APPLICATION NOTE

Characterizing Superconducting Qubits

Using AWG and digitizer setup controlled by Labber

Over the past two decades, superconducting quantum circuits have transitioned from basic research into a prominent industrial and academic platform for quantum computers. Subsequent improvements in coherence times, gate fidelities, and scalability have also motivated hardware manufacturers to enter into the quantum ecosystem. By supplying classical control hardware tailored towards quantum applications, engineers are enabled to push the quantum processor performance to higher levels.

In this application note, we provide a brief introduction to the PXI-based quantum engineering tool-kit from Keysight, used to control and readout a single transmon qubit. We will also show how the hardware can be controlled and automated using our Labber software.

A quantum bit(qubit) is the fundamental building block of quantum computer. See our application note Introduction to quantum computing for more information.

The anharmonicity of a qubit is defined as the difference in frequency (or energy) between the 01 transition and the 12 transition. It thus provides a measure of how isolated the qubits computational subspace is from higher energy levels.

Please see our application note on Quantum Computing for more information on different types of qubits.

KEYSIGHT TECHNOLOGIES

Introduction to superconducting quantum bits

In this application note, we will consider how to efficiently characterize the quantum coherence properties of a transmon qubit, using the Keysight PXI solution, controlled by the Labber automation software.

Superconducting quantum bits (qubits) are resonant, nanofabricated electrical circuits, with a resonant frequency (qubit frequency, f_{01}) set by a superconducting capacitive island and a non-linear inductance, which is usually realized by introducing one or more so-called Josephson junctions (JJs), see Figure 1(a). At cryogenic temperatures, i.e. $kT \ll hf_{01}$, the qubit has the properties of an anharmonic multi-level quantum system, in which the two qubit states $|0\rangle$ and $|1\rangle$ are typically encoded into the two lowest energy levels, see Figure 1(b).

The energy spectrum, and thus the properties of the qubit (such as frequency and anharmonicity) can be engineered via two characteristic energy scales – the charging energy E_C (associated with the capacitor) and the Josephson energy E_J (associated with the inductive Josephson junction). In typical devices, the qubit frequency is designed to be around 5 GHz and the most common qubit modality is the transmon qubit, obtained when $E_J/E_C \approx 50 - 100$. For a detailed introduction to superconducting qubits, the reader is referred to the review article written by Krantz, *et al.* (2019) [1].

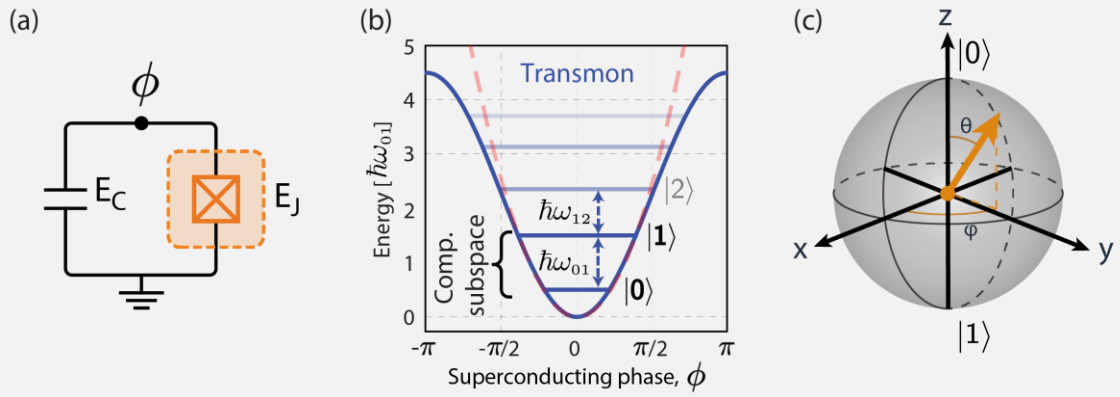

Figure 1: (a) Simplified circuit diagram, indicating the two characteristic qubit energy scales – the charging energy E_C and the Josephson energy E_J. (b) Energy-level diagram for the transmon qubit, with its computational subspace defined by its two lowest energy-levels, $|0\rangle$ and $|1\rangle$, separated by an energy $\hbar\omega_{01}$.(c) Bloch sphere representation of the qubit state, with the ground state $|0\rangle$ at the North pole and the excited state $|1\rangle$ at its South pole. Image courtesy Krantz, et al (2019).

1. P. Krantz, et al. Appl. Phys. Rev. 6 (2), 021318 (2019).

Single-qubit state manipulation using short microwave pulses

An essential criterion for any quantum processor – irrespective of its physical realization – is access to a universal gate set, in which any arbitrary quantum protocol can be compiled into a sequence of single and two-qubit gates.

In case of the transmon qubit, single-qubit rotations around the X and Y axes of the Bloch-sphere can be performed by applying a microwave pulse at the qubit transition frequency, to the qubit island via a capacitively-coupled drive line. This results in a rotation about the X-axis with an angle set by the pulse amplitude and duration. If the applied pulse is 90° phase-shifted, the rotation instead occurs about the Y-axis. This means that waveforms sent into the in-phase (I) and quadrature (Q) ports of the up-converter mixer, translates into X and Y qubit rotations, see Fig 2(a-b).

In the special case when the pulse duration and amplitude correspond to a rotation of pi-radians, we bring the state from its initial state (usually $|0\rangle$) to its excited state $|1\rangle$, see Figure 2. This, so-called π -pulse, is the case that we will consider in this application note, since the aim is to extract the energy relaxation time T_1, by varying the time duration between the applied π -pulse and the readout tone, leading to a decay from state $|1\rangle$ to state $|0\rangle$.

In this application note we focus on the hardware required for a single superconducting qubit, so we therefore limit the discussion to single-qubit gates. However, in order to run useful qubit algorithms, it is also important to have access to two-qubit gates.

