

# qSim How-To User Guide – v1.0

©GC – Dec 2022

## Table of Contents

1	Introduction.....	2
1.1	Purpose and Scope.....	2
1.2	qSim Overview.....	2
1.2.1	Main functionalities.....	2
1.2.2	Constraints and Limitations.....	3
1.2.3	Dependencies.....	3
1.3	qSim Interfaces.....	3
2	Use & Access How-To.....	4
2.1	Build.....	4
2.2	Launch.....	4
2.3	Interfaces.....	4
2.3.1	TCP/IP Socket Interface.....	4
2.3.2	General Conventions.....	5
2.4	Datastream Details.....	5
2.4.1	TCP/IP Socket Datastreams.....	5
2.4.1.1	qSim-connection Datastream.....	5
2.4.1.2	qSim-quireg-control Datastream.....	6
2.4.1.3	qSim-quireg-transformation Datastream.....	6
2.4.1.4	qSim-response Datastream.....	6
2.4.2	TCP/IP Socket Messages and Parameters.....	6
2.4.2.1	Client Register.....	6
2.4.2.2	Client Unregister.....	7
2.4.2.3	Qureg Allocate.....	7
2.4.2.4	Qureg Release.....	8
2.4.2.5	Qureg State Reset.....	9
2.4.2.6	Qureg State Set.....	9
2.4.2.7	Qureg State Transformation.....	10
2.4.2.8	Qureg State Peek.....	11
2.4.2.9	Qureg State Measurement.....	12

# 1 Introduction

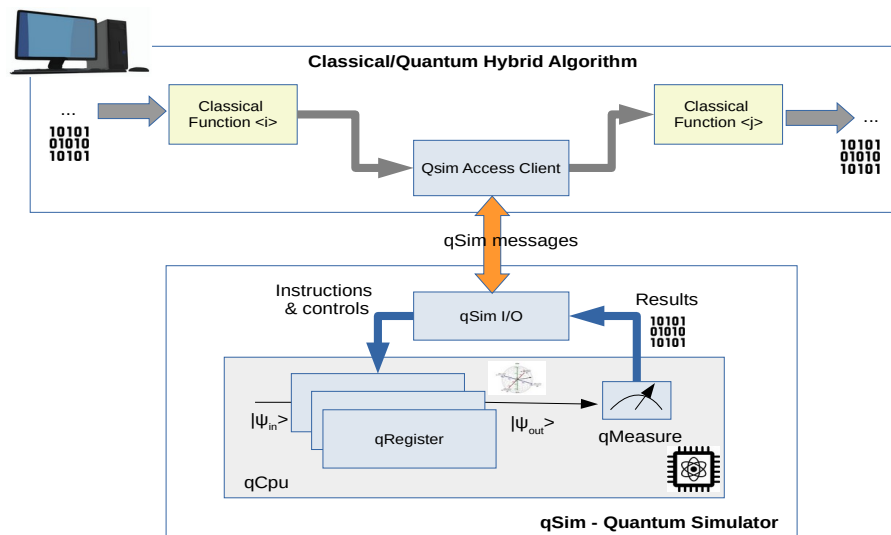
## 1.1 Purpose and Scope

The purpose of this note is to describe how to use the Quantum Simulator (qSim) from an external client for running QC algorithms. It focuses on the qSim external interfaces and the relevant data units transferred back and forth.

The scope is limited to external interfaces only and qSim internal details are not described here.

## 1.2 qSim Overview

The Quantum Simulator (qSim) is simulator able to provide an high performance universal QC machine, running on CPU or GPU based hardware.



### 1.2.1 Main functionalities

The functionalities provided by qSim are:

- qubit registers (quregs) handling
- universal 1-qubit and 2-qubits quantum gate set (e.g. Pauli, Hadamard, identity, phase shift, CX, CY, Cz, etc.)
- high level blocks for common n-qubits transformations (e.g. swap and controlled swap)
- quregs measurement

In addition, some diagnostic & test helper functionalities are also supported:

- arbitrary qureg state setup
- qureg state access (max 10 qubits qureg)

### **1.2.2 Constraints and Limitations**

The qSim limitations are the following:

- max qubit size for transformations: 20 qubits
- max qubit size for arbitrary state setup or peek: 10 qubits

### **1.2.3 Dependencies**

The qSim requires CUDA libraries installed in the target platform.

