UHD Python API

Contents

- 1 What's the UHD Python API?
- 2 How can I use it?
- 3 Example: pyuhd_rx_to_file.py
- 4 What about documentation?
- 5 FAQ

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

The pre-release announcement covers most of the information and can be found here:
http://lists.ettus.com/pipermail/usrp-users_lists.ettus.com/2017-June/025379.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
```

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?

Yes.

Does it require GNU Radio?

No.

Does it use SWIG?

No, it uses `Boost.Python`. We didn't want to add another dependency to UHD (i.e., SWIG) and Boost was already a dependency of UHD. It also doesn't require the C API.

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.

When will it be released?

It is currently on `master` branch, and will become part of the next major release.

Does it support RFNoC API?

Not yet, but it's in the pipeline. We wanted to get the basics (i.e. `multi_usrp` API) right first.

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 preallocated 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.