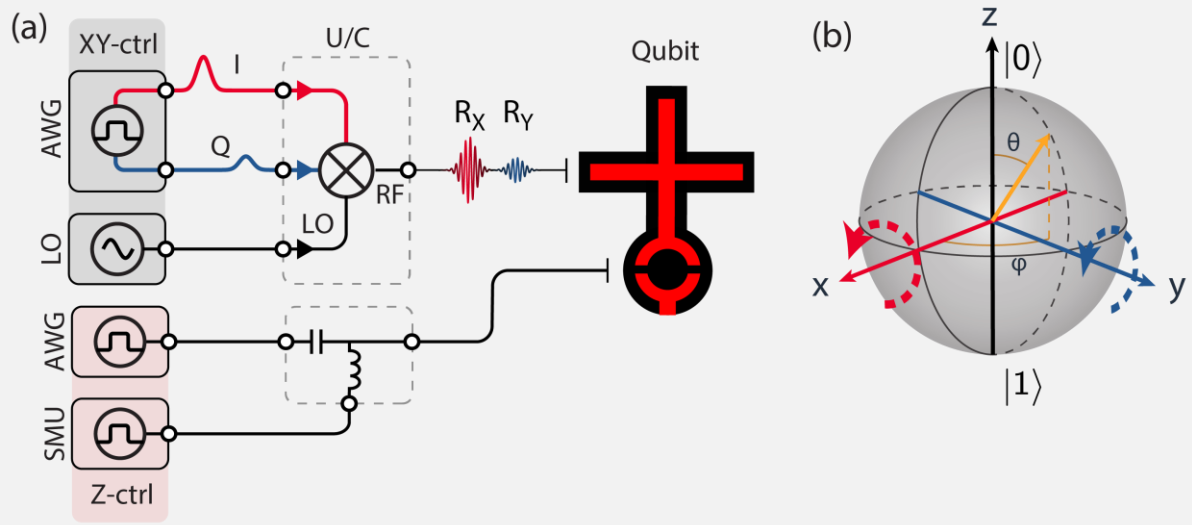

Figure 2: (a) Simplified single-qubit gate control. The XY gate control is performed by upconverting the I and Q baseband waveforms from two AWG channels by mixing them with a local oscillator (LO). For frequency-tunable superconducting qubits, their frequency can be controlled by tuning the magnetic flux through its SQUID-loop, effectively tuning the Josephson energy of the qubit. (b) Bloch-sphere representation of the X and Y rotations produced by the red and the blue pulses in (a).

Dispersive readout of superconducting qubits

In addition to controlling the state of the qubit, it is also important to be able to measure its state with high fidelity, and without altering the qubit state with the readout apparatus (non-demolition). A common technique used to perform qubit readout for superconducting processors is called *dispersive readout*. This technique is based on capacitively coupling the transmon qubit to a linear readout resonator, and then read out the microwave response of the readout resonator which carries information about the qubit state [2, 3].

In the resonant regime, *i.e.* when the frequencies of qubit and resonator are close to each other, the two systems can exchange energy, which is undesired during a quantum protocol (since the resonator would drain the qubit of energy). On the other hand, if the resonator is designed to have a resonant frequency far detuned from the qubit transition frequency, *i.e.* $\Delta = f_{01} - f_r \gg g, \kappa$, where g and κ denote the qubit-resonator coupling rate and the resonator coupling to its feedline, respectively, the two systems can no longer exchange energy with each other. However, they will still affect each other in the form of a small photon-dependent shift. This shift is called “dispersive shift” and provides a signature of the qubit state in the microwave response of the resonator. Therefore, it is possible to read out the qubit state by interrogating the resonator using a weak microwave probe tone. This way, the quantum mechanical dynamics of the system gets effectively mapped onto a classical electrical measurement of the resonator scattering response, see Figure 3. It is worth noting, however, that this dispersive regime is in fact an approximation, which only holds under the assumption that the number of photons in the resonator is kept small (typically 5-10 microwave photons), which puts some constraints on the signal-to-noise ratio of the readout amplifier chain.

In this application note, we will focus our attention on the setup requirements to perform single-qubit characterization, using the PXI AWG and PXI Digitizer modules from Keysight.

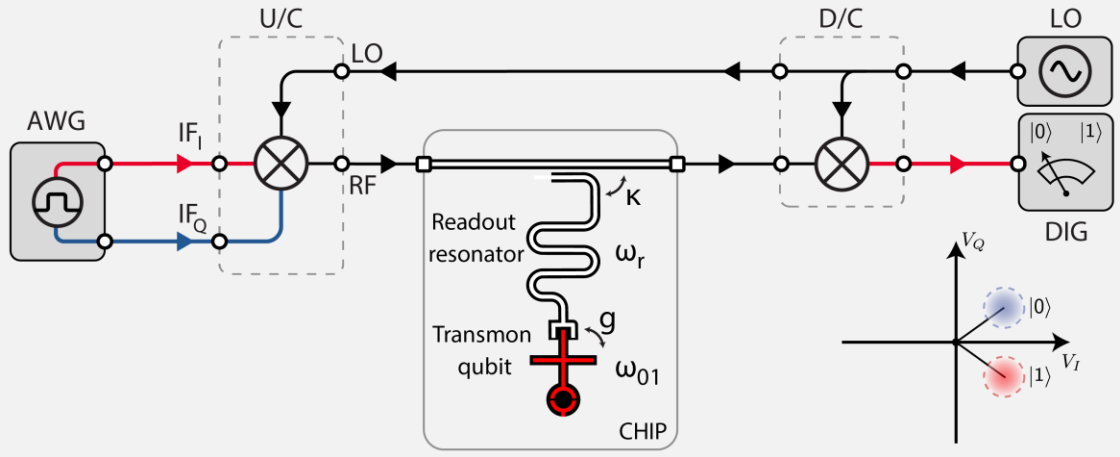

Figure 3: Simplified readout configuration. The readout resonator is probed with a microwave pulse, sent out from the upconverter. After interacting with the readout resonator, the microwave response is down converted and digitized.

