The Universal Hardware Driver

Devin Kelly

October 1st, 2011

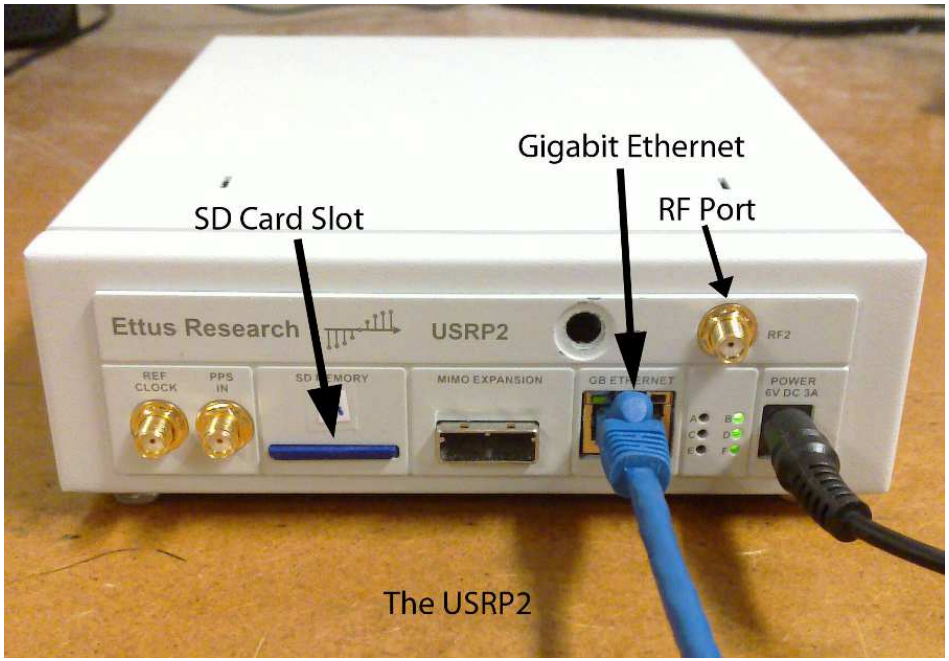
Contents

My Background

- B.S., Electrical and Computer Engineering from WPI in 2010.
- M.S., Electrical and Computer Engineering from WPI in 2011.
  - Thesis Topic: Practical Distributed Spectrum Sensing.
- Currently a staff member at MIT Lincoln Laboratory.

The USRP

- The Universal Software Radio Peripheral is software radio designed by Ettus Research, LLC.
  - `http://www.ettus.com`



The USRP2

The USRP

	USRP1	USRP2
Sample Rate	2x 64 Msps	2x 100 Msps
ADC Dynamic Range	12 bit	14 bit
PC Interface	USB 2.0	Gigabit Ethernet
RF Bandwidth	16 MHz	25 MHz
	USRP N2*	USRPE100
Sample Rate	2x 100 Msps	2x 64 Msps
ADC Dynamic Range	14 bit	12 bit
PC Interface	Gigabit Ethernet	GPMC
RF Bandwidth	50 MHz	5 MHz

Table: The USRP Product Family.

The Old Way

- Before the UHD there was only the USRP 1 and the USRP 2.
- Two *separate* C++ libraries were used to communicate with each SDR.
  - `libusrp.so` and `libusrp2.so`
- Both libraries distributed with GNU Radio
  - So both libraries are GPL licensed.
- `libusrp2.so` used raw sockets to communicate between the USRP2 and the host pc.
  - Three layer network stack (physical, data link, application)

Problems

- Different libraries for each device.
  - Must link separately.
  - Different API.
- Linux support only (no `.dll` or `.dylib`).
- GPL is an "infectious" license, if your software links to it the software must also be licensed under the GPL.
- Raw sockets require root access to open on Linux machines and require a custom driver on Windows machines.
- No network or transport layers mean that USRP2 packets are not routable.

Solution - The UHD

- The UHD is one library so there is one API.
- The UHD is cross licensed.
  - It's open source, but it can be linked to without requiring the linking software be open source.
- Linux, Windows, and Mac OSX support.
- Implements a full network stack.
  - Routable.
  - No root access or custom drivers needed.

