Quantum Control System (QCS)

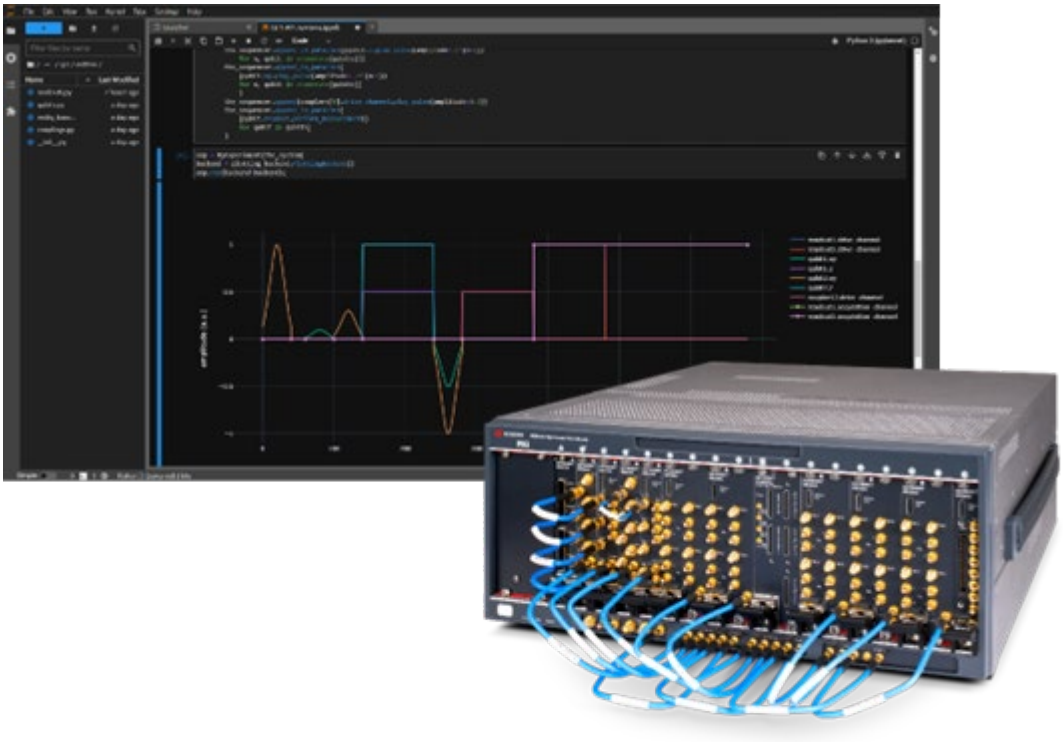
QCS software

Introduction: Accelerating Quantum Innovation

The Keysight Quantum Control System comes with powerful and intuitive software that enables easy execution of textbook quantum experiments and the flexibility to quickly design and execute custom pulse sequences, all while automatically ensuring tight timing and synchronization.

The software is controlled through the QCS API, a Python API that lets users program experiments at the level of their quantum device. To enable this, users create a system configuration file to define the mapping of their quantum system onto physical channels of the QCS. When this has been created, sequences or experiments can be written in terms of objects like qubits, couplers, and readout resonators.

This API leverages the FPGAs on the associated QCS hardware, enabling real-time performance and features without the need for complicated FPGA programming.



KEYSIGHT

Quantum Configuration

Users can define the classical components, the quantum components, and the mapping between them in a configuration YAML file. The classical components consist of Instruments that correspond to Keysight PXIe modules in the QCS. The quantum components are the objects in the quantum processor that these Instruments control, such as Qubits, Readouts, and Couplers.

Available instruments include:

- M5200A PXIe Digitizer
- M5201A PXIe Down Converter
- M5300A PXIe RF AWG

Available quantum components include:

- Tunable Qubit
- Tunable Coupler
- Readout Resonator

Additionally, the API is designed to be easily extensible to allow users to create or customize additional components to match the user's unique system. When a new quantum component and its mapping to classical channels is defined, the software can use that component throughout the stack.

Experiment Programming

When a quantum configuration is created, measurements and pulse sequences can be defined using the Experiment class. An example experiment is shown in Figure 1 below.

The QCS API provides the ability to run experiments on several different backends. This provides options for analyzing and visualizing the sequence as well as running it on hardware:

- Printing Backend
- Plotting Backend
- Waveform Upload Backend

Printing backend – The printing backend provides a complete list of the instructions and timings on each channel in the experiment. This can be used to examine fine details of the experiment and for fine debugging. After an experiment has been created, it can be executed using several different backends

```python
from keysight.qcs.experiment import Experiment
from keysight.qcs.sequence import sequence_builder
from keysight.qcs.entities import qubits

class ExampleExperiment(Experiment):
    def make_sequence(
        self,
        the_sequencer: sequence_builder.SequenceBuilder
    ):

        qubits = self.system.get_instances(qubits.TunableQubit)

        the_sequencer.append(qubits[0].xy.play_pulse(amplitude=0.5))

        the_sequencer.append_in_parallel(
            [qubit.z.play_pulse(amplitude=0.2*(n+1))
            for n, qubit in enumerate(qubits)]
            )

        the_sequencer.delay(20e-9)

        the_sequencer.append_in_parallel(
            [qubit.xy.play_pulse(amplitude=-.5*(n+1))
            for n, qubit in enumerate(qubits)]
            )
```

Figure 1. Example experiment

Backends and Experiment Execution

The QCS API provides the ability to run experiments on a number of different backends. This provides options for analyzing and visualizing the sequence as well as running it on hardware:

