The Universal Hardware Driver

Devin Kelly

October 1st, 2011
Contents

1 Introduction

2 The USRP
   ■ Communicating with the USRP

3 Getting, Building and Installing the UHD

4 Using the UHD
   ■ By Itself
   ■ With GNU Radio
   ■ With SIMULINK
   ■ With LabView

5 Conclusion
My Background

- B.S., Electrical and Computer Engineering from WPI in 2010.
- M.S., Electrical and Computer Engineering from WPI in 2011.
- 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
<table>
<thead>
<tr>
<th></th>
<th>USRP1</th>
<th>USRP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Rate</td>
<td>2x 64 Msps</td>
<td>2x 100 Msps</td>
</tr>
<tr>
<td>ADC Dynamic Range</td>
<td>12 bit</td>
<td>14 bit</td>
</tr>
<tr>
<td>PC Interface</td>
<td>USB 2.0</td>
<td>Gigabit Ethernet</td>
</tr>
<tr>
<td>RF Bandwidth</td>
<td>16 MHz</td>
<td>25 MHz</td>
</tr>
</tbody>
</table>

|                        | 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 OS X 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++ compiler, 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:

- 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:

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

```bash
```

```bash
```
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 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.

```text
host/examples/
  rx_ascii_art_dft.cpp
```


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.
Initialize the USRP

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

Set some parameters:

```cpp
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:

```cpp
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);
```
Discover a USRP

```cpp
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;
```
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.
from gnuradio import uhd

class top_block(stdgui2.std_top_block):
    __init__(self):
        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](http://www.cel.kit.edu/installation.php)

- Resources:
An Example USRP Model

Type `commusrp2` at a MATLAB prompt.
USRP Masks

Sink Block Parameters: USRP2 Transmitter

Network
- USRP2 IP address: 144.212.109.255
- Host data port: 30000
- Host control port: 30001

Control
- Center frequency (Hz): Dialog, Value: 2.45e9
- Gain (dB): Dialog, Value: 8
- Interpolation: Dialog, Value: 512

Hardware
- USRP2 revision: 04 00
- Device MAC address: 00:50:62:18:35:64
- Daughterboard name: XCVR2450 (7x)
- Minimum center frequency: 1.9e9
- Maximum center frequency: 8.4e9
- Minimum gain: 0
- Maximum gain: 30
- Gain step size: 0.04678

Source Block Parameters: USRP2 Receiver

Network
- USRP2 IP address: 144.212.109.255
- Host data port: 30000
- Host control port: 30002

Control
- Center frequency (Hz): Dialog, Value: 2.45e9
- Gain (dB): Dialog, Value: 32
- Decimation: Dialog, Value: 312

Outputs
- Enable overrun output port
- Sample time: 1
- Output data type: INT16

Hardware
- USRP2 revision: 04 00
- Device MAC address: 00:80:C2:88:38:64
- Daughterboard name: XCVR2450 (Ra)
- Minimum center frequency: 6.14e9
- Maximum center frequency: 6.14e9
- Minimum gain: 0
- Maximum gain: 92
- Gain step size: 1
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
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


Getting Help

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

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