of list to return
thresh
  Type: Microsoft.FSharp.Core.FSharpOption<Double>
  Threshold (optional=tol*1.0e+2)

Return Value
  Type: FSharpList<UInt64>
  Indicies of non-zero (tolerance defined) entries

See Also

Reference
  CVec Class
  Microsoft.Research.Liquid Namespace
CVecNormalize Method

Normalizes this vector to length 1.0.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
member Normalize : len : FSharpOption<float> -> unit
```

Parameters

len

Type: Microsoft.FSharp.Core.FSharpOption<Double>
The optional current length, if already calculated. The default is to invoke and use the result of AddMCC

See Also

Reference

CVec Class
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CVecRead Method

Reads a new vector from a binary file. Note that this routine is only intended to read vectors written with the Read method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5)

### Syntax

F#

```fsharp
static member Read :  
    br : BinaryReader -> CVec
```

### Parameters

`br`

Type: System.IO.BinaryReader  
The BinaryReader from which the vector should be read.

### Return Value

Type: CVec  
The new vector

### See Also

Reference

CVec Class  
Microsoft.Research.Liquid Namespace  
CVecWrite(BinaryWriter)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecScale Method

Scale all elements of the vector (unsafe)

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
member Scale : scale : Complex -> unit
```

### Parameters

- **scale**
  - Type: Microsoft.Research.Liquid.Complex
  - Complex scale factor

### See Also

- Reference
- CVec Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecToString Method

Converts this vector to a string. At most the first 512 entries are displayed.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0*)

**Syntax**

```fsharp
abstract ToString : unit -> string  
override ToString : unit -> string
```

**Return Value**
Type: String  
The string

**See Also**

Reference
CVec Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecWrite Method

Writes this vector to a binary file.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

**F#**

```fsharp
member Write :
  bw : BinaryWriter -> unit
```

### Parameters

* **bw**
  
  Type: `System.IO.BinaryWriter`
  
  The BinaryWriter that this vector should be written to.

### See Also

**Reference**

- CVec Class
- Microsoft.Research.Liquid Namespace
- CVecRead(BinaryReader)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecZero Method

Zeroes this vector. Note that this also densely fills in the vector.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

**Syntax**

```fsharp
member Zero : unit -> unit
```

**See Also**

Reference  
CVec Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

CVec Operators

The `CVec` type exposes the following members.

- **Operators**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiplyBang</td>
<td>Computes the Kronecker product of two vectors.</td>
</tr>
</tbody>
</table>

See Also

- **Reference**
  - `CVec Class`
  - `Microsoft.Research.Liquid Namespace`

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
CVecMultiplyBang Operator

Computes the Kronecker product of two vectors.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4*)

**Syntax**

F# does not support this operator.

**Parameters**

\( v_1 \)
- Type: Microsoft.Research.Liquid.CVec  
  Left vector

\( v_2 \)
- Type: Microsoft.Research.Liquid.CVec  
  Right vector

**Return Value**
- Type: CVec  
  The result vector

**See Also**

Reference
- CVec Class  
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Fermion Class

Hamiltonian simulation for fermionic systems.

Inheritance Hierarchy

```
System\Object  Microsoft.Research.LiquidHamiltonian
    Microsoft.Research.LiquidFermion
```

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
[<SerializableAttribute>]
type Fermion =
  class
    inherit Hamiltonian
  end
```

The Fermion type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fermion(Double, Double, Int32, Int32, Dictionary&lt;String, String&gt;, FSharpFunc&lt;Double&gt;, FSharpFunc&lt;FSharpList&lt;Qubit&gt;, Unit&gt;)</td>
<td>Creates a new Fermion instance from a prebuilt Circuit.</td>
</tr>
</tbody>
</table>

Creates a new Fermion instance from orbit overlap integrals.

## Top

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bits</td>
<td>The number of phase estimation bits. This is the bit precision plus two.</td>
</tr>
<tr>
<td>Circs</td>
<td>The built circuits, in bit order.</td>
</tr>
<tr>
<td>currentCirc</td>
<td>The last circuit that was run (ungrown).</td>
</tr>
<tr>
<td>decohereModel</td>
<td>The decoherence model for this Hamiltonian. (Inherited from Hamiltonian.)</td>
</tr>
<tr>
<td>eMax</td>
<td>The maximum energy for phase estimation.</td>
</tr>
<tr>
<td>eMin</td>
<td>The minimum energy for phase estimation.</td>
</tr>
</tbody>
</table>
| Energy       | The result of phase estimation, interpreted as an energy. This will always be between eMin and eMax. This value is only available after Run has been
Ket

The Ket vector associated with this Hamiltonian (Inherited from Hamiltonian.)

omega

The energy range, eMax - eMin, for phase estimation.

order

The trotter order for phase estimation.

Phase

The result of phase estimation, as an angle between 0 and 2*pi. This value is only available after Run has been called.

trotterN

The trotter number for phase estimation.

tTotal

The total evolution time for phase estimation. This is equal to 2*p1/omega.

Ua

The gate function that implements a full Hamiltonian time step.

Top

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>Builds either a grown circuit or an exponentiated unitary.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Clean</strong></td>
<td>Cleans out temporary files.</td>
</tr>
<tr>
<td><strong>Dump</strong></td>
<td>Dumps out information on this simulator.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Load</strong></td>
<td>Loads a Fermion test from a .dat file based on a script.</td>
</tr>
<tr>
<td><strong>LoadOrbs</strong></td>
<td>Loads orbital information from an array of strings.</td>
</tr>
<tr>
<td><strong>PhaseSetup</strong></td>
<td>Sets up phase estimation for all runs.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Prep</td>
<td>Prepares an initial state from the provided spin orbital indices.</td>
</tr>
<tr>
<td>Run(Boolean, FSharpOptionInt32)</td>
<td>Runs the simulation to obtain a phase estimate. A previous call to Build() or BuildSingle() is required.</td>
</tr>
<tr>
<td>Run(DictionaryString, String, String)</td>
<td>Runs a Fermion test from a .dat file based on a script. A basic execution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.</td>
</tr>
<tr>
<td>Run(DictionaryString, String, String)</td>
<td>Runs a pre-loaded Fermion test based on a script. A basic execution trace is sent to both the console and the log. Detailed information is sent</td>
</tr>
</tbody>
</table>
just sent to the log. See the Users Manual for details on the parameters.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run(DictionaryString, String, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Ket)</td>
<td>Runs a Fermion test from a pre-built circuit and state vector. See the Users Manual for details on the parameters.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

Remarks

This class simulates second-quantized models of electrons with a state space consisting of spin-up/spin-down pairs, with both one-body and two-body interactions. In particular, this includes second-quantized quantum chemistry, where the state pairs are molecular orbitals, and the individual states are spin orbitals. In this case, the interaction terms come from various orbital overlap integrals. See the Users Manual for more detailed information.

See Also

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
# Fermion Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

## See Also

**Reference**
- Fermion Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Fermion Constructor (Double, Double, Int32, Int32, Int32, Dictionary<String, String>, String, FSharpFuncFuncDouble, FSharpFuncFuncFSharpListListQubit, Unit)

Creates a new Fermion instance from a prebuilt Circuit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
new :  
    eMin : float *  
    eMax : float *  
    trotterN : int *  
    bits : int *  
    order : int *  
    dic : Dictionary<string, string> *  
    Ua : FSharpFunc<float, FSharpFunc<float, FSharpListListQubit, Unit>> *
```

### Parameters

**eMin**

- **Type:** System.Double
- The minimum energy to use for phase estimation.
### Remarks

The possible options to specify are: ... For more information, see the Users Manual.

### See Also

Reference
- Fermion Class
- Fermion Overload
- Microsoft.Research.Liquid Namespace

Creates a new Fermion instance from orbit overlap integrals.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
new :
    eMin : float *
    eMax : float *
    trotterN : int *
    bits : int *
    order : int *
    ij : FSharpList<TUPLE<int, int, float>>`
    ijkil : FSharpList<TUPLE<int, int, int, int, float>> *
    dic : Dictionary<String, String> *
    preps : FSharpOption<FSharpList<int>> ->
```
Parameters

**eMin**
Type: SystemDouble
The minimum energy to use for phase estimation.

**eMax**
Type: SystemDouble
The maximum energy to use for phase estimation.

**trotterN**
Type: SystemInt32
The Trotter number to use

**bits**
Type: SystemInt32
The number of bits of phase estimation accuracy desired.

**order**
Type: SystemInt32
The Trotter order, either 1 or 2 (first or second order).

**ij**
Type: Microsoft.FSharp.Collections.FSharpList<Tuple<int32, int32, double>>
A list of tuples for single-body orbital constants. Each tuple represents an Hpq term, and contains p, q, and the value of Hpq. Note that pp terms are included in this category.

**ijkl**
Type: Microsoft.FSharp.Collections.FSharpList<Tuple<int32, int32, int32, int32, double>>
A list of tuples for two-body orbital constants. Each tuple represents an Hpqrs term, and contains p, q, r, s, and the value of Hpqrs. Note that pqqp and pqqr terms are included in this category.

**dic**
Type: System.Collections.Generic.Dictionary<String, String>
A dictionary of options. See the Remarks for details.

**preps**
Type: Microsoft.FSharp.Core.FSharpOption<FSharpList<int32>>
An optional specification of a 1-based electron prep list, for diagonal fix-up. The default is no prep and no fix-up.
Remarks

The possible options to specify are: ... For more information, see the Users Manual.

See Also

Reference
Fermion Class
Fermion Overload
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Fermion Properties

The Fermion type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bits</td>
<td>The number of phase estimation bits. This is the bit precision plus two.</td>
</tr>
<tr>
<td>Circs</td>
<td>The built circuits, in bit order.</td>
</tr>
<tr>
<td>currentCirc</td>
<td>The last circuit that was run (ungrown).</td>
</tr>
<tr>
<td>decohereModel</td>
<td>The decoherence model for this Hamiltonian. (Inherited from Hamiltonian.)</td>
</tr>
<tr>
<td>eMax</td>
<td>The maximum energy for phase estimation.</td>
</tr>
<tr>
<td>eMin</td>
<td>The minimum energy for phase estimation.</td>
</tr>
<tr>
<td>Energy</td>
<td>The result of phase estimation, interpreted as an energy. This will always be between eMin and eMax. This value is only available after Run has been called.</td>
</tr>
<tr>
<td>Ket</td>
<td>Gets the Ket vector associated</td>
</tr>
</tbody>
</table>
with this Hamiltonian (Inherited from Hamiltonian.)

<table>
<thead>
<tr>
<th>omega</th>
<th>The energy range, eMax - eMin, for phase estimation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>The trotter order for phase estimation.</td>
</tr>
<tr>
<td>Phase</td>
<td>The result of phase estimation, as an angle between 0 and 2*pi. This value is only available after Run has been called.</td>
</tr>
<tr>
<td>trotterN</td>
<td>The trotter number for phase estimation.</td>
</tr>
<tr>
<td>tTotal</td>
<td>The total evolution time for phase estimation. This is equal to 2*p1/omega.</td>
</tr>
<tr>
<td>Ua</td>
<td>The gate function that implements a full Hamiltonian time step</td>
</tr>
</tbody>
</table>

See Also

Reference
Fermion Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Fermion bits Property

The number of phase estimation bits. This is the bit precision plus two.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

F#

```fsharp
member bits : int with get
```

Property Value
Type: Int32

See Also

Reference
Fermion Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
FermionCircs Property

The built circuits, in bit order.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
member Circs : FSharpList<Circuit> with get
```

Property Value
Type: FSharpList<Circuit>

See Also

Reference
Fermion Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Fermion currentCirc Property

The last circuit that was run (ungrown).

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member currentCirc : Circuit with get
```

### Property Value

Type: **Circuit**

### See Also

- Reference
  - Fermion Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
FermionMax Property

The maximum energy for phase estimation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
**Version:** 1.0.5981.24943 (1.0.0)

### Syntax

```
Fsharp

member eMax : float with get
```

**Property Value**  
Type: **Double**

### See Also

**Reference**  
Fermion Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Fermion eMin Property

The minimum energy for phase estimation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0*)

### Syntax

```fsharp
member eMin : float with get
```

**Property Value**

Type: **Double**

### See Also

**Reference**
- Fermion Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The result of phase estimation, interpreted as an energy. This will always be between eMin and eMax. This value is only available after Run has been called.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

◆ Syntax

```fsharp
member Energy : float with get
```

Property Value
Type: Double

◆ See Also

Reference
Fermion Class
Microsoft.Research.Liquid Namespace
Fermionomega Property

The energy range, eMax - eMin, for phase estimation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member omega : float with get
```

### Property Value

Type: **Double**

### See Also

- **Reference**
  - Fermion Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Fermion order Property

The trotter order for phase estimation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member order : int with get
```

### Property Value

Type: Int32

### See Also

- Reference
  - Fermion Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
FermionPhase Property

The result of phase estimation, as an angle between 0 and 2*pi. This value is only available after Run has been called.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

F#

```fsharp
member Phase : float with get
```

Property Value

Type: Double

See Also

Reference

Fermion Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Fermion	trotterN Property

The trotter number for phase estimation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.2)

### Syntax

```
F# Copy

member trotterN : int with get, set
```

### Property Value

Type: Int32

### See Also

- Reference
  - Fermion Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
**FermionTotal Property**

The total evolution time for phase estimation. This is equal to 2*p1/omega.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```
member tTotal : float with get
```

### Property Value

**Type:** Double

### See Also

- Reference
- Fermion Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
FermionUa Property

The gate function that implements a full Hamiltonian time step

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

**Syntax**

```fsharp
member Ua : FSharpFunc<float>, FSharpFunc<FSharpList<Qubit>, Unit>
```

**Property Value**

Type: `FSharpFunc` of `float`, `FSharpFunc` of `FSharpList<Qubit>`, `Unit`

**See Also**

Reference

Fermion Class  
Microsoft.Research.Liquid Namespace
Fermion Methods

The Fermion type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>Builds either a grown circuit or an exponentiated unitary.</td>
</tr>
<tr>
<td>Clean</td>
<td>Cleans out temporary files.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dumps out information on this simulator.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of</td>
</tr>
</tbody>
</table>

The Language-Integrated Quantum Operations (LIQUi|>) Simulator
the current instance. (Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>Loads a Fermion test from a .dat file based on a script.</td>
</tr>
<tr>
<td>LoadOrbs</td>
<td>Loads orbital information from an array of strings.</td>
</tr>
<tr>
<td>PhaseSetup</td>
<td>Sets up phase estimation for all runs.</td>
</tr>
<tr>
<td>Prep</td>
<td>Prepares an initial state from the provided spin orbital indices.</td>
</tr>
<tr>
<td>Run(Boolean, FSharpOptionInt32)</td>
<td>Runs the simulation to obtain a phase estimate. A previous call to Build() or BuildSingle() is required.</td>
</tr>
<tr>
<td>Run(DictionaryString, String, String)</td>
<td>Runs a Fermion test from a .dat file, based on a script. A basic execution trace is sent to both the console and the log.</td>
</tr>
</tbody>
</table>
Run(DictionaryString, String, String) | Runs a pre-loaded Fermion test based on a script. A basic execution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.

Run(DictionaryString, String, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Ket) | Runs a Fermion test from a pre-built circuit and state vector. See the Users Manual for details on the parameters.

ToString | Returns a string that represents the current object. (Inherited from Object.)
See Also

Reference
Fermion Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Fermion

Build Method

Builds either a grown circuit or an exponentiated unitary.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

F#

```fsharp
member Build : gp : GrowPars -> unit
```

Parameters

*gp*

Type: Microsoft.Research.Liquid.GrowPars
Grow parameters for Circuit.GrowGates. If the Single flag is set to true, then a single exponentiated unitary is built.

See Also

Reference

Fermion Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
FermionClean Method

Cleans out temporary files.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

## Syntax

```fsharp
member Clean : unit -> unit
```

## See Also

- Reference
- Fermion Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
FermionDump Method

Dumps out information on this simulator.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
member Dump :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<string, Unit>>, Unit>
    level : FSharpOption<int> -> unit
```

### Parameters

- **f**
  - Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<string, Unit>>, Unit>`
  - The optional output function to use. The default is `showLogInd`.

- **level**
  - Type: `Microsoft.FSharp.Core.FSharpOption<int>`
  - The optional indentation level. The default is 0.

### See Also

- Reference
  - Fermion Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

FermionLoad Method

Loads a Fermion test from a .dat file based on a script.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
static member Load :
    dic : Dictionary<string, string> *
    dataFile : string -> string
```

Parameters

dic
Type: System.Collections.Generic.Dictionary<String, String>
A dictionary of options. See the Remarks for details.

dataFile
Type: System.String
A relative or full path to the .dat file to load.

Return Value

Type: String
A single, multi-line string containing orbital information for the test number specified in the option dictionary. This string is suitable for passing (as the only element of an array) to LoadOrbs.

Remarks

The possible options to specify in the dictionary are: ... For more information, see the Users Manual.
See Also

Reference
Fermion Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
FermionLoadOrbs Method

Loads orbital information from an array of strings.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

### Syntax

```fsharp
static member LoadOrbs :
    dic : Dictionary<string, string> *
    data : string[] -> Tuple<FSharpList<Tuple<int, int, double>>, FSharpList<Tuple<int, int, int, int, double, double, double>>
```

### Parameters

**dic**

Type: `System.Collections.Generic.Dictionary<string, string>`
A dictionary of options. See the Remarks for details.

**data**

Type: `System.String`
An array of formatted strings containing orbital information. Each string corresponds to a single line in a .dat file. The string format is described in the Users Manual.

### Return Value

Type: `Tuple<FSharpList<Tuple<int, int, double>>, FSharpList<Tuple<int, int, int, int, double, double, double>>`

A tuple containing the single-orbital terms; the two-orbital terms; an informational string; and the nuclear energy. The orbital terms are in the proper format to pass to the Fermion constructor.
Remarks

The possible options to specify in the dictionary are: ... For more information, see the Users Manual.

See Also

Reference

Fermion Class
Microsoft.Research.Liquid Namespace
FermionPhaseSetup Method

Sets up phase estimation for all runs.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member PhaseSetup :
    bits : int *
    alterNoise : float *
    peType : string -> unit
```

### Parameters

**bits**
- Type: SystemInt32
- The number of bits of phase estimation accuracy desired.

**alterNoise**
- Type: SystemDouble
- Magnitude of random (additive) noise to apply to evolution angles. Set this to 0.0 for no noise.

**peType**
- Type: SystemString
- The type of phase estimation to perform. Possible values are:
  - "circ": Basic PE on the full circuit
  - "expon": Create matrix exponentiations
  - "noise": Compute noise while running (uses a single Unitary)
  - "file": Serialize matrix exponentation to disk
  - "atan": Use arc tangent to go forward and do classical post processing
- "default": Use type that naturally goes with specified GrowPars

See Also

Reference
Fermion Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Fermion Prep Method

Prepares an initial state from the provided spin orbital indices.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Prep : prep : FSharpList<int> -> string
```

### Parameters

*prep*

- **Type:** Microsoft.FSharp.Collections.FSharpList<Int32>
- The list of spin orbitals to mark as occupied.

### Return Value

**Type:** String  
The "prep state" with exactly the listed orbitals occupied.

### See Also

**Reference**  
- Fermion Class  
- Microsoft.Research.Liquid Namespace
## Fermion Run Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run(Boolean, FSharpOptionInt32)</td>
<td>Runs the simulation to obtain a phase estimate. A previous call to Build() or BuildSingle() is required.</td>
</tr>
<tr>
<td>Run(DictionaryString, String, String)</td>
<td>Runs a Fermion test from a .dat file, based on a script. A basic execution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.</td>
</tr>
<tr>
<td>Run(DictionaryString, String, String)</td>
<td>Runs a pre-loaded Fermion test based on a script. A basic execution trace is sent to</td>
</tr>
</tbody>
</table>
both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.

| Run(Dictionary<String, String, FSharpFunc<Double>, FSharpFunc<FSharpList<Qubit>, Unit, Ket)> | Runs a Fermion test from a pre-built circuit and state vector. See the Users Manual for details on the parameters. |

See Also

Reference
Fermion Class
Microsoft.Research.Liquid Namespace
FermionRun Method (Boolean, FSharpOptionInt32)

Runs the simulation to obtain a phase estimate. A previous call to Build() or BuildSingle() is required.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
member Run : single : bool *
             maxTries : FSharpOption<int> -> FSharpOpt
```

Parameters

**single**
Type: System.Boolean
Whether or not this Hamiltonian is built into a single Unitary.

**maxTries**
Type: Microsoft.FSharp.Core.FSharpOption<int>
An optional maximum number of measurements to take. The default is 40.

Return Value
Type: FSharpOption<int>
None if it succeeded, or Some (int bit position) if it failed.

See Also
Reference
Fermion Class
Run Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
FermionRun Method
(Dictionary<String, String, String>)

Runs a Fermion test from a .dat file, based on a script. A basic execution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```fsharp
module Run : 
    dic : Dictionary<string, string> *
    dataFile : string -> unit
```

Parameters

dic
    Type: System.Collections.Generic.Dictionary<String, String>
    A dictionary of options. See the Remarks for details.

dataFile
    Type: System.String
    A relative or full path to the .dat file to load.

Remarks

The possible options to specify in the dictionary are: ... For more information, see the Users Manual.
See Also

Reference
Fermion Class
Run Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
FermionRun Method
(Dictionary<String, String, String>)

Runs a pre-loaded Fermion test based on a script. A basic execution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.

**Namespace**: Microsoft.Research.Liquid
**Assembly**: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

## Syntax

```fsharp
static member Run :
    dic : Dictionary<string, string> *
    data : string[] -> unit
```

### Parameters

**dic**

Type: System.Collections.Generic.Dictionary<String, String>
A dictionary of options. See the Remarks for details.

**data**

Type: System.String
An array of formatted strings containing orbital information. Each string corresponds to a single line in a .dat file. The string format is described in the Users Manual.

## Remarks

The possible options to specify in the dictionary are: ... For more information, see the Users Manual.
See Also

Reference
Fermion Class
Run Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Fermion Run Method
(Dictionary<String, String, FSharpFunc<Double>, FSharpFunc<FSharpList<Qubit>, Unit, Ket>)

Runs a Fermion test from a pre-built circuit and state vector. See the Users Manual for details on the parameters.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member Run :
    dic : Dictionary<String, String> *
    Ua : FSharpFunc<Double, FSharpFunc<FSharpList<Qubit>, Unit, Ket >
    ket : Ket -> Tuple<int, Fermion>
```

#### Parameters

**dic**

Type: System.Collections.Generic.Dictionary<String, String>
A dictionary of options. See the Remarks for details.

**Ua**

Type: Microsoft.FSharp.Core.FSharpFunc<Double, FSharpFunc<FSharpList<Qubit>, Unit>
A gate, usually a wrapped Circuit, that implements the
Hamiltonian. The gate's parameter is the time step to evolve by.

*ket*

Type: `Microsoft.Research.LiquidKet`
The state vector to use as the initial state.

**Return Value**
Type: `Tuple<Int32, Fermion>`
A tuple of the phase estimation error bit, which will be -1 if no error occurred, and the Fermion instance that was run. The **Phase** and **Energy** properties of the Fermion instance may be read to get the phase estimation results.

**Remarks**
The possible options to specify in the dictionary are: ... For more information, see the Users Manual.

**See Also**
Reference
- Fermion Class
- Run Overload
- Microsoft.Research.Liquid Namespace
Gate Class

A quantum gate.

Inheritance Hierarchy

- SystemObject
- Microsoft.Research.LiquidGate

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

Syntax

```fsharp
[<SerializableAttribute>]
type Gate = class end
```

The Gate type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate</td>
<td>Creates a new gate from scratch.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arity</td>
<td>Get arity of the gate (based on Qubits or Mat/Kraus size).</td>
</tr>
</tbody>
</table>
### CacheDisable
Whether or not the gate cache is disabled. The cache is used if this property is false. The cache should be disabled if gates are being built in parallel.

<table>
<thead>
<tr>
<th><strong>Draw</strong></th>
<th>The drawing instructions for this gate.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Help</strong></td>
<td>The help string for this gate.</td>
</tr>
<tr>
<td><strong>Kraus</strong></td>
<td>The Kraus operator matrix list for this gate (if op type is Channel). If the gate is not a channel, this will be an empty list.</td>
</tr>
<tr>
<td><strong>Mat</strong></td>
<td>The unitary matrix for this gate. If the gate is not defined by a single unitary, this will be a 0x0 matrix.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>The name of this gate.</td>
</tr>
<tr>
<td><strong>Op</strong></td>
<td>This gate's operation.</td>
</tr>
<tr>
<td><strong>Parent</strong></td>
<td>This gate's parent, if any.</td>
</tr>
<tr>
<td><strong>Qubits</strong></td>
<td>The arity of this gate; that is, the number of qubits the gate operates on.</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>The user info associated with this gate, if any.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AddControl</td>
<td>Creates a new gate by adding one or more control qubits to an existing unitary gate.</td>
</tr>
<tr>
<td>Build</td>
<td>Gets the definition of a gate, using the cache. If the gate is already in the cache, the cached definition is returned. Otherwise, a new gate will be created, added to the cache, and returned.</td>
</tr>
<tr>
<td>CacheClear</td>
<td>Clears out the gate cache.</td>
</tr>
<tr>
<td>CacheStats</td>
<td>Gets gate cache statistics.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dumps the full gate information to the console and/or log.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>NewMat</td>
<td>Makes a new gate based on this gate but with new unitary matrix.</td>
</tr>
<tr>
<td>OptimizeKraus</td>
<td>Optimze a Channel gate with Kraus operators (checks correctness and orders by magnitude)</td>
</tr>
<tr>
<td>Run</td>
<td>Run this gate. The details depend on the current value of Ket.Mode:</td>
</tr>
<tr>
<td></td>
<td>• RunMode:</td>
</tr>
<tr>
<td></td>
<td>Apply this</td>
</tr>
</tbody>
</table>
gate operation to the supplied qubits.

- **GateMode**: Stores this gate definition into the current Ket. This is for internal use.
- **CircMode**: Compiles this gate into a Circuit. This is for internal use.

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShowMem</td>
<td>Int32, FSharpFuncUnit, String, FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean</td>
<td>Outputs memory and garbage collection statistics to the log and console.</td>
</tr>
<tr>
<td>ShowMem</td>
<td>Int32, String, FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean</td>
<td>Outputs memory and garbage collection statistics to the log and console.</td>
</tr>
</tbody>
</table>
ToString

Returns a string representation of this gate, based on the gate's name, help string, and type. (Overrides Object.ToString.)

Top

See Also

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Gate Constructor

Creates a new gate from scratch.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```fsharp
new :
    Name : FSharpOption<string> *
    Qubits : FSharpOption<int> *
    Mat : FSharpOption<CSMat> *
    Draw : FSharpOption<string> *
    Help : FSharpOption<string> *
    Op : FSharpOption<GateOp> *
    Parent : FSharpOption<FSharpOption<Gate>>*
    User : FSharpOption<FSharpOption<Object>>*
    Kraus : FSharpOption<FSharpList<KrausOp>>*
```

Parameters

Name
Type: Microsoft.FSharp.Core.FSharpOption<String>
The optional name of the gate we're creating. The default is "".

Qubits
Type: Microsoft.FSharp.Core.FSharpOption<Int32>
The optional arity of gate; that is, the number of qubits this gate operates on. If a matrix for the gate is provided using the Mat parameter, then the arity is deduced from the dimensions of the matrix. The default value is 0.

Mat
Type: Microsoft.FSharp.Core.FSharpOption<CSMat>
An optional sparse unitary matrix that implements the gate. This parameter is only used for gates that implement a unitary operation. The default is no unitary.

**Draw**

Type: `Microsoft.FSharp.Core.FSharpOption<String>
An optional string to use to render the gate. This should be a Q-Circuit drawing string, if provided. The default is "", which means that the gate doesn't show up when rendered.

**Help**

Type: `Microsoft.FSharp.Core.FSharpOption<String>
An optional help string for the gate. The default is no help string, "".

**Op**

Type: `Microsoft.FSharp.Core.FSharpOption<GateOp>
The optional gate operation to perform. The default is Normal, which is a unitary gate defined by a matrix.

**Parent**

Type: `Microsoft.FSharp.Core.FSharpOption<FSharpOption<Gate>>
The optional parent gate, if required by the gate operation type. The default is None.

**User**

Type: `Microsoft.FSharp.Core.FSharpOption<FSharpOption<Object>>
An optional user-defined field.

**Kraus**

Type: `Microsoft.FSharp.Core.FSharpOption<FSharpList<KrausOp>>
An optional list of Kraus tags and matrices for Channels.

See Also

Reference
Gate Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Gate Properties

The **Gate** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arity</td>
<td>Get arity of the gate (based on Qubits or Mat/Kraus size).</td>
</tr>
<tr>
<td>CacheDisable</td>
<td>Whether or not the gate cache is disabled. The cache is used if this property is false. The cache should be disabled if gates are being built in parallel.</td>
</tr>
<tr>
<td>Draw</td>
<td>The drawing instructions for this gate.</td>
</tr>
<tr>
<td>Help</td>
<td>The help string for this gate.</td>
</tr>
<tr>
<td>Kraus</td>
<td>The Kraus operator matrix list for this gate (if op type is Channel). If the gate is not a channel, this will be an empty list.</td>
</tr>
<tr>
<td>Mat</td>
<td>The unitary matrix for this gate. If the gate is not defined by a single unitary, this will be a 0x0 matrix.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of this gate.</td>
</tr>
<tr>
<td>Op</td>
<td>This gate's operation.</td>
</tr>
<tr>
<td><strong>Parent</strong></td>
<td>This gate's parent, if any.</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------</td>
</tr>
<tr>
<td><strong>Qubits</strong></td>
<td>The arity of this gate; that is, the number of qubits the gate operates on.</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>The user info associated with this gate, if any.</td>
</tr>
</tbody>
</table>

See Also

Reference
Gate Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateArity Property

Get arity of the gate (based on Qubits or Mat/Kraus size).