2. A. Wallraff, et al. Nature 431 (7005), p.162-167 (2004).
3. A. Blais, et al. Phys. Rev. A 69 (6), 062320 (2004).

Configuration of control instruments

The Keysight PXI modules provide a qubit control and readout hardware solution. Its modularity offers a flexible, yet compact solution that can be tailored to the specific quantum hardware at hand. In this use-case we need to control and readout a single superconducting transmon qubit, for which we can use the configuration illustrated in Figure 4 and listed in Table 1.

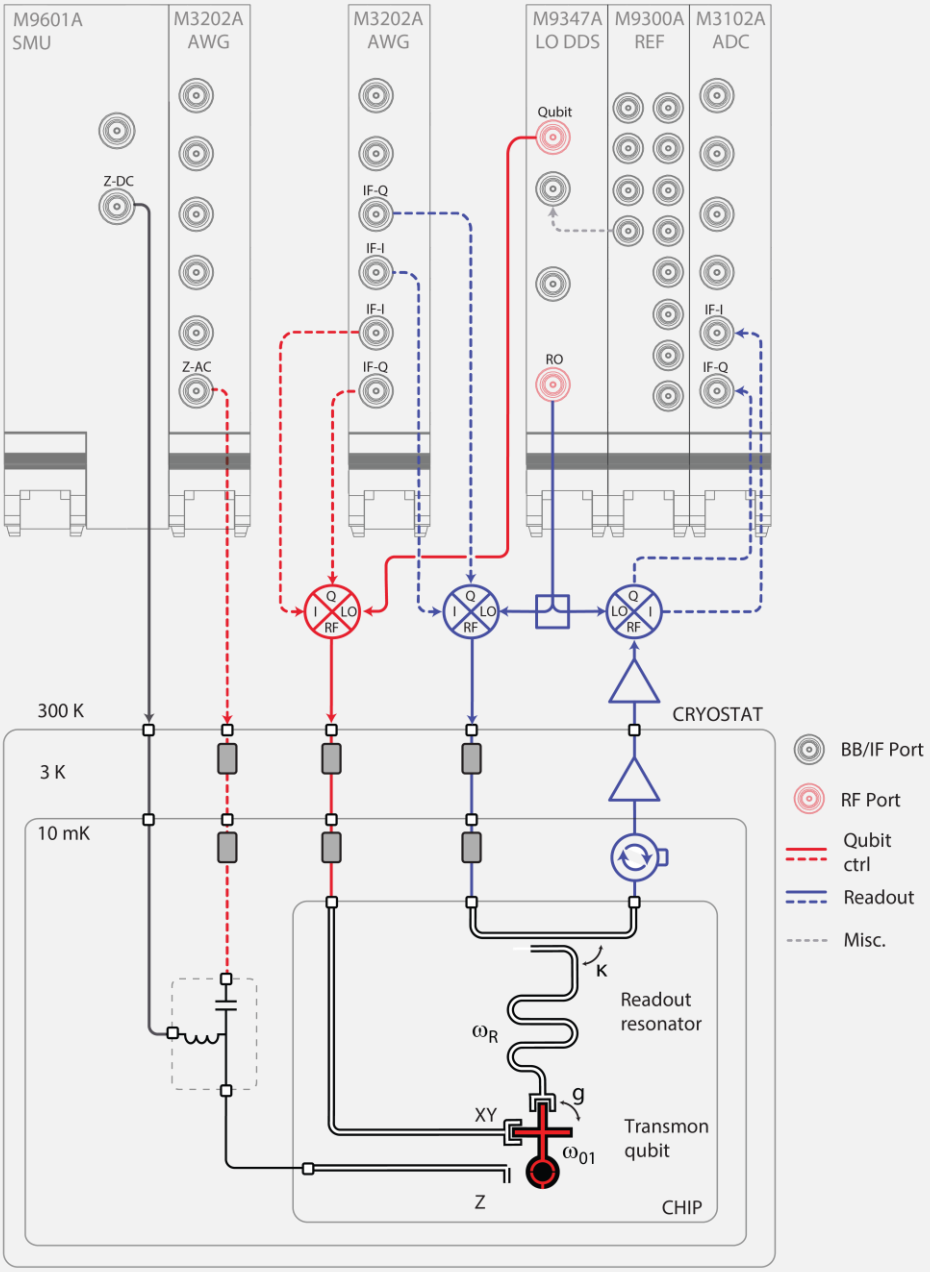

Figure 4: PXI module configuration and simplified cryogenic setup for control and readout of a transmon qubit with flux-tunable transition frequency.

Table 1. Instrument configuration in M9019A 18-slot PXI Chassis

Slot	Instrument	Model	Function	Connections	I/O
1-4	High-performance Embedded Controller	M9037A	(Optional)		
5-6	SMU	M9601A	Qubit DC-flux control	DC2: Qubit DC flux	Out
7	AWG	M3202A	Qubit AC-flux	4: Qubit IF_{flux}	Out
9	AWG	M3202A	Pulse generation for qubit and readout (RO)	1: RO IF_Q 2: RO IF_I 3: Qubit IF_I 4: Qubit IF_Q	Out Out Out Out
11	Local Oscillator (LO DDS)	M9347AH02	LO Generator for Qubit and RO	1: Qubit LO 2: 100 MHz Ref 4: RO LO	Out In Out
12	Frequency Reference	M9300A		LO DDS Ref	Out
13	Digitizer	M3102A	Data acquisition	3: RO IF_I 4: RO IF_Q	In In

For more information about the recommended options, the reader is referred to our Quantum Solutions Website

Measurement automation using Labber

After setting up the control hardware PXI Chassis, it is time to further discuss how we can control and automate the instruments to run qubit characterization measurements. In this section, we show how our lab-control software package Labber can be used to orchestrate the measurement by bringing all equipment into the same framework. Labber can either be installed on the PXI-embedded controller or on an external measurement computer and is supported for Windows, Mac OS, and Linux.

