

Q# 0.10 Language Quick Reference

Primitive Types	
64-bit integers	Int
Double-precision floats	Double
Booleans	Bool e.g.: true or false
Qubits	Qubit
Pauli basis	Pauli e.g.: PauliI, PauliX, PauliY, or PauliZ
Measurement results	Result e.g.: Zero or One
Sequences of integers	Range e.g.: 1..10 or 5..-1..0
Strings	String e.g.: "Hello Quantum!"
"Return no information" type	Unit e.g.: ()

Derived Types	
Arrays	<i>elementType</i> []
Tuples	(<i>type0</i> , <i>type1</i> , ...) e.g.: (Int, Qubit)
Functions	<i>input</i> -> <i>output</i> e.g.: ArcCos : (Double) -> Double
Operations	<i>input</i> => <i>output</i> is <i>variants</i> e.g.: H : (Qubit => Unit is Adj)

User-Defined Types	
Declare UDT with anonymous items	<code>newtype Name = (Type, Type);</code> e.g.: <code>newtype Pair = (Int, Int);</code>
Define UDT literal	<code>Name(baseTupleLiteral)</code> e.g.: <code>let origin = Pair(0, 0);</code>
Unwrap operator ! (convert UDT to underlying type)	<code>VarName!</code> e.g.: <code>let originTuple = origin!;</code> (now <code>originTuple = (0, 0)</code>)
Declare UDT with named items	<code>newtype Name = (Name1: Type, Name2: Type);</code> e.g.: <code>newtype Complex = (Re : Double, Im : Double);</code>
Accessing named items of UDTs	<code>VarName::ItemName</code> e.g.: <code>complexVariable::Re</code>
Update-and-reassign for named UDT items	<code>set VarName w/= ItemName <- val;</code> e.g.: <code>mutable p = Complex(0., 0.);</code> <code>set p w/= Re <- 1.0;</code>

Symbols and Variables	
Declare immutable symbol	<code>let varName = value</code>
Declare mutable symbol (variable)	<code>mutable varName = initialValue</code>
Update mutable symbol (variable)	<code>set varName = newValue</code>
Apply-and-reassign	<code>set varName operator= expression</code> e.g.: <code>set counter += 1;</code>

Functions and Operations	
Define function (classical routine)	<code>function Name(in0 : type0, ...)</code> <code>: returnType {</code> <code>// function body</code> <code>}</code>
Call function	<code>Name(parameters)</code> e.g.: <code>let two = Sqrt(4.0);</code>
Define operation (quantum routine) with explicitly specified body, controlled and adjoint variants	<code>operation Name(in0 : type0, ...)</code> <code>: returnType {</code> <code>body { ... }</code> <code>adjoint { ... }</code> <code>controlled { ... }</code> <code>adjoint controlled { ... }</code> <code>}</code>
Define operation with automatically generated adjoint and controlled variants	<code>operation Name(in0 : type0, ...)</code> <code>: returnType is Adj + Ctl {</code> <code>...</code> <code>}</code>
Call operation	<code>Name(parameters)</code> e.g.: <code>Ry(0.5 * PI(), q);</code>
Call adjoint operation	<code>Adjoint Name(parameters)</code> e.g.: <code>Adjoint Ry(0.5 * PI(), q);</code>
Call controlled operation	<code>Controlled Name(controlQubits, parameters)</code> e.g.: <code>Controlled Ry(controls, (0.5 * PI(), target));</code>

Control Flow	
Iterate over a range of numbers	<code>for (index in range) {</code> <code>// Use integer index</code> <code>...</code> <code>}</code> e.g.: <code>for (i in 0..N-1) { ... }</code>
While loop (within functions)	<code>while (condition) {</code> <code>...</code> <code>}</code>
Iterate over an array	<code>for (val in array) {</code> <code>// Use value val</code> <code>...</code> <code>}</code> e.g.: <code>for (q in register) { ... }</code>
Repeat-until-success loop	<code>repeat { ... }</code> <code>until (condition)</code> <code>fixup { ... }</code>
Conditional statement	<code>if (cond1) { ... }</code> <code>elif (cond2) { ... }</code> <code>else { ... }</code>
Ternary operator	<code>condition ? caseTrue caseFalse</code>
Return a value	<code>return value</code>
Stop with an error	<code>fail "Error message"</code>
Conjugations (ABA^\dagger pattern)	<code>within { ... }</code> <code>apply { ... }</code>