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```fsharp
member Arity : int with get
```

Property Value
Type: Int32

See Also

Reference
Gate Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateCacheDisable Property

Whether or not the gate cache is disabled. The cache is used if this property is false. The cache should be disabled if gates are being built in parallel.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```fsharp
static member CacheDisable : bool with get, set
```

Property Value
Type: Boolean

See Also

Reference
Gate Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateDraw Property

The drawing instructions for this gate.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

![Syntax](image)

**Property Value**
Type: String

**See Also**

Reference
Gate Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateHelp Property

The help string for this gate.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.6*)

Syntax

F#

```fsharp
member Help : string with get
```

Property Value
Type: String

See Also

Reference
Gate Class
Microsoft.Research.Liquid Namespace
GateKraus Property

The Kraus operator matrix list for this gate (if op type is Channel). If the gate is not a channel, this will be an empty list.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.2)

Syntax

```fsharp
member Kraus : FSharpList<KrausOp> with get
```

Property Value

Type: FSharpList<KrausOp>

See Also

Reference  
Gate Class  
Microsoft.Research.Liquid Namespace
GateMat Property

The unitary matrix for this gate. If the gate is not defined by a single unitary, this will be a 0x0 matrix.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
**Version:** 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Mat : CSMat with get
```

### Property Value

**Type:** CSMat

### See Also

Reference

Gate Class

Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

GateName Property

The name of this gate.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

Syntax

F#

```f#`

    member Name : string with get
```

Property Value

Type: String

See Also

Reference
Gate Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateOp Property

This gate's operation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```
F#
Copy

member Op : GateOp with get
```

**Property Value**  
**Type:** GateOp

### See Also

**Reference**  
Gate Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateParent Property

This gate's parent, if any.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member Parent : FSharpOption<Gate> with get
```

### Property Value

Type: **FSharpOptionGate**

### See Also

- Reference
  - Gate Class
  - Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

GateQubits Property

The arity of this gate; that is, the number of qubits the gate operates on.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
member Qubits : int with get
```

### Property Value

Type: **Int32**

### See Also

**Reference**  
Gate Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

GateUser Property

The user info associated with this gate, if any.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

F#

```fsharp
member User : FSharpOption<Object> with get
```

Property Value
Type: FSharpOption<Object>

See Also

Reference
Gate Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Gate Methods

The `Gate` type exposes the following members.

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>AddControl</td>
</tr>
<tr>
<td>Build</td>
</tr>
<tr>
<td>CacheClear</td>
</tr>
<tr>
<td>CacheStats</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>Dump</td>
</tr>
<tr>
<td>Equals</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>GetHashCode</td>
</tr>
<tr>
<td>GetType</td>
</tr>
<tr>
<td>NewMat</td>
</tr>
<tr>
<td>OptimizeKraus</td>
</tr>
</tbody>
</table>
Run this gate. The details depend on the current value of Ket.Mode:
- **RunMode**: Apply this gate operation to the supplied qubits.
- **GateMode**: Stores this gate definition into the current Ket. This is for internal use.
- **CircMode**: Compiles this gate into a Circuit. This is for internal use.

**ShowMem(Int32, FSharpFuncUnit, String, FSharpOptionFSharpListCircuit, FSharpOptionFSharpListCircuit, FSharpOptionFSharpListCircuit)** Outputs memory and garbage
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShowMem(Int32, String, FSharpOption FSharpList Circuit, FSharpOption Boolean, FSharpOption Boolean)</td>
<td>Outputs memory and garbage collection statistics to the log and console.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string representation of this gate, based on the gate's name, help string, and type. (Overrides ObjectToString.)</td>
</tr>
</tbody>
</table>

**See Also**

- Reference
  - Gate Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateAddControl Method

Creates a new gate by adding one or more control qubits to an existing unitary gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)  

### Syntax

```fsharp
member AddControl :  
  count : FSharpOption<int> *  
  noCache : FSharpOption<bool> -> Gate
```

### Parameters

**count**  
Type: Microsoft.FSharp.Core.FSharpOption<int32>  
The optional number of control qubits to add. The default is a single control.

**noCache**  
Type: Microsoft.FSharp.Core.FSharpOption<Boolean>  
An option to prevent caching of the new gate. The default is false, which means that the new gate should be cached. The key for the cache is a sequence of capital Cs, one for each control qubit added, followed by an underscore, '_', and then the name of the base gate.

### Return Value

Type: Gate  
The new controlled gate

### See Also

- member AddControl: count: FSharpOption<int> * noCache: FSharpOption<bool> -> Gate
GateBuild Method

Gets the definition of a gate, using the cache. If the gate is already in the cache, the cached definition is returned. Otherwise, a new gate will be created, added to the cache, and returned.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```
static member Build :
    key : string *
    gen : FSharpFunc<Unit, Gate> -> Gate
```

### Parameters

**key**

- Type: System.String
- The unique key to use to identify this gate in the cache. This may be more than just the gate name; for instance, for a rotation gate, the cache key must include the rotation angle as well as just "Rz". If the key is null or "", the cache will be bypassed.

**gen**

- Type: Microsoft.FSharp.Core.FSharpFunc<Unit, Gate>
- A function that may be used to create a definition for the gate if it is not already in the cache.

### Return Value

- Type: Gate
- The gate associated with the provided key

### See Also
Reference

Gate Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateCacheClear Method

Clears out the gate cache.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

### Syntax

```fsharp
static member CacheClear : unit -> unit
```

### See Also

- Reference
  - Gate Class
  - Microsoft.Research.Liquid Namespace
GateCacheStats Method

Gets gate cache statistics.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member CacheStats : unit -> Tuple<int, int>
```

**Return Value**

**Type:** Tuple<int, int>

A tuple of the cache hit count and cache miss count.

### See Also

- Reference
- Gate Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateDump Method

Dumps the full gate information to the console and/or log.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

## Syntax

```
F#  Copy

member Dump :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<int, String>, Unit>
    level : FSharpOption<int> -> unit
```

### Parameters

- **f**
  - *Type:* Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int, String>, Unit>
  - The optional output function to use. The default is `showLogInd`.

- **level**
  - *Type:* Microsoft.FSharp.Core.FSharpOption<int>
  - The optional indentation level. The default is 0.

## See Also

- Reference
  - Gate Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateNewMat Method

Makes a new gate based on this gate but with new unitary matrix.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
member NewMat : mat : CSMat -> Gate
```

Parameters

*mat*

Type: Microsoft.Research.LiquidCSMat
The unitary matrix for the new gate.

Return Value

Type: Gate
The new gate

See Also

Reference
Gate Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateOptimizeKraus Method

Optimize a Channel gate with Kraus operators (checks correctness and orders by magnitude)

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```fsharp
member OptimizeKraus : g2 : FSharpOption<Gate> *
nname : FSharpOption<string> *
idMult : FSharpOption<int> *
povm : FSharpOption<string> -> FSharpFunc
```

Parameters

**g2**
Type: Microsoft.FSharp.Core.FSharpOption<Gate>
Gate to append to make more complex channels
(optional=Non)  

**name**
Type: Microsoft.FSharp.Core.FSharpOption<String>
New name for the gate (optional=K####)

**idMult**
Type: Microsoft.FSharp.Core.FSharpOption<Int32>
If appending, how much to multiply the parent gate IDs by
(optional=count of appended)

**povm**
Type: Microsoft.FSharp.Core.FSharpOption<String>
Tag to use as POVM symbol for the new gate
Return Value
Type: `FSharpFunc<FSharpList<Qubit, Unit>>` New Gate Function

See Also

Reference
Gate Class
Microsoft.Research.Liquid Namespace
GateRun Method

Run this gate. The details depend on the current value of Ket.Mode:

- **RunMode**: Apply this gate operation to the supplied qubits.
- **GateMode**: Stores this gate definition into the current Ket. This is for internal use.
- **CircMode**: Compiles this gate into a Circuit. This is for internal use.

**Namespace**: Microsoft.Research.Liquid

**Assembly**: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
member Run : 
    qs : FSharpList<Qubit>  ->  unit
```

**Parameters**

**qs**

Type: `Microsoft.FSharp.Collections.FSharpList`<Qubit>

The Qubits for this gate to operate on. They are also used to identify the current Ket.

### See Also

**Reference**

- Gate Class
- Microsoft.Research.Liquid Namespace
## GateShowMem Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="ShowMem" /> ShowMem(Int32, \text{FSharpFuncUnit}, \text{String}, \text{FSharpOption} F\text{SharpListCircuit}, \text{FSharpOption} Boolean, \text{FSharpOption} Boolean)</td>
<td>Outputs memory and garbage collection statistics to the log and console.</td>
</tr>
<tr>
<td><img src="#" alt="ShowMem" /> ShowMem(Int32, \text{String}, \text{FSharpOption} F\text{SharpListCircuit}, \text{FSharpOption} Boolean, \text{FSharpOption} Boolean)</td>
<td>Outputs memory and garbage collection statistics to the log and console.</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- Gate Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateShowMem Method (Int32, FSharpFuncUnit, String, FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean)

Outputs memory and garbage collection statistics to the log and console.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

**Syntax**

```fsharp
static member ShowMem :
    seconds : int *
    f : FSharpFunc<Unit, string> *
    cs : FSharpOption<FSharpList<Circuit>> *
    deep : FSharpOption<bool> *
    collect : FSharpOption<bool> -&gt; unit
```

**Parameters**

`seconds`  
Type: System.Int32  
A minimum interval between reports. If a report was displayed within more recently, then no new report is displayed.

`f`
Type: Microsoft.FSharp.Core.FSharpFunc<Unit, String>
A function that returns a string to output as a prefix to the report.

cs
Type: Microsoft.FSharp.Core.FSharpOption<FSharpList<Circuit>>
An optional list of Circuits to analyze. The count of Circuits in the list is reported. If the "deep" parameter is true, then the total count of gates in these Circuits is also reported. The default is an empty list, which displays counts of 0.

depth
Type: Microsoft.FSharp.Core.FSharpOption<Boolean>
An option indicating whether or not to count the gates in the list of Circuits. The default is false, which indicates that the gates should not be counted and will be reported as 0.

collect
Type: Microsoft.FSharp.Core.FSharpOption<Boolean>
An option indicating whether to force a garbage collection. The default is false, to not force a collection.

See Also

Reference
Gate Class
ShowMem Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

GateShowMem Method (Int32, String, FSharpOption FSharpList Circuit, FSharpOption Boolean, FSharpOption Boolean)

Outputs memory and garbage collection statistics to the log and console.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
static member ShowMem :
    seconds : int *
    str : string *
    cs : FSharpOption<FSharpList<Circuit>> *
    deep : FSharpOption<bool> *
    collect : FSharpOption<bool> -> unit
```

Parameters

`seconds`
Type: System.Int32
A minimum interval between reports. If a report was displayed within more recently, then no new report is displayed.

`str`
Type: **System.String**
A string to output as a prefix to the report.

**cs**
Type: **Microsoft.FSharp.Core.FSharpOption**
An optional list of Circuits to analyze. The count of Circuits in the list is reported. If the "deep" parameter is true, then the total count of gates in these Circuits is also reported. The default is an empty list, which displays counts of 0.

**deep**
Type: **Microsoft.FSharp.Core.FSharpOption**
An option indicating whether or not to count the gates in the list of Circuits. The default is false, which indicates that the gates should not be counted and will be reported as 0.

**collect**
Type: **Microsoft.FSharp.Core.FSharpOption**
An option indicating whether to force a garbage collection. The default is false, to not force a collection.

**See Also**

Reference
Gate Class
ShowMem Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

GateToString Method

Returns a string representation of this gate, based on the gate's name, help string, and type.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5)

### Syntax

```
abstract ToString : unit -> string
override ToString : unit -> string
```

### Return Value

Type: String  
The string representation

### See Also

Reference  
Gate Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateOp Class

Gate operation type. This is used in Gate definitions.

Inheritance Hierarchy

- System
  - Object
    - Microsoft.Research.Liquid.GateOp

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
[<SerializableAttribute>]
type GateOp = class end
```

The GateOp type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✪ Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>✪ GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>✪ GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
</tbody>
</table>
ToString

Creates a string representation of this gate operation type.
(Overrides ObjectToString.)

### Remarks

The possible values for this type are:

- **Normal**: Implements a unitary operation and is defined by a matrix.
- **Measure**: Measures a single qubit or the joint parity of a group of qubits.
- **Channel(sym)**: Selects a Kraus operator to run (sym is a symbol to store the selected index for POVMs)
- **Reset(b)**: Reanimates a measured qubit to the Bit value b. If b is Unknown, then the qubit is reanimated to its last measured value.
- **String**: Puts a label in a diagram
- **Modify(n)**: Modifies a parent gate. The modified gate takes n more wires (input qubits) than the parent gate.
- **BCOp(n,f)**: Implements a classically-controlled gate. There are two cases: if n is greater than zero, then the control is logically based on the measured values of n qubits. If n is equal to zero, then the control is logically based on the result of one or more joint parity measurements. In either case, f is the function that determines whether or not to execute the parent gate.
- **WrapOp(f)**: Wraps multiple gates into one logical gate. This is used for multi-level circuit drawing.
- **WrapHam(pqrs,f)**: Wraps multiple gates into a single Hamiltonian term. This is used for GrowSingle.

### See Also

Reference
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

GateOp Methods

The `GateOp` type exposes the following members.

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Equals" /></td>
<td>Determines whether the specified object is equal to the current object. <em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="GetHashCode" /></td>
<td>Serves as the default hash function. <em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="GetType" /></td>
<td>Gets the <code>Type</code> of the current instance. <em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="ToString" /></td>
<td>Creates a string representation of this gate operation type. <em>(Overrides <code>Object.ToString</code>)</em></td>
</tr>
</tbody>
</table>

**Top**

**See Also**

- Reference
  - `GateOp Class`
  - `Microsoft.Research.Liquid Namespace`

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GateOpToString Method

Creates a string representation of this gate operation type.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```
abstract ToString : unit -> string
override ToString : unit -> string
```

Return Value
Type: String
The string

See Also

Reference
GateOp Class
Microsoft.Research.Liquid Namespace
GrowPars Class


Inheritance Hierarchy

- System
- Object
- Microsoft.Research.Liquid
  - GrowPars

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
[<SerializableAttribute>]
type GrowPars = class end
```

The GrowPars type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GrowPars(FSharpOption&lt;int32&gt;, FSharpOption&lt;int32&gt;, FSharpOption&lt;bool&gt;)</code></td>
<td>Creates an instance with appropriate parameters for Circuit.GrowGates, which generates a denser circuit equivalent to an existing Circuit.</td>
</tr>
</tbody>
</table>
GrowPars(Boolean, FSharpOption Int32, FSharpOption Int32, FSharpOption Int32, FSharpOption FSharpList Int32, FSharpOption Int32, FSharpOption Boolean, FSharpOption Boolean, FSharpOption Tuple Double, Boolean)

Creates an instance with appropriate parameters for Circuit.GrowSingle, which generates a single unitary operation equivalent to an existing Circuit.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowDense</td>
<td>If true, allow dense matrices to be generated by Circuit.GrowGates.</td>
</tr>
<tr>
<td>Coalesce</td>
<td>For Hamiltonian circuits, a tuple of a scale value and a flag specifying whether or not to keep rotation gates around small angles. The scale value is used if the flag is false. See the Users' Guide for details.</td>
</tr>
<tr>
<td>Diff</td>
<td>For Fermion circuits, a list of allowed differences between spin-up and spin-down counts. Effectively, this list constrains the possible values of the total net spin of valid configurations. An empty list, [], means to allow any difference.</td>
</tr>
<tr>
<td>ECnt</td>
<td>For Fermion circuits, the valid total electron count (total number of</td>
</tr>
</tbody>
</table>
occupied states). A value of 0 means that any number of electrons is valid.

<table>
<thead>
<tr>
<th>Half</th>
<th>For Fermion circuits, are the qubits ordered so that the first half are the spin-up states? The alternative is for qubits to represent alternating spin-up and spin-down states.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxWires</td>
<td>The maximum wire count for a single grown gate generated by Circuit.GrowGates</td>
</tr>
<tr>
<td>OCnt</td>
<td>For Fermion circuits, the number of electron states. In chemical simulations, this is the spin orbital count.</td>
</tr>
<tr>
<td>Parity</td>
<td>For Fermion circuits, whether to enforce conservation of angular momentum via parity.</td>
</tr>
<tr>
<td>Redund</td>
<td>For Fermion circuits, whether to ignore (set to zero) portions of the resulting unitary that don't satisfy the electron count, parity check, or net spin constraints. See the Users' Guide for details.</td>
</tr>
<tr>
<td>Single</td>
<td>If true, build a single unitary; if false, grow gates into a denser circuit.</td>
</tr>
<tr>
<td>Skip</td>
<td>Count of initial qubits to skip (because they are used for phase estimation) in Circuit.GrowSingle.</td>
</tr>
<tr>
<td>Verbose</td>
<td>The logging verbosity level. Possible</td>
</tr>
</tbody>
</table>
values are:
- **0**: No grow logging
- **1**: Final circuit logging
- **2**: Full detailed logging

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Gets a string representation of this circuit grow parameter set. (Overrides <strong>ObjectToString</strong>.)</td>
</tr>
<tr>
<td><strong>VerboseSet</strong></td>
<td>Creates a new GrowPars with a different verbosity setting.</td>
</tr>
</tbody>
</table>

### See Also

Reference

**Microsoft.Research.Liquid Namespace**
GrowPars Constructor

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GrowPars(FSharpOptionInt32, FSharpOptionInt32, FSharpOptionBoolean)</td>
<td>Creates an instance with appropriate parameters for Circuit.GrowGates, which generates a denser circuit equivalent to an existing Circuit.</td>
</tr>
<tr>
<td>GrowPars(Boolean, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionFSharpListInt32, FSharpOptionInt32, FSharpOptionBoolean, FSharpOptionBoolean, FSharpOptionTupleDouble, Boolean)</td>
<td>Creates an instance with appropriate parameters for Circuit.GrowSingle, which generates a single unitary operation equivalent to an existing Circuit.</td>
</tr>
</tbody>
</table>

See Also

Reference
GrowPars Class
Microsoft.Research.Liquid Namespace
GrowPars Constructor
(FSharpOptionInt32, FSharpOptionInt32, FSharpOptionBoolean)

Creates an instance with appropriate parameters for
Circuit.GrowGates, which generates a denser circuit equivalent to an
existing Circuit.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```
F#

new : maxWires : FSharpOption<int> *
      verbose : FSharpOption<int> *
      allowDense : FSharpOption<bool> -> GrowPars
```

Parameters

maxWires
Type: Microsoft.FSharp.Core.FSharpOption<int>
An optional value for MaxWires. The default is 11.

verbose
Type: Microsoft.FSharp.Core.FSharpOption<int>
An optional value for Verbosity. The default is 0.

allowDense
Type: Microsoft.FSharp.Core.FSharpOption<bool>
An optional value for AllowDense. The default is false.

See Also

Reference
GrowPars Class
GrowPars Overload
Microsoft.Research.Liquid Namespace
GrowPars Constructor (Boolean, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionFSharpListInt32, FSharpOptionInt32, FSharpOptionBoolean, FSharpOptionBoolean, FSharpOptionOptionTupleDouble, Boolean)

Creates an instance with appropriate parameters for Circuit.GrowSingle, which generates a single unitary operation equivalent to an existing Circuit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

**Syntax**

```fsharp
new :  
  half : bool *  
  eCnt : FSharpOption<int> *  
  oCnt : FSharpOption<int> *
```
### Parameters

**half**
- **Type:** System.Boolean
- The value for Half. This parameter is required.

**eCnt**
- **Type:** Microsoft.FSharp.Core.FSharpOption<int>
- An optional value for ECnt. The default is 0.

**oCnt**
- **Type:** Microsoft.FSharp.Core.FSharpOption<int>
- An optional value for OCnt. The default is 0.

**skip**
- **Type:** Microsoft.FSharp.Core.FSharpOption<int>
- An optional value for Skip. The default is 0.

**diff**
- **Type:** Microsoft.FSharp.Core.FSharpOption<FSharpList<int>>
- An optional value for ECnt. The default is an empty list, [].

**verbose**
- **Type:** Microsoft.FSharp.Core.FSharpOption<int>
- An optional value for Verbose. The default is 1.

**parity**
- **Type:** Microsoft.FSharp.Core.FSharpOption<bool>
- An optional value for Parity. The default is false.

**redund**
- **Type:** Microsoft.FSharp.Core.FSharpOption<bool>
- An optional value for Redund. The default is 0.

**coalesce**
- **Type:** Microsoft.FSharp.Core.FSharpOption<Tuple<float, bool>>
An optional value for Coalesce. The default is (0.0, false).

See Also

Reference
GrowPars Class
GrowPars Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

GrowPars Properties

The **GrowPars** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="" /> <strong>AllowDense</strong></td>
<td>If true, allow dense matrices to be generated by <code>Circuit.GrowGates</code>.</td>
</tr>
<tr>
<td><img src="image" alt="" /> <strong>Coalesce</strong></td>
<td>For Hamiltonian circuits, a tuple of a scale value and a flag specifying whether or not to keep rotation gates around small angles. The scale value is used if the flag is false. See the Users' Guide for details.</td>
</tr>
<tr>
<td><img src="image" alt="" /> <strong>Diff</strong></td>
<td>For Fermion circuits, a list of allowed differences between spin-up and spin-down counts. Effectively, this list constrains the possible values of the total net spin of valid configurations. An empty list, [], means to allow any difference.</td>
</tr>
<tr>
<td><img src="image" alt="" /> <strong>ECnt</strong></td>
<td>For Fermion circuits, the valid total electron count (total number of occupied states). A value of 0 means that any number of electrons is valid.</td>
</tr>
<tr>
<td><img src="image" alt="" /> <strong>Half</strong></td>
<td>For Fermion circuits, are the qubits</td>
</tr>
</tbody>
</table>
ordered so that the first half are the spin-up states? The alternative is for qubits to represent alternating spin-up and spin-down states.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxWires</td>
<td>The maximum wire count for a single grown gate generated by Circuit.GrowGates</td>
</tr>
<tr>
<td>OCnt</td>
<td>For Fermion circuits, the number of electron states. In chemical simulations, this is the spin orbital count.</td>
</tr>
<tr>
<td>Parity</td>
<td>For Fermion circuits, whether to enforce conservation of angular momentum via parity.</td>
</tr>
<tr>
<td>Redund</td>
<td>For Fermion circuits, whether to ignore (set to zero) portions of the resulting unitary that don't satisfy the electron count, parity check, or net spin constraints. See the Users' Guide for details.</td>
</tr>
<tr>
<td>Single</td>
<td>If true, build a single unitary; if false, grow gates into a denser circuit.</td>
</tr>
<tr>
<td>Skip</td>
<td>Count of initial qubits to skip (because they are used for phase estimation) in Circuit.GrowSingle.</td>
</tr>
</tbody>
</table>
| Verbose   | The logging verbosity level. Possible values are:  
  * **0**: No grow logging  
  * **1**: Final circuit logging  
  * **2**: Full detailed logging |
See Also

Reference
GrowPars Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GrowPars.AllowDense Property

If true, allow dense matrices to be generated by Circuit.GrowGates.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
**Version:** 1.0.5981.24943 (1.0*)

**Syntax**

```fsharp
member AllowDense : bool with get
```

**Property Value**

Type: **Boolean**

**See Also**

Reference
- GrowPars Class
- Microsoft.Research.Liquid Namespace
GrowPars\textit{Coalesce} Property

For Hamiltonian circuits, a tuple of a scale value and a flag specifying whether or not to keep rotation gates around small angles. The scale value is used if the flag is false. See the Users' Guide for details.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)  

### Syntax

```f#
member Coalesce : Tuple<float, bool> with get
```

**Property Value**  
Type: TupleDouble, Boolean

### See Also

**Reference**  
GrowPars Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GrowParsDiff Property

For Fermion circuits, a list of allowed differences between spin-up and spin-down counts. Effectively, this list constrains the possible values of the total net spin of valid configurations. An empty list, [], means to allow any difference.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Diff : FSharpList<int> with get
```

### Property Value

**Type:** FSharpList<int>

### See Also

**Reference**

- GrowPars Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GrowParsECnt Property

For Fermion circuits, the valid total electron count (total number of occupied states). A value of 0 means that any number of electrons is valid.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

F#

    member ECnt : int with get

See Also

Reference
GrowPars Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GrowParsHalf Property

For Fermion circuits, are the qubits ordered so that the first half are the spin-up states? The alternative is for qubits to represent alternating spin-up and spin-down states.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

**Syntax**

```fsharp
member Half : bool with get
```

**Property Value**

Type: **Boolean**

**See Also**

Reference

GrowPars Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GrowParsMaxWires Property

The maximum wire count for a single grown gate generated by Circuit.GrowGates

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member MaxWires : int with get
```

### Property Value

Type: **Int32**

### See Also

Reference
GrowPars Class  
Microsoft.Research.Liquid Namespace
GrowParsOCnt Property

For Fermion circuits, the number of electron states. In chemical simulations, this is the spin orbital count.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member OCnt : int with get
```

### Property Value

Type: **Int32**

### See Also

**Reference**  
GrowPars Class  
Microsoft.Research.Liquid Namespace
GrowParsParity Property

For Fermion circuits, whether to enforce conservation of angular momentum via parity.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

▶ Syntax

```f#
member Parity : bool with get
```

Property Value
Type: Boolean

▶ See Also

Reference
GrowPars Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GrowParsRedund Property

For Fermion circuits, whether to ignore (set to zero) portions of the resulting unitary that don't satisfy the electron count, parity check, or net spin constraints. See the Users' Guide for details.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```
member Redund : bool with get
```

### Property Value

Type: **Boolean**

### See Also

Reference  
GrowPars Class  
Microsoft.Research.Liquid Namespace
GrowParsSingle Property

If true, build a single unitary; if false, grow gates into a denser circuit.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
member Single : bool with get
```

Property Value
Type: Boolean

See Also

Reference
GrowPars Class
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

GrowParsSkip Property

Count of initial qubits to skip (because they are used for phase estimation) in Circuit.GrowSingle.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.2)

Syntax

```
member Skip : int with get
```

Property Value
Type: Int32

See Also

Reference
GrowPars Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
### GrowParsVerbose Property

The logging verbosity level. Possible values are:

- **0**: No grow logging
- **1**: Final circuit logging
- **2**: Full detailed logging

**Namespace**: Microsoft.Research.Liquid  
**Assembly**: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Verbose : int with get
```

### Property Value

Type: **Int32**

### See Also

- Reference
- GrowPars Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

## GrowPars Methods

The **GrowPars** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>).</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>).</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>).</td>
</tr>
<tr>
<td>ToString</td>
<td>Gets a string representation of this circuit grow parameter set.</td>
</tr>
<tr>
<td></td>
<td>(Overrrides <a href="#">ObjectToString</a>).</td>
</tr>
<tr>
<td>VerboseSet</td>
<td>Creates a new GrowPars with a different verbosity setting.</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - **GrowPars Class**
GrowParsToString Method

Gets a string representation of this circuit grow parameter set.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
abstract ToString : unit -> string
override ToString : unit -> string
```

Return Value

Type: String
The string

See Also

Reference
GrowPars Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
GrowParsVerboseSet Method

Creates a new GrowPars with a different verbosity setting.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5981.24943)

### Syntax

```fsharp
member VerboseSet : vNew : int -> GrowPars
```

### Parameters

**vNew**
- Type: System.Int32
- The new verbosity level

### Return Value
- Type: GrowPars
- The new GrowPars instance

### See Also

- [GrowPars Class](#)
- [Microsoft.Research.Liquid Namespace](#)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Hamiltonian Class

Base class for Hamiltonian dynamics simulators.

Inheritance Hierarchy

- System
- Microsoft.Research.Liquid
  - Microsoft.Research.Liquid.Hamiltonian
  - Microsoft.Research.Liquid.Fermion
  - Microsoft.Research.Liquid.Spin

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4+)

Syntax

```fsharp
[<SerializableAttribute>]
type Hamiltonian = class end
```

The Hamiltonian type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamiltonian</td>
<td>Initializes a new instance of the Hamiltonian class</td>
</tr>
</tbody>
</table>

Properties
**decohereModel**  The decoherence model for this Hamiltonian.

**Ket**  Gets the Ket vector associated with this Hamiltonian.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Hamiltonian Constructor

Initializes a new instance of the Hamiltonian class

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```fsharp
new : unit -> Hamiltonian
```

### See Also

Reference  
Hamiltonian Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The **Hamiltonian** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>decohereModel</td>
<td>The decoherence model for this Hamiltonian.</td>
</tr>
<tr>
<td>Ket</td>
<td>Gets the Ket vector associated with this Hamiltonian</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

- [Hamiltonian Class](#)
- [Microsoft.Research.Liquid Namespace](#)
Hamiltonian.decohereModel

Property

The decoherence model for this Hamiltonian.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5981)

### Syntax

```
member decohereModel : FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>><
```

### Property Value

**Type:** `FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>>`

### Remarks

A decoherence model is a list of gates and a decoherence probability for each gate. Gates that do not appear have a zero decoherence probability. Note that gates appearing in this list must be fully resolved; they may have no parameters other than the list of qubits.

### See Also

**Reference**
- **Hamiltonian Class**
- **Microsoft.Research.Liquid Namespace**

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Hamilto

namespace: Microsoft.Research.Liquid
assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

member Ket : Ket with get, set

Property Value
Type: Ket

See Also

Reference
Hamiltonian Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Hamiltonian Methods

The **Hamiltonian** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>

See Also

**Reference**

- Hamiltonian Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

HamiltonianGates Class

A collection of gates that are useful for Hamiltonian simulation and annealing.

Inheritance Hierarchy

- System
- Object
- Microsoft.Research.Liquid
- HamiltonianGates

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5981.24943)

Syntax

```fsharp
[<AbstractClassAttribute>]
[<SealedAttribute>]
type HamiltonianGates = class end
```

The HamiltonianGates type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![S]</td>
<td><strong>CGtheta</strong> performs a controlled global phase rotation.</td>
</tr>
<tr>
<td>![S]</td>
<td><strong>CRx</strong> performs a Controlled Pauli X rotation.</td>
</tr>
<tr>
<td>![S]</td>
<td><strong>CRy</strong> performs a Controlled Pauli Y rotation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRz</td>
<td>Performs a Controlled Pauli Z rotation.</td>
</tr>
<tr>
<td>CTtheta</td>
<td>Performs a controlled T rotation.</td>
</tr>
<tr>
<td>Entangle</td>
<td>Entangles a list of qubits. This is useful for building Jordan-Wigner strings.</td>
</tr>
<tr>
<td>Gtheta</td>
<td>Performs a global phase rotation. This is functionally equivalent to ( R_{\text{pauli}} (2.0^\ast \theta) \text{ I q}_{\text{s}} ), but has some additional drawing options.</td>
</tr>
<tr>
<td>LoadCache</td>
<td>Preload the cache with the ZZ, Ybasis, and YbasisAdj gates.</td>
</tr>
<tr>
<td>Rpauli</td>
<td>Performs an arbitrary rotation based on an existing gate. The base gate may be any unitary gate with a Hermitian, idempotent matrix. Of course, all Pauli gates satisfy this criteria.</td>
</tr>
<tr>
<td>Rx</td>
<td>Performs a Pauli X rotation. This is functionally equivalent to ( R_{\text{pauli}} \theta \text{ X q}_{\text{s}} ), but has some additional drawing options.</td>
</tr>
<tr>
<td>Ry</td>
<td>Performs a Pauli Y rotation. This is functionally equivalent to ( R_{\text{pauli}} \theta \text{ Y q}_{\text{s}} ), but has some additional drawing options.</td>
</tr>
<tr>
<td>Rz</td>
<td>Performs a Pauli Z rotation. This is functionally equivalent to ( R_{\text{pauli}} \theta \text{ Z q}_{\text{s}} ), but has some</td>
</tr>
</tbody>
</table>
additional drawing options.

