QuanTile Software Interface Control Document (ICD)

1. Introduction

QuanTile Software Interface Control Document (ICD) contains detailed information about the data items communicating QuanTile circuit components. QuanTile ICD is a data packaging protocol and thus does not depend upon a specific physical communications medium or transport protocol. Any suitable physical medium may be used. The transport protocol(s) used should depend upon performance and what is appropriate to the communications hardware and components involved.

QuanTile communication is an asynchronous operation. Unless specified, all communication messages are independent from each other. There is no requirement that any message must proceed before another. This allows any component to initiate a communication to others by sending a request or providing information. The data transition can be pulling (providing upon request) or pushing (providing without request). In pulling, a request from a connected component should not assume the request will be answered or even the connection exists. It however can set an appropriate timeout internationally and use a default answer if the answer is not provided. In pushing, a component can provide data without being asked. It makes no assumption that someone is connected or listening. This prototype is using pulling.

The ICD specifies two types of communication, peer-to-peer (P2P) and bus.

- P2P connection is used for simple service requests and responses between two components. For example, a tile may use a P2P request to query the quantum state of another tile. One component either knows exactly what the other component is or doesn't care what it is. They communicate through an exclusive channel and treat each other as a port to a data tunnel which is only connected to the two components. P2P is mainly used to send a request or pass a quantum state between two connected components on a **horizontal** line in a quantum circuit presentation. The physical layer implementation of this communication is not in the scope of this document, so any peer-to-peer implementation will work. The following figure illustrates an implementation using UART connection.

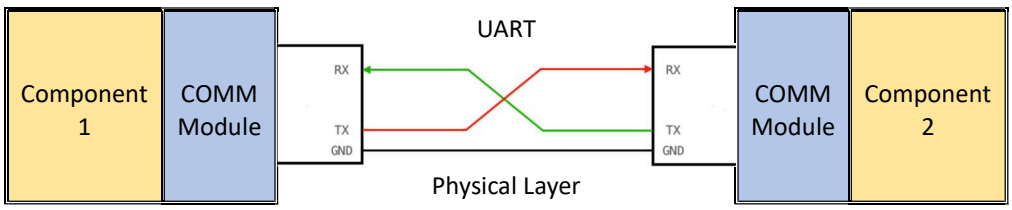

Figure 1 – P2P Connection with UART

- Bus connection is used between two or among more than two components. It is mainly used to pass quantum states and component roles along a **vertical** line of components in a quantum circuit presentation. The purpose of such communication is to handle entangled qubits. It needs to assign roles and unique identifications to the components involved. Same as P2P connection, this ICD keeps

the physical layer of bus connection out of its scope. It does not enforce its implementation details. The following illustrates an implementation using I2C.

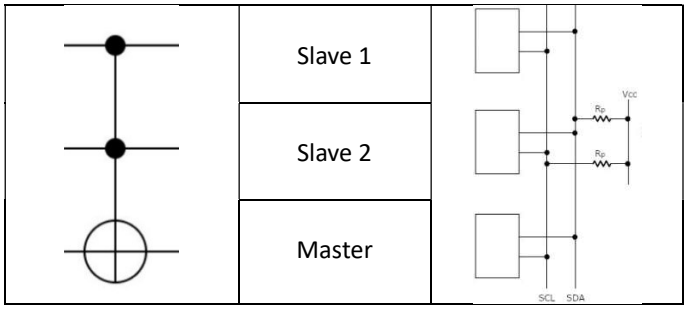

Figure 2 – Bus Connection with I2C

2. Basic Concepts

At a high-level perspective regarding message passing, each component within the QuanTile system executes identical patterns of operation. The software running on every component functions with two threads: one responsible for gathering data and calculating the output state of the tile, while the other facilitates data services to address requests from other components. Some requests expect responses, although not all requests necessitate one. However, even when a request does expect a response, there's no assurance that a response will be received, especially considering the abrupt changes in connections. When an expected response is missing, the component should follow a predefined action or assumption of the component. For instance, if a tile anticipates a response containing quantum state data but doesn't receive it before a timeout, it will default to assuming the state is $|0\rangle$. If the same missing quantum state happens to the data gateway, it will treat as no tile connected to it. Conversely, if the data gateway encounters a missing quantum state response, it will interpret this as an indication that no tile is connected to it.

2.1 Component Types

QuanTile accommodates the component types outlined in *Table 1*. Transformation matrices need not be explicitly specified. Moreover, users have the flexibility to define custom component types by providing a transformation matrix and designating the type as **USER_DEFINED**.

Enumeration Name	Symbol	Code	# of Qubits	Note
UNDEFINED		0x00	N/A	
IDENTITY	I	0x01	1	Identity
PAULI_X	X	0x02	1	Pauli X
PAULI_Y	Y	0x03	1	Pauli Y
PAULI_Z	Z	0x04	1	Pauli Z
HADAMARD	H	0x05	1	Hadamard
RX_4	$R_{X\,\pi/2}$	0x06	1	Rotate π/2 along X axis
RY_4	$R_{Y\,\pi/2}$	0x07	1	Rotate π/2 along Y axis
RZ_4	$R_{Z\,\pi/2}$	0x08	1	Rotate π/2 along Z axis
RX_8	$R_{X\,\pi/4}$	0x09	1	Rotate π/4 along X axis
RY_8	$R_{Y\,\pi/4}$	0x0a	1	Rotate π/4 along Y axis
RZ_8	$R_{Z\,\pi/4}$	0x0b	1	Rotate π/4 along Z axis
PHASE	P, S	0x0e	1	Shift phase by π/2
TWIRL	T	0x0f	1	
CONTROLLED_NOT	CNOT, CX	0x41	2	C-Not gate
CONTROLLED_Z	CZ	0x42	2	C-Z gate
SWAP	SWAP	0x43	2	
TOFFOLI	CCNOT, CCX, TOFF	0x81	3	C-C-Not gate
USER_DEFINED		0xFF	TBD	Custom transformation

