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	*`elementType`*`[]`
Tuples	`(`*`type0`*`, `*`type1`*`, ...)` e.g.: `(Int, Qubit)`
Functions	*`input`*` -> `*`output`* e.g.: `ArcCos : (Double) -> Double`
Operations	*`input`*` => `*`output`*` is `*`variants`* e.g.: `H : (Qubit => Unit is Adj)`

User-Defined Types

Declare UDT with anonymous items	`newtype `*`Name`*` = (Type, Type);` e.g.: `newtype `*`Pair`*` = (Int, Int);`
Define UDT literal	*`Name`*`(`*`baseTupleLiteral`*`)` e.g.: `let origin = Pair(0, 0);`
Unwrap operator ! (convert UDT to underlying type)	*`VarName`*`!` e.g.: `let originTuple = origin!;` (now `originTuple = (0, 0)`)

Declare UDT with named items	`newtype `*`Name`*` =`
	`(`*`Name1`*`: Type, `*`Name2`*`: Type);`
	e.g.: `newtype Complex =`
	`(Re : Double, Im : Double);`

Accessing named items of UDTs	*`VarName`*`::`*`ItemName`* e.g.: `complexVariable::Re`

Update-and-reassign for named UDT items	`set `*`VarName`*` w/= `*`ItemName`*` <- `*`val`*`;`
	e.g.: `mutable p = Complex(0., 0.);`
	`set p w/= Re <- 1.0;`

Symbols and Variables

Declare immutable symbol	`let `*`varName`*` = `*`value`*
Declare mutable symbol (variable)	`mutable `*`varName`*` = `*`initialValue`*
Update mutable symbol (variable)	`set `*`varName`*` = `*`newValue`*
Apply-and-reassign	`set `*`varName operator`*`= `*`expression`* e.g.: `set counter += 1;`

Functions and Operations

Define function (classical routine)	`function `*`Name`*`(`*`in0`*` : `*`type0`*`, ...)`
	`: `*`returnType`*` {`
	`// `*`function body`*
	`}`

Call function	*`Name`*`(`*`parameters`*`)` e.g.: `let two = Sqrt(4.0);`

Define operation (quantum routine) with explicitly specified body, controlled and adjoint variants	`operation `*`Name`*`(`*`in0`*` : `*`type0`*`, ...)`
	`: `*`returnType`*` {`
	`body { ... }`
	`adjoint { ... }`
	`controlled { ... }`
	`adjoint controlled { ... }`
	`}`

Define operation with automatically generated adjoint and controlled variants	`operation `*`Name`*`(`*`in0`*` : `*`type0`*`, ...)`
	`: `*`returnType`*` is Adj + Ctl {`
	`...`
	`}`

Call operation	*`Name`*`(`*`parameters`*`)` e.g.: `Ry(0.5 * PI(), q);`
Call adjoint operation	`Adjoint `*`Name`*`(`*`parameters`*`)` e.g.: `Adjoint Ry(0.5 * PI(), q);`

Call controlled operation	`Controlled `*`Name`*`(`*`controlQubits`*`,`
	`parameters``)`
	e.g.: `Controlled Ry(controls,`
	`(0.5 * PI(), target));`

Control Flow

Iterate over a range of numbers	`for (`*`index`*` in `*`range`*`) {`
	`// Use integer `*`index`*
	`...`
	`}`
	e.g.: `for (i in 0..N-1) { ... }`

While loop (within functions)	`while (`*`condition`*`) {`
	`...`
	`}`

Iterate over an array	`for (`*`val`*` in `*`array`*`) {`
	`// Use value `*`val`*
	`...`
	`}`
	e.g.: `for (q in register) { ... }`

Repeat-until-success loop	`repeat { ... }`
	`until (`*`condition`*`)`
	`fixup { ... }`

Conditional statement	`if (`*`cond1`*`) { ... }`
	`elif (`*`cond2`*`) { ... }`
	`else { ... }`