<table>
<thead>
<tr>
<th></th>
<th>Ttheta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Performs a phase gate rotation. This is functionally equivalent to ( \text{Rpauli} (2.0*\text{theta}) \ T \ \text{qs} ), but has some additional drawing options.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>UnEntangle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unentangles a list of qubits. This is useful for building Jordan-Wigner strings.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Ybasis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Performs a basis change from ( Z ) to ( Y ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>YbasisAdj</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Performs a basis change from ( Y ) to ( Z ). This is the adjoint of ( \text{Ybasis} ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ZR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Performs a Pauli ( Z ) rotation. This is equivalent to ( \text{Rpauli} (2.0*\text{theta}) \ Z \ \text{qs} ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ZZ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Performs a ZZ gate: Pauli Zs on consecututive wires. This is used for coupling strength.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ZZR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Performs a Pauli ZZ rotation; that is, a simultaneous ( Z ) rotation of two qubits. This is equivalent to ( \text{Rpauli} (2.0*\text{theta}) \ ZZ \ \text{qs} ).</td>
</tr>
</tbody>
</table>

See Also

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
# HamiltonianGates Methods

The `HamiltonianGates` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CGtheta</code></td>
<td>Performs a controlled global phase rotation.</td>
</tr>
<tr>
<td><code>CRx</code></td>
<td>Performs a Controlled Pauli X rotation.</td>
</tr>
<tr>
<td><code>CRy</code></td>
<td>Performs a Controlled Pauli Y rotation.</td>
</tr>
<tr>
<td><code>CRz</code></td>
<td>Performs a Controlled Pauli Z rotation.</td>
</tr>
<tr>
<td><code>CTtheta</code></td>
<td>Performs a controlled T rotation.</td>
</tr>
<tr>
<td><code>Entangle</code></td>
<td>Entangles a list of qubits. This is useful for building Jordan-Wigner strings.</td>
</tr>
<tr>
<td><code>Gtheta</code></td>
<td>Performs a global phase rotation. This is functionally equivalent to <code>R_{\text{Pauli}} \cdot \left(2.0^*\theta\right) \cdot I^\otimes q\cdot s</code>, but has some additional drawing options.</td>
</tr>
<tr>
<td><code>LoadCache</code></td>
<td>Preload the cache with the ZZ, Ybasis, and YbasisAdj gates.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Operation</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| ![R]
| R\text{pauli} | Performs an arbitrary rotation based on an existing gate. The base gate may be any unitary gate with a Hermitian, idempotent matrix. Of course, all Pauli gates satisfy this criteria. |
| ![Rx]
| Rx    | Performs a Pauli X rotation. This is functionally equivalent to \text{Rpauli \theta X qs}, but has some additional drawing options. |
| ![Ry]
| Ry    | Performs a Pauli Y rotation. This is functionally equivalent to \text{Rpauli \theta Y qs}, but has some additional drawing options. |
| ![Rz]
| Rz    | Performs a Pauli Z rotation. This is functionally equivalent to \text{Rpauli \theta Z qs}, but has some additional drawing options. |
| ![T theta]
| T\theta | Performs a phase gate rotation. This is functionally equivalent to \text{Rpauli (2.0*\theta) T qs}, but has some additional drawing options. |
| ![UnEntangle]
| Un\text{Entangle} | Unentangles a list of qubits. This is useful for building Jordan-Wigner strings. |
| ![Y basis]
| Y\text{basis} | Performs a basis change from Z to Y. |
| ![Y basis Adj]
| Y\text{basis Adj} | Performs a basis change from Y to Z. This is the adjoint of Y\text{basis}. |
| ![ZR]
| ZR    | Performs a Pauli Z rotation. This is |
equivalent to $R_{\text{pauli}} (2.0*\theta) Z$ $q$s.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>ZZ</th>
<th>Performs a ZZ gate: Pauli Zs on consectutive wires. This is used for coupling strength.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>ZZR</td>
<td>Performs a Pauli ZZ rotation; that is, a simultaneous Z rotation of two qubits. This is equivalent to $R_{\text{pauli}} (2.0*\theta) Z$ $q$s.</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- HamiltonianGates Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

HamiltonianGatesCGtheta Method

Performs a controlled global phase rotation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1*)

### Syntax

```
static member CGtheta :  
    theta : float *  
    factor : float *  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**theta**  
Type: System.Double  
The rotation angle. 2*pi is a full rotation.

**factor**  
Type: System.Double  
A multiplier, used to compute the gate label.

**qs**  
Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
The first qubit in the list is the control; the second qubit is the target.

### See Also

Reference  
HamiltonianGates Class
HamiltonianGatesCRx Method

Performs a Controlled Pauli X rotation.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0.0)

Syntax

```
static member CRx :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

*theta*
Type: System.Double
The rotation angle. 4*pi is a full rotation.

*factor*
Type: System.Double
A multiplier, used to compute the gate label.

*subs*
Type: System.String
A subscript to attach to the gate label. Use an empty string, """, if no subscript is desired.

*qs*
Type: Microsoft.FSharp.Collections.FSharpList<FSharpList<Qubit>>
The first qubit in the list is the control; the second qubit is the target.
See Also

Reference

HamiltonianGates Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
HamiltonianGatesCRy Method

Performs a Controlled Pauli Y rotation.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
static member CRy :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

### Parameters

**theta**

Type: SystemDouble
The rotation angle. 4*pi is a full rotation.

**factor**

Type: SystemDouble
A multiplier, used to compute the gate label.

**subs**

Type: SystemString
A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

**qs**

Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
The first qubit in the list is the control; the second qubit is the target.
See Also

Reference

HamiltonianGates Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
HamiltonianGatesCRz Method

Performs a Controlled Pauli Z rotation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member CRz :  
    theta : float *  
    factor : float *  
    subs : string *  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

* **theta**
  - Type: SystemDouble
  - The rotation angle. 4*pi is a full rotation.

* **factor**
  - Type: SystemDouble
  - A multiplier, used to compute the gate label.

* **subs**
  - Type: SystemString
  - A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

* **qs**
  - Type: Microsoft.FSharp.CollectionsFSharpListQubit
  - The first qubit in the list is the control; the second qubit is the target.
See Also

Reference
HamiltonianGates Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQui|>) Simulator

HamiltonianGatesCTtheta Method

Performs a controlled T rotation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

⚠️ Syntax

```fsharp
static member CTtheta :  
  theta : float *  
  factor : float *  
  qs : FSharpList<Qubit> -> unit
```

**Parameters**

- **theta**  
  Type: SystemDouble  
  The rotation angle. 2*pi is a full rotation.

- **factor**  
  Type: SystemDouble  
  A multiplier, used to compute the gate label.

- **qs**  
  Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
  The first qubit in the list is the control; the second qubit is the target.

⚠️ See Also

- Reference
  HamiltonianGates Class
HamiltonianGatesEntangle Method

Entangles a list of qubits. This is useful for building Jordan-Wigner strings.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.∗)

### Syntax

```fsharp
static member Entangle :
    ladder : FSharpList<Tuple<int, int>> *  
    qs : FSharpList<Qubit> -> unit
```

**Parameters**

- **ladder**
  Type: Microsoft.FSharp.CollectionsFSharpList<Tuple<int, int>>
  A list of pairs of qubit indices to entangle. Each entry should have a tuple of two indices that refer to qubits in the qs list. The entangle gate will wrap a sequence of CNOT gates, one for each tuple, each with the control being the first qubit in the tuple and the target being the second in the tuple. The CNOT gates are built in ladder list order.

- **qs**
  Type: Microsoft.FSharp.CollectionsFSharpList<Qubit>
  The qubits to build the ladder from.

### See Also
Reference
HamiltonianGates Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
HamiltonianGatesGtheta Method

Performs a global phase rotation. This is functionally equivalent to $R_{\text{Pauli}}(2.0*\text{theta}) I_{qs}$, but has some additional drawing options.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member Gtheta :
    theta : float *
    factor : float *
    qs : FSharpList<Qubit> -> unit
```

### Parameters

#### theta
Type: `System.Double`
The rotation angle. $2\pi$ is a full rotation.

#### factor
Type: `System.Double`
A multiplier, used to compute the gate label.

#### qs
Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
The first qubit in the list is rotated.

### See Also

Reference

- HamiltonianGates Class
- Microsoft.Research.Liquid Namespace
HamiltonianGatesLoadCache Method

Preload the cache with the ZZ, Ybasis, and YbasisAdj gates.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member LoadCache : unit -> unit
```

### See Also

**Reference**  
HamiltonianGates Class  
Microsoft.Research.Liquid Namespace
HamiltonianGatesRPauli Method

Performs an arbitrary rotation based on an existing gate. The base gate may be any unitary gate with a Hermitian, idempotent matrix. Of course, all Pauli gates satisfy this criteria.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
static member RPauli :  
    theta : float *  
    f : FSharpFunc<FSharpList<Qubit>, Unit> *  
    qs : FSharpList<Qubit> -> unit
```

Parameters

**theta**  
Type: System.Double  
The rotation angle. 4*pi is a full rotation.

**f**  
Type: Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>  
The gate to base the rotation on.

**qs**  
Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
The first qubit in the list is rotated.

See Also

Reference
| HamiltonianGates Class | Microsoft.Research.Liquid Namespace |

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
HamiltonianGatesRx Method

Performs a Pauli X rotation. This is functionally equivalent to \( \text{R\text{pauli}} \ \theta \ X \ \text{qs} \), but has some additional drawing options.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member Rx :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

### Parameters

**\( \theta \)**
- **Type:** SystemDouble
- The rotation angle. \( 4\pi \) is a full rotation.

**\( \text{factor} \)**
- **Type:** SystemDouble
- A multiplier, used to compute the gate label.

**\( \text{subs} \)**
- **Type:** SystemString
- A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

**\( \text{qs} \)**
- **Type:** Microsoft.FSharp.Collections.FSharpList\( <\text{Qubit}> \)
- The first qubit in the list is rotated.
See Also

Reference
HamiltonianGates Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
**HamiltonianGatesRy Method**

Performs a Pauli Y rotation. This is functionally equivalent to `R pauli theta Y qs`, but has some additional drawing options.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
Version: 1.0.5981.24943 (1.0.)*

### Syntax

F#  
```fsharp
static member Ry :  
  theta : float *  
  factor : float *  
  subs : string *  
  qs : FSharpList<Qubit> -> unit
```

### Parameters

**theta**  
Type: SystemDouble  
The rotation angle. 4*pi is a full rotation.

**factor**  
Type: SystemDouble  
A multiplier, used to compute the gate label.

**subs**  
Type: SystemString  
A subscript to attach to the gate label. Use an empty string, """, if no subscript is desired.

**qs**  
Type: Microsoft.FSharp.CollectionsFSharpListQubit  
The first qubit in the list is rotated.
See Also

Reference
HamiltonianGates Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

HamiltonianGatesRz Method

Performs a Pauli Z rotation. This is functionally equivalent to \( R_{\text{Pauli}}^{\theta}Z qs \), but has some additional drawing options.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
Version: 1.0.5981.24943 (1.0.4)

▲ Syntax

```fsharp
static member Rz :  
    theta : float *  
    factor : float *  
    subs : string *  
    qs : FSharpList<Qubit> -> unit
```

**Parameters**

\( \theta \)

Type: SystemDouble  
The rotation angle. 4*pi is a full rotation.

\( \text{factor} \)

Type: SystemDouble  
A multiplier, used to compute the gate label.

\( \text{subs} \)

Type: SystemString  
A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

\( q\)

Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
The first qubit in the list is rotated.
See Also

Reference
HamiltonianGates Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
HamiltonianGatesTtheta Method

Performs a phase gate rotation. This is functionally equivalent to \texttt{R\_pauli (2.0*theta) \_T \_qs}, but has some additional drawing options.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
static member Ttheta :  
    theta : float *  
    factor : float *  
    subs : string *  
    qs : FSharpList\<\Qubit\>  ->  unit
```

#### Parameters

**theta**
- Type: System\Double
- The rotation angle. 2*pi is a full rotation.

**factor**
- Type: System\Double
- A multiplier, used to compute the gate label.

**subs**
- Type: System\String
- A subscript to attach to the gate label. Use an empty string, "\"", if no subscript is desired.

**qs**
- Type: Microsoft.FSharp.Collections\FSharpList\\<\Qubit\>
- The first qubit in the list is rotated.
See Also

Reference
HamiltonianGates Class
Microsoft.Research.Liquid Namespace
HamiltonianGatesUnEntangle Method

Unentangles a list of qubits. This is useful for building Jordan-Wigner strings.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

## Syntax

```
static member UnEntangle :
    ladder : FSharpList<Tuple<int, int>> *
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**ladder**
- **Type:** Microsoft.FSharp.Collections.FSharpList<Tuple<int, int>>
- **Description:** A list of pairs of qubit indices to entangle. Each entry should have a tuple of two indices that refer to qubits in the *qs* list. The entangle gate will wrap a sequence of CNOT gates, one for each tuple, each with the control being the first qubit in the tuple and the target being the second in the tuple. The CNOT gates are built in the reverse order of the ladder list.

**qs**
- **Type:** Microsoft.FSharp.Collections.FSharpList<Qubit>
- **Description:** The qubits to build the ladder from.

## See Also
Reference

HamiltonianGates Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
HamiltonianGatesYbasis Method

Performs a basis change from Z to Y.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member Ybasis : 
    qs : FSharpList<Qubit> -> unit
```

### Parameters

- **qs**
  
  Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
  
  The first qubit in the list has its basis changed.

### See Also

- **Reference**
  
  HamiltonianGates Class
  
  Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
HamiltonianGatesYbasisAdj Method

Performs a basis change from Y to Z. This is the adjoint of Ybasis.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)  

**Syntax**

```fsharp
static member YbasisAdj :  
    qubits : FSharpList<Qubit> -> unit
```

**Parameters**

- `qubits`  
  Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
  The first qubit in the list has its basis changed.

**See Also**

- Reference  
  HamiltonianGates Class  
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
HamiltonianGatesZR Method

Performs a Pauli Z rotation. This is equivalent to \( R_{\text{pauli}} (2.0*\theta) Z \) qs.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member ZR :
  theta : float *
  qs : FSharpList<Qubit> -> unit
```

### Parameters

**theta**
- Type: System.Double
- The rotation angle. 2*pi is a full rotation.

**qs**
- Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
- The first qubit in the list is rotated.

### See Also

**Reference**
- HamiltonianGates Class
- Microsoft.Research.Liquid Namespace
HamiltonianGatesZZ Method

Performs a ZZ gate: Pauli Zs on consecutive wires. This is used for coupling strength.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member ZZ :
    qs : FSharpList<Qubit> -> unit
```

### Parameters

- **qs**
  - Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
  - The first two qubits will have Z performed on them.

### See Also

- **Reference**
  - HamiltonianGates Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
HamiltonianGates ZZR Method

Performs a Pauli ZZ rotation; that is, a simultaneous Z rotation of two qubits. This is equivalent to $R_{\text{pauli}}(2.0\cdot\theta)\; ZZ\; \text{qs}$.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```
static member ZZR : 
    theta : float * 
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

- **theta**
  - Type: System.Double
  - The rotation angle. $2\pi$ is a full rotation.

- **qs**
  - Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
  - The first two qubits in the list are rotated.

### See Also

- **Reference**
  - HamiltonianGates Class  
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Ket Class

Represents a state vector.

Inheritance Hierarchy

```
SystemObject  Microsoft.Research.LiquidKet
```

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
[<SerializableAttribute>]
type Ket = class end
```

The Ket type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ket</td>
<td>Creates an empty Ket vector.</td>
</tr>
<tr>
<td>Ket(FSharpOptionInt32, FSharpOptionBit)</td>
<td>Creates a populated Ket vector.</td>
</tr>
</tbody>
</table>

Properties
<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Count of Qubits in state</td>
</tr>
<tr>
<td>Item</td>
<td>Get a specific qubit in our state (by qubit ID)</td>
</tr>
<tr>
<td>MaxEntangled</td>
<td>Max entangled that we've seen (reset during Reset())</td>
</tr>
<tr>
<td>Qubits</td>
<td>Get all qubits in ID order</td>
</tr>
<tr>
<td>RandSeed</td>
<td>Force the pseudo-random number gen to a known initial state</td>
</tr>
<tr>
<td>Rnd</td>
<td>Ask for a random number generator</td>
</tr>
<tr>
<td>Symbol</td>
<td>Symbol table used to store the results of joint parity measurements.</td>
</tr>
<tr>
<td>TraceRun</td>
<td>Trace a circuit run (0=none 1=to log 2=to log and console, 3=Ket dumps)</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(Bit)</td>
<td>Adds a new qubit to the state with a classical Bit value. The new qubit is unentangled and has the provided state. Note that the new qubit is not considered measured.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Add(CVec)</td>
<td>Adds a new qubit to the state with a complex state vector. The new qubit is unentangled and has the provided state.</td>
</tr>
<tr>
<td>Add(Int32, Bit)</td>
<td>Adds multiple qubits to the state, all with the same classical Bit value. The new qubits are unentangled and have the provided state. Note that the new qubits are not considered measured.</td>
</tr>
<tr>
<td>Copy</td>
<td>Makes a new Ket that is a deep copy of this Ket.</td>
</tr>
<tr>
<td>Decohere</td>
<td>Applies a function to each Qubit in the state vector, returning the list of qubit IDs for which the function returned true. This method is usually used to apply noise to the state.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dump this state vector.</td>
</tr>
<tr>
<td>DumpKP</td>
<td>Dump the portion of this state vector that includes a specific qubit and all of the qubits it is entangled with. If this Ket is a product state, then only the factor containing the specified qubit is dumped.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Entropy</td>
<td>Get the entanglement entropy</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.     (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.                                        (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.                                      (Inherited from Object.)</td>
</tr>
<tr>
<td>Join</td>
<td>Joins another Ket to the end of this one (added Ket is NOT usable after this)</td>
</tr>
<tr>
<td>NormDiff</td>
<td>Get L2 Norm of the difference of two ket vectors</td>
</tr>
<tr>
<td>Prob1</td>
<td>Get the probability of measuring 1 for a single qubit</td>
</tr>
<tr>
<td>ProbOdd</td>
<td>Get the probability of measuring odd parity for a set of qubits</td>
</tr>
<tr>
<td>Probs</td>
<td>Get the state probabilities for a list of qubits (i.e. 10 qubits)</td>
</tr>
<tr>
<td>Purity</td>
<td>Finds the purity of each qubit in this Ket. NOT</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Read(BinaryReader)</strong></td>
<td>Reads a new state vector from a binary stream. Note that this routine is only intended to read vectors written with the <strong>Write</strong> method.</td>
</tr>
<tr>
<td><strong>Read(String)</strong></td>
<td>Reads a new state vector from a file. Note that this routine is only intended to read vectors written with the <strong>Write</strong> method.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets this Ket to an initial state with a specified number of qubits.</td>
</tr>
<tr>
<td><strong>Single</strong></td>
<td>Treats this Ket as if all of its qubits are entangled.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Gets a string representation of this Ket. This string may be extremely long; in general, it will have $2^N$ lines if there are $N$ qubits in the Ket. Generally it is better to use the <strong>Dump</strong> method instead. (Overrides <strong>ObjectToS</strong></td>
</tr>
<tr>
<td><strong>Write(BinaryWriter)</strong></td>
<td>Writes this state vector to a binary stream.</td>
</tr>
<tr>
<td><strong>Write(String)</strong></td>
<td>Writes this state vector to a file.</td>
</tr>
</tbody>
</table>
## Ket Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ket</td>
<td>Creates an empty Ket vector.</td>
</tr>
<tr>
<td>Ket(FSharpOptionInt32, FSharpOptionBit)</td>
<td>Creates a populated Ket vector.</td>
</tr>
</tbody>
</table>

### See Also

Reference

Ket Class
Microsoft.Research.Liquid Namespace
Ket Constructor

Creates an empty Ket vector.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
new : unit -> Ket
```

### See Also

- **Reference**
  - Ket Class
  - Ket Overload
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Ket Constructor (FSharpOptionInt32, FSharpOptionBit)

Creates a populated Ket vector.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```fsharp
new : count : FSharpOption<int> * init : FSharpOption<Bit> -> Ket
```

Parameters

`count`
Type: Microsoft.FSharp.Core.FSharpOption<int32>
Number of qubits to allocate.

`init`
Type: Microsoft.FSharp.Core.FSharpOption<Bit>
An optional initial value for the qubits, either Zero or One. The default is to initialize all qubits to Zero.

See Also

Reference
Ket Class
Ket Overload
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Ket Properties

The **Ket** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>Count of Qubits in state</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Get a specific qubit in our state (by qubit ID)</td>
</tr>
<tr>
<td><strong>MaxEntangled</strong></td>
<td>Max entangled that we've seen (reset during Reset())</td>
</tr>
<tr>
<td><strong>Qubits</strong></td>
<td>Get all qubits in ID order</td>
</tr>
<tr>
<td><strong>RandSeed</strong></td>
<td>Force the pseudo-random number gen to a known initial state</td>
</tr>
<tr>
<td><strong>Rnd</strong></td>
<td>Ask for a random number generator</td>
</tr>
<tr>
<td><strong>Symbol</strong></td>
<td>Symbol table used to store the results of joint parity measurements.</td>
</tr>
<tr>
<td><strong>TraceRun</strong></td>
<td>Trace a circuit run (0=none 1=to log 2=to log and console, 3=Ket dumps)</td>
</tr>
</tbody>
</table>
KetCount Property

Count of Qubits in state

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
**Version:** 1.0.5981.24943 (1.0.*)

**Syntax**

```fsharp
member Count : int with get
```

**Property Value**

Type: `Int32`

**See Also**

**Reference**
- Ket Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetItem Property

Get a specific qubit in our state (by qubit ID)

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
member Item : Qubit with get
```

**Parameters**

`i`

Type: System.Int32

**Property Value**

Type: Qubit

### See Also

**Reference**

- Ket Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetMaxEntangled Property

Max entangled that we've seen (reset during Reset())

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```fsharp
member MaxEntangled : int with get
```

Property Value

Type: Int32

See Also

Reference

Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

KetQubits Property

Get all qubits in ID order

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
member Qubits : FSharpList<Qubit> with get
```

Property Value
Type: FSharpListQubit

See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetRandSeed Property

Force the pseudo-random number gen to a known initial state

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0."

Syntax

```fsharp
member RandSeed : int with get, set
```

Property Value
Type: Int32

See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetRnd Property

Ask for a random number generator

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.6)

### Syntax

```fsharp
member Rnd : Random with get
```

**Property Value**  
Type: Random

### See Also

**Reference**  
Ket Class  
Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetSymbol Property

Symbol table used to store the results of joint parity measurements.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
member Symbol : int with get, set
```

**Parameters**

- **name**
  Type: System.String

**Property Value**

Type: Int32

### See Also

**Reference**
- Ket Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetTraceRun Property

Trace a circuit run (0=none 1=to log 2=to log and console, 3=Ket dumps)

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

**Syntax**

```fsharp
copy

member TraceRun : int with get, set
```

**Property Value**

Type: **Int32**

**See Also**

- **Reference**
  - Ket Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
# Ket Methods

The **Ket** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add(Bit)</strong></td>
<td>Adds a new qubit to the state with a classical Bit value. The new qubit is unentangled and has the provided state. Note that the new qubit is not considered measured.</td>
</tr>
<tr>
<td><strong>Add(CVec)</strong></td>
<td>Adds a new qubit to the state with a complex state vector. The new qubit is unentangled and has the provided state.</td>
</tr>
<tr>
<td><strong>Add(Int32, Bit)</strong></td>
<td>Adds multiple qubits to the state, all with the same classical Bit value. The new qubits are unentangled and have the provided state. Note that the new qubits are not considered measured.</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Makes a new Ket that is a deep copy of this Ket.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Decohere</strong></td>
<td>Applies a function to each Qubit in the state vector, returning the list of qubit IDs for which the function returned true. This method is usually used to apply noise to the state.</td>
</tr>
<tr>
<td><strong>Dump</strong></td>
<td>Dump this state vector.</td>
</tr>
<tr>
<td><strong>DumpKP</strong></td>
<td>Dump the portion of this state vector that includes a specific qubit and all of the qubits it is entangled with. If this Ket is a product state, then only the factor containing the specified qubit is dumped.</td>
</tr>
<tr>
<td><strong>Entropy</strong></td>
<td>Get the entanglement entropy</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Join</strong></td>
<td>Joins another Ket to the end of this one (added Ket is NOT usable after this)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>NormDiff</strong></td>
<td>Get L2 Norm of the difference of two ket vectors</td>
</tr>
<tr>
<td><strong>Prob1</strong></td>
<td>Get the probability of measuring 1 for a single qubit</td>
</tr>
<tr>
<td><strong>ProbOdd</strong></td>
<td>Get the probability of measuring odd parity for a set of qubits</td>
</tr>
<tr>
<td><strong>Probs</strong></td>
<td>Get the state probabilities for a list of qubits (i.e. 10 qubits)</td>
</tr>
<tr>
<td><strong>Purity</strong></td>
<td>Finds the purity of each qubit in this Ket. NOT OPTIMIZED.</td>
</tr>
<tr>
<td><strong>Read(BinaryReader)</strong></td>
<td>Reads a new state vector from a binary stream. Note that this routine is only intended to read vectors written with the Write method.</td>
</tr>
<tr>
<td><strong>Read(String)</strong></td>
<td>Reads a new state vector from a file. Note that this routine is only intended to read vectors written with the Write method.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets this Ket to an initial state with a specified number of qubits.</td>
</tr>
</tbody>
</table>
| **Single** | Treats this Ket as if all of its
qubits are entangled.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ToString</strong></td>
<td>Gets a string representation of this Ket. This string may be extremely long; in general, it will have (2^N) lines if there are (N) qubits in the Ket. Generally it is better to use the Dump method instead. (Overrides ObjectToString.)</td>
</tr>
<tr>
<td><strong>Write(BinaryWriter)</strong></td>
<td>Writes this state vector to a binary stream.</td>
</tr>
<tr>
<td><strong>Write(String)</strong></td>
<td>Writes this state vector to a file.</td>
</tr>
</tbody>
</table>

See Also

Reference

Ket Class

Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
### KetAdd Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add(Bit)</strong></td>
<td>Adds a new qubit to the state with a classical Bit value. The new qubit is unentangled and has the provided state. Note that the new qubit is not considered measured.</td>
</tr>
<tr>
<td><strong>Add(CVec)</strong></td>
<td>Adds a new qubit to the state with a complex state vector. The new qubit is unentangled and has the provided state.</td>
</tr>
<tr>
<td><strong>Add(Int32, Bit)</strong></td>
<td>Adds multiple qubits to the state, all with the same classical Bit value. The new qubits are unentangled and have the provided state. Note that the new qubits are not considered measured.</td>
</tr>
</tbody>
</table>

---

### See Also

**Reference**
- Ket Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

KetAdd Method (Bit)

Adds a new qubit to the state with a classical Bit value. The new qubit is unentangled and has the provided state. Note that the new qubit is not considered measured.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
member Add : 
    b : Bit -> Qubit
```

### Parameters

- **b**
  - Type: Microsoft.Research.LiquidBit
  - The initial state of the added qubit.

### Return Value

- Type: Qubit
- The new Qubit

### See Also

- Reference
  - Ket Class
  - Add Overload
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetAdd Method (CVec)

Adds a new qubit to the state with a complex state vector. The new qubit is unentangled and has the provided state.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Add : cv : CVec -> Qubit
```

### Parameters

- `cv`
  - **Type:** Microsoft.Research.Liquid.CVec  
    - The state of the added qubit.

### Return Value

- **Type:** Qubit  
  - The new Qubit

### See Also

- **Reference**
  - Ket Class
  - Add Overload
  - Microsoft.Research.Liquid Namespace
KetAdd Method (Int32, Bit)

Adds multiple qubits to the state, all with the same classical Bit value. The new qubits are unentangled and have the provided state. Note that the new qubits are not considered measured.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member Add : 
    cnt : int * 
    b : Bit -> FSharpList<Qubit>
```

### Parameters

**cnt**
- Type: System.Int32
  - The number of qubits to add.

**b**
- Type: Microsoft.Research.Liquid.Bit
  - The initial state of the added qubits.

### Return Value
- Type: FSharpList<Qubit>
  - The new Qubits

### See Also

- Reference
  - Ket Class
  - Add Overload
KetCopy Method

Makes a new Ket that is a deep copy of this Ket.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

Syntax

```
F# Copy

member Copy : unit -> Ket
```

Return Value
Type: Ket
The new ket vector

See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

KetDecohere Method

Applies a function to each Qubit in the state vector, returning the list of qubit IDs for which the function returned true. This method is usually used to apply noise to the state.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
F#    Copy

member Decohere :
    decohere : FSharpFunc<Qubit, bool> -> FSharpList<Int32>
```

### Parameters

`decohere`

Type: `Microsoft.FSharp.Core.FSharpFunc<Qubit, Boolean>`
The function to apply to each qubit. This function is allowed to modify the state of the qubit.

### Return Value

Type: `FSharpList<Int32>`
The list of qubits where the function returned true

### See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetDump Method

Dump this state vector.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```fsharp
member Dump :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<int, FSharpFunc<String, Unit>>>>
    level : FSharpOption<int>  *
    doMCC : FSharpOption<bool>  *
    doSort : FSharpOption<bool>  -> unit
```

### Parameters

- **f**  
  Type: `Microsoft.FSharp.Core.FSharpOption<Microsoft.FSharp.Core.FSharpFunc<int, Microsoft.FSharp.Core.FSharpFunc<int, Microsoft.FSharp.Core.FSharpFunc<String, Unit>>>>`  
  The optional output function to use. The default is `showLogInd`

- **level**  
  Type: `Microsoft.FSharp.Core.FSharpOption<int>`  
  The optional indentation level. The default is 0.

