QuantumLabPython Manual

Ruggero

December 4, 2021

Controlling the experiment

The AWGs (Arbitrary Waveform Generators) trigger the cameras, provide AOM's MOD-IN and FREQ-IN when they need to be dynamically changed, and control the magnetic field by commanding the relay.
It is possible to communicate with the electronic instruments (Basler cameras, Tektronix oscilloscopes, Keysight AWGs) using appropriate scripts and connecting the instruments via usb cables to the computer. All the scripts are implemented in Python 3.6.4. Python has been installed with the Python distribution Anaconda (v. 5.1.0).
List of scripts for experiment control, in folder 'ExerimentMOT':

- MultiResources.py – defines classes to interface with AWG and oscilloscopes
- CameraResources.py – defines classes to interface with Cameras
- Visa_Instrument_Initialisation.py – Detects and test all the Visa resources connected to the PC
- UseCameraResources.py – example on how to use cameras
- ExperimentMOT.py – example of a code used to run experiments
- CreateArbWithMultiResources.py – Used to Create and Trigger Arbitrary Waveform with the AWGs.
- BuiltInFunctionWithMultiResources.py – Script used to output simple built-in functions from AWGs.

Cameras

Cameras are controlled using a python library called **pypylon**. It is a wrapper of the camera API Pylon, released by Basler. It is nearly undocumented. Basically, you can have a look at the examples and try to get what the functions does. There is a pylon.py somewhere but it is not so useful. To modify camera acquisition parameters you can basically use the same syntax of the C++ API that you find in the manual, or on-line at this link (it contains more things than the manual itself):
`https://docs.baslerweb.com/#t=en%2Ftrigger_selector.htm%23Specifics&rhsearch=acA2040-90%20umnir`

There is a GitHub community that sometimes answer your questions:
`https://github.com/basler/pypylon/issues?page=2&q=is%3Aissue+is%3Aopen`

Moreover, if you open Pylon and you are 'Expert user' (you can modify you user level in Tools -> Options) you can go to the question mark in the top bar and look for 'C++ Programmer's Guide'.

Here some examples of Python sintax:
camera.TriggerSelector.SetValue("FrameBurstStart")
camera.TriggerSource.SetValue("Line1")
camera.TriggerMode.SetValue("On")

Images are first Grabbed ('StartGrabbing()') and data are stored in a buffer. Then you can retrieve them to be available to your application ('RetrieveResult()'). Together with the image you can retrieve lots of metadata (chunk data) if enabled.

AWGs and Oscilloscopes

We use **PyVisa** which is a Python frontend of the VISA backend (the API). VISA is provided by different companies producing electronic equipment in order to allow the user to interact with their instruments through a computer. VISA is distributed, for free, by Tektronix, National Instrument, Keysight. We use NyVisa (v. 17.5), provided by National Instruments, because this is the one that is officially used as a backend of PyVisa. The VISA versions of other suppliers are not always compatible. PyVisa is well documented and can be access at this link:
`https://pyvisa.readthedocs.io/en/master/`

VISA API uses the 'VISA shared components' already included in every Windows operative systems. These compontents are provided by the IVI foundation and a 64 bit machine has both the 32 and 64 bit versions.
When using Pyvisa, you need just to have a version of VISA installed, and Python finds it automatically. PyVisa provides a framework to write SCPI commands to interact with the instrument. SCPI commands are similar for different models, but it is better if you find the 'programmer manual' of your instrument (or instrument series). Resources (Devices) are managed by the 'Resource Manager' class. Basically, PyVisa allows the user to open and close the communication sessions with instruments and to actually communicate with them. To do so, you basically perform 2 operations which are 'write' and 'query'. With the first one you tell the instrument what to do, with the second one you ask about the value of an instrument parameter. 'Write' and 'query' use SCPI commands.

Oscilloscope
With the oscilloscopes you have take care of 3 things:

- tell the oscilloscope to start the acquisition ('ACQ'). The channel(s) has to be selected manually on the oscilloscope
- define the way data are stored in the oscilloscope in order to allow your program to interpret them ('DATA')
- retrieve the data ('WFM', 'CURVE')

AWGs
SCPI here is built so that for every command that can be referred to a channel output, you can specify, in each command, which channel you are referring to. If not specified, default channel is 1.

An arbitrary function can be created using a dedicated software (Benchlink basic/pro), but you can easily build it in a CSV file (es: using excel).
Note: When using the AWGS always set the correct load to match the impedance of the appliance. If the load is 50 Ω the maximum output voltage is 5 V. If the load is 1 MΩ or higher, is 10 V.
Note: Each AWG's output is defined in an Excel file. After the experiment is performed always save the excel file of the experiment and copy and paste the console output in a .txt file.