

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)[Qiskit / qiskit-terra](#) Public[Watch](#) 217[Fork](#) 1.6k[Star](#) 3.2k[Code](#) [Issues 593](#) [Pull requests 185](#) [Actions](#) [Projects 11](#) [Security](#) [Insights](#)

# Qasm exporter creates invalid Qasm for subcircuits with classical registers

## #XXXX

[Edit](#)[New issue](#)[Open](#) **ANONYMOUS** opened this issue 8 days ago · 1 comment**ANONYMOUS** commented 8 days ago

### Environment

- Qiskit Terra version: 0.19.1
- Python version: 3.8
- Operating system: Ubuntu 18.04.6 LTS

### What is happening?

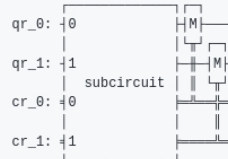
Converting a circuit with a subcircuit to qasm fails, when this subcircuit contains also classical registers.

### How can we reproduce the issue?

Run this:

```
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
qr = QuantumRegister(2, name='qr')
cr = ClassicalRegister(2, name='cr')
qc = QuantumCircuit(qr, cr, name='qc')
subcircuit = QuantumCircuit(qr, cr, name='subcircuit')
subcircuit.x(qr[0])
qc.append(subcircuit, qargs=qr, cargs=cr)
qc.measure(qr, cr)
qc.draw(fold=-1)
```

Output:



Convert to qasm:

```
qc.qasm(formatted=True)
```

Output:

```
OPENQASM 2.0;
include "qelib1.inc";
gate subcircuit q0,q1 { x q0; }
qreg qr[2];
creg cr[2];
subcircuit qr[0],qr[1],cr[0],cr[1];
measure qr[0] -> cr[0];
measure qr[1] -> cr[1];
```

Now, if we want to reuse this qasm and load it:

```
qc = QuantumCircuit.from_qasm_str(qc.qasm())
```

We get the following error:

```
qiskit/circuit/quantumcircuit.py in from_qasm_str(qasm_str)
    2362     """
    2363     qasm = Qasm(data=qasm_str)
-> 2364     return _circuit_from_qasm(qasm)
    2365
    2366     @property
qiskit/circuit/quantumcircuit.py in circuit from qasm(qasm)
```

### Assignees

No one assigned

### Labels

[bug](#) [help wanted](#) [qasm](#)

### Projects

None yet

### Milestone

No milestone

### Development

No branches or pull requests

### Notifications

[Customize](#)[Unsubscribe](#)

You're receiving notifications because you authored the thread.

2 participants

```

4695     from qiskit.converters import dag_to_circuit
4696
-> 4697     ast = qasm.parse()
4698     dag = ast_to_dag(ast)
4699     return dag_to_circuit(dag)

qiskit/qasm/qasm.py in parse(self)
51         with QasmParser(self._filename) as qasm_p:
52             qasm_p.parse_debug(False)
--> 53             return qasm_p.parse(self._data)

qiskit/qasm/qasmparser.py in parse(self, data)
1138     def parse(self, data):
1139         """Parse some data."""
-> 1140         self.parser.parse(data, lexer=self.lexer, debug=self.parse_deb)
1141         if self.qasm is None:
1142             raise QasmError("Uncaught exception in parser; " + "see previous messages for details.")

ply/yacc.py in parse(self, input, lexer, debug, tracking, tokenfunc)
331         return self.parseopt(input, lexer, debug, tracking, tokenfunc)
332     else:
-> 333         return self.parseopt_notrack(input, lexer, debug, tracking, tokenfunc)
334
335

ply/yacc.py in parseopt_notrack(self, input, lexer, debug, tracking, tokenfunc)
1118             del symstack[-plen:]
1119             self.state = state
-> 1120             p.callable(pslice)
1121             del statestack[-plen:]
1122             symstack.append(sym)

qiskit/qasm/qasmparser.py in p_unitary_op_2(self, program)
704         """
705         program[0] = node.CustomUnitary([program[1], program[2]])
-> 706         self.verify_as_gate(program[1], program[2])
707         self.verify_reg_list(program[2], "qreg")
708         self.verify_distinct([program[2]])

qiskit/qasm/qasmparser.py in verify_as_gate(self, obj, bitlist, arglist)
158
159         if g_sym.n_bits() != bitlist.size():
-> 160             raise QasmError(
161                 "Gate or opaque call to '" + obj.name + "' uses",
162                 str(bitlist.size()),

QasmError: "Gate or opaque call to 'subcircuit' uses 4 qubits but is declared for 2 qubits line 6 file "

```

### What should happen?

I would have expected the exported qasm to be a valid qasm (aka to be read again) and also give me back the original exported circuit if I decide to store my circuit on a file as qasm and pass it to someone else.

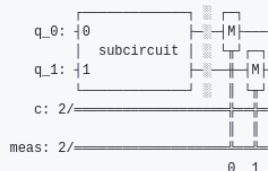
### Any suggestions?

I would have expected the following QASM:

```

qasm_expected = """
OPENQASM 2.0;
include "qelib1.inc";
gate subcircuit q0,q1 { x q0; }
qreg q[2];
creg c[2];
creg meas[2];
subcircuit q[0],q[1];
barrier q[0],q[1];
measure q[0] -> meas[0];
measure q[1] -> meas[1];
"""
qc = QuantumCircuit.from_qasm_str(qasm_expected)
qc.draw(fold=-1)

```



But this is also inaccurate, since in the original `subcircuit` might as well use the classical registers to perform some measurement, whereas with this qasm representation it is not possible to express this.

Thus, I am not sure on how we should proceed and I very curious to listen to your feedback on this.

Thanks in advance



ANONYMOUS added the bug label 8 days ago

QISKIT DEV commented 8 days ago • edited

Contributor

There is no possible valid OpenQASM 2 programme for the circuit you're describing; there's no subroutine-like construct that can take classical parameters. The error message could be better, but there is no way to produce valid OpenQASM 2 in this situation, because the object has no way of being represented. The bug here is that the QASM 2 exporter should have rejected the circuit out-of-hand.

I haven't looked at the QASM 2 exporter code for a while, but I think it tries to export all `Instruction` instances currently, whereas it should fail on anything other than a special case or a gate, due to limitations in the QASM 2 language. Probably the fix involves modifying the exporter step that tries to find the definition of each object in a circuit so that it goes through the following steps:

- if a `Gate` instance, proceed as it currently does
- if `Barrier`, `Measure` or `Reset`, output the specialist QASM 2 statements
- if not a `Gate` (i.e. an unknown `Instruction`), then fail with a message saying that QASM 2 cannot represent non-unitary operations (except for the built-in `measure` and `reset`).

The way you have added the subcircuit to your circuit, it's as an `Instruction` not a `Gate`, so even if it had *no* classical registers, in my new scheme it would still be rejected as non-unitary (that's the difference between `Instruction` and `Gate`). You should call `QuantumCircuit.to_gate` to avoid that. This is an important note for API stability in the bug fix, though - people may be relying on the QASM 2 exporter working in similar situations, so we should take care to check that we don't break anybody's workflow.

OpenQASM 3 can represent non-unitary subroutines, but at the moment Terra's QASM 3 exporter is quite limited in what it can support, since that language spec is still evolving and so is Terra's capability to represent dynamic circuits.



**QISKIT DEV** added `help wanted` `qasm` labels 8 days ago

Write Preview

H B I

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Close issue

Comment

Remember, contributions to this repository should follow its [contributing guidelines](#) and [code of conduct](#).