- **doMCC**  
  Type: `Microsoft.FSharp.Core.FSharpOption<bool>`  
  An option that, if true, causes probabilities (complex magnitude squared) to be output for each state entry, rather than the complex amplitude. The default is to output the complex amplitude for each state.

- **doSort**  
  Type: `Microsoft.FSharp.Core.FSharpOption<bool>`
An option that, if true, causes the output to be sorted in qubit ID order. The default is to sort the output.

See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

KetDumpKP Method

Dump the portion of this state vector that includes a specific qubit and all of the qubits it is entangled with. If this Ket is a product state, then only the factor containing the specified qubit is dumped.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```fsharp
member DumpKP : q : Qubit * f : FSharpFunc<int, FSharpFunc<string, Unit>, Unit>
level : int * doMCC : FSharpOption<bool> -> unit
```

Parameters

q
Type: Microsoft.Research.LiquidQubit
The Qubit that identifies the product state factor.

f
Type: Microsoft.FSharp.Core.FSharpFunc<Int32, FSharpFunc<String, Unit>
The optional output function to use. The default is showLogInd.

level
Type: System.Int32
The optional indentation level. The default is 0.

doMCC
Type: Microsoft.FSharp.Core.FSharpOption<bool>
An option that, if true, causes probabilities (complex magnitude
squared) to be output for each state entry, rather than the complex amplitude. The default is to output the complex amplitude for each state.

See Also

Reference

Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetEntropy Method

Get the entanglement entropy

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

**Syntax**

```fsharp
member Entropy : 
    alpha : int * 
    qPos : int * 
    order : FSharpOption<FSharpList<int>>  ->
```

**Parameters**

- **alpha**  
  Type: SystemInt32  
  Denotes the generalized entropy parameter: 1 for Von Neumann or 2 for Reny

- **qPos**  
  Type: SystemInt32  
  qubit position, 0 to nQubits (based on order)

- **order**  
  Type: Microsoft.FSharp.Core.FSharpOption.FSharpList<int>  
  Int list of qubit order to wind up in []=use results of previous call (optional=[0..N-1])

**Return Value**

Type: Double  
Entropy value
See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetJoin Method

Joins another Ket to the end of this one (added Ket is NOT usable after this)

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```f#
member Join : kAdd : Ket -> Ket
```

**Parameters**

*kAdd*
- Type: Microsoft.Research.Liquid.Ket
  - ket to add at the end

**Return Value**
- Type: Ket
  - Resulting ket

### See Also

**Reference**
Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetNormDiff Method

Get L2 Norm of the difference of two ket vectors

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member NormDiff : v2 : CVec -> float
```

### Parameters

- **v2**
  - Type: Microsoft.Research.Liquid.CVec
  - Target ket vector to diff (obtained by ket.Single())

### Return Value

- **Type:** Double
- L2 Norm of the difference

### See Also

- **Reference**
  - Ket Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetProb1 Method

Get the probability of measuring 1 for a single qubit

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

**Syntax**

```
F# Copy

member Prob1 :
    q : Qubit -> float
```

**Parameters**

q
Type: Microsoft.Research.Liquid.Qubit
Qubit to "fake" measure

**Return Value**

Type: Double
probability

**See Also**

Reference

Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetProbOdd Method

Get the probability of measuring odd parity for a set of qubits

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```fsharp
member ProbOdd :
    qs : FSharpList<Qubit> *
    basis : FSharpOption<string> -> float
```

Parameters

*qs*
Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
Qubits to fake measure

*basis*
Type: Microsoft.FSharp.Core.FSharpOption<String>
String of basis to use for each qubit (optional="Z")

Return Value
Type: Double
Probability of odd parity

See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetProbs Method

Get the state probabilities for a list of qubits (i.e. 10 qubits)

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```
F# Copy

member Probs :  
    qs : FSharpList<Qubit> -> float[]
```

### Parameters

* qs
  * Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
  * Qubits to fake measure

### Return Value

* Type: `Double`
  * Array of state values (low bit=first qubit requested)

### See Also

**Reference**

Ket Class  
Microsoft.Research.Liquid Namespace
KetPurity Method

Finds the purity of each qubit in this Ket. NOT OPTIMIZED.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```
F#  Copy

member Purity : unit -> float[]
```

Return Value
Type: Double
An array of purities

See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace
## KetRead Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="BinaryReader" alt="Read" /></td>
<td>Reads a new state vector from a binary stream. Note that this routine is only intended to read vectors written with the Write method.</td>
</tr>
<tr>
<td>![Read(String)]</td>
<td>Reads a new state vector from a file. Note that this routine is only intended to read vectors written with the Write method.</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - Ket Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetRead Method (BinaryReader)

Reads a new state vector from a binary stream. Note that this routine is only intended to read vectors written with the `Write` method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)&

### Syntax

```fsharp
static member Read :  
br : BinaryReader -> Ket
```

**Parameters**

*br*

- **Type:** System.IO.BinaryReader
  - The stream from which the vector should be read.

**Return Value**

- **Type:** Ket
  - The new Ket

### See Also

- **Reference**
  - Ket Class
  - Read Overload
  - Microsoft.Research.Liquid Namespace
  - KetRead(String)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetRead Method (String)

Reads a new state vector from a file. Note that this routine is only intended to read vectors written with the **Write** method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
static member Read :  
    file : string -> Ket
```

### Parameters

**file**
- **Type:** System.String  
  The name of the file from which the state vector should be read.

### Return Value
- **Type:** Ket  
  The new Ket

### See Also

- Reference
  Ket Class  
  Read Overload  
  Microsoft.Research.Liquid Namespace  
  KetRead(BinaryReader)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetReset Method

Resets this Ket to an initial state with a specified number of qubits.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
member Reset :
    want : FSharpOption<int> *
    init : FSharpOption<Bit> -> FSharpList<Qubit>
```

### Parameters

**want**

Type: Microsoft.FSharp.Core.FSharpOption<int>
An optional new qubit count. The default is to keep the same number of qubits.

**init**

Type: Microsoft.FSharp.Core.FSharpOption<Bit>
An optional initial value for all qubits. The default is Zero.

### Return Value

Type: FSharpList<Qubit>
The new Qubits in this Ket

---

### See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace
KetSingle Method

Treats this Ket as if all of its qubits are entangled.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

F#

```fsharp
member Single :
  order : FSharpOption<FSharpList<int>> -> CVec
```

### Parameters

`order`

Type: Microsoft.FSharp.Core.FSharpOption<FSharpList<int>>

Optionally, a list specifying a new order for the qubits. The default is for qubits to remain in their current order.

### Return Value

Type: CVec

A CVec that contains the single state vector. Note that it is unsafe to modify this vector.

### See Also

**Reference**

Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetToString Method

Gets a string representation of this Ket. This string may be extremely long; in general, it will have $2^N$ lines if there are $N$ qubits in the Ket. Generally it is better to use the Dump method instead.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```
abstract ToString : unit -> string
override ToString : unit -> string
```

Return Value
Type: String
The string representation

See Also

Reference
Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
## KetWrite Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write(BinaryWriter)</td>
<td>Writes this state vector to a binary stream.</td>
</tr>
<tr>
<td>Write(String)</td>
<td>Writes this state vector to a file.</td>
</tr>
</tbody>
</table>

See Also

Reference

Ket Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KetWrite Method (BinaryWriter)

Writes this state vector to a binary stream.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.2)

### Syntax

```fsharp
member Write :  
  bw : BinaryWriter -> unit
```

### Parameters

**bw**

Type: System.IO.BinaryWriter  
The stream that this Ket should be written to.

### See Also

Reference

Ket Class  
Write Overload  
Microsoft.Research.Liquid Namespace  
KetWrite(String)  
KetRead(BinaryReader)
KetWrite Method (String)

Writes this state vector to a file.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4*)

### Syntax

```
F#

member Write : 
    file : string -> unit
```

**Parameters**

- **file**
  - Type: System.String
  - The name of the file that this Ket should be written to.

### See Also

Reference
- Ket Class
- Write Overload
- Microsoft.Research.Liquid Namespace
- Ket.Write(IO, BinaryWriter)
- KetRead(String)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KrausOp Class

Entries for Kraus operators in Channel Gate type

Inheritance Hierarchy

- System
  - Object

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0*)

Syntax

```fsharp
[<SealedAttribute>] [<SerializableAttribute>]
type KrausOp =
    class
        interface IEquatable<KrausOp>
        interface IStructuralEquatable
    end
```

The KrausOp type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mat</td>
<td>Kraus operator matrix</td>
</tr>
<tr>
<td>tag</td>
<td>Name to use for POVM application (usually a user provided index)</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

KrausOp Properties

The KrausOp type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mat</td>
<td>Kraus operator matrix</td>
</tr>
<tr>
<td>tag</td>
<td>Name to use for POVM application (usually a user provided index)</td>
</tr>
</tbody>
</table>

See Also

Reference

KrausOp Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KrausOpmat Property

Kraus operator matrix

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member mat : CSMat with get
```

### Property Value

Type: CSMat

### See Also

**Reference**

KrausOp Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
KrausOptag Property

Name to use for POVM application (usually a user provided index)

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```fsharp
member tag : int with get
```

Property Value
Type: Int32

See Also

Reference
KrausOp Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

KrausOp Methods

The KrausOp type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>

See Also

Reference
KrausOp Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Noise Class

A complete noise model for a specific circuit.

Inheritance Hierarchy

SystemObject  Microsoft.Research.LiquidNoise

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

Syntax

```fsharp
[<SerializableAttribute>]
type Noise = class end
```

The Noise type exposes the following members.

 Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ] Noise</td>
<td>Creates a new Noise instance.</td>
</tr>
</tbody>
</table>

 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ] DampProb</td>
<td>The probability of amplitude-damping noise on a qubit. This allows different qubits to have</td>
</tr>
</tbody>
</table>
different amplitude damping probabilities.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DampProbs</strong></td>
<td>(Set only) The amplitude damping probability for all qubits. Use <strong>DampProb</strong> to get or set the amplitude damping probability for a single qubit.</td>
</tr>
<tr>
<td><strong>ECgates</strong></td>
<td>The list of wrap gates that are part of error-correcting circuits, by name. A name may end with &quot;*&quot; to indicate a wildcard.</td>
</tr>
<tr>
<td><strong>Idle</strong></td>
<td>(Set only) The idle gate. By default, I (the identity) is the idle gate.</td>
</tr>
<tr>
<td><strong>LogGates</strong></td>
<td>Whether or not to log gate execution to noise statistics during Run(). Gate execution log entries will be marked with detail=&quot;!&quot;.</td>
</tr>
<tr>
<td><strong>Models</strong></td>
<td>The noise models from last run. This provides access to the detailed statistics.</td>
</tr>
<tr>
<td><strong>NoNoise</strong></td>
<td>The list of noiseless gates, by name. A name may end with &quot;*&quot; to indicate a wildcard.</td>
</tr>
<tr>
<td><strong>Stats</strong></td>
<td>The error statistics from the last run. Statistics are kept in reverse time order.</td>
</tr>
<tr>
<td><strong>TraceNoise</strong></td>
<td>Whether or not to trace noise to the log as it's inserted.</td>
</tr>
<tr>
<td><strong>TraceWrap</strong></td>
<td>Whether or not to trace wrap gates</td>
</tr>
</tbody>
</table>
to the log as they're executed.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultNoise</td>
<td>Creates a default noise model. The new noise model has depolarizing noise on all qubits, and all gates have unit expected duration.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dumps noise statistics from the last run.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Run</td>
<td>Runs a circuit with this noise model.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>
See Also

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Noise Constructor

Creates a new Noise instance.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
new :  
circ : Circuit  
ket : Ket  
models : FSharpList<NoiseModel> -> Noise
```

### Parameters

**circ**

Type: Microsoft.Research.LiquidCircuit  
The circuit to apply noise to.

**ket**

Type: Microsoft.Research.LiquidKet  
The Ket to use for noise model.

**models**

Type: Microsoft.FSharp.CollectionsFSharpList<NoiseModel>  
A list of noise models to use, in decreasing precedence order.

### See Also

Reference

Noise Class  
Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
# Noise Properties

The **Noise** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![DampProb]</td>
<td>The probability of amplitude-damping noise on a qubit. This allows different qubits to have different amplitude damping probabilities.</td>
</tr>
<tr>
<td>![DampProbs]</td>
<td>(Set only) The amplitude damping probability for all qubits. Use <strong>DampProb</strong> to get or set the amplitude damping probability for a single qubit.</td>
</tr>
<tr>
<td>![ECgates]</td>
<td>The list of wrap gates that are part of error-correcting circuits, by name. A name may end with &quot;*&quot; to indicate a wildcard.</td>
</tr>
<tr>
<td>![Idle]</td>
<td>(Set only) The idle gate. By default, <strong>I</strong> (the identity) is the idle gate.</td>
</tr>
<tr>
<td>![LogGates]</td>
<td>Whether or not to log gate execution to noise statistics during Run(). Gate execution log entries will be marked with detail=&quot;!&quot;.</td>
</tr>
<tr>
<td>![Models]</td>
<td>The noise models from last run. This provides access to the detailed</td>
</tr>
</tbody>
</table>
NoNoise

The list of noiseless gates, by name. A name may end with "*" to indicate a wildcard.

Stats

The error statistics from the last run. Statistics are kept in reverse time order.

TraceNoise

Whether or not to trace noise to the log as it's inserted.

TraceWrap

Whether or not to trace wrap gates to the log as they're executed.

Top

See Also

Reference
Noise Class
Microsoft.Research.Liquid Namespace
NoiseDampProb Property

The probability of amplitude-damping noise on a qubit. This allows different qubits to have different amplitude damping probabilities.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

## Syntax

```fsharp
member DampProb : float with get, set
```

### Parameters

**id**

Type: **System.Int32**

The ID of the qubit

### Property Value

Type: **Double**

## See Also

**Reference**

- Noise Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseDampProbs Property

(Set only) The amplitude damping probability for all qubits. Use **DampProb** to get or set the amplitude damping probability for a single qubit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```
F# Copy

member DampProbs : float with set
```

### Property Value

Type: **Double**

### See Also

**Reference**  
Noise Class  
Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseECgates Property

The list of wrap gates that are part of error-correcting circuits, by name. A name may end with "*" to indicate a wildcard.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```fsharp
member ECgates : FSharpList<string> with get, set
```

### Property Value

Type: **FSharpList<String>**

### See Also

Reference  
**Noise Class**  
**Microsoft.Research.Liquid Namespace**

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoisIdle Property

(Set only) The idle gate. By default, I (the identity) is the idle gate.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

⚠️ Syntax

```fsharp
member Idle : Gate with set
```

Property Value

Type: Gate

⚠️ See Also

Reference
Noise Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseLogGates Property

Whether or not to log gate execution to noise statistics during Run(). Gate execution log entries will be marked with detail="!".

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

**Syntax**

```fsharp
member LogGates : bool with get, set
```

**Property Value**

Type: Boolean

**See Also**

Reference
- Noise Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseModels Property

The noise models from last run. This provides access to the detailed statistics.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

⚠️ Syntax

```fsharp
member Models : FSharpList<NoiseModel> with get
```

⚠️ Property Value

Type: `FSharpList<NoiseModel>`

⚠️ See Also

- Reference
  - Noise Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseNoNoise Property

The list of noiseless gates, by name. A name may end with "*" to indicate a wildcard.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
member NoNoise : FSharpList<string> with get, set
```

### Property Value

Type: **FSharpList<string>**

### See Also

- Reference
  - Noise Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseStats Property

The error statistics from the last run. Statistics are kept in reverse time order.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

**Syntax**

```
member Stats : FSharpList<NoiseStat> with get
```

**Property Value**

Type: `FSharpList<NoiseStat>`

**See Also**

Reference  
Noise Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Noise Trace Noise Property

Whether or not to trace noise to the log as it's inserted.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```fsharp
member TraceNoise : bool with get, set
```

### Property Value

Type: **Boolean**

### See Also

- **Reference**
- Noise Class
- Microsoft.Research.Liquid Namespace
NoiseTraceWrap Property

Whether or not to trace wrap gates to the log as they're executed.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

Syntax

```fsharp
member TraceWrap : bool with get, set
```

Property Value
Type: Boolean

See Also

Reference
Noise Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Noise Methods

The **Noise** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultNoise</td>
<td>Creates a default noise model. The new noise model has depolarizing noise on all qubits, and all gates have unit expected duration.</td>
</tr>
<tr>
<td>Dump</td>
<td>Dumps noise statistics from the last run.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Run</td>
<td>Runs a circuit with this noise model.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>
See Also

Reference
Noise Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseDefaultNoise Method

Creates a default noise model. The new noise model has depolarizing noise on all qubits, and all gates have unit expected duration.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
static member DefaultNoise :  
    prob : float -> NoiseModel
```

Parameters

- **prob**
  
  Type: System.Double
  
  The probability of depolarizing noise occurring per unit of time.

Return Value

Type: NoiseModel

The new noise model.

See Also

- Reference
- Noise Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseDump Method

Dumps noise statistics from the last run.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
member Dump :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<int>>  
    level : FSharpOption<int> *  
    doStats : FSharpOption<bool> -&gt; unit
```

### Parameters

- **f**  
  Type: Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int>>, FSharpFunc<int>>  
  The optional output function to use. The default is showLogInd.

- **level**  
  Type: Microsoft.FSharp.Core.FSharpOption<int>  
  The optional indentation level. The default is 0.

- **doStats**  
  Type: Microsoft.FSharp.Core.FSharpOption<bool>  
  An option to dump full statistics. The default is false, in which case summary statistics per gate pattern are displayed.

### See Also

- Reference
  Noise Class
NoiseRun Method

Runs a circuit with this noise model.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)  

### Syntax

```fsharp
member Run : ket2 : FSharpOption<Ket> -> unit
```

### Parameters

**ket2**

Type: `Microsoft.FSharp.Core.FSharpOption<Ket>`  
An optional state vector to use to re-initialize execution. If provided, this Ket is used for the run and all times and statistics are reset. If not provided, then execution continues from the state after the last Run.

### See Also

Reference  
Noise Class  
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseEvents Class

Noise statistics that are tracked for normal and error-correcting gates.

Inheritance Hierarchy

System Object  Microsoft.Research.Liquid

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0."

Syntax

```
[<SealedAttribute>]  
[<SerializableAttribute>]  

type NoiseEvents =  
    class  
        interface IEquatable<NoiseEvents>  
        interface IStructuralEquatable  
        interface IComparable<NoiseEvents>  
        interface IComparable  
        interface IStructuralComparable  
    end
```

The NoiseEvents type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>applied</td>
<td>The number of times noise has been applied</td>
</tr>
</tbody>
</table>
**count**  The number of times the model has been executed

**events**  The number of noise events

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accum</strong></td>
<td>Increments this instance with counts from another instance.</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>Creates a new instance with all counts initialized to 0.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets all counters back to 0.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object.  (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>
See Also

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseEvents Properties

The NoiseEvents type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>applied</td>
<td>The number of times noise has been applied</td>
</tr>
<tr>
<td>count</td>
<td>The number of times the model has been executed</td>
</tr>
<tr>
<td>events</td>
<td>The number of noise events</td>
</tr>
</tbody>
</table>

See Also

Reference

NoiseEvents Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseEventsapplied Property

The number of times noise has been applied

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5981.24943)

Syntax

F#

member applied : int with get, set

Property Value
Type: Int32

See Also

Reference
NoiseEvents Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseEventscount Property

The number of times the model has been executed

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

F#

```fsharp
member count : int with get, set
```

Property Value

Type: Int32

See Also

Reference

NoiseEvents Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseEventsevents Property

The number of noise events

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

### Syntax

```
F#

member events : int with get, set
```

### Property Value

Type: **Int32**

### See Also

**Reference**
- NoiseEvents Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
# NoiseEvents Methods

The `NoiseEvents` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Accum</code></td>
<td>Increments this instance with counts from another instance.</td>
</tr>
<tr>
<td><code>Default</code></td>
<td>Creates a new instance with all counts initialized to 0.</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>Reset</code></td>
<td>Resets all counters back to 0.</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
See Also

Reference
NoiseEvents Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseEventsAccum Method

Increments this instance with counts from another instance.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

F#

```fsharp
member Accum : y : NoiseEvents -> unit
```

Parameters

y

Type: Microsoft.Research.LiquidNoiseEvents
The instance containing counts to add to this instance's counts.

See Also

Reference

NoiseEvents Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseEventsDefault Method

Creates a new instance with all counts initialized to 0.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member Default : unit -> NoiseEvents
```

**Return Value**

Type: **NoiseEvents**

The new instance.

### See Also

**Reference**

- NoiseEvents Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseEventsReset Method

Resets all counters back to 0.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
member Reset : unit -> unit
```

### See Also

- Reference
  - NoiseEvents Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
## NoiseModel Class

A noise model for a particular type of gate (or set of gates).

### Inheritance Hierarchy

- System
- Object
- Microsoft.Research.Liquid
- NoiseModel

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
**Version:** 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
[<SealedAttribute>]  
[<SerializableAttribute>]

type NoiseModel =   
class end
```

The `NoiseModel` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecEvents</td>
<td>Accumulated noise statistics for error-correcting gates.</td>
</tr>
<tr>
<td>func</td>
<td>The noise function to execute.</td>
</tr>
<tr>
<td>gate</td>
<td>The name of the gate that this noise model applies to. The name may end with an asterisk, &quot;*&quot;, to indicate a wildcard match.</td>
</tr>
</tbody>
</table>
- **gateEvents**: Accumulated noise statistics for normal gates.
- **maxQs**: The maximum number of qubits to apply noise to.
- **time**: The expected duration of the gate. By convention, an idle gate takes 1.0 units.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
<td>Creates a default noise model that will apply to all gates.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
See Also

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseModel Properties

The NoiseModel type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecEvents</td>
<td>Accumulated noise statistics for error-correcting gates.</td>
</tr>
<tr>
<td>func</td>
<td>The noise function to execute.</td>
</tr>
<tr>
<td>gate</td>
<td>The name of the gate that this noise model applies to. The name may end with an asterisk, '*', to indicate a wildcard match.</td>
</tr>
<tr>
<td>gateEvents</td>
<td>Accumulated noise statistics for normal gates.</td>
</tr>
<tr>
<td>maxQs</td>
<td>The maximum number of qubits to apply noise to.</td>
</tr>
<tr>
<td>time</td>
<td>The expected duration of the gate. By convention, an idle gate takes 1.0 units.</td>
</tr>
</tbody>
</table>
NoiseModel.ecEvents Property

Accumulated noise statistics for error-correcting gates.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

### Syntax

```fsharp
member ecEvents : NoiseEvents with get
```

### Property Value

**Type:** NoiseEvents

### See Also

**Reference**
- NoiseModel Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseModelfunc Property

The noise function to execute.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
member func : FSharpFunc<float, FSharpFunc<float,
```

Property Value

Type: FSharpFunc<Double, FSharpFunc<Double, FSharpFunc<FSharpList<Qubit>, FSharpList<Tuple<FSharpList<Qubit>, FSharpOption<String>}}

See Also

Reference
NoiseModel Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseModel\texttt{gate} Property

The name of the gate that this noise model applies to. The name may end with an asterisk, '*', to indicate a wildcard match.

\textbf{Namespace:} Microsoft.Research.Liquid
\textbf{Assembly:} Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

\section*{Syntax}

\begin{verbatim}
member gate : string with get
\end{verbatim}

\section*{Property Value}

Type: \texttt{String}

\section*{See Also}

\textbf{Reference}
\texttt{NoiseModel Class}
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseModelGateEvents Property

Accumulated noise statistics for normal gates.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0*)

Syntax

F#

```fsharp
member gateEvents : NoiseEvents with get
```

Property Value
Type: NoiseEvents

See Also

Reference
NoiseModel Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseModelmaxQs Property

The maximum number of qubits to apply noise to.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.***)

### Syntax

```
member maxQs : int with get
```

### Property Value

Type: **Int32**

### See Also

- **Reference**
  - NoiseModel Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseModel time Property

The expected duration of the gate. By convention, an idle gate takes 1.0 units.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member time : float with get
```

### Property Value

**Type:** Double

### See Also

- Reference
  - NoiseModel Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseModel Methods

The `NoiseModel` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Creates a default noise model that will apply to all gates.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

`NoiseModel Class`
NoiseModelDefault Method

Creates a default noise model that will apply to all gates.

**Namespace**: Microsoft.Research.Liquid

**Assembly**: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.\*)

### Syntax

```fsharp
static member Default :
    modelFunc : FSharpFunc<float, FSharpFunc<float>
```

### Parameters

- **modelFunc**
  
  Type: `Microsoft.FSharp.Core.FSharpFunc<Double, FSharpFunc<Double, FSharpFunc<FSharpList<Qubit>, FSharpList<Tuple<FSharpList<Qubit>, FSharpOption<String>>>}}`  
  The noise function to apply.

### Return Value

Type: **NoiseModel**

A noise model that will apply the given function to all gates, to a single qubit, and initialized with zero statistics.

### See Also

- **Reference**
  - NoiseModel Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseStat Class

Statistics tracked for each time that noise is applied.

Inheritance Hierarchy


Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5)

Syntax

```fsharp
[<SealedAttribute>]
[<SerializableAttribute>]
type NoiseStat = class end
```

The NoiseStat type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>detail</td>
<td>User-defined noise details from the noise function output.</td>
</tr>
<tr>
<td>dur</td>
<td>The duration of the noise application.</td>
</tr>
<tr>
<td>ecGate</td>
<td>Whether or not the gate was flagged as an error correcting gate.</td>
</tr>
<tr>
<td>model</td>
<td>The noise model that was applied.</td>
</tr>
</tbody>
</table>
qs  The qubits that noise was applied to.

time  When noise was applied during the simulated execution.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.type">Type</a> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string representation of this noise statistic.</td>
</tr>
<tr>
<td></td>
<td>(Overrides <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">ObjectToS</a>.)</td>
</tr>
</tbody>
</table>

## See Also

[Reference](https://docs.microsoft.com/en-us/dotnet/api)

NoiseStat Properties

The NoiseStat type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>detail</td>
<td>User-defined noise details from the noise function output.</td>
</tr>
<tr>
<td>dur</td>
<td>The duration of the noise application.</td>
</tr>
<tr>
<td>ecGate</td>
<td>Whether or not the gate was flagged as an error correcting gate.</td>
</tr>
<tr>
<td>model</td>
<td>The noise model that was applied.</td>
</tr>
<tr>
<td>qs</td>
<td>The qubits that noise was applied to.</td>
</tr>
<tr>
<td>time</td>
<td>When noise was applied during the simulated execution.</td>
</tr>
</tbody>
</table>

See Also

Reference
- NoiseStat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseStat det<e>

tail Property

User-defined noise details from the noise function output.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5)

**Syntax**

```fsharp
member detail : string with get
```

**Property Value**

Type: **String**

**See Also**

Reference

- NoiseStat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseStatdur Property

The duration of the noise application.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

**Syntax**

```f#
member dur : float with get
```

**Property Value**

Type: **Double**

**See Also**

Reference

- NoiseStat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseStatecGate Property

Whether or not this was flagged as an error correcting gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```
F# Copy

member ecGate : bool with get
```

### Property Value

Type: **Boolean**

### See Also

**Reference**

- NoiseStat Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoiseStatmodel Property

The noise model that was applied.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.9)

Syntax

F#

```fsharp
member model : NoiseModel with get
```

Property Value

Type: NoiseModel

See Also

Reference

NoiseStat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseStat property

The qubits that noise was applied to.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member qs : FSharpList<Qubit>  with get
```

### Property Value

Type: `FSharpList<Qubit>`

### See Also

Reference

- NoiseStat Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseStat time Property

When noise was applied during the simulated execution.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member time : float with get
```

**Property Value**

**Type:** Double

### See Also

**Reference**

NoiseStat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseStat Methods

The NoiseStat type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string representation of this noise statistic.</td>
</tr>
<tr>
<td></td>
<td>(Overrides Object.ToString.)</td>
</tr>
</tbody>
</table>

See Also

Reference
NoiseStat Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
NoiseStatToString Method

Returns a string representation of this noise statistic.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
abstract ToString : unit -> string
override ToString : unit -> string
```

### Return Value

**Type:** String  
The string

### See Also

**Reference**
- NoiseStat Class
- Microsoft.Research.Liquid Namespace
NoisyMats Class

Utility class for computing a Pauli rotation matrix. This is used to run quantum chemistry circuits with noise injected.

Inheritance Hierarchy

```
```

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0.*)

Syntax

```
[<SerializableAttribute>]
type NoisyMats = class end
```

The `NoisyMats` type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.    (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GenRot</td>
<td>Generates a Pauli-based rotation matrix for an arbitrary angle.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
</tbody>
</table>
(Inherited from \texttt{Object}.)

\begin{tabular}{ll}
\textbf{\texttt{Get Type}} & Gets the \texttt{Type} of the current instance. \hfill (Inherited from \texttt{Object}.)
\end{tabular}

\begin{tabular}{ll}
\textbf{\texttt{To String}} & Returns a string that represents the current object. \hfill (Inherited from \texttt{Object}.)
\end{tabular}

\textbf{Top}

\section*{See Also}

\textbf{Reference}

\texttt{Microsoft.Research.Liquid Namespace}

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

NoisyMats Methods

The NoisyMats type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GenRot</td>
<td>Generates a Pauli-based rotation matrix for an arbitrary angle.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>

See Also

Reference

NoisyMats Class
NoisyMatsGenRot Method

Generates a Pauli-based rotation matrix for an arbitrary angle.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

F#

```fsharp
static member GenRot :
    theta : float *
    mParent : CSMat -> CSMat
```

Parameters

**theta**
Type: SystemDouble
The rotation angle.

**mParent**
Type: Microsoft.Research.LiquidCSMat
The base matrix for the rotation. This must be an idempotent Hermitian matrix.

Return Value
Type: CSMat
A rotation matrix, exp(i*theta/2*mParent).

See Also

Reference
NoisyMats Class
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Operations Class

The Operations module provides definitions of basic gates. It also includes some handy operators for manipulating qubit lists, and some operations for building gates from existing gates.

Inheritance Hierarchy

- System
- Object
- Microsoft.Research.Liquid

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
[<AbstractClassAttribute>]
[<SealedAttribute>]
type Operations = class end
```