Ternary operator	*`condition`*` ? `*`caseTrue`*` | `*`caseFalse`*
Return a value	`return `*`value`*
Stop with an error	`fail "`*`Error message`*`"`
Conjugations ($ABA^\dagger$ pattern)	`within { ... }` `apply { ... }`

Arrays

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

Slicing (subarray)	*`name`*`[`*`sliceRange`*`]`
	e.g.: if `t = [1, 2, 3, 4, 5]`, then
	`t[1 .. 3]` is `[2, 3, 4]`
	`t[3 ...]` is `[4, 5]`
	`t[... 1]` is `[1, 2]`
	`t[0 .. 2 ...]` is `[1, 3, 5]`
	`t[...-1...]` is `[5, 4, 3, 2, 1]`

Debugging (classical)

Print a string	`Message("Hello Quantum!")`
Print an interpolated string	`Message($"Value = {`*`val`*`}")`

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 (reg = Qubit[N]) {`
	`    // Qubits in reg start in |0⟩.`
	`    ...`
	`    // Qubits must be returned to |0⟩.`
	`}`

Allocate one qubit	`using (one = Qubit()) { ... }`

Allocate a mix of qubit registers and individual qubits	`using ((x, y, ... ) =`
	`    (Qubit[N], Qubit(), ... )) {`
	`    ...`
	`}`

Debugging (quantum)

Print amplitudes of wave function	`DumpMachine("dump.txt")`
Assert that a qubit is in $\|0\rangle$ or $\|1\rangle$ state	`AssertQubit(Zero, zeroQubit)` `AssertQubit(One, oneQubit)`

Measurements

Measure qubit in Pauli Z basis	`M(oneQubit)` yields a `Result` (`Zero` or `One`)
Reset qubit to $\|0\rangle$	`Reset(oneQubit)`
Reset an array of qubits to $\|0..0\rangle$	`ResetAll(register)`

Working with Q# from command line

Command Line Basics

Change directory	`cd dirname`
Go to home	`cd ~`
Go up one directory	`cd ..`
Make new directory	`mkdir dirname`
Open current directory in VS Code	`code .`

Working with Q# Projects

Create new project	`dotnet new console -lang Q#` `--output project-dir`
Change directory to project directory	`cd project-dir`
Build project	`dotnet build`
Run all unit tests	`dotnet test`

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	$\overline{a+bi} = a - bi$
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	$\begin{array}{cc} & n \text{ columns} \\ m \text{ rows} & \begin{bmatrix} a_{0,0} & \cdots & a_{0,n-1} \\ \vdots & \ddots & \vdots \\ a_{m-1,0} & \cdots & a_{m-1,n-1} \end{bmatrix} \end{array}$
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 $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}^\dagger$	$\begin{bmatrix} \overline{a} & \overline{e} \\ \overline{b} & \overline{d} \\ \overline{c} & \overline{f} \end{bmatrix}$
Inner product $\left\langle \begin{bmatrix} a \\ b \end{bmatrix}, \begin{bmatrix} c \\ d \end{bmatrix} \right\rangle$	$\begin{bmatrix} a \\ b \end{bmatrix}^\dagger \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \overline{a} & \overline{b} \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \overline{a}c + \overline{b}d$
Outer product $\begin{bmatrix} a \\ b \end{bmatrix}$ and $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}^\dagger = \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} \overline{x} & \overline{y} & \overline{z} \end{bmatrix} =$ $= \begin{bmatrix} a\cdot\overline{x} & a\cdot\overline{y} & a\cdot\overline{z} \\ b\cdot\overline{x} & b\cdot\overline{y} & b\cdot\overline{z} \end{bmatrix}$