Labber is a powerful, yet user-friendly software package for instrument control and lab automation, with a focus on qubit measurements. The structure of Labber is schematically illustrated in Figure 5. In this section, we will introduce each module of the software, applying the use-case of characterizing our superconducting qubit.

Labber can be downloaded from the Keysight website. The trial period is 30 days, after which a license needs to be purchased. To purchase a Labber license, please contact labber@keysight.com.

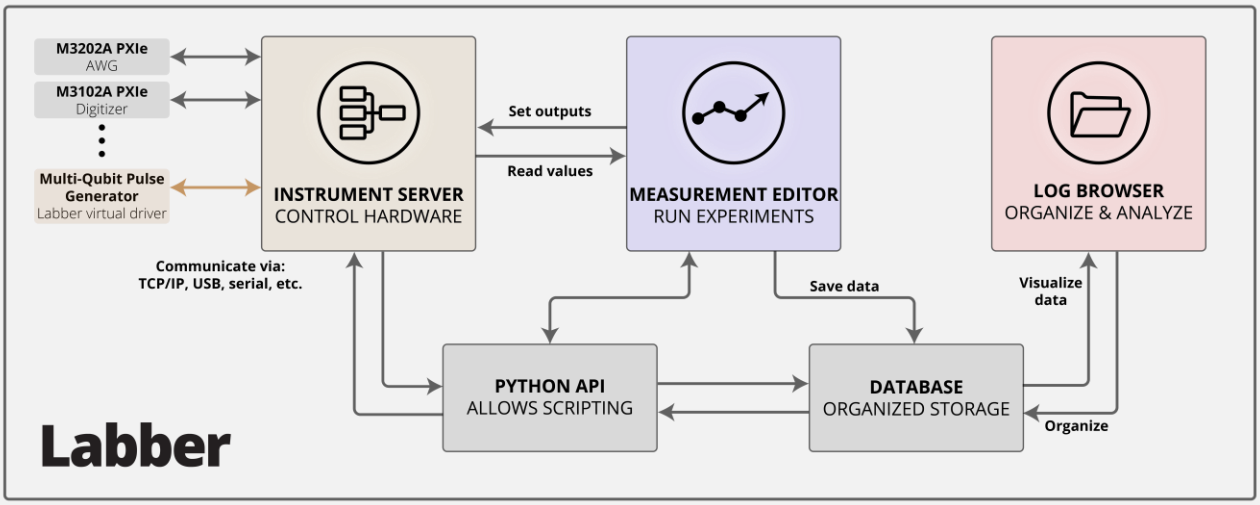

Figure 5: Schematic overview of the Labber software package.

Instrument Server

Before we can start our measurements, we first need to establish communication with all the PXI modules. As illustrated in Figure 5, all instruments communicate with Labber via the Instrument Server. In addition to the physical instruments, Labber also allows users to employ virtual instrument drivers, containing software routines to perform specific tasks, such as waveform creation, on-the-fly analysis protocols, or any other function specific to the user needs.

To add an instrument to the server, click the **Add**-button and provide the **instrument type, name, communication interface**, and **address** for each instrument in the configuration. See Figs. 6-8 below.

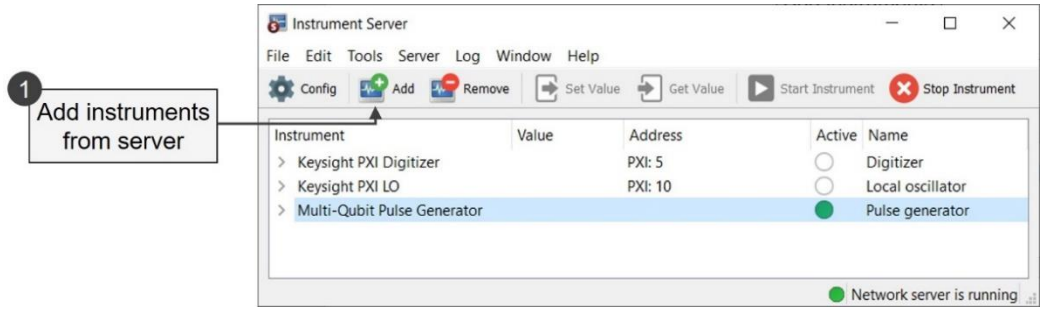

Figure 6: The Instrument server lists the connected instruments. Instruments can easily be added or removed from the list. The green light indicates active communication with Labber.

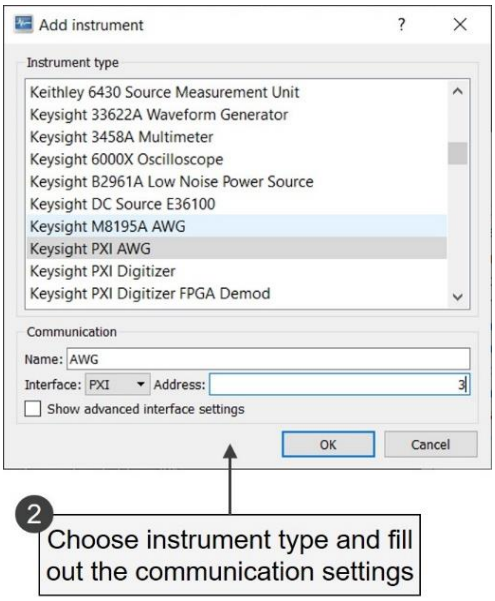

Figure 7: An instrument is added by first choosing instrument type and then filling out the communication settings. When done, click **OK**.