Table 1: Component Types and Codes

3. Synchronization Header

Given the resource constraints of microcontrollers in the tiles, only UART and I2C hardware communication protocols are utilized in this project. Despite their simplicity, QuanTile demonstrates robustness in handling abrupt power-up, power-down, connection, and disconnection activities of components. To maintain communication data alignment, a 4-byte synchronization header is implemented. Before transmitting any message, the sender must include this synchronization header. Similarly, the receiver must successfully receive all 4 bytes of the synchronization header in sequence before interpreting the subsequent message.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1: 0x7F								2: 0xFF								3: 0xFF								4: 0xFF							

4. Data Block Types

The following data block types are supported, with details provided in subsequent subsections.

Command ID	ID in Hex	Function

1	0x01	Quantum state request
2	0x02	Component type request
3	0x03	Component type assignment
129	0x81	Quantum state response
130	0x82	Component type response

Table 2: Data Block Types and Codes

4.1 Data Block Header

All message start with a 4-byte data header as the following. Some messages may require only 4 bytes, while others may require additional length.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1: command = [1,255]								2: TBD								3: TBD								4: ICD revision							

The first byte (command) depicts the message type. The 4th byte specifies the ICD revision which may be used to detect foreseeable compatibility issue in the future. The meaning of the 2nd and 3rd bytes depends on the message type. During the development and pre-release phases, the ICD revision is always set to 0.

4.2 Quantum State Request

This message serves to request a quantum state from another component, typically a tile. The data block contains no additional data beyond the header. However, it employs the second byte to specify the anticipated number of qubits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1: command = 1								2: # of qubits								3: format								4: ICD revision							

1. Command ID = 0x01
2. Number of qubits in the request. It is set to 1 in a single qubit tile and can exceed 1 in the case of a multi-qubit gate tile. This operation assumes that connected tiles share the same number of qubits. In case of a mismatch, tiles should visually (LED) or audibly (buzzer) indicate that the connection is not valid if such features are available.
3. Data format: This data field specifies the preferred data format of the requesting component. However, the connected component providing the response is not obliged to agree with this format. Instead, the responding entity can specify the format in the response itself. Possible values are:

   0 → undefined. The receiver of this message decides the data format when responds.

   1 → theta-phi. The requester prefers theta-phi quantum state format.

   Other formats to be determined in future development.

4. ICD revision

4.3 Component Type Request

This message sends a request to another component for its type. The data block contains no additional data beyond the data block header.

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
1: command=2	2: Spare	3: Spare	4: ICD revision

1. Command ID = 0x02
2. Spare
3. Spare
4. ICD revision

4.4 Component Type Assignment

This message sends a request to change the component type of its connected entity. The assignment is a request for change, but the connected entity is not required to comply with the request.

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
1: command=3	2: Component type	3: Data length	4: ICD revision
5: User-defined data :			

1. Command ID = 0x03
2. Component type: One of the values in the component type *Table 1*.
3. Data length: The length of data in bytes in the user-defined data. This number is used to determine the size of the data block, which is the number plus 1. If data field 2 is not USER_DEFINED, data field 3 should be set to 0. The data format is TBD when it is not 0.
4. ICD revision
5. User-defined data: When the component type is used-defined, user data is passed to the requester.

4.5 Quantum State Response

This is a general message to send a quantum state to another component. It consists of a data block header to specify number of qubits of the connection line and the format of this response. The length of the data follows depends on the value of the 2nd and 3rd data fields.

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
1: command = 129	2: # of qubits	3: format	4: ICD revision
5: data :			

1. Command ID = 0x81
2. Number of qubits in the response.
3. Data format: This data field specifies the data format of the quantum state in the response. Possible values are:

   0 → undefined. The receiver of this message decides the data format when responds.

   1 → theta-phi. The requester prefers theta-phi quantum state format.

   Other formats to be determined in future development.

4. ICD revision
5. Data: data format is determined by the 3rd byte and data length is determined by both 2nd and 3rd data fields.

For example, a single-qubit and double-qubit quantum state messages with theta-phi format would look like the following.

Single-qubit example:

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
1: command = 129	2: # of qubits = 1	3: format = 1 (theta-phi)	4: ICD revision = 0
5: theta			
6: phi			

5. Theta in radians
6. Phi in radians

Two-qubit example:

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
1: command = 129	2: # of qubits = 2	3: format = 1 (theta-phi)	4: ICD revision = 0
5: theta_0			
6: phi_0			
7: theta_1			
8: phi_1			

5. Theta of the lowest qubit in radians
6. Phi of the lowest qubit in radians
7. Theta of the second lowest qubit in radians
8. Phi of the second lowest qubit in radians

4.6 Component Type Response

This message sends a component type to the requester.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1: command=130								2: Component type								3: Data length								4: ICD revision							
5: User-defined data :																															

1. Command ID = 0x82
2. Component type: One of the values in the component type in *Table 1*.
3. Data length: The length of data in bytes in the user-defined data. This number is used to determine the size of the data block, which is the number plus 1. If data field 2 is not USER_DEFINED, data field 3 should be set to 0. The data format is TBD when it is not 0.
4. ICD revision
5. User-defined data: When the component type is used-defined, user data is passed to the requester.

In revision 0, we only support predefined component types for quantum gates. The communication data block would appear as follows.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1: command=130								2: Component type								3: Data length=0								4: ICD revision							