```
# Quantum Fourier Transform:
QFT | qureg

# Quantum arithmetic:
# Note: quint is a quantum register representing an integer
# Add integer a:
AddConstant(a) | quint
# Add a modulo N:
AddConstantModN(a, N) | quint
# Multiply by a modulo N:
MultiplyByConstantModN(a, N) | quint

# Apply exp(-i*t*hamiltonian)
# Hamiltonians can be specified using QubitOperators:
hamiltonian = QubitOperator("X0 X1") + QubitOperator("Y0 Y1")
TimeEvolution(hamiltonian, t) | qureg

# Turns state |0> into state sum_j beta_j |j>:
StatePreparation(beta) | qureg
```

Listing 1: Some of ProjectQ's n-qubit gates.

ProjectQ syntax	Name	Definition	Symbol
`X | qubit`	Pauli X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\oplus$
`Y | qubit`	Pauli Y	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\boxed{\mathrm{Y}}$
`Z | qubit`	Pauli Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\boxed{\mathrm{Z}}$
`S | qubit`	S gate	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	$\boxed{\mathrm{S}}$
`Sdagger | qubit`	inverse of S	$\mathrm{S}^\dagger$	$\boxed{\mathrm{S}^\dagger}$
`T | qubit`	T gate	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$	$\boxed{\mathrm{T}}$
`Tdagger | qubit`	inverse of T	$\mathrm{T}^\dagger$	$\boxed{\mathrm{T}^\dagger}$
`H | qubit`	Hadamard	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\boxed{\mathrm{H}}$
`Ph(theta) | qubit`	(global) Phase gate	$\begin{pmatrix} e^{i\theta} & 0 \\ 0 & e^{i\theta} \end{pmatrix}$	$\boxed{\mathrm{Ph}_\theta}$
`R(theta) | qubit`	Phase-shift gate	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$	$\boxed{\mathrm{R}_\theta}$
`Rx(theta) | qubit`	Rotation around x	$e^{-i\theta X/2} = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$	$\boxed{\mathrm{Rx}_\theta}$
`Ry(theta) | qubit`	Rotation around y	$e^{-i\theta Y/2} = \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$	$\boxed{\mathrm{Ry}_\theta}$
`Rz(theta) | qubit`	Rotation around z	$e^{-i\theta Z/2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$	$\boxed{\mathrm{Rz}_\theta}$
`Measure | qubit`	Measurement in computational basis $\|0\rangle$, $\|1\rangle$		

Table 1: Some of the standard single qubit gates available in ProjectQ.

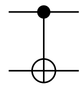
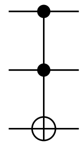
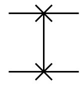
ProjectQ syntax	Definition	Symbol
`CNOT | (ctrl_qubit, qubit)`	$\underbrace{\|1\rangle\langle 1\|}_{\texttt{ctrl\_qubit}} \otimes \underbrace{X}_{\texttt{qubit}} + \|0\rangle\langle 0\| \otimes I$	
`Toffoli | (ctrl_qureg, qubit)`	$\underbrace{\|11\rangle\langle 11\|}_{\texttt{ctrl\_qureg}} \otimes \underbrace{X}_{\texttt{qubit}} + (I - \|11\rangle\langle 11\|) \otimes I$	
`Swap | (qubit0, qubit1)`	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	

Table 2: Some of the standard two and three qubit gates available in ProjectQ.

Meta-instructions

```
eng = projectq.MainEngine()
...
with Loop(eng, 10):
    U | qubits
```

```
get_inverse(T) | qubit # Equivalent to Tdagger | qubit
```

```
eng = projectq.MainEngine()
...
with Dagger(eng):
    # Anything executed in this context will be inverted
    T | qubit
    S | qubit
```

```
C(X, 2) | (ctrl_qureg, qubit)  # ctrl_qureg contains two qubits
```

```python
with Control(eng, ctrl_qureg):
    # Anything executed in this context will be controlled
    # on all  qubits in ctrl_qureg being in state 1
    T | qubit
    S | qubit
```

```python
eng = projectq.MainEngine()
...
with Compute(eng):
    U | qureg
V | qureg
Uncompute(eng)
```

```python
eng = projectq.MainEngine()
...
with Compute(eng):
    U | qureg
V | qureg
with CustomUncompute(eng):
    Udagger | qureg
```