## **1.3 qSim Interfaces**

The qSim supports the following client access interfaces:

- TCP/IP Socket, for direct access to qSim

Their description is provided in section 2.3 .

## 2 Use & Access How-To

### 2.1 Build

The qSim executable can be built in two ways:

1. from Eclipse importing the repository
2. manually, using the Makefile provided in the build\_make folder

In both cases the CUDA development toolkit for the target hardware has to be available and configured in the include and library build variables using the actual paths and CUDA capability, namely:

- NVCC := nvcc
- CUDA\_ARCH := -gencode arch=compute\_61,code=sm\_61
- CUDA\_INC := /usr/local/cuda-9.2/include
- CUDA\_LIB := /usr/local/cuda-9.2/lib64

### 2.2 Launch

The qSim executable can be launched with optional command line arguments which allows specific functionalities setup, when needed. More specifically:

- silent mode (default)  
`./qSim_sk`
- diagnostic mode  
`./qSim_sk -verbose`

### 2.3 Interfaces

qSim provides the following interfaces to external clients for accessing and use the QC functionalities provided:

- TCP/IP socket based access

#### 2.3.1 TCP/IP Socket Interface

The default connection parameters for this interface are the following.

Parameter	Value
IP address	127.0.0.1 (localhost)
Port	27020
Connection initiator	<client>

Stream type	ASCII
-------------	-------

The messages are in ASCII format and their syntax is the following:

```
// coding format: ASCII string with "|" as field separators
//   <encoded_instruction> = <counter>"|"<id>"|"<params>
//
// with ":" as param tag-value pair separator
//   <params> = <tag_1>=<value_1>:<tag_2>=<value_2>: ...<tag_n>=<value_n>:
```

## 2.3.2 General Conventions

The data types supported in qSim messages are the following:

- integer values: encoded as string value
- double values: encoded in decimal format with decimal digits as per required precision
- index range values: encoded as string value pair in parenthesis “(<index<sub>1</sub>>, <index<sub>2</sub>>)”
- complex state values: encoded in decimal pair format for real and imaginary parts, with decimal digits as per required precision “(<real\_val>, <imag\_val>)”
- complex state value array: encoded as comma separated sequence of complex state values “(<real\_val<sub>1</sub>>, <imag\_val<sub>1</sub>>), ..., (<real\_val<sub>n</sub>>, <imag\_val<sub>n</sub>>)”

## 2.4 Datastream Details

### 2.4.1 TCP/IP Socket Datastreams

This case allows a client to exchange socket messages with qSim, through the following datastreams:

- client => qSim
  - qSim-connection
  - qSim-quireg-control
  - qSim-qreg-transformation
- qSim => client
  - qSim-response

The client <==> qSim handshake is pretty simple and it foresees a synchronous exchange of messages from client to qSim and back.

#### 2.4.1.1 qSim-connection Datastream

This datastream allows a client to establish and manage the connection with qSim. The messages part of the datastream are:

- Client Register
- Client Unregister

Message details are in 2.4.2 .

### **2.4.1.2 *qSim-quireg-control Datastream***

This datastream allows a client to setup and manage qureg within qSim. The messages part of the datastream are:

- Qureg Allocate
- Qureg Release
- Qureg State Reset
- Qureg State Set
- Qureg State Measurement
- Qureg State Peek (diagnostic)

Message details are in 2.4.2 .

### **2.4.1.3 *qSim-quireg-transformation Datastream***

This datastream allows a client to perform qureg transformations with qSim. The messages part of the datastream are:

- Qureg State Transform

Message details are in 2.4.2 .

### **2.4.1.4 *qSim-response Datastream***

This datastream allows a client to receive responses for all the messages sent to qSim. The messages part of the datastream are:

- Message Response

Message details are in 2.4.2 .

## **2.4.2 TCP/IP Socket Messages and Parameters**

The description of the qSim messages is provided in this section. Each sub-para includes the client => qSim message and the relevant response.