The Operations type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![S] AD</td>
<td>Amplitude Damping Channel using two Kraus operators (always put the most probable one first)</td>
</tr>
<tr>
<td>![S] Adj</td>
<td>Performs the adjoint of the parent gate. This only works if the parent gate is a unitary gate.</td>
</tr>
<tr>
<td>Gate</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>BC</td>
<td>Performs a gate under classical control. The parent gate will be executed if the control qubit has a measured value of One.</td>
</tr>
<tr>
<td>BCany</td>
<td>Performs a gate under classical control. This gate allows an arbitrary condition to be specified. The parent gate will be executed if the condition function evaluates to true.</td>
</tr>
<tr>
<td>CCgate</td>
<td>Performs a parent gate under two-qubit quantum control. The parent must be a unitary gate.</td>
</tr>
<tr>
<td>CCNOT</td>
<td>Performs a Toffoli or Controlled-Controlled-NOT gate.</td>
</tr>
<tr>
<td>Cgate</td>
<td>Performs a parent gate under quantum control. The parent must be a unitary gate.</td>
</tr>
<tr>
<td>CgateNC</td>
<td>Performs a parent gate under quantum control. The parent must be a unitary gate. The resulting gate is not cached.</td>
</tr>
<tr>
<td>CNOT</td>
<td>Performs a quantum-controlled NOT gate.</td>
</tr>
<tr>
<td>CZ</td>
<td>Performs a quantum-controlled Pauli Z gate.</td>
</tr>
<tr>
<td>DP</td>
<td>Depolarizing channel using four Kraus operators (always put the most probable one first).</td>
</tr>
<tr>
<td>H</td>
<td>Performs a Hadamard gate.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><strong>I</strong></td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><strong>JM</strong></td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><strong>JMx</strong></td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><strong>JMz</strong></td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><strong>Label</strong></td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><strong>LabelC</strong></td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><strong>LabelCD</strong></td>
</tr>
</tbody>
</table>
will have a box drawn tightly around it.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="LabelID Icon" /></td>
<td>LabelD</td>
<td>Adds a label to a circuit drawing. The label will appear below the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.</td>
</tr>
<tr>
<td><img src="image" alt="LabelL Icon" /></td>
<td>LabelL</td>
<td>Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the left side of the current column. The label will take up no logical space; it will not fill the current cell.</td>
</tr>
<tr>
<td><img src="image" alt="LabelR Icon" /></td>
<td>LabelR</td>
<td>Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the right side of the current column. The label will take up no logical space; it will not fill the current cell.</td>
</tr>
<tr>
<td><img src="image" alt="LabelRaw Icon" /></td>
<td>LabelRaw</td>
<td>Adds a raw label to a circuit drawing. The provided string should contain the Q-Circuit commands used to draw the label. Note that labels drawn using this gate will only appear when rendered to &quot;qc&quot; format.</td>
</tr>
<tr>
<td><img src="image" alt="LabelU Icon" /></td>
<td>LabelU</td>
<td>Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.</td>
</tr>
</tbody>
</table>
the current cell.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![LoadCache]</td>
<td>LoadCache</td>
<td>Preload the cache with basic gates: Paulis, measurement, Hadamard, phase, CNOT, Toffoli, swap, and T.</td>
</tr>
<tr>
<td>![M]</td>
<td>M</td>
<td>Performs a measurement of a single qubit in the computational basis. The result of the measurement is stored in the measured qubit; see Qubit.Bit. The state vector is collapsed according to the result of the measurement. The measured qubit becomes &quot;classical&quot; and must be reset before any further quantum gates are performed on it.</td>
</tr>
<tr>
<td>![Native]</td>
<td>Native</td>
<td>Performs any desired native operations at this point in the circuit.</td>
</tr>
<tr>
<td>![NativeDbg]</td>
<td>NativeDbg</td>
<td>Performs any desired native operations at this point in the circuit. This version doesn't appear in a circuit drawing.</td>
</tr>
<tr>
<td>![PC]</td>
<td>PC</td>
<td>Performs a gate based on a classical condition. The parent gate is executed if the test function returns true. Typically this condition is based on the results of one or more joint parity measurements.</td>
</tr>
<tr>
<td>![R]</td>
<td>R</td>
<td>Performs a $2\pi/2^K$ rotation gate.</td>
</tr>
<tr>
<td>![Reset]</td>
<td>Reset</td>
<td>Resets a qubit to a specified initial state after it has been measured. This allows quantum gates to be performed on it.</td>
</tr>
</tbody>
</table>
performed against the collapsed qubit. See the M gate.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Restore Resets a qubit after it has been measured. This allows quantum gates to be performed against the collapsed qubit. The initial state of the qubit will be its last measured state. See the M gate.</td>
</tr>
<tr>
<td>S</td>
<td>S Performs a phase gate.</td>
</tr>
<tr>
<td>S</td>
<td>SWAP Performs a swap gate, exchanging the quantum states of two qubits.</td>
</tr>
<tr>
<td>S</td>
<td>T Performs a pi/8 gate.</td>
</tr>
<tr>
<td>S</td>
<td>T_BC Performs a transverse classically-controlled gate.</td>
</tr>
<tr>
<td>S</td>
<td>Transverse Expands a parent gate to a transverse version.</td>
</tr>
<tr>
<td>S</td>
<td>X Performs a Pauli X gate.</td>
</tr>
<tr>
<td>S</td>
<td>Y Performs a Pauli Y gate.</td>
</tr>
<tr>
<td>S</td>
<td>Z Performs a Pauli Z gate.</td>
</tr>
</tbody>
</table>

Top

▲ Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>BangBang In F# code, this operator is named !!. Builds a list of Qubits from a wide variety of</td>
</tr>
</tbody>
</table>
### Possible inputs.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BangLess</strong></td>
<td>In F# code, this operator is named !&lt;. Gets the gate definition from a gate function. This is usually used to discover the &quot;parent&quot; gate.</td>
</tr>
<tr>
<td><strong>GreaterBangLessa</strong></td>
<td>In F# code, this operator is named &gt;!. Applies a function to each qubit in a list, passing a parameter to the function as well as each Qubit. The parameter values may be a list of a single value which is then passed to each invocation.</td>
</tr>
<tr>
<td><strong>GreaterLess</strong></td>
<td>In F# code, this operator is named &gt;&gt;=. Applies a function to each qubit in a list of Qubits.</td>
</tr>
</tbody>
</table>

### Top

#### See Also

Reference

Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

## Operations Methods

The *Operations* type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>Amplitude Damping Channel using two Kraus operators (always put the most probable one first)</td>
</tr>
<tr>
<td>Adj</td>
<td>Performs the adjoint of the parent gate. This only works if the parent gate is a unitary gate.</td>
</tr>
<tr>
<td>BC</td>
<td>Performs a gate under classical control. The parent gate will be executed if the control qubit has a measured value of One.</td>
</tr>
<tr>
<td>BCany</td>
<td>Performs a gate under classical control. This gate allows an arbitrary condition to be specified. The parent gate will be executed if the condition function evaluates to true.</td>
</tr>
<tr>
<td>CCgate</td>
<td>Performs a parent gate under two-qubit quantum control. The parent must be a unitary gate.</td>
</tr>
<tr>
<td>CCNOT</td>
<td>Performs a Toffoli or Controlled-Controlled-NOT gate</td>
</tr>
<tr>
<td>Cgate</td>
<td>Performs a parent gate under</td>
</tr>
</tbody>
</table>
quantum control. The parent must be a unitary gate.

<table>
<thead>
<tr>
<th>Action</th>
<th>Gate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CgateNC</td>
<td>Performs a parent gate under quantum control. The parent must be a unitary gate. The resulting gate is not cached.</td>
</tr>
<tr>
<td></td>
<td>CNOT</td>
<td>Performs a quantum-controlled NOT gate</td>
</tr>
<tr>
<td></td>
<td>CZ</td>
<td>Performs a quantum-controlled Pauli Z gate.</td>
</tr>
<tr>
<td></td>
<td>DP</td>
<td>Depolarizing channel using four Kraus operators (always put the most probable one first)</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>Performs a Hadamard gate.</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>Performs a Pauli I (identify) gate.</td>
</tr>
<tr>
<td></td>
<td>JM</td>
<td>Performs a joint parity measurement in the given basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.</td>
</tr>
<tr>
<td></td>
<td>JMx</td>
<td>Performs a joint parity measurement in the X basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.</td>
</tr>
<tr>
<td></td>
<td>JMz</td>
<td>Performs a joint parity measurement in the computational basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.</td>
</tr>
</tbody>
</table>
The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.</td>
</tr>
<tr>
<td>LabelC</td>
<td>Adds a label to a circuit drawing. The label be centered in and will fill the current cell.</td>
</tr>
<tr>
<td>LabelCD</td>
<td>Adds a framed label to a circuit drawing. The label be centered in and will fill the current cell. The label will have a box drawn tightly around it.</td>
</tr>
<tr>
<td>LabelD</td>
<td>Adds a label to a circuit drawing. The label will appear below the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.</td>
</tr>
<tr>
<td>LabelL</td>
<td>Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the left side of the current column. The label will take up no logical space; it will not fill the current cell.</td>
</tr>
<tr>
<td></td>
<td>LabelR</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>LabelRaw</td>
</tr>
<tr>
<td></td>
<td>LabelU</td>
</tr>
<tr>
<td></td>
<td>LoadCache</td>
</tr>
<tr>
<td></td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>Native</td>
</tr>
</tbody>
</table>
operations at this point in the circuit.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NativeDbg</td>
<td>Performs any desired native operations at this point in the circuit. This version doesn't appear in a circuit drawing.</td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>Performs a gate based on a classical condition. The parent gate is executed if the test function returns true. Typically this condition is based on the results of one or more joint parity measurements.</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Performs a $2\pi/2^K$ rotation gate.</td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>Resets a qubit to a specified initial state after it has been measured. This allows quantum gates to be performed against the collapsed qubit. See the M gate.</td>
<td></td>
</tr>
<tr>
<td>Restore</td>
<td>Resets a qubit after it has been measured. This allows quantum gates to be performed against the collapsed qubit. The initial state of the qubit will be its last measured state. See the M gate.</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>Performs a phase gate.</td>
<td></td>
</tr>
<tr>
<td>SWAP</td>
<td>Performs a swap gate, exchanging the quantum states of two qubits.</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Performs a $\pi/8$ gate.</td>
<td></td>
</tr>
<tr>
<td>T_BC</td>
<td>Performs a transverse classically-</td>
<td></td>
</tr>
</tbody>
</table>
controlled gate.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Gate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🟡</td>
<td>Transverse</td>
<td>Expands a parent gate to a transverse version.</td>
</tr>
<tr>
<td>🟡</td>
<td>X</td>
<td>Performs a Pauli X gate.</td>
</tr>
<tr>
<td>🟡</td>
<td>Y</td>
<td>Performs a Pauli Y gate.</td>
</tr>
<tr>
<td>🟡</td>
<td>Z</td>
<td>Performs a Pauli Z gate.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsAD Method

Amplitude Damping Channel using two Kraus operators (always put the most probable one first)

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

### Syntax

```fsharp
static member AD :  
  prob : float *  
  qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**prob**
- Type: System.Double  
  Probability of damping

**qs**
- Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
  The list of qubits that the Kraus operators touch

### See Also

Reference  
Operations Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsAdj Method

Performs the adjoint of the parent gate. This only works if the parent gate is a unitary gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.6)

### Syntax

```fsharp
static member Adj :
    f : FSharpFunc<FSharpList<Qubit>, Unit> ',
    qs : FSharpList<Qubit> -> unit
```

### Parameters

- **f**
  
  Type: **Microsoft.FSharp.Core.FSharpFunc**

- **qs**

  A list of qubits which are passed to the adjoint.

### See Also

Reference  
Operations Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

OperationsBC Method

Performs a gate under classical control. The parent gate will be executed if the control qubit has a measured value of One.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

```
 static member BC :
    f : FSharpFunc<FSharpList<Qubit>, Unit> ,
    qs : FSharpList<Qubit> -> unit
```

Parameters

- **f**
  Type: `Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>`
  The parent gate to control

- **qs**
  Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
  A list of qubits. The head qubit is the control qubit; its measured state determines whether or not the parent gate is executed. The tail of the list is passed to the parent gate if it is executed.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Operations

BCany Method

Performs a gate under classical control. This gate allows an arbitrary condition to be specified. The parent gate will be executed if the condition function evaluates to true.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
static member BCany :
    cnt : int *
    tst : FSharpFunc<FSharpList<Qubit>, bool>
    f : FSharpFunc<FSharpList<Qubit>, Unit>
    qs : FSharpList<Qubit> -> unit
```

Parameters

cnt

Type: System.Int32
The count of binary control bits. These bits will not be passed to the parent gate.

tst

Type: Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Boolean>
The condition function that controls execution of the parent gate.

f

Type: Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>
The parent gate to control

qs
Type: `Microsoft.FSharp.Collections.FSharpList` `Qubit`
A list of qubits. The initial `cnt` qubits are provided to the condition function, and the remainder are passed to the parent gate if it is executed.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace
OperationsBC(FSharpFuncFSharpListQubit, Unit, FSharpListQubit)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Operations CCgate Method

Performs a parent gate under two-qubit quantum control. The parent must be a unitary gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member CCgate :
    f : FSharpFunc<FSharpList<Qubit>, Unit>, 'qs :
    FSharpList<Qubit> -> unit
```

### Parameters

**f**

Type: `Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>`  
The gate to control.

**qs**

Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`  
The first two qubits are the control, and the remainder are passed to the parent gate.

### See Also

**Reference**

Operations Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Operations CCNOT Method

Performs a Toffoli or Controlled-Controlled-NOT gate

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member CCNOT :  
    qs : FSharpList<Qubit> -> unit
```

#### Parameters

- **qs**
  - Type: Microsoft.FSharp.CollectionsFSharpListQubit
  - The first two qubits in the list are the control qubits, and the third qubit is the target.

### See Also

#### Reference

- [Operations Class](#)
- [Microsoft.Research.Liquid Namespace](#)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsCgate Method

Performs a parent gate under quantum control. The parent must be a unitary gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member Cgate :  
    f : FSharpFunc<FSharpList<Qubit>, Unit>  
    qs : FSharpList<Qubit> -&gt; unit
```

### Parameters

**f**  
Type: `Microsoft.FSharp.CorexFSharpFuncxFSharpListQubit, Unit`  
The gate to control.

**qs**  
Type: `Microsoft.FSharp.CollectionsxFSharpListQubit`  
The first qubit is the control qubit, and the remainder are passed to the parent gate.

### See Also

**Reference**  
Operations Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsCgateNC Method

Performs a parent gate under quantum control. The parent must be a unitary gate. The resulting gate is not cached.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member CgateNC :
    f : FSharpFunc<FSharpList<Qubit>, Unit> ->
    qs : FSharpList<Qubit> -> unit
```

### Parameters

- **f**
  - Type: `Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>`
  - The gate to control.

- **qs**
  - Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
  - The first qubit is the control qubit, and the remainder are passed to the parent gate.

### See Also

- Reference
  - Operations Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Operations

CNOT Method

Performs a quantum-controlled NOT gate

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
static member CNOT : qs : FSharpList<Qubit> -> unit
```

Parameters

`qs`

Type: Microsoft.FSharp.Collections.FSharpList<Qubit>

The head qubit is the control qubit, the second qubit is the target.

See Also

Reference

Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

OperationsCZ Method

Performs a quantum-controlled Pauli Z gate.

Namespaces: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

Syntax

F#

```fsharp
static member CZ :
    qs : FSharpList<Qubit> -> unit
```

Parameters

`qs`

Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
The head qubit is the control qubit, the second qubit is the target.

See Also

Reference

Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsDP Method

Depolarizing channel using four Kraus operators (always put the most probable one first)

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member DP :
    prob : float *
    qs : FSharpList<Qubit> -> unit
```

### Parameters

- **prob**
  
  Type: `System.Double`
  
  Probability of depolarizing in X, Y or Z

- **qs**
  
  Type: `Microsoft.FSharp.Collections.FSharpList`<`Qubit`>
  
  The list of qubits that the Kraus operators touch

### See Also

- Reference
  - Operations Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Operations

H Method

Performs a Hadamard gate.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5981.24943)

Syntax

\[
\text{static member } H : \quad \text{qs : FSharpList<Qubit> \rightarrow unit}
\]

Parameters

qs

Type: Microsoft.FSharp.Collections.FSharpListQubit
The head qubit of this list is operated on.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Operations\texttt{I} Method

Performs a Pauli $I$ (identify) gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member I :  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

\(qs\)

Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`  
The head qubit of this list is operated on.

### See Also

- **Reference**  
  - Operations Class  
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Operations

JM Method

Performs a joint parity measurement in the given basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.

Namespace: Microsoft.Research.Liquid

Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
static member JM :
    tag : string *
    basis : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

**tag**

Type: System.String

If not empty, the symbol name to store the measurement result under. See **Ket.Symbol**.

**basis**

Type: System.String

The basis to measure in. This may be a string of any length, including empty, made up of X, Y, and Z characters. Each character is used to specify the basis for the corresponding qubit being measured. If there are more qubits than bases specified, then the last basis character is repeated. If this is an empty string, then it is treated as "Z", which performs a joint measurement of all qubits in the computational basis.

**qs**
Type: Microsoft.FSharp.Collections.FSharpList{Qubit}The list of qubits to jointly measure.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsJMx Method

Performs a joint parity measurement in the X basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
static member JMx :
    tag : string *
    qs : FSharpList<Qubit>  ->  unit
```

Parameters

*tag*

Type: System.String
If not empty, the symbol name to store the measurement result under. See Ket.Symbol.

*qs*

Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
The list of qubits to jointly measure.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Operations

JMz Method

Performs a joint parity measurement in the computational basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
static member JMz :
    tag : string *
    qs : FSharpList<Qubit> -> unit
```

### Parameters

**tag**  
Type: `System.String`  
If not empty, the symbol name to store the measurement result under. See `Ket.Symbol`.

**qs**  
Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`  
The list of qubits to jointly measure.

### See Also

**Reference**  
Operations Class  
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

OperationsLabel Method

Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

Syntax

```fsharp
static member Label :
    nam : string *
    qs : FSharpList<Qubit>  ->  unit
```

Parameters

**nam**
Type: System.String
The string to use as a label.

**qs**
Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
A list of qubits, the first of which determines the vertical positioning of the label.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsLabelC Method

Adds a label to a circuit drawing. The label be centered in and will fill the current cell.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member LabelC :
    nam : string *
    qs : FSharpList<Qubit>  ->  unit
```

**Parameters**

`nam`  
Type: `System.String`
The string to use as a label.

`qs`  
Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
A list of qubits, the first of which determines the vertical positioning of the label.

### See Also

**Reference**

Operations Class  
Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

OperationsLabelCD Method

Adds a framed label to a circuit drawing. The label be centered in and will fill the current cell. The label will have a box drawn tightly around it.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

## Syntax

```fsharp
static member LabelCD :
    nam : string *
    qs : FSharpList<Qubit> -> unit
```

### Parameters

**nam**

*Type:* System.String

The string to use as a label.

**qs**

*Type:* Microsoft.FSharp.Collections.FSharpList<Qubit>

A list of qubits, the first of which determines the vertical positioning of the label.

## See Also

**Reference**

Operations Class

Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsLabelD Method

Adds a label to a circuit drawing. The label will appear below the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1*)

### Syntax

```fsharp
static member LabelD :
    nam : string *
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**nam**
- **Type:** System.String  
  The string to use as a label.

**qs**
- **Type:** Microsoft.FSharp.Collections.FSharpList<Qubit>  
  A list of qubits, the first of which determines the vertical positioning of the label.

### See Also

**Reference**
- Operations Class  
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsLabelL Method

Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the left side of the current column. The label will take up no logical space; it will not fill the current cell.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

### Syntax

```fsharp
static member LabelL :  
    nam : string *  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**nam**
- Type: System.String
- The string to use as a label.

**qs**
- Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
- A list of qubits, the first of which determines the vertical positioning of the label.

### See Also

- Reference
  - Operations Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsLabelR Method

Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the right side of the current column. The label will take up no logical space; it will not fill the current cell.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4+)

### Syntax

```fsharp
static member LabelR :  
    nam : string *  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**nam**

Type: `System.String`  
The string to use as a label.

**qs**

A list of qubits, the first of which determines the vertical positioning of the label.

### See Also

Reference  
Operations Class  
Microsoft.Research.Liquid Namespace
Operations.LabelRaw Method

Adds a raw label to a circuit drawing. The provided string should contain the Q-Circuit commands used to draw the label. Note that labels drawn using this gate will only appear when rendered to "qc" format.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
static member LabelRaw :
  cmd : string *
  qs : FSharpList<Qubit> -> unit
```

Parameters

- **cmd**
  - Type: System.String
  - The string to use to draw the label.

- **qs**
  - Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
  - A list of qubits, the first of which determines the vertical positioning of the label.

See Also

Reference
- Operations Class
- Microsoft.Research.Liquid Namespace
OperationsLabelU Method

Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member LabelU :  
    nam : string *  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**nam**  
Type: System.String  
The string to use as a label.

**qs**  
Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
A list of qubits, the first of which determines the vertical positioning of the label.

### See Also

**Reference**  
Operations Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsLoadCache Method

Preload the cache with basic gates: Paulis, measurement, Hadamard, phase, CNOT, Toffoli, swap, and T.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

⚠️ Syntax

```fsharp
static member LoadCache : unit -> unit
```

⚠️ See Also

Reference  
Operations Class  
Microsoft.Research.Liquid Namespace
OperationsM Method

Performs a measurement of a single qubit in the computational basis. The result of the measurement is stored in the measured qubit; see Qubit.Bit. The state vector is collapsed according to the result of the measurement. The measured qubit becomes "classical" and must be reset before any further quantum gates are performed on it.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
static member M : 
    qs : FSharpList<Qubit>  ->  unit
```

Parameters

`qs`
Type: Microsoft.FSharp.Collections.FSharpList{Qubit}
A list of qubits; the head of the list is measured.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Operations
Native Method

Performs any desired native operations at this point in the circuit.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1*)

Syntax

```
static member Native :
    f : FSharpFunc<FSharpList<Qubit>, Unit>
    qs : FSharpList<Qubit> -> unit
```

Parameters

*f*

Type: `Microsoft.FSharp.Core.FSharpFunc`<br>

The native function to call.

*qs*

Type: `Microsoft.FSharp.Collections.FSharpList`<br>

The list of qubits to pass to the native operation.

See Also

Reference

Operations Class
Microsoft.Research.Liquid Namespace
OperationsNativeDbg(FSharpFuncFSharpListQubit, Unit, FSharpListQubit)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsNativeDbg Method

Performs any desired native operations at this point in the circuit. This version doesn't appear in a circuit drawing.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4*)

### Syntax

```fsharp
static member NativeDbg :  
    f : FSharpFunc<FSharpList<Qubit>, Unit> ,  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**f**

- **Type:** Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>
- The native function to call.

**qs**

- **Type:** Microsoft.FSharp.Collections.FSharpList<Qubit>
- The list of qubits to pass to the native operation.

### See Also

- **Reference**  
  Operations Class  
  Microsoft.Research.Liquid Namespace  
  OperationsNative(FSharpFunc<FSharpList<Qubit>, Unit, FSharpList<Qubit>)
OperationsPC Method

Performs a gate based on a classical condition. The parent gate is executed if the test function returns true. Typically this condition is based on the results of one or more joint parity measurements.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```
static member PC :
    lbl : string *
    tst : FSharpFunc<FSharpList<Qubit>, bool>
    f : FSharpFunc<FSharpList<Qubit>, Unit> *
    qs : FSharpList<Qubit> -> unit
```

### Parameters

**lbl**
Type: `System.String`
A label for drawing.

**tst**
Type: `Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Boolean>`
The test function used to control execution of the parent gate.

**f**
Type: `Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>`
The parent gate to control

**qs**
Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
A list of qubits which will be passed to the parent gate if it is executed.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace
Ket.Symbol(String, Int32)
OperationsR Method

Performs a \( \frac{2\pi}{2^k} \) rotation gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member R :  
    k : int *  
    qs : FSharpList<Qubit> -> unit
```

### Parameters

- **k**  
  Type: System.Int32  
  The rotation parameter

- **qs**  
  Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
  The head qubit of this list is operated on.

### See Also

- Reference  
  Operations Class  
  Microsoft.Research.Liquid Namespace
OperationsReset Method

Resets a qubit to a specified initial state after it has been measured. This allows quantum gates to be performed against the collapsed qubit. See the M gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member Reset :  
    b : Bit  *  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

- **b**
  
  Type: Microsoft.Research.LiquidBit  
  The initial state of the reset qubit, either Zero or One.

- **qs**
  
  Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
  A list of qubits; the head of the list is reset.

### See Also

- Reference  
  Operations Class  
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Operations

Restore Method

Resets a qubit after it has been measured. This allows quantum gates to be performed against the collapsed qubit. The initial state of the qubit will be its last measured state. See the M gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member Restore :  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

*qs*

Type: **Microsoft.FSharp.Collections.FSharpList**<Qubit>

A list of qubits; the head of the list is reset.

### See Also

**Reference**

Operations Class  
Microsoft.Research.Liquid Namespace
Operations S Method

Performs a phase gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

**Syntax**

```fsharp
static member S :
    qs : FSharpList<Qubit>  ->  unit
```

**Parameters**

`qs`  
Type: **`Microsoft.FSharp.Collections.FSharpList<Qubit>`**  
The head qubit of this list is operated on.

**See Also**

**Reference**
- Operations Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsSWAP Method

Performs a swap gate, exchanging the quantum states of two qubits.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```fsharp
static member SWAP :
    qs : FSharpList<Qubit> -> unit
```

Parameters

`qs`

Type: Microsoft.FSharp.Collections.FSharpList<Qubit>

The states of the first two qubits in the list are swapped.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsT Method

Performs a pi/8 gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```
static member T :
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

- **qs**
  - **Type:** Microsoft.FSharp.Collections.FSharpList<Qubit>
  - The head qubit of this list is operated on.

### See Also

- **Reference**  
  - Operations Class  
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Operations

T_BC Method

Performs a transverse classically-controlled gate.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member T_BC :
    decode : FSharpFunc<FSharpList<Qubit>, Tuple Bit, Int32>
    f : FSharpFunc<FSharpList<Qubit>, Unit>
    qs : FSharpList<Qubit> -> unit
```

### Parameters

**decode**

*Type:* Microsoft.FSharp.Core.FSharpFunc<>Qubit, Tuple Bit, Int32

The code-specific decode function that takes a list of measured qubits and returns a logical value (One or Zero) and the Hamming distance from the current state to a valid code state.

**f**

*Type:* Microsoft.FSharp.Core.FSharpFunc<>Qubit, Unit

The (transverse) parent gate to control

**qs**

*Type:* Microsoft.FSharp.Collections.FSharpList<Qubit>

The first codeSize qubits in the list are passed to the *decode* function to compute a logical Zero or One state. If the decoded state is One, then the remaining qubits are passed to the parent gate.
See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Operations

Transverse Method

Expands a parent gate to a transverse version.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
cnt

static member Transverse : 
    cCnt : int * 
    f : FSharpFunc<FSharpList<Qubit>, Unit> ' 
    qs : FSharpList<Qubit> -> unit
```  

### Parameters

**cCnt**
- Type: System.Int32
  - The total number of qubits in code

**f**
- Type: Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>
- The parent gate to expand

**qs**
- Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
- The qubits that final gate will operate on

### Remarks

If the parent gate operates on a single qubit, then it is applied to each qubit in the code. If the parent gate applies to two qubits, then it is applied to pairs of qubits selected from the first and second half
of the list. For example, if the code contained 3 qubits, so that the qubits to operate on would contain 6 qubits, then the gate would be applied to pairs 0 and 3, 1 and 4, and 2 and 5. It is an error if the parent gate takes more than two qubits.

The transverse gate's name is the parent's name with a "_T" suffix.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsX Method

Performs a Pauli X gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```
static member X :  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

- `qs`
  
  Type: Microsoft.FSharp.Collections.FSharpList`Qubit`
  
  The head qubit of this list is operated on.

### See Also

- Reference
  - Operations Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Operations

**Y Method**

Performs a Pauli Y gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

**Syntax**

```fsharp
static member Y :  
    qs : FSharpList<Qubit>  ->  unit
```

**Parameters**

**qs**

Type: Microsoft.FSharp.Collections.FSharpList<Qubit>

The head qubit of this list is operated on.

**See Also**

Reference
- Operations Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsZ Method

Performs a Pauli Z gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
static member Z :  
    qs : FSharpList<Qubit> -> unit
```

### Parameters

`qs`

*Type:* Microsoft.FSharp.Collections.FSharpList<Qubit>*

The head qubit of this list is operated on.

### See Also

Reference

- Operations Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Operations Operators

The Operations type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BangBang</td>
<td>In F# code, this operator is named <code>!!</code>. Builds a list of Qubits from a wide variety of possible inputs.</td>
</tr>
<tr>
<td>BangLess</td>
<td>In F# code, this operator is named <code>!</code>. Gets the gate definition from a gate function. This is usually used to discover the &quot;parent&quot; gate.</td>
</tr>
<tr>
<td>GreaterBangLess</td>
<td>In F# code, this operator is named <code>&gt;!</code>. Applies a function to each qubit in a list, passing a parameter to the function as well as each Qubit. The parameter values may be a list of a single value which is then passed to each invocation.</td>
</tr>
<tr>
<td>GreaterLess</td>
<td>In F# code, this operator is named <code>&gt;&gt;</code>. Applies a function to each qubit in a list of Qubits.</td>
</tr>
</tbody>
</table>
See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Operations

BangBang Operator

In F# code, this operator is named !!. Builds a list of Qubits from a wide variety of possible inputs.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
F# does not support this operator.
```

### Parameters

`qs`

Type: `System.Object`

The value to interpret as a list of Qubits.

