UHD Python API

Contents

As the name suggests, it exposes the UHD API into Python. We use `pybind11` to generate a Python module which exposes most of the C++ API, and some extra features. The Python API is part of stable releases.

The USRP Hardware Driver and USRP Manual covers most information about the UHD Python API and can be found here: https://files.ettus.com/manual/page_python.html

In order to test the Python API, check out the `master` branch and build it like always. When running CMake, make sure that the Python API was enabled (`-DENABLE_PYTHON_API=ON`).

The output from CMake should look something like this:

```
-- ##################################################
-- # UHD enabled components
-- ##################################################
--   * LibUHD
--   * LibUHD - C API
--   * LibUHD - Python API
--   * Examples
--   * Utils
--   * Tests
--   * USB
--   * B100
--   * B200
--   * USRP1
--   * USRP2
--   * X300
--   * N230
--   * OctoClock
--   * Manual
--   * API/Doxygen
--   * Man Pages
--
-- ##################################################
-- # UHD disabled components
-- ##################################################
--   * GPSD
--   * E100
--   * E300
```

Please refer to the USRP Manual for extended instructions especially when installing on Windows. Once it's built and installed, you'll be able to import the `uhd` Python module.

We have some examples in `host/examples/python`. The examples are very simple, but concise.

This Python example is based on the C++ example `uhd/host/examples/rx_samples_to_file.cpp`.

Documentation is currently pretty sparse. The best we can do right now is to ask users to infer the documentation from the C++ API. For example, the Python has an object called `MultiUSRP` which is an equivalent of the C++ `multi_usrp` API. The methods on both classes are the same, and take the same arguments.

Does it support Python 2 and 3?

Starting with UHD 4, Python 2 support has been removed.

Does it require GNU Radio?

No.

Does it use SWIG?

No, it uses `pybind11`. It also doesn't require the C API. Pybind11 is vendored with UHD so as to not require installing another dependency.

How does this relate to the Python API in gr-uhd?

It serves an entirely different purpose. This Python API is for people writing standalone applications for USRPs that *don't* use GNU Radio. `gr-uhd` is staying the way it is, and is going nowhere. If you're using GNU Radio, you probably don't care about this.

Are the UHD Python API and the gr-uhd Python API compatible?

Short answer: No. Long answer: There are very few cases where it makes sense to mix these APIs, so no. However, this means that a `TimeSpec` from the `Boost.Python` API is not convertible into a `time_spec_t` from the `gr-uhd` API.

Does it support RFNoC API?

For sure!

What's the streaming performance?

Worse than straight C++, but not a lot, thanks to NumPy. You can run `host/examples/benchmark_rate.py` if you want to see for yourself. Overall, `recv()` calls are pretty efficient if you've pre-allocated a NumPy array, because we can cast that to a straight pointer (and also skip any type checking!) and then it's not that different from a `recv()` call in a C++ app. However, consuming the data is limited by how fast you can handle that in Python.