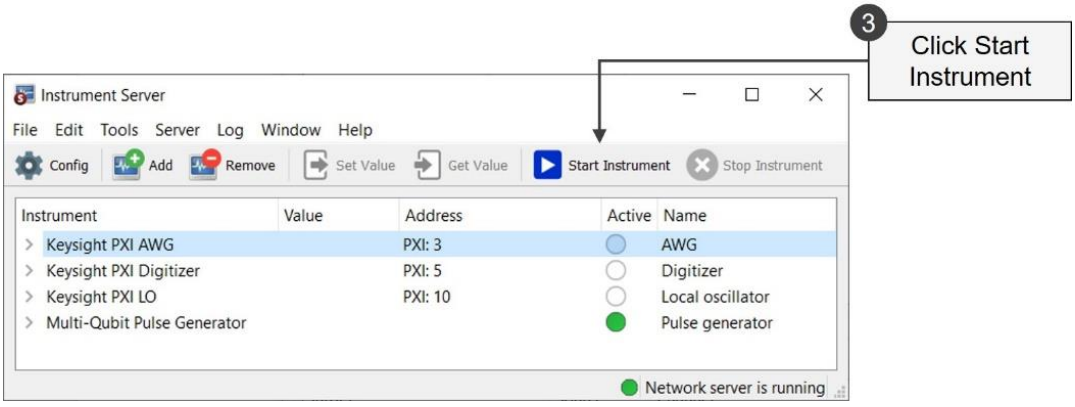

Figure 8: Finally, the instrument communication is initiated by clicking **Start Instrument**.

Virtual driver: Multi-qubit pulse generator

To perform qubit measurement, we also need a tool to define the duration, shape, and timing of microwave pulses, both for qubit control and readout. To do this, Labber provides a virtual driver called the *Multi-qubit pulse generator*. This driver contains many of the features needed to control a quantum processor of a few qubits, with individual (but multiplexed) readout resonators.

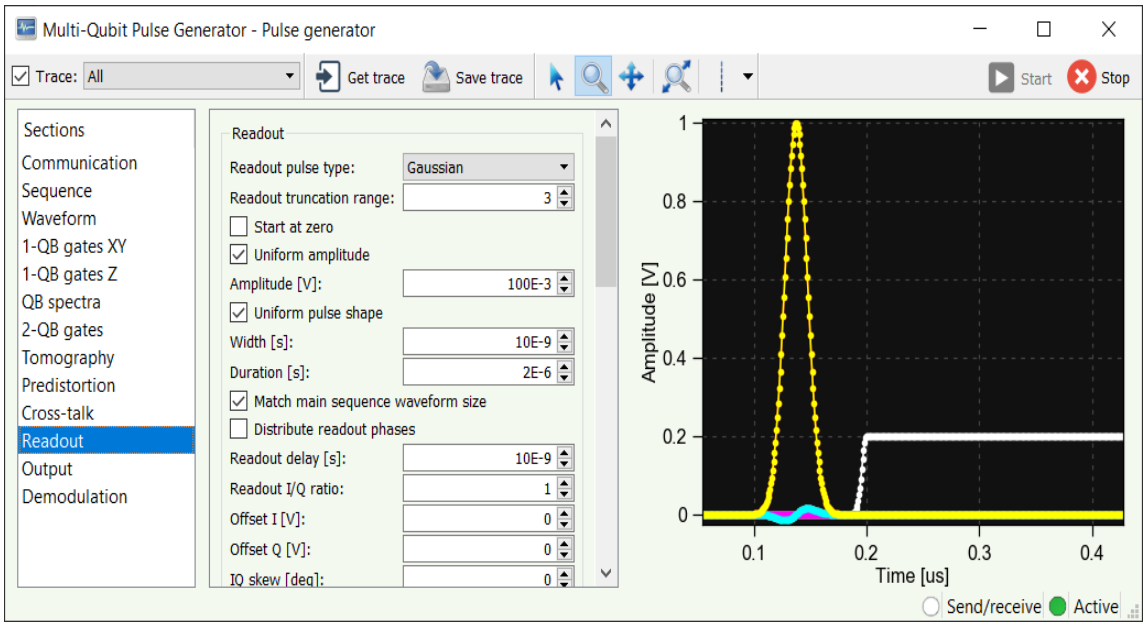

Figure 9: GUI for the Multi-qubit pulse generator driver, allowing the user to access to the required settings for defining control and readout pulses for qubit measurements.

Measurement Editor

Once all instruments are added to the server and communication has been established, the Measurement Editor – in which we will define the measurement instructions – can be started. Following the steps outlined in Figure 10, we can configure the measurement by selecting instrument settings. Using **Signal Connections**, we can route the correct waveform to the intended target channel on the AWGs.