### Return Value

Type: `FSharpList<Qubit>`

A list of Qubits

### Remarks

The inputs that are interpreted by this operator are:

- **A Ket**, which is interpreted as the complete list of Qubits in the Ket.
- **A single Qubit**, which is interpreted as a single-element list.
- **A two-tuple of Qubits**, which is interpreted as a two-element list.
- **A three-tuple of Qubits**, which is interpreted as a three-element list.
- **A list of Qubits**, which is returned directly.
- A two-tuple of lists of Qubits, which are concatenated into a single list.
- A three-tuple of lists of Qubits, which are concatenated into a single list.
- A list of lists of Qubits, which are concatenated into a single list.
- A two-tuple of a list of Qubits and a single Qubit, which are concatenated into a single list.
- A two-tuple of a single Qubit and a list of Qubits, which are concatenated into a single list.
- A two-tuple of a list of Qubits and an integer, which is interpreted as the single-element list containing the item in the list indexed by the integer.
- A three-tuple of a list of Qubits and two integers, which is interpreted as the two-element list containing the items in the list indexed by the two integers.
- A four-tuple of a list of Qubits and three integers, which is interpreted as the three-element list containing the item in the list indexed by the three integers.
- A two-tuple of a list of Qubits and a list of integers, which is interpreted as the list containing the items in the Qubit list indexed by the elements in the integer list.

See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsBangLess Operator

In F# code, this operator is named !<. Gets the gate definition from a gate function. This is usually used to discover the "parent" gate.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.★)

### Syntax

```fsharp
does not support this operator.
```

### Parameters

\( f \)
- **Type:** Microsoft.FSharp.Core.FSharpFuncFSharpListQubit, Unit
- The gate function we want the Gate for.

\( qs \)
- **Type:** Microsoft.FSharp.Collections.FSharpListQubit
- A list of Qubits that can be used to find the gate.

### Return Value

- **Type:** Gate
- The Gate corresponding to the gate function. Note that if \( f \) is not a gate function, this function will raise an exception.

### See Also

- **Reference**
  - Operations Class
  - Microsoft.Research.Liquid Namespace
Operations

GreaterBangLessa

Operator

In F# code, this operator is named >!<. Applies a function to each qubit in a list, passing a parameter to the function as well as each Qubit. The parameter values may be a list of a single value which is then passed to each invocation.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

F# does not support this operator.

Parameters

\( f \)

Type: Microsoft.FSharp.Core.FSharpFunc<
FSharpFunc<FSharpList<Qubit>, Unit>
The function to call. The extra parameter must be the first argument to the function.

\( \text{args} \)

Type: System.Object
Either a list of arguments, one per Qubit, or a single value that is passed with each Qubit.

\( \text{qs} \)

Type: Microsoft.FSharp.Collections.FSharpList<Qubit>
The list of qubits to iterate over.

Type Parameters
See Also

Reference
Operations Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
OperationsGreaterLess Operator

In F# code, this operator is named `><`. Applies a function to each qubit in a list of Qubits.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

▲ **Syntax**

F# does not support this operator.

**Parameters**

- `f`
  - Type: `Microsoft.FSharp.Core.FSharpFunc<FSharpList<Qubit>, Unit>`
  - The gate function to call.

- `qs`
  - Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`
  - The list of qubits to iterate over.

▲ **See Also**

Reference
- Operations Class  
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QECC Class

Base class for quantum error correcting codes.

## Inheritance Hierarchy

```
SystemObject  Microsoft.Research.LiquidQECC
             Microsoft.Research.LiquidSteane7
```

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
**Version:** 1.0.5981.24943 (1.0.*)

## Syntax

```fsharp
[<AbstractClassAttribute>]
[<SerializableAttribute>]
type QECC = class end
```

The `QECC` type exposes the following members.

## Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com" alt="QECC" /></td>
<td>Initializes a Quantum Error Correcting Circuit (QECC). This base constructor must be called by all derived constructors.</td>
</tr>
</tbody>
</table>

## Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Circuit</strong></td>
<td>The error-correcting circuit built by Compile. The circuit will be built now if it hasn't already been.</td>
</tr>
<tr>
<td><strong>Ket</strong></td>
<td>The state vector for the compiled code.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compile</td>
<td>Compiles the target circuit into an error-correcting version.</td>
</tr>
<tr>
<td>Decode</td>
<td>Decodes a set of measured physical qubits to get the measured value for a logical qubit.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetMeasured</td>
<td>Gets the measured values of the physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Inject</strong></td>
<td>Injects dephasing errors with the given probability into the error-correcting circuit.</td>
</tr>
<tr>
<td><strong>Log2Phys</strong></td>
<td>Gets the physical qubits that make up a logical qubit.</td>
</tr>
<tr>
<td><strong>Prep</strong></td>
<td>A gate function that prepares a logical</td>
</tr>
<tr>
<td><strong>Replace</strong></td>
<td>Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit.</td>
</tr>
<tr>
<td><strong>Syndrome</strong></td>
<td>A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUId>) Simulator

QECC Constructor

Initializes a Quantum Error Correcting Circuit (QECC). This base constructor must be called by all derived constructors.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
new :
   aCnt : int *
   cCnt : int *
   tgt : Circuit -> QECC
```

Parameters

*aCnt*
Type: System.Int32
The number of ancilla qubits. By convention ancillae come first in the state vector, before data and syndrome qubits. Ancilla qubits are shared across all logical qubits and are for use during decoding, measurement, and other operations; they are not syndrome qubits.

*cCnt*
Type: System.Int32
The code size; that is, the number of physical qubits (data and syndrome) per logical qubit.

*tgt*
Type: Microsoft.Research.Liquid.Circuit
The target Circuit to build an error-correcting circuit for.
See Also

Reference
QECC Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QECC Properties

The QECC type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit</td>
<td>The error-correcting circuit built by Compile. The circuit will be built now if it hasn't already been.</td>
</tr>
<tr>
<td>Ket</td>
<td>The state vector for the compiled code.</td>
</tr>
</tbody>
</table>

## See Also

Reference
- QECC Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QECC Circuit Property

The error-correcting circuit built by Compile. The circuit will be built now if it hasn't already been.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

▲ Syntax

```
member Circuit : Circuit with get
```

**Property Value**  
Type: Circuit

▲ See Also

Reference  
QECC Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QECC Ket Property

The state vector for the compiled code.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.9)

Syntax

F#

```fsharp
member Ket : Ket with get
```

Property Value
Type: Ket

See Also

Reference
QECC Class
Microsoft.Research.Liquid Namespace
# QECC Methods

The **QECC** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compile</td>
<td>Compiles the target circuit into an error-correcting version.</td>
</tr>
<tr>
<td>Decode</td>
<td>Decodes a set of measured physical qubits to get the measured value for a logical qubit.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetMeasured</td>
<td>Gets the measured values of the physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Inject</td>
<td>Injects dephasing errors with the given probability into the error-correcting circuit.</td>
</tr>
<tr>
<td>Log2Phys</td>
<td>Gets the physical qubits that make up a logical qubit.</td>
</tr>
<tr>
<td>Prep</td>
<td>A gate function that prepares a logical</td>
</tr>
<tr>
<td>Replace</td>
<td>Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit.</td>
</tr>
<tr>
<td>Syndrome</td>
<td>A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
QECC Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

QECC Compile Method

Compiles the target circuit into an error-correcting version.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```
F# Copy

member Compile : unit -> unit
```

### See Also

Reference  
QECC Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QECCDecode Method

Decodes a set of measured physical qubits to get the measured value for a logical qubit.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*/)

### Syntax

```fsharp
abstract Decode : qs : FSharpList<Qubit> -> Tuple<Bit, int>
```

### Parameters

**qs**
- **Type:** Microsoft.FSharp.Collections.FSharpList<Qubit>
- The physical qubits to decode. They must already have been measured.

### Return Value

**Type:** Tuple<Bit, Int32>
- A tuple containing the logical measured value of the logical qubit, either Zero or One, and the Hamming distance from the physical state to the code space.

### See Also

- Reference
  - QECC Class
  - Microsoft.Research.Liquid Namespace
QECC Get Measured Method

Gets the measured values of the physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member GetMeasured : 
    qs : FSharpList<Qubit> -> int
```

### Parameters

**qs**

Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`

The set of qubits to read. This should be the set of physical (data and syndrome) qubits for one logical qubit.

### Return Value

Type: `Int32`

The total measured value of the qubits. The value of the first qubit in the list goes into the left-most (most significant) bit in the result.

### See Also

Reference
QECC Class
Microsoft.Research.Liquid Namespace
QECCInject Method

Injects dephasing errors with the given probability into the error-correcting circuit.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Inject : prob : float -> Tuple<Circuit, FSharpList<int>>
```

### Parameters

**prob**

Type: System.Double

The probability of an error on execution of a gate. Each qubit input to the gate has this same chance of a dephasing error. If an error is injected, it will be either an X, Y, or Z error, with equal probability. Note that errors are never injected on ancilla qubits.

### Return Value

Type: Tuple<Circuit, FSharpList<int>>

A tuple containing the new Circuit with errors injected as the first item and a list of injected error counts as the second. The error count list contains the count of X, Y, and Z errors injected, in that order.

### Remarks

Note that the errors are inserted as explicit gates into a new Circuit,
rather than randomly being injected on each run. That is, if the resulting circuit is executed multiple times, the same errors will be injected each time. To get a different set of random errors, this routine must be re-run and a new Circuit generated. Also note that errors are only inserted before wrapped and extended gates (Gate types Wrap and Ext). In particular, this means that errors appear before logical gates rather than before physical gates. If the original non-error correcting circuit included wrapped or extended gates, then errors may be injected before those gates as well.

See Also

Reference
QECC Class
Microsoft.Research.Liquid Namespace
QECCLog2Phys Method

Gets the physical qubits that make up a logical qubit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Log2Phys : w : int -> FSharpList<Qubit>
```

**Parameters**

- **w**
  
  Type: `System.Int32`
  
  The wire ID of the logical qubit in the original non-error correcting circuit.

**Return Value**

Type: `FSharpList<Qubit>`

A list of the physical qubits that implement the logical qubit.

### See Also

- **Reference**
  - QECC Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QECC Prep Method

A gate function that prepares a logical $|0\rangle$ qubit. This gets compiled into the error-correcting circuit by the Compile method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
abstract Prep :
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**qs**

*Type: Microsoft.FSharp.CollectionsFSharpListQubit*

The physical qubits for the logical qubit. This contains both data and syndrome qubits, but no ancillae.

### See Also

- Reference
- QECC Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QECCReplace Method

Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
abstract Replace : g : Gate -> FSharpOption<FSharpFunc<FSharpList<Qubit, Unit>>
override Replace : g : Gate -> FSharpOption<FSharpFunc<FSharpList<Qubit, Unit>>
```

### Parameters

\(g\)

Type: Microsoft.Research.LiquidGate  
The logical gate function to replace.

### Return Value

Type: FSharpOptionFSharpFuncFSharpListQubit, Unit  
An option holding a physical gate that implements the logical gate. If there is no physical implementation of the logical gate, then None is returned.

### Remarks

The default implementation replaces the Pauli gates, phase gate, Hadamard gate, identity gate, CNOT, and measurement with transverse versions of these gates. Classically-controlled versions of these gates are replaced by transverse versions, controlled by the
decoded (logical) value of the classical control.

See Also

Reference
QECC Class
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

QECC Syndrome Method

A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
abstract Syndrome :
    qs : FSharpList<Qubit> -> unit
```

**Parameters**

- `qs`  
  Type: Microsoft.FSharp.Collections.FSharpList<Qubit>  
  The ancilla qubits followed by physical qubits for the logical qubit. The number of ancillae was passed to the QECC constructor.

### See Also

- Reference  
  QECC Class  
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Qubit Class

Represents a quantum bit. New Qubits are created using the Ket Add methods.

Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidQubit

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

**F#**

```fsharp
<SerializableAttribute>
type Qubit = class end
```

The **Qubit** type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>Bit</td>
</tr>
<tr>
<td>Entangled</td>
<td></td>
</tr>
<tr>
<td>![ ]</td>
<td>Id</td>
</tr>
<tr>
<td>![ ]</td>
<td>Ket</td>
</tr>
</tbody>
</table>
One  A Complex vector that represents the state |1> in the computational basis.

Prob1  The probability of this qubit being 1.

State  The state vector for this qubit, if it is unentangled. This property will raise an exception if the qubit is entangled.

Type  The type of the qubit. This is only used for noise modeling.

Zero  A Complex vector that represents the state |0> in the computational basis.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dump</td>
<td>Dump this qubit's state.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ReAnimate(Bit)</td>
<td>Reanimates this qubit and</td>
</tr>
</tbody>
</table>
sets its state vector to $|0\rangle$ or $|1\rangle$. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReAni_mz(C_Vec)</td>
<td>Reanimates this qubit and sets its state vector. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.</td>
</tr>
<tr>
<td>Show_Mag</td>
<td>Creates a string representation of this Qubit. The representation shows the probabilities of measuring this Qubit in the computational $</td>
</tr>
<tr>
<td>State_Set(Bit)</td>
<td>Sets the state vector for this qubit to $</td>
</tr>
<tr>
<td>State_Set(C_Vec)</td>
<td>Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>StateSet(Complex, Complex)</code></td>
<td>Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.</td>
</tr>
<tr>
<td><code>StateSet(Double, Double, Double, Double)</code></td>
<td>Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Creates a string representation of this Qubit. The representation shows the current state of the qubit if it is unentangled. (Overrides <code>Object.ToString</code>.)</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Qubit Properties

The Qubit type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>The measured value of a qubit in the computational basis. This will be Unknown if the qubit has not been measured since it last interacted.</td>
</tr>
<tr>
<td>Entangled</td>
<td>Whether or not this qubit is entangled.</td>
</tr>
<tr>
<td>Id</td>
<td>Wire number in Ket vector</td>
</tr>
<tr>
<td>Ket</td>
<td>State we belong to</td>
</tr>
<tr>
<td>One</td>
<td>A Complex vector that represents the state</td>
</tr>
<tr>
<td>Prob1</td>
<td>The probability of this qubit being 1.</td>
</tr>
<tr>
<td>State</td>
<td>The state vector for this qubit, if it is unentangled. This property will raise an exception if the qubit is entangled.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of the qubit. This is only used for noise modeling.</td>
</tr>
<tr>
<td>Zero</td>
<td>A Complex vector that represents the state</td>
</tr>
</tbody>
</table>
See Also

Reference
Qubit Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitBit Property

The measured value of a qubit in the computational basis. This will be Unknown if the qubit has not been measured since it last interacted.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

⚠️ Syntax

```fsharp
member Bit : Bit with get, set
```

Property Value

Type: `Bit`

⚠️ See Also

Reference

Qubit Class  
Microsoft.Research.Liquid Namespace
QubitEntangled Property

Whether or not this qubit is entangled.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

**Syntax**

```fsharp
member Entangled : bool with get
```

**Property Value**  
Type: **Boolean**

**See Also**

**Reference**  
**Qubit Class**  
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

QubitId Property

Wire number in Ket vector

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Id : int with get
```

### Property Value

**Type:** Int32

### See Also

**Reference**  
Qubit Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitKet Property

State we belong to

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Ket : Ket with get
```

### Property Value

Type: Ket

---

### See Also

- Reference
- Qubit Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

QubitOne Property

A Complex vector that represents the state $|1\rangle$ in the computational basis.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

⚠️ Syntax

```fsharp
static member One : CVec with get
```

Property Value

Type: *CVec*

⚠️ See Also

Reference

- Qubit Class  
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitProb1 Property

The probability of this qubit being 1.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0*)

Syntax

```
member Prob1 : float with get
```

Property Value
Type: Double

See Also

Reference
Qubit Class
Microsoft.Research.Liquid Namespace
The state vector for this qubit, if it is unentangled. This property will raise an exception if the qubit is entangled.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
member State : CVec with get
```

**Property Value**

Type: `CVec`

### See Also

- **Reference**
  - Qubit Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitType Property

The type of the qubit. This is only used for noise modeling.

**Namespace**: Microsoft.Research.Liquid  
**Assembly**: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

⚠️Syntax

```fsharp
member Type : string with get, set
```

**Property Value**

Type: **String**

⚠️See Also

**Reference**

Qubit Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitZero Property

A Complex vector that represents the state $|0>$ in the computational basis.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```fsharp
static member Zero : CVec with get
```

**Property Value**  
Type: CVec

### See Also

Reference  
Qubit Class  
Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Qubit Methods

The Qubit type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dump</td>
<td>Dump this qubit's state.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ReAnimate(Bit)</td>
<td>Reanimates this qubit and sets its state vector to</td>
</tr>
<tr>
<td>ReAnimate(CVec)</td>
<td>Reanimates this qubit and sets its state vector.</td>
</tr>
</tbody>
</table>
Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShowMag</strong></td>
<td>Creates a string representation of this Qubit. The representation shows the probabilities of measuring this Qubit in the computational $</td>
</tr>
<tr>
<td><strong>StateSet(Bit)</strong></td>
<td>Sets the state vector for this qubit to $</td>
</tr>
<tr>
<td><strong>StateSet(CVec)</strong></td>
<td>Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.</td>
</tr>
<tr>
<td><strong>StateSet(Complex, Complex)</strong></td>
<td>Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.</td>
</tr>
<tr>
<td><strong>StateSet(Double, Double, Double)</strong></td>
<td>Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.</td>
</tr>
</tbody>
</table>
### See Also

- **Reference**
  - **Qubit Class**
  - **Microsoft.Research.Liquid Namespace**
QubitDump Method

Dump this qubit's state.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```fsharp
member Dump :
    f : FSharpOption<FSharpFunc<int, FSharpFunc<int>, Unit>
level : FSharpOption<int> -> unit
```

Parameters

- **f**
  Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<int>, Unit>`
  The optional output function to use. The default is `showLogInd`.

- **level**
  Type: `Microsoft.FSharp.Core.FSharpOption<int>`
  The optional indentation level. The default is 0.

See Also

Reference
- Qubit Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
# QubitReAnimate Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReAnimate(Bit)</td>
<td>Reanimates this qubit and sets its state vector to (</td>
</tr>
<tr>
<td>ReAnimate(CVec)</td>
<td>Reanimates this qubit and sets its state vector. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

- Qubit Class
- Microsoft.Research.Liquid Namespace
Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitReAnimate Method (Bit)

Reanimates this qubit and sets its state vector to |0> or |1>. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

## Syntax

```fsharp
member ReAnimate : b : Bit -> unit
```

### Parameters

\(b\)

- **Type:** Microsoft.Research.LiquidBit
  - The Bit value to set this qubit's state to.

## See Also

**Reference**
- **Qubit Class**
- **ReAnimate Overload**
- **Microsoft.Research.Liquid Namespace**

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitReAnimate Method (CVec)

Reanimates this qubit and sets its state vector. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

**F#**

```fsharp
member ReAnimate :
    v : CVec -> unit
```

### Parameters

**v**

Type: Microsoft.Research.Liquid.CVec

The Complex vector to set this qubit's state to.

### See Also

Reference

- Qubit Class
- ReAnimate Overload
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitShowMag Method

Creates a string representation of this Qubit. The representation shows the probabilities of measuring this Qubit in the computational |0> and |1> states.

**Namespace**: Microsoft.Research.Liquid  
**Assembly**: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

<table>
<thead>
<tr>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>member ShowMag : unit -&gt; string</td>
</tr>
</tbody>
</table>

**Return Value**

**Type**: String  
The string representation

### See Also

**Reference**

Qubit Class  
Microsoft.Research.Liquid Namespace

# QubitStateSet Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StateSet(Bit)</td>
<td>Sets the state vector for this qubit to</td>
</tr>
<tr>
<td>StateSet(CVec)</td>
<td>Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.</td>
</tr>
<tr>
<td>StateSet(Complex, Complex)</td>
<td>Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.</td>
</tr>
<tr>
<td>StateSet(Double, Double, Double, Double)</td>
<td>Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.</td>
</tr>
</tbody>
</table>
QubitStateSet Method (Bit)

Sets the state vector for this qubit to |0> or |1>, if it is unentangled. This method will raise an exception if the qubit is entangled.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```fsharp
member StateSet :  
  b : Bit -> unit
```

### Parameters

**b**

Type: Microsoft.Research.Liquid.Bit  
The Bit value to set this qubit's state to.

### See Also

**Reference**  
Qubit Class  
StateSet Overload  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitStateSet Method (CVec)

Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member StateSet : v : CVec -> unit
```

**Parameters**

- `v`  
  Type: Microsoft.Research.Liquid.CVec  
  The Complex vector to set this qubit's state to.

### See Also

**Reference**

- Qubit Class
- StateSet Overload
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

QubitStateSet Method (Complex, Complex)

Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

⚠️ **Syntax**

```fsharp
member StateSet :
    c1 : Complex  *
    c2 : Complex  -> unit
```

**Parameters**

`c1`

Type: `Microsoft.Research.LiquidComplex`  
The complex $|0\rangle$ amplitude to set this qubit's state to.

`c2`

Type: `Microsoft.Research.LiquidComplex`  
The complex $|1\rangle$ amplitude to set this qubit's state to.

⚠️ **See Also**

Reference

- Qubit Class
- StateSet Overload
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitStateSet Method (Double, Double, Double, Double, Double)

Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member StateSet :
    r1 : float *
    i1 : float *
    r2 : float *
    i2 : float -> unit
```

### Parameters

- **r1**
  - Type: System.Double
  - The real part of the |0> amplitude to set this qubit's state to.

- **i1**
  - Type: System.Double
  - The imaginary part of the |0> amplitude to set this qubit's state to.

- **r2**
  - Type: System.Double
  - The real part of the |1> amplitude to set this qubit's state to.

- **i2**
  - Type: System.Double
  - The imaginary part of the |1> amplitude to set this qubit's state to.
See Also

Reference
Qubit Class
StateSet Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
QubitToString Method

Creates a string representation of this Qubit. The representation shows the current state of the qubit if it is unentangled.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.∗)

### Syntax

```fsharp
abstract ToString : unit -> string
override ToString : unit -> string
```

**Return Value**

Type: **String**  
The string representation

### See Also

**Reference**

Qubit Class  
Microsoft.Research.Liquid Namespace  

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

RunMode Class

Trotterization types.

Inheritance Hierarchy

```
SystemObject  Microsoft.Research.LiquidRunMode
```

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
[<SealedAttribute>]                          
[<SerializableAttribute>]                    
type  RunMode  =                               
    class                                       
        interface  IEquatable<RunMode>          
        interface  IStructuralEquatable        
        interface  IComparable<RunMode>         
        interface  IComparable                 
        interface  IStructuralComparable       
    end                                         
```

The RunMode type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>

### Remarks

Possible values are:
- **Trotter1**: First order Trotter
- **Trotter1X**: First order Trotter splitting X to each side
- **Trotter1R**: Reverse mode (implies reversing list and negating angles)
- **Trotter1XR**: Reverse mode with X splitting (implies reversing list and negating angles)
- **Trotter2**: Second order Trotter (includes X splitting)
- **Trotter2R**: Second order reversed (includes X splitting)

### See Also

Reference
- `Microsoft.Research.Liquid Namespace`
RunMode Methods

The RunMode type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>

See Also

Reference

RunMode Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Spin Class

Hamiltonian for spin systems, such as the Ising model or a spin glass.

Inheritance Hierarchy

```
SystemObject  Microsoft.Research.LiquidHamiltonian
    Microsoft.Research.LiquidSpin
```

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
[<SerializableAttribute>]  
type Spin =  
    class  
        inherit Hamiltonian  
    end
```

The Spin type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spin(DictionaryInt32, Double, DictionaryTupleInt32, Int32, Double)</td>
<td>Creates a spin Hamiltonian for simple systems that only have single spin biases and two-spin couplings. The count of spins is inferred.</td>
</tr>
</tbody>
</table>
Spin(FSharpListSpinTerm, Int32, FSharpOptionRunMode) Creates a spin Hamiltonian from a set of spin terms.

- **currentCirc**: The last (grown) circuit that was run.
- **decohereModel**: The decoherence model for this Hamiltonian. (Inherited from Hamiltonian.)
- **Ket**: Gets the Ket vector associated with this Hamiltonian (Inherited from Hamiltonian.)
- **lastAnneal**: The last set of annealing coefficients that were applied.
- **lastRawCirc**: The last (ungrown) circuit that was run.
- **runMode**: The run mode. Note that setting this property will force a new circuit to be generated.
- **time**: The current simulation time. Note that setting this property will force a new circuit to be generated.
- **trotterN**: The Trotter number. Note that setting this property will force a
new circuit to be generated.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Returns an instance of Spin with the same Hamiltonian parameters as this instance. Simulation parameters such as run time and Trotter number are copied.</td>
</tr>
<tr>
<td>EnergyExpectation</td>
<td>Finds the expectation value of the Hamiltonian. That is, given a state vector $</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the object.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Ferro</code></td>
<td>Test adiabatic evolution with a ferro-magnetic chain</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the type of the current instance. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td><code>Peek</code></td>
<td>Peeks at the current Hamiltonian matrix. This is based on the most recently grown circuit.</td>
</tr>
<tr>
<td><code>Prep</code></td>
<td>Prepares the qubit state for a run. All qubits are reset and initialized to the ground state.</td>
</tr>
<tr>
<td><code>Run</code></td>
<td>Runs the simulation.</td>
</tr>
</tbody>
</table>
is a highly optimized implementation.

**Step**

Performs a single timestep of simulation.

**Test(String, Int32, Int32, FSharpList<Tuple<Int32, Double, Int32, Spin, FSharpOption<Boolean>, FSharpOption<FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>>})**

Executes a set of simulation runs for a spin Hamiltonian.

**Test(String, Int32, Int32, Dictionary<Int32, Double>, Dictionary<Tuple<Int32, Int32, Double>, Int32, Double>, FSharpList<Tuple<Int32, Double, Double, Int32, FSharpOption<Boolean>, FSharpOption<FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>>})**

Executes a set of simulation runs for a spin Hamiltonian.

**ToString**

Returns a string that represents the current object. (Inherited from `Object`.)

**See Also**

Reference

*Microsoft.Research.Liquid Namespace*
## Spin Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spin(DictionaryInt32, Double, DictionaryTupleInt32, Int32, Double)</td>
<td>Creates a spin Hamiltonian for simple systems that only have single spin biases and two-spin couplings. The count of spins is inferred.</td>
</tr>
<tr>
<td>Spin(FSharpListSpinTerm, Int32, FSharpOptionRunMode)</td>
<td>Creates a spin Hamiltonian from a set of spin terms.</td>
</tr>
</tbody>
</table>

### See Also

- Reference
- Spin Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Spin Constructor
(DictionaryInt32, Double,
DictionaryTupleInt32, Int32,
Double)

Creates a spin Hamiltonian for simple systems that only have single spin biases and two-spin couplings. The count of spins is inferred.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
new : 

  hs : Dictionary<int, float> * 
  Js : Dictionary<Tuple<int, int>, float> .
```

### Parameters

**hs**

A Dictionary that maps a qubit id to that qubit's bias strength (Z term coefficient).

**Js**

A Dictionary that maps a pair of qubit id's to the pair's coupling strength (ZZ coefficient).
See Also

Reference
Spin Class
Spin Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Spin Constructor
(FSharpListSpinTerm, Int32, FSharpOptionRunMode)

Creates a spin Hamiltonian from a set of spin terms.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

**Syntax**

```
new :
    _spinTerms : FSharpList<SpinTerm> *
    _numSpins : int *
    _runMode : FSharpOption<RunMode> -> Spin
```

**Parameters**

- **_spinTerms**
  Type: **Microsoft.FSharp.CollectionsFSharpListSpinTerm**
  The list of SpinTerms that together make up the spin Hamiltonian.

- **_numSpins**
  Type: **SystemInt32**
  The number of spins in the system.

- **_runMode**
  Type: **Microsoft.FSharp.CoreFSharpOptionRunMode**
  The Trotterization type to use. This must be one of Trotter1, Trotter1X, or Trotter2.
See Also

Reference
Spin Class
Spin Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Spin Properties

The Spin type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>currentCirc</code></td>
<td>The last (grown) circuit that was run.</td>
</tr>
<tr>
<td><code>decohereModel</code></td>
<td>The decoherence model for this Hamiltonian. (Inherited from Hamiltonian.)</td>
</tr>
<tr>
<td><code>Ket</code></td>
<td>Gets the Ket vector associated with this Hamiltonian (Inherited from Hamiltonian.)</td>
</tr>
<tr>
<td><code>lastAnneal</code></td>
<td>The last set of annealing coefficients that were applied.</td>
</tr>
<tr>
<td><code>lastRawCirc</code></td>
<td>The last (ungrown) circuit that was run.</td>
</tr>
<tr>
<td><code>runMode</code></td>
<td>The run mode. Note that setting this property will force a new circuit to be generated.</td>
</tr>
<tr>
<td><code>time</code></td>
<td>The current simulation time. Note that setting this property will force a new circuit to be generated.</td>
</tr>
<tr>
<td><code>trotterN</code></td>
<td>The Trotter number. Note that</td>
</tr>
</tbody>
</table>
setting this property will force a new circuit to be generated.

See Also

Reference
Spin Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

SpincurrentCirc Property

The last (grown) circuit that was run.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

Syntax

```fsharp
member currentCirc : Circuit with get
```

Property Value

Type: Circuit

See Also

Reference
Spin Class
Microsoft.Research.Liquid Namespace
SpinlastAnneal Property

The last set of annealing coefficients that were applied.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member lastAnneal : float[] with get
```

### Property Value

Type: **Double**

### See Also

- **Reference**
  - Spin Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinlastRawCirc Property

The last (ungrown) circuit that was run.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)&

Syntax

```fsharp
member lastRawCirc : Circuit with get
```

Property Value
Type: Circuit

See Also

Reference
Spin Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinrunMode Property

The run mode. Note that setting this property will force a new circuit to be generated.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1*)

Syntax

```fsharp
member runMode : RunMode with get, set
```

Property Value
Type: RunMode

Remarks

Possibilities are Trotter1 (fully first-order); Trotter1X (does X/2 on both sides, but fully first-order in other terms); and Trotter2 (fully second-order). The initial default value for new Hamiltonians is Trotter1.

See Also

Reference
Spin Class
Microsoft.Research.Liquid Namespace
**Spintime Property**

The current simulation time. Note that setting this property will force a new circuit to be generated.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member time : int with get, set
```

**Property Value**

Type: **Int32**

### See Also

- **Reference**
  - Spin Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

SpintrotterN Property

The Trotter number. Note that setting this property will force a new circuit to be generated.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0*)

### Syntax

```fsharp
member trotterN : int with get, set
```

### Property Value

Type: `Int32`

### See Also

- Reference
- Spin Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Spin Methods

The Spin type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Returns an instance of Spin with the same Hamiltonian parameters as this instance. Simulation parameters such as run time and Trotter number are copied.</td>
</tr>
<tr>
<td>EnergyExpectation</td>
<td>Finds the expectation value of the Hamiltonian. That is, given a state vector $</td>
</tr>
<tr>
<td>Equals</td>
<td>Determin...</td>
</tr>
</tbody>
</table>
whether the specified object is equal to the current object. (Inherited from `Object`.)