### **2.4.2.1 *Client Register***

This message allow a client to register to qSim for being allowed to operate on quregs. It receives a token to be provided in all qreg related messages. The relevant arguments and parameters are the following.

**Client => qSim message**

Message ID	1
------------	---

Parameter	Tag	Type	Value
Client ID	id	string	<client mnemonic id>

#### **qSim => Client response**

Message ID	20
------------	----

Parameter	Tag	Type	Value
Result	result	string	<message exec result (Ok/Not-Ok)>
Error	error	string	<error details> only for result Not-Ok
Client token	token	string	<client registration token>

### **2.4.2.2 Client Unregister**

This message allow a client to deregister to qSim and to stop the connection in a clean way. The relevant arguments and parameters are the following.

#### **Client => qSim message**

Message ID	2
------------	---

Parameter	Tag	Type	Value
Client token	token	string	<client registration token>

#### **qSim => Client response**

Message ID	20
------------	----

Parameter	Tag	Type	Value
Result	result	string	<message exec result (Ok/Not-Ok)>
Error	error	string	<error details> only for result Not-Ok

### **2.4.2.3 Qureg Allocate**

This message allow a client to allocate a qureg in qSim, and to receive an handler to be used to reference the qureg for further operations. The relevant arguments and parameters are the following.

#### **Client => qSim message**

Message ID	10
------------	----

Parameter	Tag	Type	Value
Client token	token	string	<client registration token>
Qureg size	qr_n	int	<qureg size in qubits>

#### **qSim => Client response**

Message ID	20
------------	----

Parameter	Tag	Type	Value
Result	result	string	<message exec result (Ok/Not-Ok)>
Error	error	string	<error details> only for result Not-Ok
Qureg handler	qr_h	int	<qureg handler>

### **2.4.2.4 Qureg Release**

This message allow a client to deallocate a qureg in qSim, based on given handler. The relevant arguments and parameters are the following.

#### **Client => qSim message**

Message ID	11
------------	----

Parameter	Tag	Type	Value
Client token	token	string	<client registration token>
Qureg handler	qr_h	int	<qureg handler>

#### **qSim => Client response**

Message ID	20
------------	----

Parameter	Tag	Type	Value
Result	result	string	<message exec result (Ok/Not-Ok)>
Error	error	string	<error details> only for result Not-Ok



### 2.4.2.5 Qureg State Reset

This message allow a client to reset the state of a qureg in qSim, based on given handler. The relevant arguments and parameters are the following.

#### Client => qSim message

Message ID	12
------------	----

Parameter	Tag	Type	Value
Client token	token	string	<client registration token>
Qureg handler	qr_h	int	<qureg handler>

#### qSim => Client response

Message ID	20
------------	----

Parameter	Tag	Type	Value
Result	result	string	<message exec result (Ok/Not-Ok)>
Error	error	string	<error details> only for result Not-Ok

### 2.4.2.6 Qureg State Set

This message allow a client to set an excited state or an arbitrary state (optional – diagnostic case) in a qureg in qSim, based on given handler. The relevant arguments and parameters are the following.

#### Client => qSim message

Message ID	13
------------	----

Parameter	Tag	Type	Value
Client token	token	string	<client registration token>
Qureg handler	qr_h	int	<qureg handler>
Qureg state values	qr_stVal	Complex array	<qureg state values> - optional for arbitrary state setup only

#### qSim => Client response

Message ID	20
------------	----

Parameter	Tag	Type	Value
Result	result	string	<message exec result (Ok/Not-Ok)>
Error	error	string	<error details> only for result Not-Ok

### 2.4.2.7 Qureg State Transformation

This message allow a client to apply a transformation function to a qureg states in qSim, based on given handler. The relevant arguments and parameters are the following.