Arrays											
Allocate array	<code>mutable name = new Type[Length]</code> e.g.: <code>mutable b = new Bool[2];</code>										
Get array length	<code>Length(name)</code>										
Access k-th element	<code>name[k]</code> NB: indices are 0-based										
Assign k-th element (copy-and-update)	<code>set name w/= k <- value</code> e.g.: <code>set b w/= 0 <- true;</code>										
Array literal	<code>[value0, value1, ...]</code> e.g.: <code>let b = [true, false, true];</code>										
Array concatenation	<code>array1 + array2</code> e.g.: <code>let t = [1, 2, 3] + [4, 5];</code>										
Slicing (subarray)	<code>name[sliceRange]</code> e.g.: <code>if t = [1, 2, 3, 4, 5], then</code> <table><tr><td><code>t[1 .. 3]</code></td><td>is <code>[2, 3, 4]</code></td></tr><tr><td><code>t[3 ...]</code></td><td>is <code>[4, 5]</code></td></tr><tr><td><code>t[... 1]</code></td><td>is <code>[1, 2]</code></td></tr><tr><td><code>t[0 .. 2 ...]</code></td><td>is <code>[1, 3, 5]</code></td></tr><tr><td><code>t[...-1...]</code></td><td>is <code>[5, 4, 3, 2, 1]</code></td></tr></table>	<code>t[1 .. 3]</code>	is <code>[2, 3, 4]</code>	<code>t[3 ...]</code>	is <code>[4, 5]</code>	<code>t[... 1]</code>	is <code>[1, 2]</code>	<code>t[0 .. 2 ...]</code>	is <code>[1, 3, 5]</code>	<code>t[...-1...]</code>	is <code>[5, 4, 3, 2, 1]</code>
<code>t[1 .. 3]</code>	is <code>[2, 3, 4]</code>										
<code>t[3 ...]</code>	is <code>[4, 5]</code>										
<code>t[... 1]</code>	is <code>[1, 2]</code>										
<code>t[0 .. 2 ...]</code>	is <code>[1, 3, 5]</code>										
<code>t[...-1...]</code>	is <code>[5, 4, 3, 2, 1]</code>										

Debugging (classical)	
Print a string	<code>Message("Hello Quantum!")</code>
Print an interpolated string	<code>Message(\$"Value = {val}")</code>

Resources

Documentation	
Quantum Development Kit	https://docs.microsoft.com/quantum
Q# Language Reference	https://docs.microsoft.com/quantum/language
Q# Libraries Reference	https://docs.microsoft.com/qsharp/api

Q# Code Repositories	
QDK Samples	https://github.com/microsoft/quantum
QDK Libraries	https://github.com/microsoft/QuantumLibraries
Quantum Katas (tutorials)	https://github.com/microsoft/QuantumKatas
Q# compiler and extensions	https://github.com/microsoft/qsharp-compiler
Simulation framework	https://github.com/microsoft/qsharp-runtime
Jupyter kernel and Python host	https://github.com/microsoft/iqsharp
Source code for the documentation	https://github.com/MicrosoftDocs/quantum-docs-pr

Qubit Allocation	
Allocate a register of N qubits	using <code>(reg = Qubit[N]) { // Qubits in <i>reg</i> start in $0\rangle$. ... // Qubits must be returned to $0\rangle$. }</code>
Allocate one qubit	using <code>(one = Qubit()) { ... }</code>
Allocate a mix of qubit registers and individual qubits	using <code>((x, y, ...) = (Qubit[N], Qubit(), ...)) { ... }</code>

Debugging (quantum)	
Print amplitudes of wave function	<code>DumpMachine("dump.txt")</code>
Assert that a qubit is in $ 0\rangle$ or $ 1\rangle$ state	<code>AssertQubit(Zero, <i>zeroQubit</i>) AssertQubit(One, <i>oneQubit</i>)</code>

Measurements	
Measure qubit in Pauli Z basis	<code>M(<i>oneQubit</i>)</code> yields a Result (Zero or One)
Reset qubit to $ 0\rangle$	<code>Reset(<i>oneQubit</i>)</code>
Reset an array of qubits to $ 0..0\rangle$	<code>ResetAll(<i>register</i>)</code>

Working with Q# from command line

Command Line Basics	
Change directory	<code>cd <i>dirname</i></code>
Go to home	<code>cd ~</code>
Go up one directory	<code>cd ..</code>
Make new directory	<code>mkdir <i>dirname</i></code>
Open current directory in VS Code	<code>code .</code>

Working with Q# Projects	
Create new project	<code>dotnet new console -lang Q# --output <i>project-dir</i></code>
Change directory to project directory	<code>cd <i>project-dir</i></code>
Build project	<code>dotnet build</code>
Run all unit tests	<code>dotnet test</code>