Test adiabatic evolution with a ferro-magnetic chain.

Serves as the default hash function. (Inherited from `Object`.)

Gets the type of the current instance. (Inherited from `Object`.)

Peeks at the current Hamiltonian matrix. This is based on the most recently grown circuit.

Prepares the qubit state for a run. All qubits are reset and initialized to the ground state.
Run

Runs the simulation. Is a highly optimized implementation.

Step

Performs a single timestep of simulation.

Test(String, Int32, Int32, FSharpList<Tuple<Int32, Double, Int32, Spin, FSharpOption<Boolean>, FSharpOption<FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>>)), Double, Int32, Spin, FSharpOption<Boolean>, FSharpOption<FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>>)), Double, Int32, Spin, FSharpOption<Boolean>, FSharpOption<FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>>))

Executes a set of simulation runs for a spin Hamiltonian.

Test(String, Int32, Int32, Dictionary<Int32, Double>, Dictionary<Tuple<Int32, Int32, Double, Double, Int32, FSharpOption<Boolean>, FSharpOption<FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>>), Int32, Double, Double, Int32, FSharpOption<Boolean>, FSharpOption<FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>>), Int32, Double, Double, Int32, FSharpOption<Boolean>, FSharpOption<FSharpList<Tuple<FSharpFunc<Qubit, Unit, Double>>>))

Executes a set of simulation runs for a spin Hamiltonian.

ToString

Returns a string that represents the current object. (Inherits from Object.)

See Also

Reference
Spin Class
Microsoft.Research.Liquid Namespace
SpinCopy Method

Returns an instance of Spin with the same Hamiltonian parameters as this instance. Simulation parameters such as run time and Trotter number are not copied.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member Copy : unit -> Spin
```

### Return Value

Type: Spin  
Copy of this instance

### See Also

Reference  
Spin Class  
Microsoft.Research.Liquid Namespace
SpinEnergyExpectation Method

Finds the expectation value of the Hamiltonian. That is, given a state vector $|\psi\rangle$, this method computes $<\psi|H|\psi\rangle$.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member EnergyExpectation :  
  stdev : bool *  
  anneal : FSharpOption<float[]> *  
  qubits : FSharpOption<FSharpList<Qubit>>
```

### Parameters

- **stdev**
  - Type: System.Boolean
  - Whether or not to evaluate the standard deviation as well.

- **anneal**
  - Type: Microsoft.FSharp.Core.FSharpOption<Double>
  - An optional array of annealing values to use. The default is to use the most recent annealing values.

- **qubits**
  - Type: Microsoft.FSharp.Core.FSharpOption<FSharpList<Qubit>>
  - An optional state vector to take the expectation value against. The default is to use this.Ket.Qubits; that is, the qubits from the current Ket vector.

### Return Value

- Type: Tuple<Double, Double>
A tuple of the expectation value and standard deviation. The standard deviation will be 0.0 if it was not computed.

See Also

Reference
Spin Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinFerro Method

Test adiabatic evolution with a ferro-magnetic chain

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
static member Ferro :  
    repeat : int * 
    sCnt : int * 
    h0 : float * 
    hn : float * 
    J : float * 
    gammalambda : FSharpOption<FSharpList<Tuple>>(
    runonce : FSharpOption<bool> * 
    decohereModel : FSharpOption<FSharpList<
```

### Parameters

**repeat**  
Type: SystemInt32  
How many tests to run

**sCnt**  
Type: SystemInt32  
How many qubits

**h0**  
Type: SystemDouble  
-1,0,1 = left most down, random, up

**hn**  
Type: SystemDouble  
-1,0,1 = right most down, random, up
\( J \)

Type: `System.Double`
coupling: 1=ferro -1=anti 0=none

\( \gamma_\lambda \)

Type: `Microsoft.FSharp.Core.FSharpOption<FSharpList<Tuple<int, double, double>>>`
List of (time,\( \gamma \),\( \lambda \)) tuples that define the annealing schedule (optional=[(30,0.0,1.0)])

\( runonce \)

Type: `Microsoft.FSharp.Core.FSharpOption<bool>`
true=run the simulation once and perform repeated 'virtual measurements'; false=run the simulation and measure each time (optional=false)

\( decohereModel \)

Type: `Microsoft.FSharp.Core.FSharpOption<FSharpList<Tuple<FSharpFunc<Unit, double>, double>>>`
Decoherence probability per qubit per timestep [(gate,prob) list] optional=None

See Also

Reference
Spin Class
Microsoft.Research.Liquid Namespace
SpinPeek Method

Peeks at the current Hamiltonian matrix. This is based on the most recently grown circuit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.5)

### Syntax

```fsharp
member Peek :  
anneal : FSharpOption<float[]> -> CSMat
```

**Parameters**

**anneal**

Type: `Microsoft.FSharp.Core.FSharpOption<Double>`
An optional array of annealing values to use. The default is to use the most recent annealing values.

**Return Value**

Type: `CSMat`
A sparse matrix representation of the Hamiltonian

### See Also

**Reference**  
Spin Class  
Microsoft.Research.Liquid Namespace
SpinPrep Method

Prepares the qubit state for a run. All qubits are resets and initialized to the ground state.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
member Prep : unit -> unit
```

See Also

Reference
Spin Class
Microsoft.Research.Liquid Namespace
SpinRun Method

Runs the simulation. This is a highly optimized implementation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member Run : dt : int * schedule : FSharpList<Tuple<int, float[]>> * gp : FSharpOption<GrowPars> -> int
```

### Parameters

**dt**

Type: `System.Int32`

The number of timesteps to run.

**schedule**

Type: `Microsoft.FSharp.Collections.FSharpList<Tuple<int, float[]>>`

The annealing schedule to use. Each entry in the list is a tuple whose first entry is a time step and whose second entry is an array of annealing values. Annealing values for time steps in between entries are computed by linearly interpolating between those for the previous and next entries. The list must be in ascending order by time step.

**gp**

Type: `Microsoft.FSharp.Core.FSharpOption<GrowPars>`

Optional grow parameters for the resulting circuit. The default is GrowGates with maxWires of 11. See `GrowPars` for more
Return Value
Type: Int32
The number of decoherence events which occurred during the step. Note that this will be zero unless a decoherence model has been set on this instance.

See Also
Reference
Spin Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinStep Method

Performs a single timestep of simulation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll)  
**Version:** 1.0.5981.24943 (1.0*)

### Syntax

```
F#

member Step :
  angles : float[] *
  gp : FSharpOption<GrowPars> -> int
```

### Parameters

**angles**

Type: System.Double  
The vector of annealing coefficients to be used in this step.

**gp**

Type: Microsoft.FSharp.Core.FSharpOption<GrowPars>  
Optional grow parameters for the resulting circuit. The default is GrowGates with maxWires of 11. See GrowPars for more details.

### Return Value

Type: Int32  
The number of decoherence events which occurred during the step.  
Note that this will be zero unless a decoherence model has been set on this instance.

### See Also

Reference
# Spin Test Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

## See Also

- **Reference**
  - Spin Class
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

SpinTest Method (String, Int32, Int32, FSharpList<Tuple<Int32, Double, Int32, Spin>, FSharpOption<Boolean>, FSharpOption<FSharpList<Tuple<FSharpFunc<Unit, Double>, Int32>*)

Executes a set of simulation runs for a spin Hamiltonian.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
static member Test :
    tag : string *
    repeats : int *
    trotter : int *
    schedule : FSharpList<Tuple<int, float[]>> *
    res : int *
    spin : Spin *
    runonce : FSharpOption<bool> *
    decohereModel : FSharpOption<FSharpList<]]
```

Parameters

*tag*
Type: System.String
The output label for logging.

*repeats*
Type: `SystemInt32`
The number of simulations to run.

`trotter`
Type: `SystemInt32`
The Trotter number to use.

`schedule`
Type: `Microsoft.FSharp.Collections.FSharpList<Tuple<int32, double>>`
The annealing schedule to use. Each entry in the list is a tuple whose first entry is a time step and whose second entry is an array of annealing values. Annealing values for time steps in between entries are computed by linearly interpolating between those for the previous and next entries. The list must be in ascending order by time step.

`res`
Type: `SystemInt32`
The resolution of the simulation, in time steps. Larger values may increase speed but will reduce the granularity of output. Note that this must evenly divide the final time in the annealing schedule.

`spin`
Type: `Microsoft.Research.LiquidSpin`
The actual Hamiltonian to be simulated.

`runonce`
Type: `Microsoft.FSharp.Core.FSharpOption<bool>`
An option to only run the simulation once and perform repeated 'virtual measurements'. The default is false, which means to run the simulation and measure each time.

`decohereModel`
Type: `Microsoft.FSharp.Core.FSharpOption<FSharpList<Tuple<Microsoft.FSharp.Core.Unit, double>>>`
An optional decoherence model to use for this simulation. See `Hamiltonian.decohereModel` for more information. The default is no decoherence.

See Also
Reference
Spin Class
Test Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinTest Method (String, Int32, Int32, Dictionary[Int32, Double], Dictionary[Tuple[Int32, Int32], Double], Int32, Dictionary[Tuple[Int32, Double], FSharpList<Tuple[Int32, Double], FSharpOption:Boolean>, FSharpList<Tuple[Int32, Double], FSharpOption:Boolean>, FSharpList<Tuple[Int32, Double], FSharpOption:Boolean>))

Executes a set of simulation runs for a spin Hamiltonian.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
<table>
<thead>
<tr>
<th>tag</th>
<th>repeats</th>
<th>trotter</th>
<th>hs</th>
<th>Js</th>
<th>gammalambda</th>
<th>res</th>
<th>runonce</th>
<th>decohereModel</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>int</td>
<td>int</td>
<td>Dictionary&lt;int, float&gt;</td>
<td>Dictionary&lt;Tuple&lt;int, int&gt;, float&gt;</td>
<td>FSharpList&lt;Tuple&lt;int, float&gt;&gt;</td>
<td>int</td>
<td>FSharpOption&lt;bool&gt;</td>
<td>FSharpList&lt;int&gt;</td>
</tr>
</tbody>
</table>
```

### Parameters

- **tag**
Type: **System.String**
The output label for logging.

**repeats**
Type: **System.Int32**
The number of simulations to run.

**trotter**
Type: **System.Int32**
The Trotter number to use.

**hs**
Type: **System.Collections.Generic.Dictionary<Int32, Double>**
A Dictionary that maps a qubit id to that qubit's bias strength (Z term coefficient).

**Js**
Type: **System.Collections.Generic.Dictionary<Tuple<Int32, Int32>, Double>**
A Dictionary that maps a pair of qubit id's to the pair's coupling strength (ZZ coefficient).

**gammalambda**
Type: **Microsoft.FSharp.Collections.FSharpList<Tuple<Int32, Double, Double>>**
List of (time,gamma,lambda) tuples that define the annealing schedule. Each entry in the list is a tuple whose first entry is a time step and whose second entry is an array of annealing values. Annealing values for time steps in between entries are computed by linearly interpolating between those for the previous and next entries. The list must be in ascending order by time step.

**res**
Type: **System.Int32**
The resolution of the simulation, in time steps. Larger values may increase speed but will reduce the granularity of output. Note that this must evenly divide the final time in the annealing schedule.

**runonce**
Type: **Microsoft.FSharp.Core.FSharpOption<Boolean>**
An option to only run the simulation once and perform repeated 'virtual measurements'. The default is false, which means to run
the simulation and measure each time.

**decohereModel**
Type: **Microsoft.FSharp.Core**
**FSharpOption**
**FSharpList**
**Tuple**
**FSharpFunc**
**Unit**, **Double**

An optional decoherence model to use for this simulation. See **Hamiltonian.decohereModel** for more information. The default is no decoherence.

**See Also**

Reference
*Spin Class*
*Test Overload*
*Microsoft.Research.Liquid Namespace*

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinTerm Class

A single term in a Spin Hamiltonian.

Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidSpinTerm

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

Syntax

```
[<SerializableAttribute>]
type SpinTerm = class end
```

The SpinTerm type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="SpinTerm(Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Double)" /></td>
<td>Initializes a new SpinTerm instance.</td>
</tr>
<tr>
<td><img src="image" alt="SpinTerm(Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, FSharpListInt32, Double)" /></td>
<td>Initializes a new SpinTerm instance.</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amplitude</td>
<td>The numerical coefficient ('strength') leading this term.</td>
</tr>
<tr>
<td>Schedule</td>
<td>The numerical id (zero-based) of the annealing schedule corresponding to this term.</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
See Also

Reference

Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinTerm Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpinTerm(Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Double)</td>
<td>Initializes a new SpinTerm instance.</td>
</tr>
<tr>
<td>SpinTerm(Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, FSharpListInt32, Double)</td>
<td>Initializes a new SpinTerm instance.</td>
</tr>
</tbody>
</table>

See Also

Reference

SpinTerm Class

Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinTerm Constructor (Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Double)

Initializes a new SpinTerm instance.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

**Syntax**

```fsharp
new :  
    s : int *  
    o : FSharpFunc<float, FSharpFunc<FSharpList<int, Unit>>>  
    a : float -> SpinTerm
```

**Parameters**

- **s**  
  Type: **SystemInt32**  
  The numerical id, zero-based, of the annealing schedule corresponding to this term.

- **o**  
  Type: **Microsoft.FSharp.Core.FSharpFunc<Double, FSharpFuncFSharpListQubit, Unit**  
  A function which performs the operation of this term over a given angle.

- **a**
Type: `SystemDouble`
The numerical coefficient ('strength') leading this term.

See Also

Reference
SpinTerm Class
SpinTerm Overload
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

SpinTerm Constructor (Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, FSharpListInt32, Double)

Initializes a new SpinTerm instance.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
new : s : int * o : FSharpFunc<float, FSharpFunc<FSharpList<int> * idx : FSharpList<int> * a : float -> SpinTerm
```

Parameters

s
Type: SystemInt32
The numerical id, zero-based, of the annealing schedule corresponding to this term.

o
Type: Microsoft.FSharp.Core.FSharpFunc<Double, FSharpFunc<FSharpList<Qubit>, Unit
A function which performs the operation of this term over a given angle.
**idx**

Type: `Microsoft.FSharp.Collections.FSharpList`Int32
A list of integer indices specifying the qubits to apply this term to.

**a**

Type: `System.Double`
The numerical coefficient ('strength') leading this term.

### See Also

- **Reference**
  - SpinTerm Class
  - SpinTerm Overload
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinTerm Properties

The SpinTerm type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amplitude</td>
<td>The numerical coefficient ('strength') leading this term.</td>
</tr>
<tr>
<td>Schedule</td>
<td>The numerical id (zero-based) of the annealing schedule corresponding to this term.</td>
</tr>
</tbody>
</table>

See Also

Reference
SpinTerm Class
Microsoft.Research.Liquid Namespace
SpinTermAmplitude Property

The numerical coefficient ('strength') leading this term.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```
member Amplitude : float with get
```

**Property Value**

Type: **Double**

### See Also

**Reference**

SpinTerm Class
Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
SpinTermSchedule Property

The numerical id (zero-based) of the annealing schedule corresponding to this term.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
member Schedule : int with get
```

**Property Value**

Type: **Int32**

### See Also

**Reference**

SpinTerm Class  
Microsoft.Research.Liquid Namespace
SpinTerm Methods

The SpinTerm type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>

See Also

Reference
SpinTerm Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Stabilizer Class


Inheritance Hierarchy

- System
  - Object
  - Microsoft.Research.Liquid
  - Stabilizer

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

```fsharp
[<SerializableAttribute>]
type Stabilizer = class end
```

The Stabilizer type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🦄 Stabilizer</td>
<td>Creates a tableau to run a circuit in the stabilizer simulator.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📔 Item</td>
<td>The current measured value for a qubit.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Gaussian</td>
<td>Performs a Gaussian elimination to put the tableau in quasi upper triangular form.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Run</td>
<td>Runs the circuit.</td>
</tr>
<tr>
<td>ShowState</td>
<td>Dumps the stabilizer tableau.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
The Language-Integrated Quantum Operations (LIQU|>) Simulator

Stabilizer Constructor

Creates a tableau to run a circuit in the stabilizer simulator.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
new :
    circ : Circuit *
    ket : Ket -> Stabilizer
```

### Parameters

- **circ**
  
  Type: Microsoft.Research.LiquidCircuit
  
  The circuit that will be run in stabilizer mode.

- **ket**
  
  Type: Microsoft.Research.LiquidKet
  
  The state vector that will be used for execution.

### See Also

- **Reference**
  
  Stabilizer Class
  
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>|) Simulator

Stabilizer Properties

The `Stabilizer` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>The current measured value for a qubit.</td>
</tr>
</tbody>
</table>

## See Also

Reference
- `Stabilizer` Class
- `Microsoft.Research.Liquid` Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

StabilizerItem Property

The current measured value for a qubit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member Item : Tuple<bool, Bit> with get
```

### Parameters

- **i**
  - Type: System.Int32
  - The index of the qubit in the simulator's state vector.

### Property Value

Type: TupleBoolean, Bit

### Remarks

This property has a tuple value. The first item is a flag indicating whether the last measurement was random or forced, and the second item is the result of the last measurement as a Bit. As with any qubit, the Bit will be Unknown if the qubit has interacted since it was last measured.

### See Also

- Reference
  - Stabilizer Class
  - Microsoft.Research.Liquid Namespace
**Stabilizer Methods**

The **Stabilizer** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Gaussian</strong></td>
<td>Performs a Gaussian elimination to put the tableau in quasi upper triangular form.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Run</strong></td>
<td>Runs the circuit.</td>
</tr>
<tr>
<td><strong>ShowState</strong></td>
<td>Dumps the stabilizer tableau.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>
StabilizerGaussian Method

Performs a Gaussian elimination to put the tableau in quasi upper triangular form.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```
member Gaussian : unit -> unit
```

### See Also

Reference  
Stabilizer Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
StabilizerRun Method

Runs the circuit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
member Run :  
    circuit : FSharpOption<Circuit>  *  
    dumpAll : FSharpOption<bool>  *  
    gaussian : FSharpOption<bool>  ->  unit
```

### Parameters

#### circuit

Type: `Microsoft.FSharp.Core.FSharpOption<Circuit>`  
An optional different circuit to run. This allows the tableau to be reused for different circuits. The default is to used the circuit the tableau was created with.

#### dumpAll

Type: `Microsoft.FSharp.Core.FSharpOption<bool>`  
An option indicating that the tableau state should be written to the log after each gate application. The default is false, indicating not to log.

#### gaussian

Type: `Microsoft.FSharp.Core.FSharpOption<bool>`  
An option indicating that the tableau state should be simplified using Gaussian elimination before being logged. This is ignored if `dumpAll` is false. The default is false, indicating not to perform Gaussian elimination.
See Also

Reference
Stabilizer Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
**StabilizerShowState Method**

Dumps the stabilizer tableau.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```
F#

member ShowState :  
    f : FSharpFunc<int, FSharpFunc<string, Unit>>,  
    level : int -> unit
```

### Parameters

- **f**  
  Type: Microsoft.FSharp.Core.FSharpFunc<int, FSharpFunc<string, Unit>>  
  The output function to use. A common output function is showLogInd.

- **level**  
  Type: System.Int32  
  The indentation level.

### See Also

- **Reference**  
  Stabilizer Class  
  Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Steane7 Class

Implementation of a Steane 7-bit quantum error correcting code, [[7,1,3]], based on the QECC class.

Inheritance Hierarchy

```
System
  Object
  Microsoft.Research.Liquid
    QECC
    Microsoft.Research.Liquid
      Steane7
```

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

Syntax

```
[<SerializableAttribute>]
type Steane7 =
  class
    inherit QECC
  end
```

The Steane7 type exposes the following members.

Constructors

```
Name      Description
---------  ---------------------------------------
Steane7    Constructs a Steane code implementation for a circuit.
```

Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit</td>
<td>The error-correcting circuit built by <code>Compile</code>. The circuit will be built now if it hasn't already been. (Inherited from <code>QECC</code>.)</td>
</tr>
<tr>
<td>Ket</td>
<td>The state vector for the compiled code. (Inherited from <code>QECC</code>.)</td>
</tr>
<tr>
<td>NumFixed</td>
<td>The number of syndrome fixups performed.</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compile</td>
<td>Compiles the target circuit into an error-correcting version. (Inherited from <code>QECC</code>.)</td>
</tr>
<tr>
<td>Decode</td>
<td>Decodes a set of measured physical qubits to get the measured value for a logical qubit. (Overrides <code>QECCDecode(FSharpListQubit)</code>.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetMeasured</td>
<td>Gets the measured values of the</td>
</tr>
</tbody>
</table>
physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit. (Inherited from QECC.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Inject</strong></td>
<td>Injects dephasing errors with the given probability into the error-correcting circuit. (Inherited from QECC.)</td>
</tr>
<tr>
<td><strong>Log2Phys</strong></td>
<td>Gets the physical qubits that make up a logical qubit. (Inherited from QECC.)</td>
</tr>
<tr>
<td><strong>Prep</strong></td>
<td>A gate function that prepares a logical</td>
</tr>
<tr>
<td><strong>Replace</strong></td>
<td>Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit. (Inherited from QECC.)</td>
</tr>
<tr>
<td><strong>Syndrome</strong></td>
<td>A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method. (Overrides</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Test1</td>
<td>Tests the Steane7 QECC on a single logical qubit with forced X, Y, and Z error injections.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

**Remarks**

This code uses 7 physical qubits per logical qubit. It also requires 6 ancillae.

**See Also**

Reference

Microsoft.Research.Liquid Namespace
Microsoft.Research.LiquidQECC
Steane7 Constructor

Constructs a Steane code implementation for a circuit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

### Syntax

```fsharp
new : tgt : Circuit -> Steane7
```

### Parameters

- **tgt**
  - Type: Microsoft.Research.Liquid.Circuit
  - The target Circuit to build an error-correcting circuit for.

### See Also

- Reference  
  - Steane7 Class  
  - Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Steane7 Properties

The Steane7 type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit</td>
<td>The error-correcting circuit built by Compile. The circuit will be built now if it hasn't already been. (Inherited from QECC.)</td>
</tr>
<tr>
<td>Ket</td>
<td>The state vector for the compiled code. (Inherited from QECC.)</td>
</tr>
<tr>
<td>NumFixed</td>
<td>The number of syndrome fixups performed.</td>
</tr>
</tbody>
</table>

See Also

Reference
Steane7 Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Steane7NumFixed Property

The number of syndrome fixups performed.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member NumFixed : int with get, set
```

### Property Value

Type: Int32

### See Also

**Reference**  
Steane7 Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQui|>) Simulator

Steane7 Methods

The Steane7 type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compile</td>
<td>Compiles the target circuit into an error-correcting version. (Inherited from QECC.)</td>
</tr>
<tr>
<td>Decode</td>
<td>Decodes a set of measured physical qubits to get the measured value for a logical qubit. (Overrides QECCDecode(FSharpListQubit).)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetMeasured</td>
<td>Gets the measured values of the physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit. (Inherited from QECC.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Inject</td>
<td>Injects dephasing errors with the given probability into the error-correcting circuit. (Inherited from QECC.)</td>
</tr>
<tr>
<td>Log2Phys</td>
<td>Gets the physical qubits that make up a logical qubit. (Inherited from QECC.)</td>
</tr>
<tr>
<td>Prep</td>
<td>A gate function that prepares a logical</td>
</tr>
<tr>
<td>Replace</td>
<td>Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit. (Inherited from QECC.)</td>
</tr>
<tr>
<td>Syndrome</td>
<td>A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method. (Overrides QECCSyndrome(FSharpListQubit).)</td>
</tr>
<tr>
<td>Test1</td>
<td>Tests the Steane7 QECC on a single logical qubit with forced X, Y, and Z error injections.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
See Also

Reference
Steane7 Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Steane7Decode Method

Decodes a set of measured physical qubits to get the measured value for a logical qubit.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
abstract Decode : qs : FSharpList<Qubit> -> Tuple<Bit, int>
override Decode : qs : FSharpList<Qubit> -> Tuple<Bit, int>
```

### Parameters

**qs**
- **Type:** Microsoft.FSharp.Collections.FSharpList<Qubit>
- The physical qubits to decode. They must already have been measured.

### Return Value
- **Type:** Tuple<Bit, Int32>
- A tuple containing the logical measured value of the logical qubit, either Zero or One, and the Hamming distance from the physical state to the code space.

### See Also

- Reference: Steane7 Class
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Steane7 Prep Method

A gate function that prepares a logical $|0\rangle$ qubit. This gets compiled into the error-correcting circuit by the Compile method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
abstract Prep :  
    qs : FSharpList<Qubit>  ->  unit
override Prep :  
    qs : FSharpList<Qubit>  ->  unit
```

### Parameters

**qs**

Type: `Microsoft.FSharp.Collections.FSharpList<Qubit>`

The physical qubits for the logical qubit. This contains both data and syndrome qubits, but no ancillae.

### See Also

**Reference**

Steane7 Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Steane7Syndrome Method

A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

## Syntax

```fsharp
abstract Syndrome :  
   qs : FSharpList<Qubit>  ->  unit
override Syndrome :  
   qs : FSharpList<Qubit>  ->  unit
```

### Parameters

- **qs**
  Type: Microsoft.FSharp.Collections.FSharpList Qubit  
The ancilla qubits followed by physical qubits for the logical qubit. The number of ancillae was passed to the QECC constructor.

## See Also

- **Reference**  
  Steane7 Class  
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Steane7Test1 Method

Tests the Steane7 QECC on a single logical qubit with forced X, Y, and Z error injections.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.~)

### Syntax

```
static member Test1 : unit -> unit
```

### Remarks

The test verifies that the Steane 7 code properly decodes a prepared logical qubit in the face of all possible single-qubit dephasing errors on each of the 7 physical qubits.

### See Also

- Reference
- Steane7 Class  
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests Class

A collection of sample Liquid simulations and tests, plus some utility routines to make it easier to write new samples.

▲ Inheritance Hierarchy

SystemObject → Microsoft.Research.LiquidTests

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

▲ Syntax

F#

```fsharp
[<AbstractClassAttribute>]
[<SealedAttribute>]
type Tests = class end
```

The Tests type exposes the following members.

▲ Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>![ ] __Big</td>
</tr>
<tr>
<td>![ ]</td>
<td>![ ] __Chem</td>
</tr>
</tbody>
</table>
Trotter, 28 bits of accuracy, and no additional options. See the Users Manual for more information.

<table>
<thead>
<tr>
<th></th>
<th>__ChemFull</th>
<th>Runs a quantum chemistry simulation. See the Users Manual for more information.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>__Correct</td>
<td>Test various permutations for correctness using teleport</td>
</tr>
<tr>
<td></td>
<td>__EIGS</td>
<td>Validates that LAPACK is properly installed.</td>
</tr>
<tr>
<td></td>
<td>__Entangle1</td>
<td>Runs a simple gate sequence that entangles all the qubits in the state vector.</td>
</tr>
<tr>
<td></td>
<td>__Entangle2</td>
<td>Runs a simple gate sequence that entangles all the qubits in the state vector. The sequence is run three different ways to demonstrate the difference in timings.</td>
</tr>
<tr>
<td></td>
<td>__Entangles</td>
<td>Runs 100 entanglement tests on 16 qubits to show the statistics on the bits measured.</td>
</tr>
<tr>
<td></td>
<td>__EntEnt</td>
<td>Demonstrates a couple of simple entanglement entropy calculations.</td>
</tr>
<tr>
<td></td>
<td>__EPR</td>
<td>Renders a small EPR circuit that entangles two qubits.</td>
</tr>
<tr>
<td></td>
<td>__Ferro</td>
<td>Simulates a ferromagnetic chain using a first-quantized Hamiltonian.</td>
</tr>
</tbody>
</table>
JointCNOT  Demonstrates and tests various ways of implementing CNOT from joint measurements and single-qubit Clifford gates.

Kraus  Show how to add noise to Teleport with Kraus operators

Noise1  Demonstrates the use of a complex noise model with error correction. See the Users Manual for more information.

NoiseAmp  Demonstrates a complex noise model. See the Users Manual for more information.

QECC  Demonstrates some examples of error correction and stabilizer simulation. See the Users Manual for details.

QFTbench  Benchmarks the Quantum Fourier Transform at the heart of the Shor algorithm.

QLSA  Demonstrates the Quantum Linear Algebra algorithm from Harrow, Hassidim, and Lloyd. See the Users Manual for more information.