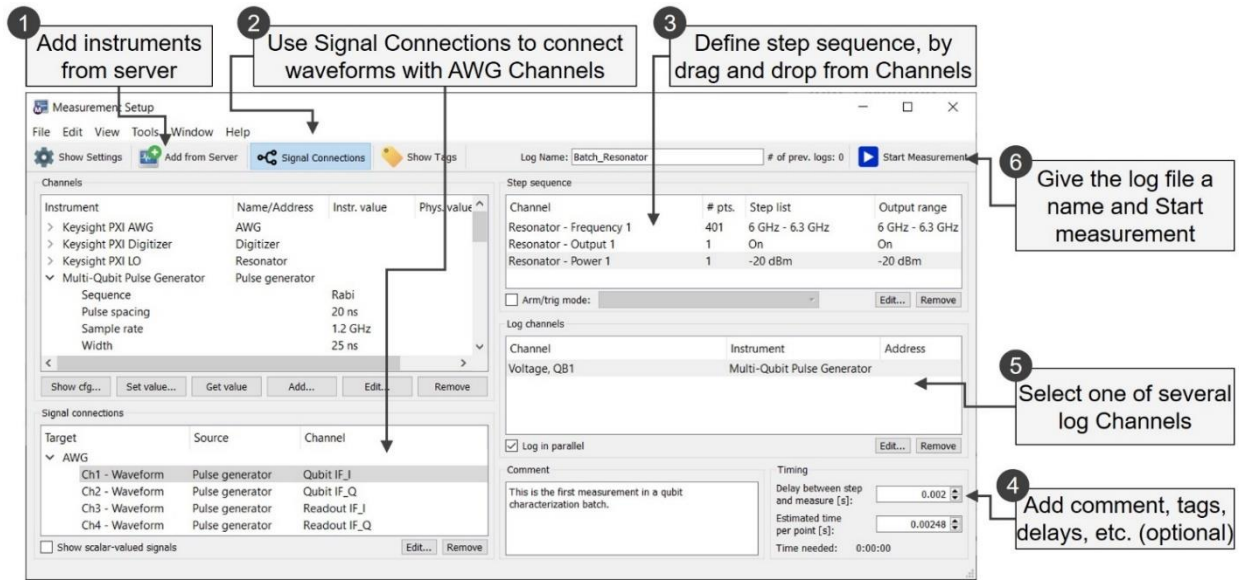

Figure 10: The Measurement Editor is used to build up the configuration file that will contain the measurement instructions. The steps indicate their typical order.

Measurement automation using the Labber Python API

In many realistic laboratory scenarios, experiments require not only a single measurement configuration, but rather a set of subsequent measurements, interleaved with various steps of data fittings and parameter extractions. The Python API can be used when the input of a certain measurements depends on the output of an earlier measurement. This is the case when characterizing the coherence properties of a qubit, since it requires some calibration information among the settings of the measurement configuration.

In this section, we discuss how the Python API can be used to execute the set of measurements and data fitting steps, listed in Table. 2. The list contains four different measurement configuration files, as well as four data fitting steps. See also Figure 11 for the corresponding data and fits. To add one additional degree of freedom, we would like to extract the T_1-time for each value of an applied magnetic flux (which will move both the qubit and the resonator in frequency), thus requiring a new tune-up for each step.

In this application note, we only give a brief introduction to the Python API. See our Labber documentation for more details.

Table 2: Example of components in a batch script, with the aim to extract the energy relaxation time for a qubit. The black text represents measurements, whereas the entries marked in red are data fitting steps.

Step	Measurement	Description	Fit output
1	Scan resonator	Sweep resonator probe frequency	
2	Data fitting	Identify readout probe frequency	Freq – res. [f_r]
3	Qubit spectroscopy	Fix resonator probe frequency [f_r] and sweep the qubit probe frequency	
4	Data fitting	Identify the qubit frequency	Freq – qubit [f_{01}]
5	Rabi Oscillations	Using [f_r] and [f_{01}], sweep qubit pulse duration	
6	Data fitting	Extract duration for π-pulse	Pulse duration [τ_π]
7	T1-decay	Using [f_r], [f_{01}], and [τ_π], sweep the readout delay	
8	Data fitting	Fit energy relaxation time T_1	Relaxation time [T_1]

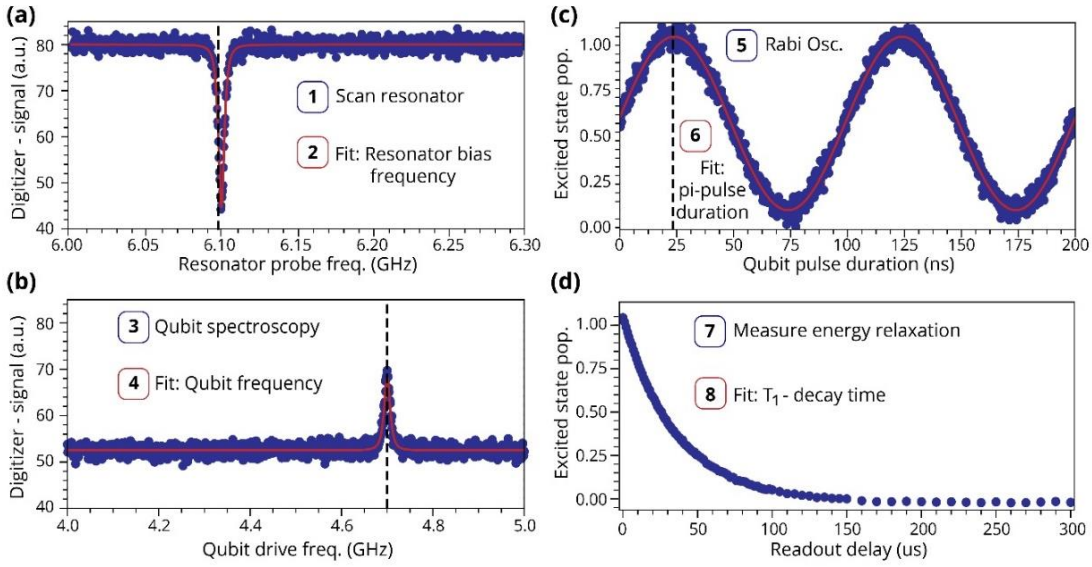

Figure 11: Python API batch measurement scheme used for extracting the T_1-decay time of a qubit. The blue dots represent data, whereas the red lines are fits, from which calibration parameters are determined and fed into the next measurement.

Key functions to construct a batch-script

The Python API calls the measurement executable for performing the measurements. Therefore, the first step when constructing a script is to run through the different measurements manually once one-by-one, thus defining a new configuration file for each measurement type required for the script. These configuration files will serve as the addressable objects in the script. Note that the name of the log file needs to be the same as the name written in the script.