Getting the UHD

- The UHD is not in any software repositories.
  - There aren't software repositories for Windows or Mac anyway.
- Get the dependencies,
- Get the source.
- Build and install the source.

Dependencies

- Git.
- C++ compilier, One of GCC, CLang (LLVM), MSVC.
- CMake version 2.6 or higher.
- Boost version 1.36 or higher (Mac/Linux) version 1.40 or higher (Windows).
- LibUSB version 1.0 or higher.
- Python version 2.6 or higher.
- Cheetah version 2.0 or higher.
- Doxygen.
- Docutils.

Getting the UHD

- Clone from the Git repository:
  - `git clone git://code.ettus.com/ettus/uhd.git`
- Download a release from GitHub:
  - `https://github.com/EttusResearch/UHD-Mirror/archives/master`
- Directory structure:

```
firmware/
fpga/
host/
images/
```

Building and Installing the UHD

Linux install instructions.

1. `$ cd host; mkdir build; cd build`
2. `$ cmake -DCMAKE_INSTALL_PREFIX=/opt/uhd ../`
3. `$ make`
4. `$ make test`
5. `$ make install`
6. `$ export LD_LIBRARY_PATH=/opt/uhd/path/to/libuhd.so`
7. `# ldconfig`

Mac install instructions are the same except:

- `$ export DYLD_LIBRARY_PATH=/opt/uhd/path/to/libuhd.so`

Building the UHD

Check the output from CMake:

```
##################################################
# UHD enabled components
##################################################
* LibUHD
* Examples
* Utils
* Tests
* Manual
* Doxygen
* USRP-E Utils
##################################################
# UHD disabled components
##################################################
```

Confirming the Install

- To check the UHD install plug in your USRP and open a shell.
- For a USRP2 or N series type,
- `$ ifconfig eth1 192.168.10.1 netmask 255.255.255.0`
  - The default address for the USRP is 192.168.10.2
  - Make sure you pick the correct Ethernet device.
- Next, type
- `$ uhd_find_devices`, or
- `$ uhd_usrp_probe`
- Either command should find your USRP device.
  - If neither work, start to troubleshoot by pinging the USRP device.

The API

- The UHD is written in C++, so that is the most direct way to access the API.
- Before writing any code consider reviewing the examples and doxygen documentation.
- `host/examples/`
  - `rx_ascii_art_dft.cpp`
- `http://files.ettus.com/uhd_docs/doxygen/html/annotated.html`
- When browsing the doxygen documentation these classes
  - uhd::usrp::multi_usrp
  - uhd::device

Important Classes

- `uhd::usrp::multi_usrp`
  - This class provides a high level interface to one or multiple USRPs.
  - This is the class that is used to manipulate the RF parameters USRP.
  - For example, this class is used to control center frequency, bandwidth, gain, channel, daughtercard selection (for the USRP1), and the antenna.
  - This class is also used to start transmit or receive streams.
- `uhd::device`
  - This is the low level interface to the USRP.
  - This API is used for
    - Discovering USRP devices
    - Reading and writing device parameters
    - "Low level" data streams, *i.e.* transmit/receive samples with meta-data.

uhd::usrp::multi_usrp

- Initialize the USRP

```
std::string args = "192.168.10.1";
uhd::usrp::multi_usrp::sptr u = \
uhd::usrp::multi_usrp::make(args);
```

- Set some parameters:

```
size_t channel_number = 0; double bw = 500000;
double tune_freq = 2450000000;
u->set_rx_freq(tune_freq, channel_number);
u->set_rx_bandwidth(bw, channel_number);  ...
```

- Start receiving:

```
uhd::stream_cmd_t \
sc(uhd::stream_cmd_t::STREAM_MODE_NUM_SAMPS_AND_DONE);
sc.num_samps = 1000000;
sc.stream_now = true;
u->issue_stream_command(sc);
```

uhd::device

- Discover a USRP

```
std::string arg = "192.168.10.1";
uhd::device_addrs_t device_addr = \
    uhd::device::find(arg);
if(device_addr.size() == 0)
    std::cerr << "No USRP Device Found" << std::endl;
else
    std::cout << "USRP Device Found" << std::endl \
    << device_addr[0].to_pp_string() << std::endl;
```

Using the UHD with GNU Radio

- GNU Radio provides blocks one can use to access the UHD.
- The blocks are written in C++, but GNU Radio provides Python bindings via SWIG.
- You can write your SDR in the GNU Radio framework in C++ *or* Python.
- GNU Radio also provides a tool called GNU Radio Companion (GRC).
  - GRC is a graphical frontend to GNU Radio, where one can organize blocks together to make their SDR.