QuAM  Demonstrates the Quantum Associative Memory algorithm from Ventura and Martinez. See the Users Manual for more information.
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>![logo]</td>
<td>__QWalk</td>
</tr>
<tr>
<td>![logo]</td>
<td>__Ramsey33</td>
</tr>
<tr>
<td>![logo]</td>
<td>__SG</td>
</tr>
<tr>
<td>![logo]</td>
<td>__Shor</td>
</tr>
<tr>
<td>![logo]</td>
<td>__Steane7</td>
</tr>
<tr>
<td>![logo]</td>
<td>__Teleport</td>
</tr>
<tr>
<td>![logo]</td>
<td>__TSP</td>
</tr>
<tr>
<td>![logo]</td>
<td>RenderTest</td>
</tr>
</tbody>
</table>

Top

**See Also**

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Tests Methods

The Tests type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__Big</td>
<td>Runs an entanglement test for state vectors from 16 to 22 qubits in size.</td>
</tr>
<tr>
<td>__Chem</td>
<td>Runs a simple quantum chemistry simulation. This is the same as the __ChemFull sample, with common values for the detailed parameters: test 0, 32 Trotter steps, first-order Trotter, 28 bits of accuracy, and no additional options. See the Users Manual for more information.</td>
</tr>
<tr>
<td>__ChemFull</td>
<td>Runs a quantum chemistry simulation. See the Users Manual for more information.</td>
</tr>
<tr>
<td>__Correct</td>
<td>Test various permutations for correctness using teleport</td>
</tr>
<tr>
<td>__EIGS</td>
<td>Validates that LAPACK is properly installed.</td>
</tr>
<tr>
<td>__Entangle1</td>
<td>Runs a simple gate sequence that entangles all the qubits in the state vector.</td>
</tr>
<tr>
<td></td>
<td><strong>__Entangle2</strong></td>
</tr>
<tr>
<td>-----</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td><strong>__Entangles</strong></td>
</tr>
<tr>
<td></td>
<td><strong>__EntEnt</strong></td>
</tr>
<tr>
<td></td>
<td><strong>__EPR</strong></td>
</tr>
<tr>
<td></td>
<td><strong>__Ferro</strong></td>
</tr>
<tr>
<td></td>
<td><strong>__JointCNOT</strong></td>
</tr>
<tr>
<td></td>
<td><strong>__Kraus</strong></td>
</tr>
<tr>
<td></td>
<td><strong>__Noise1</strong></td>
</tr>
<tr>
<td></td>
<td><strong>__NoiseAmp</strong></td>
</tr>
<tr>
<td>Tool</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>__QECC</td>
<td>Demonstrates some examples of error correction and stabilizer simulation. See the Users Manual for details.</td>
</tr>
<tr>
<td>__QFTbench</td>
<td>Benchmarks the Quantum Fourier Transform at the heart of the Shor algorithm.</td>
</tr>
<tr>
<td>__QLSA</td>
<td>Demonstrates the Quantum Linear Algebra algorithm from Harrow, Hassidim, and Lloyd. See the Users Manual for more information.</td>
</tr>
<tr>
<td>__QuAM</td>
<td>Demonstrates the Quantum Associative Memory algorithm from Ventura and Martinez. See the Users Manual for more information.</td>
</tr>
<tr>
<td>__QWalk</td>
<td>Demonstrates the Quantum PageRank algorithm from Paparo and Martin-Delgado. See the Users Manual for more information.</td>
</tr>
<tr>
<td>__Ramsey33</td>
<td>Solves for the (3,3) Ramsey number. See this paper for details.</td>
</tr>
<tr>
<td>__SG</td>
<td>Simulates a spin glass using a first-quantized Hamiltonian.</td>
</tr>
<tr>
<td>__Shor</td>
<td>Runs the classic Shor factoring algorithm.</td>
</tr>
<tr>
<td>__Steane7</td>
<td>Validates that the Steane 7 code is correct. See Steane7.Test1 for</td>
</tr>
</tbody>
</table>
more details.

| Teleport | Renders and runs the classic quantum teleportation algorithm. |
| TSP | Solves the traveling salesman problem. |
| RenderTest | Renders and dumps test circuits in a variety of formats and detail levels. |

See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Tests__Big Method

Runs an entanglement test for state vectors from 16 to 22 qubits in size.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member __Big : unit -> unit
```

### See Also

Reference

Tests Class
Microsoft.Research.Liquid Namespace
Tests___Chem Method

Runs a simple quantum chemistry simulation. This is the same as the __ChemFull sample, with common values for the detailed parameters: test 0, 32 Trotter steps, first-order Trotter, 28 bits of accuracy, and no additional options. See the Users Manual for more information.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member __Chem : string -> unit
```

### Parameters

`mol`

Type: **System.String**

The name of the molecule to simulate. Entering an empty string, "", will display the list of available molecules.

### See Also

**Reference**

Tests Class
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests — ChemFull Method

Runs a quantum chemistry simulation. See the Users Manual for more information.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

### Syntax

```f#
static member __ChemFull :  
mol : string *  
test : int *  
opts : string *  
trot : int *  
bits : int *  
order : int -> unit
```

### Parameters

- **mol**  
  Type: **SystemString**  
  The name of the molecule to simulate. Entering an empty string, """, will display the list of available molecules.

- **test**  
  Type: **SystemInt32**  
  The test number to run, from the .dat file.

- **opts**  
  Type: **SystemString**  
  A string of any options you want to set. See the Users Manual for more information.

- **trot**  
  Type: **SystemInt32**
The Trotter parameter; that is, number of Trotter steps per time step.

**bits**
Type: `SystemInt32`
The number of bits of accuracy desired in the phase estimation.

**order**
Type: `SystemInt32`
The order of Trotter approximation to use, 1 or 2.

## See Also

**Reference**
Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__Correct Method

Test various permutations for correctness using teleport

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

▲ Syntax

```fsharp
static member __Correct : unit -> unit
```

▲ See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Tests__EIGS Method

Validates that LAPACK is properly installed.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

static member __EIGS : unit -> unit

See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace
Tests__Entangle1 Method

Runs a simple gate sequence that entangles all the qubits in the state vector.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member __Entangle1 : entSiz : int -> unit
```

### Parameters

- **entSiz**
  - Type: System.Int32  
  - The number of qubits desired in the state vector.

### Remarks

This sample displays detailed timing and memory usage information, and so allows you to see how Liquid scales as the state vector size grows.

### See Also

- Reference
  - Tests Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__Entangle2 Method

Runs a simple gate sequence that entangles all the qubits in the state vector. The sequence is run three different ways to demonstrate the difference in timings.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.2)

![Syntax](https://via.placeholder.com/150)

```fsharp
static member __Entangle2 : entSiz : int -> unit
```

Parameters

- **entSiz**
  - Type: SystemInt32
  - The number of qubits desired in the state vector.

Remarks

This test executes the same gate sequence in three different ways:
- As a simple sequence of gates. This is essentially the same as the __Entangle1 sample.
- As a compiled circuit.
- As a compiled circuit that has been optimized by calling GrowGates.

See Also

- Reference
  - Tests Class
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__Entangles Method

Runs 100 entanglement tests on 16 qubits to show the statistics on the bits measured.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member __Entangles : unit -> unit
```

### See Also

Reference

Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__EntEnt Method

Demonstrates a couple of simple entanglement entropy calculations.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.1)

Syntax

```
static member __EntEnt : unit -> unit
```

See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__EPR Method

Renders a small EPR circuit that entangles two qubits.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▶ Syntax

```fsharp
static member __EPR : unit -> unit
```

▶ See Also

Reference  
Tests Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__Ferro Method

Simulates a ferromagnetic chain using a first-quantized Hamiltonian.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

F#

```fsharp
static member __Ferro :  
  full : bool *  
  runonce : bool -> unit
```

### Parameters

**full**

Type: `SystemBoolean`  
Whether to run all chain variations (isolated, ferromagnetic, anti-ferromagnetic, freeze up, freese down, and freeze up/down) or just the last variation, freeze up/down.

**runonce**

Type: `SystemBoolean`  
For each variation, whether to run the circuit once and then do "virtual sampling" from the state vector, or to run the circuit and perform full simulated measurements each time.

### See Also

**Reference**  
Tests Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Tests__JointCNOT Method

Demonstrates and tests various ways of implementing CNOT from joint measurements and single-qubit Clifford gates.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

```
static member __JointCNOT : unit -> unit
```

See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__Kraus Method

Show how to add noise to Teleport with Kraus operators

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```
F#
static member __Kraus :
    nRuns : int *
    probAD : float *
    probDP : float *
    verbose : bool -> unit
```

Parameters

nRuns
Type: SystemInt32
How many runs to gather statistics over

probAD
Type: SystemDouble
Probability of Amplitude Damping on any single qubit

probDP
Type: SystemDouble
Probability of Depolarizing noise on any single qubit

verbose
Type: SystemBoolean
Output detailed stats and drawings

See Also

Reference
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__Noise1 Method

Demonstrates the use of a complex noise model with error correction. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

```fsharp
static member __Noise1 :
    depth : int *
    iters : int *
    prob : float -> unit
```

Parameters

- **depth**
  - Type: SystemInt32
  - The number of idle gates to include in the circuit.

- **iters**
  - Type: SystemInt32
  - The number of executions to run, for statistical purposes.

- **prob**
  - Type: SystemDouble
  - The probability of an error occurring.

See Also

Reference

Tests Class
Microsoft.Research.Liquid Namespace
Tests__NoiseAmp Method

Demonstrates a complex noise model. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

```fsharp
static member __NoiseAmp : unit -> unit
```

See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Tests__QECC Method

Demonstrates some examples of error correction and stabilizer simulation. See the Users Manual for details.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▽ **Syntax**

```fsharp
static member __QECC : unit -> unit
```

▽ **See Also**

Reference  
Tests Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__QFTbench Method

Benchmarks the Quantum Fourier Transform at the heart of the Shor algorithm.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member __QFTbench : unit -> unit
```

### See Also

**Reference**  
Tests Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Tests__QLSA Method

Demonstrates the Quantum Linear Algebra algorithm from Harrow, Hassidim, and Lloyd. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid  
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
F#

static member __QLSA : unit -> unit
```

See Also

Reference
Tests Class  
Microsoft.Research.Liquid Namespace
Tests__QuAM Method

Demonstrates the Quantum Associative Memory algorithm from Ventura and Martinez. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

```
static member __QuAM : unit -> unit
```

See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace
Tests__QWalk Method

Demonstrates the Quantum PageRank algorithm from Paparo and Martin-Delgado. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

⚠️ Syntax

```fsharp
static member __QWalk : string -> unit
```

Parameters

`which`

Type: System.String

The web graph to use. There are three built-in graphs: tiny, tree, and graph. Alternatively, the path to a .graph file may be provided.

⚠️ See Also

Reference

Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
**Tests__Ramsey33 Method**

Solves for the (3,3) Ramsey number. See [this paper](#) for details.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```
static member __Ramsey33 : unit -> unit
```

### See Also

- Reference  
  Tests Class  
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Tests__SG Method

Simulates a spin glass using a first-quantized Hamiltonian.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
static member __SG : unit -> unit
```

### See Also

- Reference
  - Tests Class
  - Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Tests__Shor Method

Runs the classic Shor factoring algorithm.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```fsharp
static member __Shor : 
    N : int * 
    doCirc : bool -> unit
```

Parameters

N
Type: System.Int32
The number to factor.

doCirc
Type: System.Boolean
Whether or not to optimize the circuit.

See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Tests__Steane7 Method

Validates that the Steane 7 code is correct. See Steane7.Test1 for more details.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

Syntax

F#

```fsharp
static member __Steane7 : unit -> unit
```

See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__Teleport Method

Renders and runs the classic quantum teleportation algorithm.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

Syntax

```
static member __Teleport : unit -> unit
```

See Also

Reference
Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests__TSP Method

Solves the traveling salesman problem.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

Syntax

```fsharp
static member __TSP :
  nCities : int -> unit
```

Parameters

*nCities*
  Type: System.Int32
  The number of cities to include in the map, from 5 to 8.

See Also

Reference
  Tests Class
  Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Tests

RenderTest Method

Renders and dumps test circuits in a variety of formats and detail levels.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

## Syntax

```fsharp
static member RenderTest :  
  name : string *  
  circ : Circuit *  
  ket : Ket -> unit
```

### Parameters

- **name**
  - Type: System.String
  - The base name for the output files.

- **circ**
  - Type: Microsoft.Research.LiquidCircuit
  - The Circuit to render.

- **ket**
  - Type: Microsoft.Research.LiquidKet
  - The state vector for the circuit.

## Remarks

Three versions of the circuit will be dumped to the log file:

- The base circuit.
- The circuit aggressively folded to maximize parallelism.
- The circuit with gates grown to a maximum of 11 inputs, and
then folded.

Four pairs of graphics files will be created. For each version, both an SVG version, with a .htm extension, and a Tik-Z version, with a .tex, will be created.

- The base circuit.
- The circuit normally folded to use "easy" parallelism.
- The circuit aggressively folded to maximize parallelism.
- The circuit with gates grown to a maximum of 11 inputs, and then folded.

See Also

Reference

Tests Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

Util Class

General utilities used by the rest of the system

Inheritance Hierarchy

System
Object    Microsoft.Research.LiquidUtil

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4*)

Syntax

```
[<AbstractClassAttribute>]
[<SealedAttribute>]
type Util = class end
```

The Util type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>copyright</td>
<td>Copyright notice for Liquid.</td>
</tr>
<tr>
<td>interactive</td>
<td></td>
</tr>
<tr>
<td>outputPrefix</td>
<td>Global override for the prefix used for output functions (show and variants).</td>
</tr>
<tr>
<td>sqrt2</td>
<td>Highly accurate square root of 2, for use building unitary matrices.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__show</td>
<td>Shows a line of output. This is a command line-callable routine.</td>
</tr>
<tr>
<td>getFlgPar</td>
<td>Extracts a boolean macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>getFltPar</td>
<td>Extracts a float macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>getIntPar</td>
<td>Extracts an integer macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>getIntsPar</td>
<td>Extracts an integer list macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>getStrPar</td>
<td>Extracts a string macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>Normal</td>
<td>Add a Normal distribution to the System.Random class. This is an extension method and may be used as if it were an instance method on Random.</td>
</tr>
<tr>
<td>PermAry</td>
<td>Create a random permutation array (call .permute with</td>
</tr>
</tbody>
</table>
results) This is an extension method and may be used as if it were an instance method on Random.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>procStats</td>
<td>Gets current process memory usage statistics.</td>
</tr>
<tr>
<td>showa</td>
<td>Shows a line of output to the console and the log file.</td>
</tr>
<tr>
<td>showBareInd</td>
<td>Shows an indented line of output, with no prefix, to the console and log file. This routine is used by or with various Dump() routines.</td>
</tr>
<tr>
<td>showBareLogInd</td>
<td>Shows an indented line of output, with no prefix, to the log file. This routine is used by or with various Dump() routines.</td>
</tr>
<tr>
<td>showDump</td>
<td>Dumps out a string with with an optional indentation.</td>
</tr>
<tr>
<td>showInd</td>
<td>Shows an indented line of output to the console and log file. This routine is used by or with various Dump() routines.</td>
</tr>
<tr>
<td>showLoga</td>
<td>Shows a line of output to the log file.</td>
</tr>
<tr>
<td>showLogInd</td>
<td>Shows an indented line of output to the log file. This routine is used by or with various Dump() routines.</td>
</tr>
</tbody>
</table>
showProcStats  Print process memory usage statistics to the console and log file.

See Also

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
## Util Properties

The **Util** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>copyright</code></td>
<td>Copyright notice for Liquid.</td>
</tr>
<tr>
<td><code>interactive</code></td>
<td></td>
</tr>
<tr>
<td><code>outputPrefix</code></td>
<td>Global override for the prefix used for output functions.</td>
</tr>
<tr>
<td><code>sqrt2</code></td>
<td>Highly accurate square root of 2, for use building matrices.</td>
</tr>
</tbody>
</table>

### See Also

Reference

Util Class

Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Utilcopyright Property

Copyright notice for Liquid.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
static member copyright : FSharpFunc<Unit, Unit>
```

**Property Value**

Type: FSharpFunc<Unit, Unit>

### See Also

Reference

Util Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Utilinteractive Property


**Namespace:**  Microsoft.Research.Liquid  
**Assembly:**  Liquid1 (in Liquid1.dll)  
**Version:**  1.0.5981.24943 (1.0.)*

▶ **Syntax**

```fsharp
static member interactive : bool with get
```

**Property Value**

Type:  **Boolean**

▶ **See Also**

Reference

Util Class
Microsoft.Research.Liquid Namespace
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

UtiloutputPrefix Property

Global override for the prefix used for output functions (show and variants).

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```
static member outputPrefix : string with get, set
```

### Property Value

Type: **String**

### See Also

**Reference**  
Util Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Utilsqrt2 Property

Highly accurate square root of 2, for use building unitary matrices.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
static member sqrt2 : float with get
```

Property Value
Type: Double

See Also

Reference
Util Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
## Util Methods

The **Util** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>show</strong></td>
<td>Shows a line of output. This is a command line-callable routine.</td>
</tr>
<tr>
<td>getFlgPar</td>
<td>Extracts a boolean macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>getFltPar</td>
<td>Extracts a float macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>getIntPar</td>
<td>Extracts an integer macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>getIntsPar</td>
<td>Extracts an integer list macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>getStrPar</td>
<td>Extracts a string macro definition from a command option dictionary.</td>
</tr>
<tr>
<td>Normal</td>
<td>Add a Normal distribution to the System.Random class. This is an extension method and may...</td>
</tr>
</tbody>
</table>
be used as if it were an instance method on Random.

<table>
<thead>
<tr>
<th>S</th>
<th>PermAry</th>
<th>Create a random permutation array (call .permute with results) This is an extension method and may be used as if it were an instance method on Random.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>procStats</td>
<td>Gets current process memory usage statistics.</td>
</tr>
<tr>
<td>S</td>
<td>showa</td>
<td>Shows a line of output to the console and the log file.</td>
</tr>
<tr>
<td>S</td>
<td>showBareInd</td>
<td>Shows an indented line of output, with no prefix, to the console and log file. This routine is used by or with various Dump() routines.</td>
</tr>
<tr>
<td>S</td>
<td>showBareLogInd</td>
<td>Shows an indented line of output, with no prefix, to the log file. This routine is used by or with various Dump() routines.</td>
</tr>
<tr>
<td>S</td>
<td>showDump</td>
<td>Dumps out a string with with an optional indentation.</td>
</tr>
<tr>
<td>S</td>
<td>showInd</td>
<td>Shows an indented line of output to the console and log file. This routine is used by or with various Dump() routines.</td>
</tr>
<tr>
<td>S</td>
<td>showLoga</td>
<td>Shows a line of output to the log file.</td>
</tr>
<tr>
<td>showLogInd</td>
<td>Shows an indented line of output to the log file. This routine is used by or with various Dump() routines.</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>showProcStats</td>
<td>Print process memory usage statistics to the console and log file.</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference

Util Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Util__show Method

Shows a line of output. This is a command line-callable routine.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.4)

Syntax

```fsharp
static member __show :
    str : string -> unit
```

Parameters

- `str`
  - Type: System.String
  - String argument to show

See Also

Reference
Util Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
**UtilgetFlgPar Method**

Extracts a boolean macro definition from a command option dictionary.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
static member getFlgPar :
  dic : Dictionary<String, String> *
  nam : string *
  def : bool -> bool
```

**Parameters**

- **dic**
  - The dictionary to search

- **nam**
  - Type: System.String
  - The macro name

- **def**
  - Type: System.Boolean
  - The default value to return if the name is not found in the dictionary

**Return Value**

- Type: Boolean
  - Macro value if found, or else default value

### See Also
Reference

Util Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilgetFltPar Method

Extracts a float macro definition from a command option dictionary.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

F#

```fsharp
static member getFltPar :
    dic : Dictionary<string, string> *
    nam : string *
    def : float -> float
```

Parameters

dic
    Type: System.Collections.GenericDictionary<String, String>
    The dictionary to search

nam
    Type: System.String
    The macro name

def
    Type: System.Double
    The default value to return if the name is not found in the dictionary

Return Value
    Type: Double
    Macro value if found, or else default value

See Also
Reference

Util Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilgetIntPar Method

Extracts an integer macro definition from a command option dictionary.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
static member getIntPar :  
    dic : Dictionary<string, string> *  
    nam : string *  
    def : int -> int
```

**Parameters**

- `dic`
  - Type: System.Collections.Generic.Dictionary`<string, string>`
  - The dictionary to search

- `nam`
  - Type: System.String
  - The macro name

- `def`
  - Type: System.Int32
  - The default value to return if the name is not found in the dictionary

**Return Value**

- Type: Int32
  - Macro value if found, or else default value

### See Also
Reference

Util Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Util.getIntsPar Method

Extracts an integer list macro definition from a command option dictionary.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```fsharp
static member getIntsPar :  
dic : Dictionary<string, string>  *  
   nam : string  *  
def : FSharpList<int>  ->  FSharpList<int>
```

**Parameters**

*dic*
Type: System.Collections.Generic.Dictionary<string, string>  
The dictionary to search

*nam*
Type: System.String  
The macro name

*def*
Type: Microsoft.FSharp.Collections.FSharpList<Int32>  
The default value to return if the name is not found in the dictionary

**Return Value**

Type: FSharpList<Int32>  
Macro value if found, or else default value
See Also

Reference

Util Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilgetStrPar Method

Extracts a string macro definition from a command option dictionary.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member getStrPar :
  dic : Dictionary&lt;string, string&gt; *
  nam : string *
  def : string -&gt; string
```

**Parameters**

- **dic**
  
  Type: System.Collections.Generic.Dictionary&lt;string, string&gt;, String  
  The dictionary to search

- **nam**
  
  Type: System.String  
  The macro name

- **def**
  
  Type: System.String  
  The default value to return if the name is not found in the dictionary

**Return Value**

Type: String  
Macro value if found, or else default value

### See Also
Reference
Util Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilNormal Method

Add a Normal distribution to the System.Random class. This is an extension method and may be used as if it were an instance method on Random.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
static member Normal :  
    x : Random * 
    mean : float * 
    sd : float -> float
```

### Parameters

- **x**
  - Type: SystemRandom
  - The instance of System.Random (ignore)

- **mean**
  - Type: SystemDouble
  - Mean of returned value

- **sd**
  - Type: SystemDouble
  - Standard deviation of returned value

### Return Value

- Type: Double
  - Random Gaussian value

### See Also
UtilPermAry Method

Create a random permutation array (call .permute with results) This is an extension method and may be used as if it were an instance method on Random.

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0. *)

⚠️ Syntax

```fsharp
static member PermAry :  
  x : Random *  
  len : int -> int[]
```

Parameters

- **x**
  - Type: System.Random
  - The instance of System.Random (ignore)

- **len**
  - Type: System.Int32
  - Length of array to create

Return Value

- Type: Int32
- Randomized array

See Also

Reference
- Util Class
- Microsoft.Research.Liquid Namespace
UtilprocStats Method

Gets current process memory usage statistics.

**Namespace:** Microsoft.Research.Liquid
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member procStats : doCollect : bool -> UtilprocStatsT
```

### Parameters

**doCollect**

Type: System.Boolean

Do a garbage collection before reporting?

### Return Value

Type: UtilprocStatsT

procStatsT struct

### See Also

**Reference**

Util Class
Microsoft.Research.Liquid Namespace
Microsoft.Research.LiquidUtilprocStatsT

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
Utilshowa Method

Shows a line of output to the console and the log file.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

## Syntax

```
static member show : 
  fmt : PrintfFormat<'a, Unit, string, Unit
```

### Parameters

**fmt**

Type: Microsoft.FSharp.CorePrintfFormata, Unit, string, Unit

Printf format parameters

### Type Parameters

**a**

Internal type of the printf string. The F# compiler will deduce this.

### Return Value

Type: a

Internal printfformat.

## See Also

Reference

- Util Class
- Microsoft.Research.Liquid Namespace
Util.showLoga(PrintfFormat, Unit, String, Unit)
Util.logOpen
Util.logClose

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilshowBareInd Method

Shows an indented line of output, with no prefix, to the console and log file. This routine is used by or with various Dump() routines.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member showBareInd :  
    level : int *  
    line : string -> unit
```

### Parameters

- **level**  
  Type: `System.Int32`  
  The indentation level. Each level represents a two space indent.  

- **line**  
  Type: `System.String`  
  The text to show.

### See Also

**Reference**  
- Util Class  
- Microsoft.Research.Liquid Namespace  
  - `UtilshowInd(Int32, String)`  
  - `UtilshowLogInd(Int32, String)`  
  - `UtilshowBareLogInd(Int32, String)`

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilshowBareLogInd Method

Shows an indented line of output, with no prefix, to the log file. This routine is used by or with various Dump() routines.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member showBareLogInd :  
    level : int *  
    line : string -> unit
```

**Parameters**

- **level**
  
  Type: **System.Int32**  
  The indentation level. Each level represents a two space indent.

- **line**
  
  Type: **System.String**  
  The text to show.

### See Also

**Reference**

- Util Class  
- Microsoft.Research.Liquid Namespace  
- UtilshowInd(Int32, String)  
- UtilshowLogInd(Int32, String)  
- UtilshowBareInd(Int32, String)
Util showDump Method

Dumps out a string with an optional indentation.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0*)

### Syntax

```fsharp
static member showDump :  
    fO : FSharpOption<FSharpFunc<int, FSharpFunc<string, unit>>, Int32> *  
    lO : FSharpOption<int>  
    str : string -> unit
```

### Parameters

- **fO**
  - Type: `Microsoft.FSharp.Core.FSharpOption<FSharpFunc<int, FSharpFunc<string, unit>>, Int32>`
  - The optional output function to use. The default is `showLogInd`.

- **lO**
  - Type: `Microsoft.FSharp.Core.FSharpOption<int>`
  - The optional indentation level. The default is 0.

- **str**
  - Type: `System.String`
  - String to output

### See Also

**Reference**
- Util Class
- Microsoft.Research.Liquid Namespace
UtilshowInd Method

Shows an indented line of output to the console and log file. This routine is used by or with various Dump() routines.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member showInd :  
  level : int *  
  line : string -> unit
```

### Parameters

**level**

Type: **SystemInt32**  
The indentation level. Each level represents a two space indent.

**line**

Type: **SystemString**  
The text to show.

### See Also

**Reference**

Util Class  
Microsoft.Research.Liquid Namespace  
UtilshowLogInd(Int32, String)  
UtilshowBareInd(Int32, String)  
UtilshowBareLogInd(Int32, String)
UtilShowLog 

**Method**

Shows a line of output to the log file.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0*)

### Syntax

```fsharp
static member showLog : fmt : PrintfFormat<'a, Unit, string, Unit
```

**Parameters**

- **fmt**
  
  Type: Microsoft.FSharp.CorePrintfFormata, Unit, String, Unit

- **Type Parameters**
  
  - **a**

  Internal type of the printf string. The F# compiler will deduce this.

- **Return Value**

  Type: a

  Internal printf format

### See Also

- Reference
- Util Class
- Microsoft.Research.Liquid Namespace
Util.showa(PrintfFormata, Unit, String, Unit)
Util.logOpen
Util.logClose

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilshowLogInd Method

Shows an indented line of output to the log file. This routine is used by or with various Dump() routines.

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member showLogInd :  
    level : int *  
    line : string -> unit
```

### Parameters

- **level**
  - Type: System.Int32
  - The indentation level. Each level represents a two space indent.

- **line**
  - Type: System.String
  - The text to show.

### See Also

**Reference**
- Util Class
- Microsoft.Research.Liquid Namespace
- UtilshowInd(Int32, String)
- UtilshowBareInd(Int32, String)
- UtilshowBareLogInd(Int32, String)

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilshowProcStats Method

Print process memory usage statistics to the console and log file.

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
static member showProcStats : nam : string -> unit
```

### Parameters

`nam`

Type: `System.String`

Tag for this log entry, up to 12 characters

### See Also

- Reference
- Util Class
- Microsoft.Research.Liquid Namespace

---

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

UtilLQDAttribute Class

Allows a function to be visible from a LIQUiD script or the command line

Inheritance Hierarchy

```
SystemObject  SystemAttribute
    Microsoft.Research.LiquidUtilLQDAttribute
```

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```
[<SerializableAttribute>]
type LQDAttribute =
    class
        inherit Attribute
    end
```

The UtilLQDAttribute type exposes the following members.

Constructors

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon" alt="UtilLQDAttribute" /></td>
<td>Initializes a new instance of the UtilLQDAttribute class</td>
</tr>
</tbody>
</table>
```

Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TypeId</strong></td>
<td>When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)</td>
</tr>
</tbody>
</table>

**Top**

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns the hash code for this instance. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>IsDefaultAttribute</strong></td>
<td>When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>Match</strong></td>
<td>When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)</td>
</tr>
</tbody>
</table>
ToString

Returns a string that represents the current object. (Inherited from Object.)

See Also

Reference
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilLQDAttribute Constructor

Initializes a new instance of the UtilLQDAttribute class

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.)*

### Syntax

```fsharp
new : unit -> LQDAttribute
```

### See Also

- Reference
- UtilLQDAttribute Class
- Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
LQDAttribute Properties

The UtilLQDAttribute type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typeld</td>
<td>When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)</td>
</tr>
</tbody>
</table>

### See Also

Reference

UtilLQDAttribute Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The **UtilLQDAttribute** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Returns a value that indicates whether this instance is equal to a specified object. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns the hash code for this instance. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
<tr>
<td>Match</td>
<td>When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that</td>
</tr>
</tbody>
</table>
represents the current object.
(Inherited from Object.)

See Also

Reference
UtilLQDAttribute Class
Microsoft.Research.Liquid Namespace
UtilprocStatsT Class

Current process memory usage statistics. Returned by the procStates function.

**Inheritance Hierarchy**

System\Object \ Microsoft.Research.Liquid\UtilprocStatsT

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

**Syntax**

```fsharp
[<SealedAttribute>]
[<SerializableAttribute>]
type procStatsT =
    class
        interface IEquatable<UtilprocStatsT>
        interface IStructuralEquatable
        interface IComparable<UtilprocStatsT>
        interface IComparable
        interface IStructuralComparable
    end
```

The **UtilprocStatsT** type exposes the following members.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="peakVMMB" /></td>
<td>Peak virtual memory in megabytes</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>peakWSMB</td>
<td>Peak working set in megabytes</td>
</tr>
<tr>
<td>privMB</td>
<td>Private memory in megabytes</td>
</tr>
<tr>
<td>wsetMB</td>
<td>Working set in megabytes</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

Microsoft.Research.Liquid Namespace
UtilprocStats(Boolean)
procStatsT Properties

The UtilprocStatsT type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>peakVMMB</td>
<td>Peak virtual memory in megabytes</td>
</tr>
<tr>
<td>peakWSMB</td>
<td>Peak working set in megabytes</td>
</tr>
<tr>
<td>privMB</td>
<td>Private memory in megabytes</td>
</tr>
<tr>
<td>wsetMB</td>
<td>Working set in megabytes</td>
</tr>
</tbody>
</table>

See Also

Reference
UtilprocStatsT Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilprocStatsTpeakVMMB Property

Peak virtual memory in megabytes

**Namespace:** Microsoft.Research.Liquid  
**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.0)

### Syntax

```fsharp
member peakVMMB : int with get
```

### Property Value

Type: Int32

### See Also

**Reference**

UtilprocStatsT Class  
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilprocStatsTpeakWSMB

Property

Peak working set in megabytes

**Namespace:** Microsoft.Research.Liquid

**Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

### Syntax

```fsharp
member peakWSMB : int with get
```

### Property Value

**Type:** Int32

### See Also

**Reference**

UtilprocStatsT Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilprocStatsTprivMB Property

Private memory in megabytes

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
member privMB : int with get
```

Property Value
Type: Int32

See Also

Reference
UtilprocStatsT Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
UtilprocStatsTwsetMB Property

Working set in megabytes

Namespace: Microsoft.Research.Liquid
Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax

```fsharp
member wsetMB : int with get
```

Property Value
Type: Int32

See Also

Reference
UtilprocStatsT Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)
The Language-Integrated Quantum Operations (LIQUi|>) Simulator

procStatsT Methods

The UtilprocStatsT type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>

See Also

Reference

UtilprocStatsT Class
Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)