Math reference

Complex Arithmetic	
i^2	-1
$(a + bi) + (c + di)$	$(a + c) + (b + d)i$
$(a + bi)(c + di)$	$a \cdot c + a \cdot di + b \cdot ci + (b \cdot d)i^2 =$ $= (a \cdot c - b \cdot d) + (a \cdot d + b \cdot c)i$
Complex conjugate	$a + bi = a - bi$
Division $\frac{a+bi}{c+di}$	$\frac{a+bi}{c+di} \cdot 1 = \frac{a+bi}{c+di} \cdot \frac{c-di}{c-di} = \frac{(a+bi)(c+di)}{c^2+d^2}$
Modulus $ a + bi $	$\sqrt{a^2 + b^2}$
$e^{i\theta}$	$\cos \theta + i \sin \theta$
e^{a+bi}	$e^a \cdot e^{bi} = e^a \cos b + ie^a \sin b$
r^{a+bi}	$r^a \cdot r^{bi} = r^a \cdot e^{bi \ln r} =$ $= r^a \cos(b \ln r) + i \cdot r^a \sin(b \ln r)$
Polar form $re^{i\theta}$ to Cartesian form $a + bi$	$a = r \cos \theta$ $b = r \sin \theta$
Cartesian form $a + bi$ to polar form $re^{i\theta}$	$r = \sqrt{a^2 + b^2}$ $\theta = \arctan(\frac{b}{a})$

Linear Algebra	
$m \times n$ matrix	$\begin{matrix} & \xrightarrow{n \text{ columns}} \\ \begin{matrix} \downarrow m \text{ rows} \end{matrix} & \begin{bmatrix} a_{0,0} & \cdots & a_{0,n-1} \\ \vdots & \ddots & \vdots \\ a_{m-1,0} & \cdots & a_{m-1,n-1} \end{bmatrix} \end{matrix}$
Vector of size n	$\begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix}$
Addition $\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix}$	$\begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$
Scalar product $a \cdot \begin{bmatrix} b & c \\ d & e \end{bmatrix}$	$\begin{bmatrix} a \cdot b & a \cdot c \\ a \cdot d & a \cdot e \end{bmatrix}$
Matrix product $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\begin{bmatrix} a \cdot x + b \cdot y + c \cdot z \\ d \cdot x + e \cdot y + f \cdot z \end{bmatrix}$
Transpose $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}^T$	$\begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$
Adjoint $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}^\dagger$	$\begin{bmatrix} \bar{a} & \bar{d} \\ \bar{b} & \bar{e} \\ \bar{c} & \bar{f} \end{bmatrix}$
Inner product $\left\langle \begin{bmatrix} a \\ b \end{bmatrix}, \begin{bmatrix} c \\ d \end{bmatrix} \right\rangle$	$\begin{bmatrix} a \\ b \end{bmatrix}^\dagger \begin{bmatrix} c \\ d \end{bmatrix} = [\bar{a} \quad \bar{b}] \begin{bmatrix} c \\ d \end{bmatrix} = \bar{a}c + \bar{b}d$
Outer product $\begin{bmatrix} a \\ b \end{bmatrix}$ and $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}^\dagger = \begin{bmatrix} a \\ b \end{bmatrix} [\bar{x} \quad \bar{y} \quad \bar{z}] =$ $= \begin{bmatrix} a \cdot \bar{x} & a \cdot \bar{y} & a \cdot \bar{z} \\ b \cdot \bar{x} & b \cdot \bar{y} & b \cdot \bar{z} \end{bmatrix}$