- Printing Backend
- Plotting Backend
- Waveform Upload Backend

Printing backend – The printing backend provides a complete list of the instructions and timings on each channel in the experiment. This can be used to examine fine details of the experiment and for fine debugging.

```
readout1.drive_channel
    DelayInstruction(length=4e-08)
    DelayInstruction(length=4e-08)
    DelayInstruction(length=2e-08)
    DelayInstruction(length=4e-08)
qubit1.xy
    PulseInstruction(pulse=GaussianPulse, length=4e-08, parameters={'amplitude': 0.5})
    DelayInstruction(length=4e-08)
    DelayInstruction(length=2e-08)
    PulseInstruction(pulse=GaussianPulse, length=4e-08, parameters={'amplitude': -0.5})
qubit1.z
    DelayInstruction(length=4e-08)
    PulseInstruction(pulse=GaussianPulse, length=4e-08, parameters={'amplitude': 0.2})
    DelayInstruction(length=2e-08)
    DelayInstruction(length=4e-08)
readout1.acquisition_channel
    DelayInstruction(length=4e-08)
    DelayInstruction(length=4e-08)
    DelayInstruction(length=2e-08)
    DelayInstruction(length=4e-08)
```

Figure 2. Printing backend

Plotting Backend – The plotting backend displays a graphical preview of the pulse sequence. This backend is useful for quickly reviewing sequences to make sure they match your desired output.

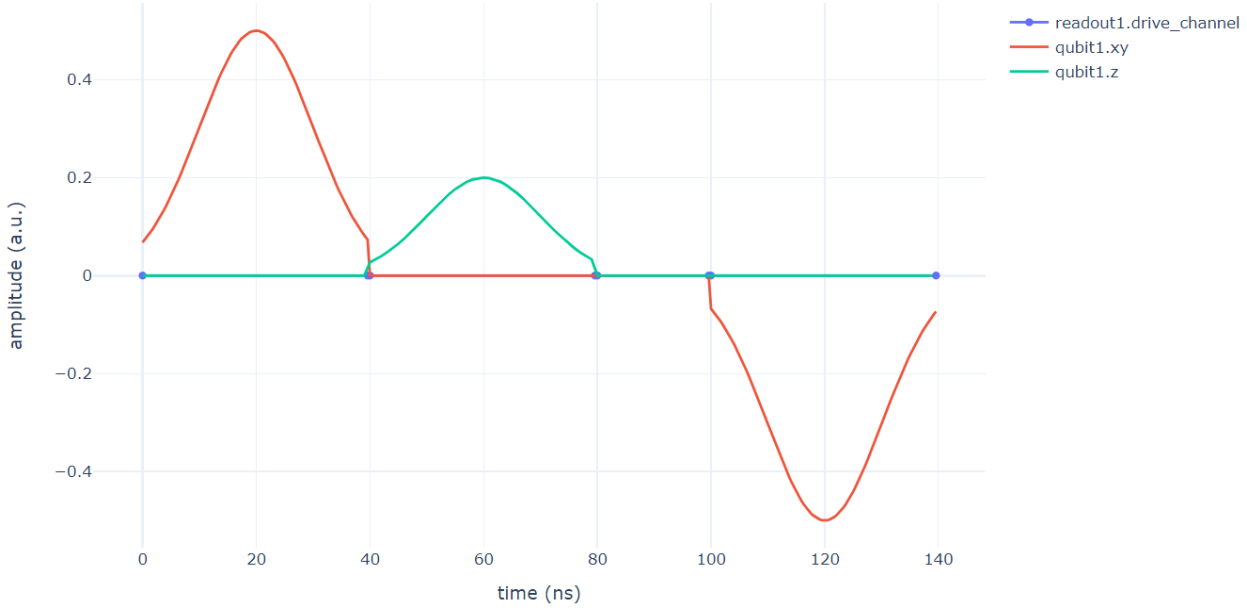


Figure 3. Plotting backend

Waveform Upload Backend – This backend is used for executing the experiment on hardware. It requires the Labber Instrument Server to be running. When running, this backend will add the appropriate instruments to the server (if not already present), connect to them, and orchestrate waveform upload and data acquisition. Synchronization between different modules is handled automatically by this backend. The results are returned because of this run method.

Features

The QCS software contains the following features.

- Quantum configuration file allows users to define mapping between quantum components and classical channels
- QCS API enables programmatic creation of experiments and pulse sequences
- Multiple backends allow easy switching setup, execution, and visualization of quantum sequences
- Separation of calibration parameters simplifies tune-up routines
- Tight integration with QCS hardware leverages FPGA acceleration and synchronization through the python API

Required Licenses

To use all features of the QCS SW detailed in this document, the following products and licenses are required.

Table 1. Required licenses

Product	Description	One license required per
M5401LUNA	Labber Quantum Software License	System
KS2201A	PathWave Test Sync Executive License	Modular instrument
KF9001B	PathWave FPGA Run-Time License	M5300A or M5200A
M5400B	Quantum FPGA IP Library License	M5300A or M5200A
KS2401A	PathWave Test Station Manager License	System

Conclusion

The Keysight Quantum Control System (QCS) provides a natural software interface for users to program quantum experiments. Pulse sequences are written in the languages of the qubits, couplers, and resonators rather than AWG and digitizer channels. This enables users to program at the level of their quantum hardware rather than worrying about the details of their classical instrumentation.