The Python API uses a python package called **ScriptTools.py**, and works around a few, but important commands, shown in below sample script below (performing steps 1-3 in the batch measurement):

```
import os
import numpy as np

from Labber import ScriptTools

# Define list of flux points [V]
vFlux = np.linspace(-1E-3, 1E-3, 101)

# Define measurement objects for step 1 and 3
sPath = os.path.dirname(os.path.abspath(__file__))
MeasResonator = ScriptTools.MeasurementObject(\
                os.path.join(sPath, 'Batch_Resonator.hdf5'),
                os.path.join(sPath, 'Batch_ResonatorOut.hdf5'))
MeasQubit = ScriptTools.MeasurementObject(\
            os.path.join(sPath, 'Batch_Qubit.hdf5'),
            os.path.join(sPath, 'Batch_QubitOut.hdf5'))
# Set the master channel that defines the third data dimension. In our case the flux
bias.
MeasResonator.setMasterChannel('Flux bias')
MeasQubit.setMasterChannel('Flux bias')

# Go through list of points
for n1, value_1 in enumerate(vFlux):
    print('Flux [mA]:', 1000*value_1)
    # Set flux bias
    MeasResonator.updateValue('Flux bias', value_1)
    MeasQubit.updateValue('Flux bias', value_1)
    # Measure resonator
    (x,y) = MeasResonator.performMeasurement()
    # For this example, y is complex, take absolute value
    y = abs(y)
    # look for peak position
    print('Resonator position [GHz]:', x[np.argmax(y)]/1E9)
    # set new frequency position
    MeasQubit.updateValue('RF - Frequency', x[np.argmax(y)])
    # measure qubit
    (x,y) = MeasQubit.performMeasurement()
```

Note that this is a simplified script (to illustrate the functions of ScriptTools) in which the fitting routine is only identifying a peak in the resonator spectrum. In a real script, this fitting routine can be replaced for a more advanced one.

Log Browser

Before we conclude this application note, we will look at the Log Browser of Labber. Each measurement that is run in the Measurement Editor gets stored as a struct-file of hdf5 format, containing both the acquired data as well as the full measurement configuration. Figure 12 shows the main log browser GUI.

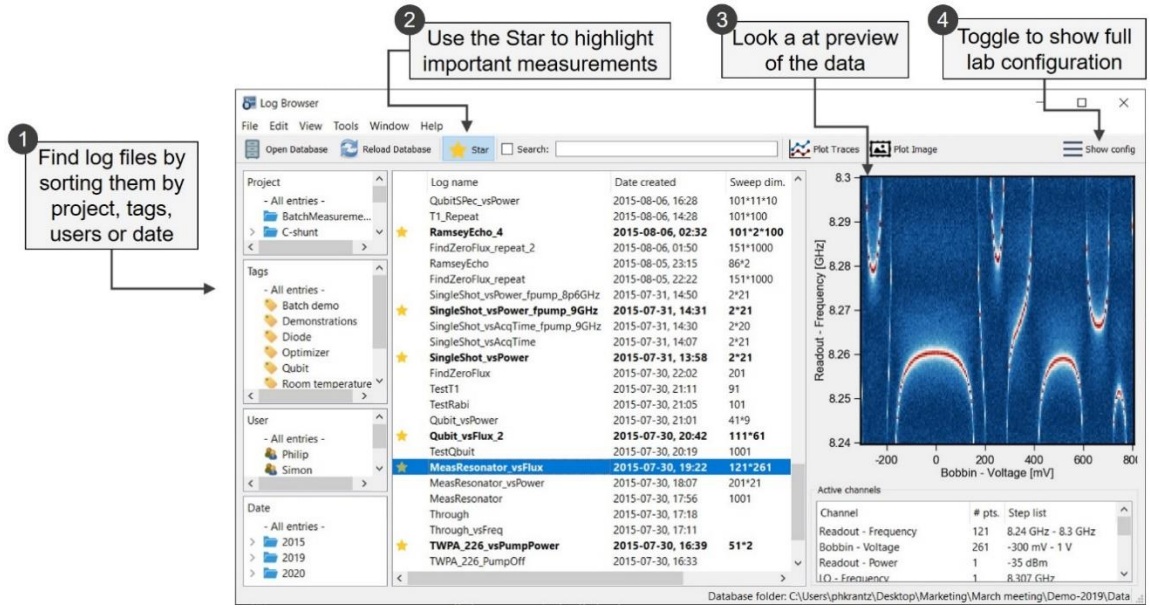

Figure 12: Log Browser interface, storing the acquired measurement data and configurations.

Restore earlier measurement scenarios

The LogBrowser has several key advantages. In addition to storing all data in a structured and standardized way, it also stores all instrument settings used for the measurement. This allows the user to drag and drop log files into the measurement editor in order to restore earlier configurations.

Fast data browsing and analysis

By double-clicking on a log file, the data is displayed in the log viewer mode, see Figs. 13 and 14. This mode allows users to fast screen through measured data and to extract information, without any need to export the data into another programming environment just for plotting.

Export data and plots

After adjusting plots of traces or 2D maps using the built-in tools in the log viewer, the plots can easily be saved and exported to logbooks or publications. It is also possible to export the raw data to different formats for further analysis.