#### Client => qSim message

Message ID	14
------------	----

Parameter	Tag	Type	Value
Client token	token	string	<client registration token>
Qureg handler	qr_h	int	<qureg handler>
Function type	f_type	int	<function type> from following list <i>1-qubit</i> <ul style="list-style-type: none"> <li>• Q1_I = 0</li> <li>• Q1_H = 1</li> <li>• Q1_X = 2</li> <li>• Q1_Y = 3</li> <li>• Q1_Z = 4</li> <li>• Q1_SX = 5</li> <li>• Q1_PS = 6</li> <li>• Q1_T = 7</li> <li>• Q1_S = 8</li> <li>• Q1_Rx = 9</li> <li>• Q1_Ry = 10</li> <li>• Q1_Rz = 11</li> </ul> <i>2 qubits</i> <ul style="list-style-type: none"> <li>• Q2_CU = 12</li> <li>• Q2_CX = 13</li> <li>• Q2_CY = 14</li> <li>• Q2_CZ = 15</li> </ul> <i>n qubits</i> <ul style="list-style-type: none"> <li>• Qn_MCSLRU = 16</li> </ul> <i>blocks</i> <ul style="list-style-type: none"> <li>• Q1_SWAP = 100</li> <li>• Qn_SWAP = 101</li> <li>• Q1_CSWAP = 102</li> <li>• Qn_CSWAP = 103</li> </ul>
Function size	f_size	int	<function size in states>

Function repetitions	f_rep	int	<function repetitions in the qureg>
Function LSQ	f_lqs	int	<function LSQ index in the qureg>
Function control range	f_cRange	Int (start, stop) interval	<function control start-stop indexes> only for n-qubits case
Function target range	f_tRange	Int (start, stop) interval	<function target start-stop indexes> only for n-qubit case
Controlled U-function type	f_uType	int	<controlled U-function type>, only for n-qubit case
Function arguments	f_args	Double list	<function or controlled U-function arguments> <i>1-qubit function</i> <ul style="list-style-type: none"> <li>arg #0: phase value (PS, Rx, Ry, Rz cases)</li> </ul> <i>2-qubit controlled U-function</i> <ul style="list-style-type: none"> <li>arg #0: controlled U-function control range</li> <li>arg #1: controlled U-function target range</li> <li>arg #2: phase value (PS, Rx, Ry, Rz cases)</li> </ul>

#### **qSim => Client response**

Message ID	20
------------	----

Parameter	Tag	Type	Value
Result	result	string	<message exec result (Ok/Not-Ok)>
Error	error	string	<error details> only for result Not-Ok

### **2.4.2.8 Qureg State Peek**

This message allow a client to peek the state values of a qureg in qSim, based on given handler. The relevant arguments and parameters are the following.

#### **Client => qSim message**

Message ID	15
------------	----

Parameter	Tag	Type	Value
Client token	token	string	<client registration token>
Qureg handler	qr_h	int	<qureg handler>

### **qSim => Client response**

Message ID	20
------------	----

Parameter	Tag	Type	Value
Result	result	string	<message exec result (Ok/Not-Ok)>
Error	error	string	<error details> only for result Not-Ok
Qureg state values	qr_stVal	Complex array	<qureg state values>

Note: the usage of this message is constrained by the size of the qureg to inspect, with a limit of 10 qubits size applied.

### ***2.4.2.9 Qureg State Measurement***

This message allow a client to measure fully or partially the state a qureg in qSim, based on given handler, and collapsing the state due to the measure. The relevant arguments and parameters are the following.

### **Client => qSim message**

Message ID	16
------------	----

Parameter	Tag	Type	Value
Client token	token	string	<client registration token>
Qureg handler	qr_h	int	<qureg handler>
Measure qubit start index	qr_mQidx	int	<(sub)qureg start index>
Measure qubit length	qr_mQlen	int	<(sub)qureg length in qubits>
Measure random flag	qr_mRand	int	<random vs. expectation measure flag >
Measure state collapse flag	qr_mStColl	int	<real-state collapse vs. simulated measure flag>

### **qSim => Client response**

Message ID	20
------------	----

Parameter	Tag	Type	Value
-----------	-----	------	-------

Result	result	string	<message exec result (Ok/Not-Ok)>
Error	error	string	<error details> only for result Not-Ok
Measure state index	m_stIdx	int	<measure state index>
Measure state expectation	m_stExp	double	<measure state expectation value>
Measure state collapse indexes	qr_mStColl	Int list	<list of collapsed qureg state indexes>