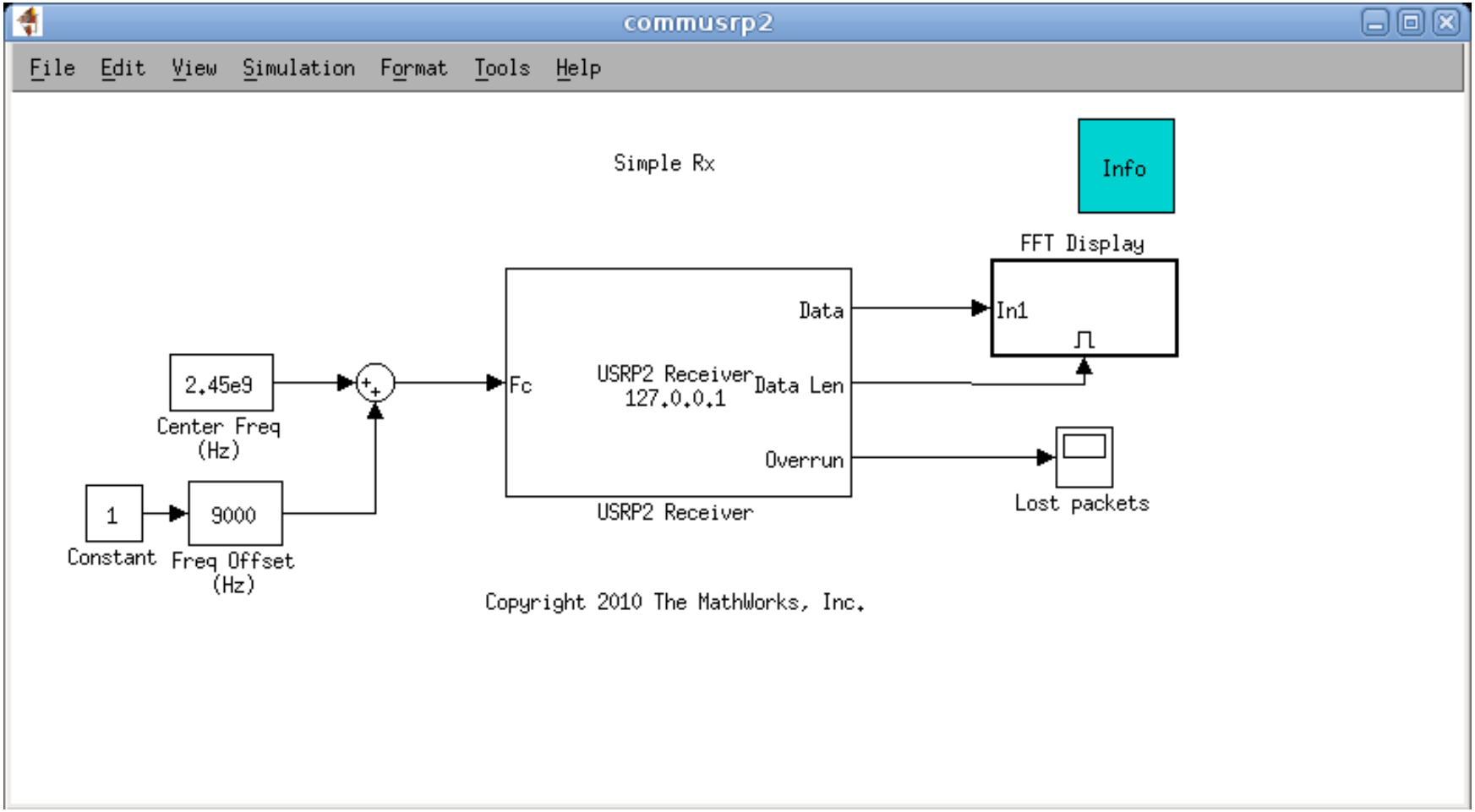
Python Example

```
from gnuradio import uhd
class top_block(stdgui2.std_top_block):
  __init__():
    self.u = uhd.usrp_source(address,\
    uhd.io_type.COMPLEX_FLOAT32, 1)
    self.u.set_samp_rate(40000000)
    self.scope = fftsink2.fft_sink_c (panel, \
                512, 4000000, ...)
    self.connect(self.u, self.scope)
if __name__ == '__main__()':
   app = stdgui2.stdapp(top_block,"FFT",nstatus=1)
   app.MainLoop()
```