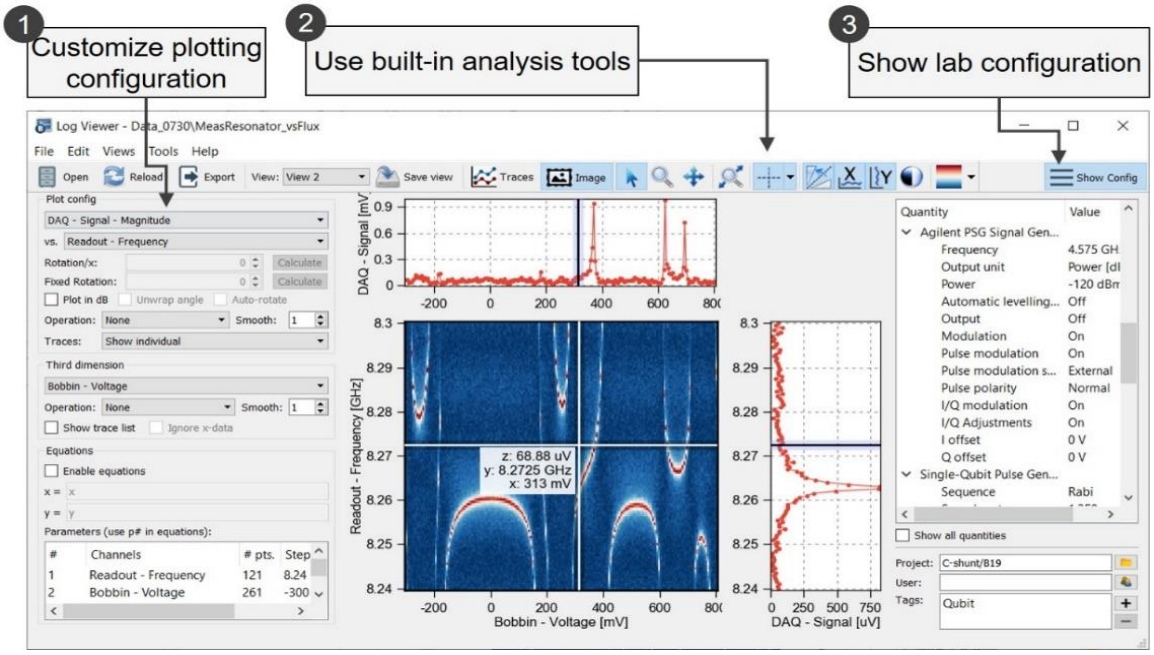

Figure 13: The Log Viewer allows users to perform fast data analysis and screening, without the need to export data from Labber.

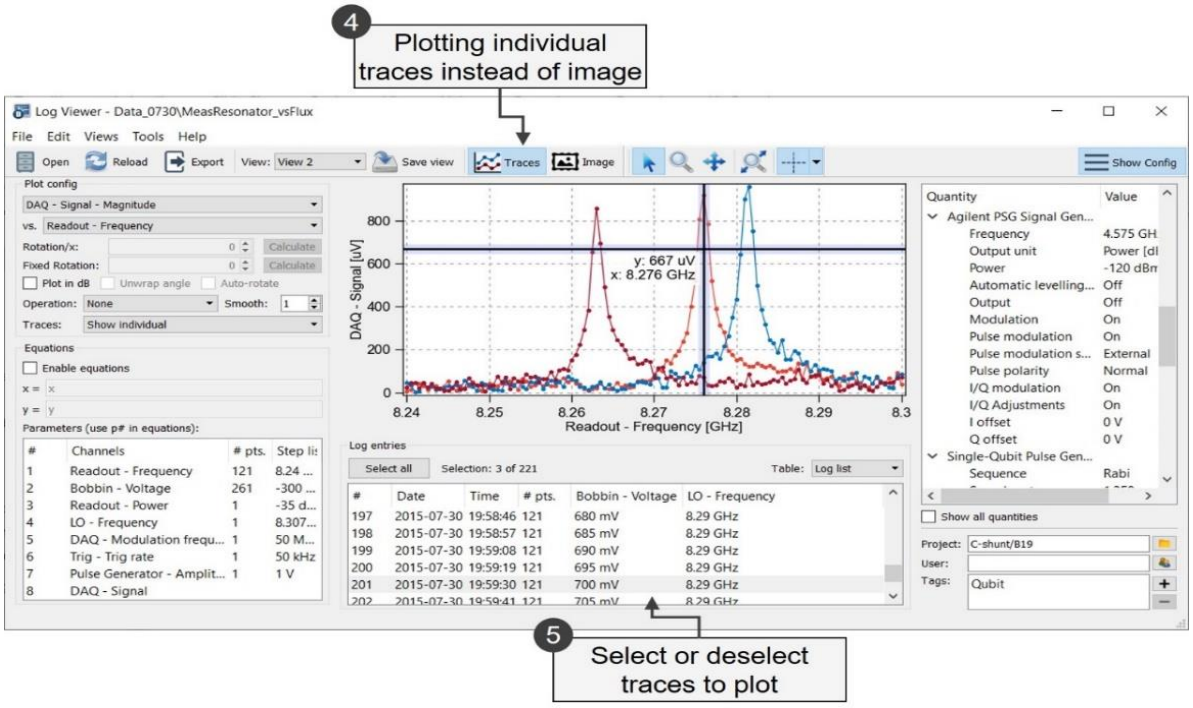

Figure 14: By selecting **Traces** instead of **Image**, individual traces are plotted and can be analyzed.

Conclusion: Faster qubit measurements using Keysight hardware and Labber

In this application note, we have investigated a typical use-case for superconducting qubit platforms, namely, how to extract qubit coherence properties. The Keysight PXI solution provides a modular and compact hardware platform, which scales well as the number of qubits is increased.

The scripting capability of the Labber API provides a fast, yet flexible, solution for fast and automated system tune-up, without the need to write many lines of code. In turn, this enables quantum engineers to conveniently extract device parameters and advance their scientific frontier.

Further reading

- General application note on Quantum Computing
- Trapped-ion quantum computing

Learn more at: www.keysight.com

For more information on Keysight Technologies' products, applications or services, please contact your local Keysight office. The complete list is available at: www.keysight.com/find/contactus

KEYSIGHT TECHNOLOGIES