Gates reference

Single Qubit gates

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

Two-qubit gates

Gate	Matrix Representation	Ket-Bra Representation	Applying to $\vert\psi\rangle = \alpha\vert 00\rangle + \beta\vert 01\rangle + \gamma\vert 10\rangle + \delta\vert 11\rangle$	Applying to basis states	
CNOT	$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & \mathbf{0} & \mathbf{1}\\ 0 & 0 & \mathbf{1} & \mathbf{0}\end{bmatrix}$	$\vert 00\rangle\langle 00\vert + \vert 01\rangle\langle 01\vert + \vert 11\rangle\langle 10\vert + \vert 10\rangle\langle 11\vert$ Or $\vert 0\rangle\langle 0\vert \otimes I + \vert 1\rangle\langle 1\vert \otimes X$	CNOT $\vert\psi\rangle = \alpha\vert 00\rangle + \beta\vert 01\rangle + \boldsymbol{\delta\vert 10\rangle + \gamma\vert 11\rangle}$	CNOT $\vert 00\rangle = \vert 00\rangle$ CNOT $\vert 01\rangle = \vert 01\rangle$	CNOT $\vert 10\rangle = \vert 11\rangle$ CNOT $\vert 11\rangle = \vert 10\rangle$
SWAP	$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \mathbf{0} & \mathbf{1} & 0\\ 0 & \mathbf{1} & \mathbf{0} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$	$\vert 00\rangle\langle 00\vert + \vert 01\rangle\langle 10\vert + \vert 10\rangle\langle 01\vert + \vert 11\rangle\langle 11\vert$	SWAP $\vert\psi\rangle = \alpha\vert 00\rangle + \boldsymbol{\gamma\vert 01\rangle + \beta\vert 10\rangle} + \delta\vert 11\rangle$	SWAP $\vert 00\rangle = \vert 00\rangle$ SWAP $\vert 01\rangle = \vert 10\rangle$	SWAP $\vert 10\rangle = \vert 01\rangle$ SWAP $\vert 11\rangle = \vert 11\rangle$
Controlled U	$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & a_{0,0} & a_{0,1}\\ 0 & 0 & a_{1,0} & a_{1,1}\end{bmatrix}$	$\vert 0\rangle\langle 0\vert \otimes I + \vert 1\rangle\langle 1\vert \otimes U$	C(U) $\vert\psi\rangle = \alpha\vert 00\rangle + \gamma\vert 01\rangle + (a_{0,0} + a_{0,1})\vert 10\rangle +$ $(a_{1,0} + a_{1,1})\vert 11\rangle$	C(U) $\vert 00\rangle = \vert 00\rangle$ C(U) $\vert 01\rangle = \vert 01\rangle$	C(U) $\vert 10\rangle = \vert 1\rangle \otimes (\alpha\vert 0\rangle + \gamma\vert 1\rangle)$ C(U) $\vert 11\rangle = \vert 1\rangle \otimes (\beta\vert 0\rangle + \delta\vert 1\rangle)$

Toffoli (CCNOT) gate

Gate	Matrix Representation	Ket-Bra Representation	Applying to $\vert\psi\rangle = \alpha\vert 000\rangle + \beta\vert 001\rangle + \gamma\vert 010\rangle +$ $\delta\vert 011\rangle + \epsilon\vert 100\rangle + \lambda\vert 101\rangle + \eta\vert 110\rangle + \kappa\vert 111\rangle$	Applying to basis states	
CCNOT	$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1}\\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0}\end{bmatrix}$	$(I_2 - \vert 11\rangle\langle 11\vert) \otimes I_1 + \vert 11\rangle\langle 11\vert \otimes X$	CCNOT $\vert\psi\rangle = \alpha\vert 000\rangle + \beta\vert 001\rangle + \gamma\vert 010\rangle +$ $\delta\vert 011\rangle + \epsilon\vert 100\rangle + \lambda\vert 101\rangle + \boldsymbol{\kappa\vert 110\rangle + \eta\vert 111\rangle}$	CCNOT $\vert 000\rangle = \vert 000\rangle$ CCNOT $\vert 001\rangle = \vert 001\rangle$ CCNOT $\vert 010\rangle = \vert 010\rangle$ CCNOT $\vert 011\rangle = \vert 011\rangle$	CCNOT $\vert 100\rangle = \vert 100\rangle$ CCNOT $\vert 101\rangle = \vert 101\rangle$ CCNOT $\vert 110\rangle = \vert 111\rangle$ CCNOT $\vert 111\rangle = \vert 110\rangle$