Using the UHD with Simulink

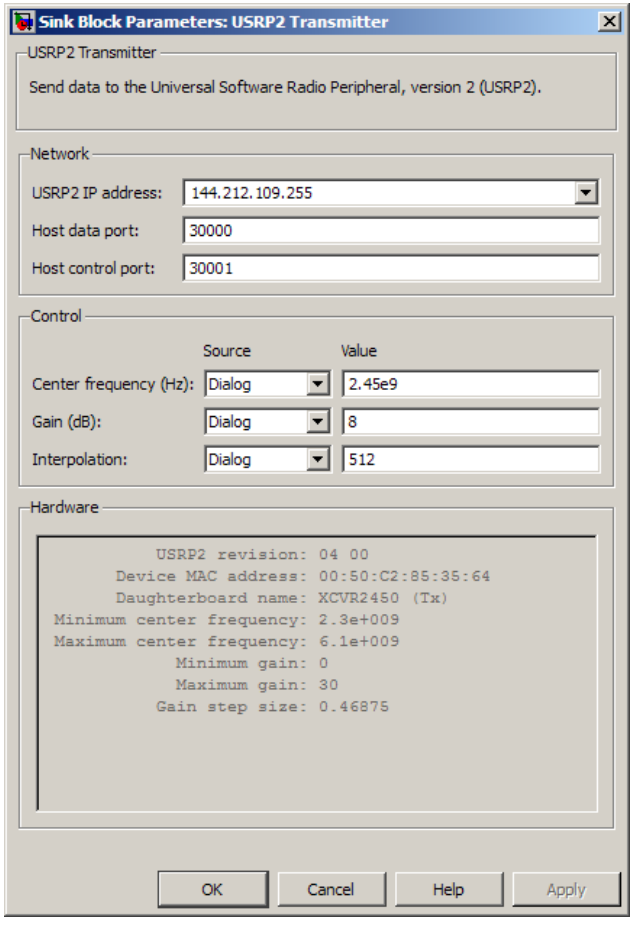
- Simulink is a graphical system for designing models, it is distributed by the MathWorks.
- There is a Simulink block that is distributed with MATLAB/Simulink starting with 2011a that can be used with the USRP2.
- There is a block distributed by Karlsruher Institute for Technology for the USRP1.
  - `http://www.cel.kit.edu/installation.php`
- Resources:
  `http://www.mathworks.com/discovery/sdr/usrp.html`
  `http://www.mathworks.com/help/toolbox/comm/ref/usrp2transmitter.html`
  `http://www.mathworks.com/help/toolbox/comm/ref/usrp2receiver.html`