Gates reference

Single Qubit gates						
Gate	Matrix representation	Ket-bra representation	Applying to $ \psi\rangle = \alpha 0\rangle + \beta 1\rangle$	Applying to basis states:	$ 0\rangle, 1\rangle, +\rangle, -\rangle$ and	$ \pm i\rangle = \frac{1}{\sqrt{2}}(0\rangle \pm i 1\rangle)$
X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$ 0\rangle\langle 1 + 1\rangle\langle 0 $	$ \psi\rangle = \alpha 1\rangle + \beta 0\rangle$	$X 0\rangle = 1\rangle$ $X 1\rangle = 0\rangle$	$X +\rangle = +\rangle$ $X -\rangle = - -\rangle$	$X i\rangle = i -i\rangle$ $X -i\rangle = -i i\rangle$
Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	$i(1\rangle\langle 0 - 0\rangle\langle 1)$	$Y \psi\rangle = i(\alpha 1\rangle - \beta 0\rangle)$	$Y 0\rangle = i 1\rangle$ $Y 1\rangle = -i 0\rangle$	$Y +\rangle = -i -\rangle$ $Y -\rangle = i +\rangle$	$Y i\rangle = i\rangle$ $Y -i\rangle = - -i\rangle$
Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$ 0\rangle\langle 0 - 1\rangle\langle 1 $	$Z \psi\rangle = \alpha 0\rangle - \beta 1\rangle$	$Z 0\rangle = 0\rangle$ $Z 1\rangle = - 1\rangle$	$Z +\rangle = -\rangle$ $Z -\rangle = +\rangle$	$Z i\rangle = -i\rangle$ $Z -i\rangle = i\rangle$
I	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$ 0\rangle\langle 0 + 1\rangle\langle 1 $	$I \psi\rangle = \psi\rangle$			
H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	$ 0\rangle\langle + + 1\rangle\langle - $	$H \psi\rangle = \alpha +\rangle + \beta -\rangle =$ $= \frac{\alpha+\beta}{\sqrt{2}} 0\rangle + \frac{\alpha-\beta}{\sqrt{2}} 1\rangle$	$H 0\rangle = +\rangle$ $H 1\rangle = -\rangle$	$H +\rangle = 0\rangle$ $H -\rangle = 1\rangle$	$H i\rangle = e^{i\pi/4} -i\rangle$ $H -i\rangle = e^{-i\pi/4} i\rangle$
S	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$	$ 0\rangle\langle 0 + i 1\rangle\langle 1 $	$S \psi\rangle = \alpha 0\rangle + i\beta 1\rangle$	$S 0\rangle = 0\rangle$ $S 1\rangle = i 1\rangle$	$S +\rangle = i\rangle$ $S -\rangle = -i\rangle$	$S i\rangle = -\rangle$ $S -i\rangle = +\rangle$
T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	$ 0\rangle\langle 0 + e^{i\pi/4} 1\rangle\langle 1 $	$T \psi\rangle = \alpha 0\rangle + e^{i\pi/4}\beta 1\rangle$	$T 0\rangle = 0\rangle$ $T 1\rangle = e^{i\pi/4} 1\rangle$		
$R_x(\theta)$	$\begin{bmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}$	$\cos \frac{\theta}{2} 0\rangle\langle 0 - i \sin \frac{\theta}{2} 1\rangle\langle 0 -$ $- i \sin \frac{\theta}{2} 0\rangle\langle 1 + \cos \frac{\theta}{2} 1\rangle\langle 1 $	$R_x(\theta) \psi\rangle = (\alpha \cos \frac{\theta}{2} - i\beta \sin \frac{\theta}{2}) 0\rangle +$ $+ (\beta \cos \frac{\theta}{2} - i\alpha \sin \frac{\theta}{2}) 1\rangle$	$R_x(\theta) 0\rangle =$ $= \cos \frac{\theta}{2} 0\rangle - i \sin \frac{\theta}{2} 1\rangle$	$R_x(\theta) 1\rangle =$ $= \cos \frac{\theta}{2} 1\rangle - i \sin \frac{\theta}{2} 0\rangle$	
$R_y(\theta)$	$\begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}$	$\cos \frac{\theta}{2} 0\rangle\langle 0 + \sin \frac{\theta}{2} 1\rangle\langle 0 -$ $- \sin \frac{\theta}{2} 0\rangle\langle 1 + \cos \frac{\theta}{2} 1\rangle\langle 1 $	$R_y(\theta) \psi\rangle = (\alpha \cos \frac{\theta}{2} - \beta \sin \frac{\theta}{2}) 0\rangle +$ $+ (\beta \cos \frac{\theta}{2} + \alpha \sin \frac{\theta}{2}) 1\rangle$	$R_y(\theta) 0\rangle =$ $= \cos \frac{\theta}{2} 0\rangle + \sin \frac{\theta}{2} 1\rangle$	$R_y(\theta) 1\rangle =$ $= \cos \frac{\theta}{2} 1\rangle - \sin \frac{\theta}{2} 0\rangle$	
$R_z(\theta)$	$\begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$	$e^{-i\theta/2} 0\rangle\langle 0 + e^{i\theta/2} 1\rangle\langle 1 $	$R_z(\theta) \psi\rangle =$ $= \alpha e^{-i\theta/2} 0\rangle + \beta e^{i\theta/2} 1\rangle$	$R_z(\theta) 0\rangle = e^{-i\theta/2} 0\rangle$	$R_z(\theta) 1\rangle = e^{i\theta/2} 1\rangle$	
$R_1(\theta)$	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$	$ 0\rangle\langle 0 + e^{i\theta} 1\rangle\langle 1 $	$R_1(\theta) \psi\rangle = \alpha 0\rangle + \beta e^{i\theta} 1\rangle$	$R_1(\theta) 0\rangle = 0\rangle$	$R_1(\theta) 1\rangle = e^{i\theta} 1\rangle$	