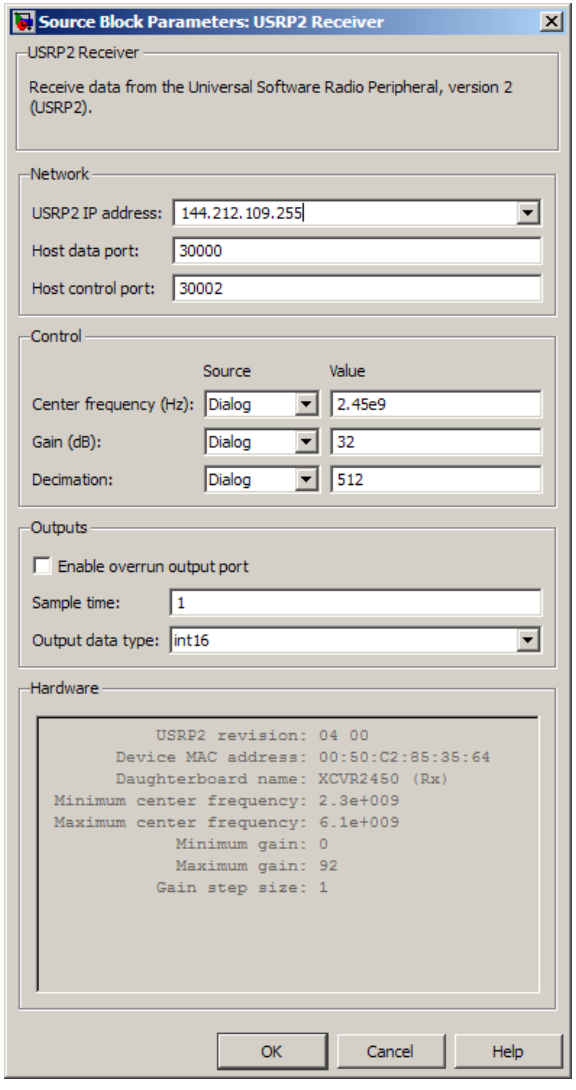
An Example USRP Model



Type `commusrp2` at a MATLAB prompt.

USRP Masks


Sink Block Parameters: USRP2 Transmitter
USRP2 Transmitter
Send data to the Universal Software Radio Peripheral, version 2 (USRP2).
Network
USRP2 IP address: 144.212.109.255
Host data port: 30000
Host control port: 30001
Control
Source
Value
Center frequency (Hz): Dialog 2.45e9
Gain (dB): Dialog 8
Interpolation: Dialog 512
Hardware
USRP2 revision: 04 00
Device MAC address: 00:50:C2:85:35:64
Daughterboard name: XCVR2450 (Tx)
Minimum center frequency: 2.3e+009
Maximum center frequency: 6.1e+009
Minimum gain: 0
Maximum gain: 30
Gain step size: 0.46875
OK
Cancel
Help
Apply


Source Block Parameters: USRP2 Receiver
USRP2 Receiver
Receive data from the Universal Software Radio Peripheral, version 2 (USRP2).
Network
USRP2 IP address: 144.212.109.255
Host data port: 30000
Host control port: 30002
Control
Source
Value
Center frequency (Hz): Dialog 2.45e9
Gain (dB): Dialog 32
Decimation: Dialog 512
Outputs
Enable overrun output port
Sample time: 1
Output data type: int16
Hardware
USRP2 revision: 04 00
Device MAC address: 00:50:C2:85:35:64
Daughterboard name: XCVR2450 (Rx)
Minimum center frequency: 2.3e+009
Maximum center frequency: 6.1e+009
Minimum gain: 0
Maximum gain: 92
Gain step size: 1
OK
Cancel
Help

USRP with LabView

- NI Re-brands the USRP as USRP-2920 and USRP-2921
- **Web**
  - `http://zone.ni.com/devzone/cda/tut/p/id/12985`
  - `http://www.ni.com/academic/usrp.htm`
- **Mail**
  - `niusrp@ni.com`

Closing Remarks

- The UHD has solved many problems for the USRP2 family of products.
  - USRP traffic is now routable and doesn't require root access.
  - The USRP can be used on all three major platforms.
  - Proprietary software can link to the UHD and not have to open source their code.

Resources and Getting Help

Resources

- `http://files.ettus.com/uhd_docs/manual/html/index.html`
- `http://code.ettus.com/redmine/ettus/projects/uhd/wiki`

Getting Help

- Mailing lists
  - USRP Users `usrp-users@lists.ettus.com`
  - GNU Radio Discuss `discuss-gnuradio@gnu.org`
- Doxygen
- `pydoc` and `pydoc -p`

Questions

Questions

Thanks

For a copy of the slide deck mail dwwkelly@gmail.com.