Getting Started with UHD and C++

Contents

- 1 Application Note Number
- 2 Revision History
- 3 Abstract
- 4 Overview
- 5 UHD Manual
- 6 Including Header Files
- 7 UHD_SAFE_MAIN()
- 8 set_thread_priority_safe()
- 9 Create Variables
- 10 Creating a USRP Object
- 11 Setting the Motherboard Clocks
- 12 Selecting the Sub Device
- 13 Getting the selected Sub Device
- 14 Setting the Sample Rate
- 15 Getting the Sample Rate
- 16 Setting Center Frequency
- 17 Getting the Center Frequency
- 18 Setting the RF Gain
- 19 Reading the RF Gain
- 20 Setting the IF Filter Bandwidth
- 21 Getting the IF Filter Bandwidth
- 22 Selecting the Antenna
- 23 Getting the Antenna
- 24 Exiting
- 25 Full Example
- 26 CMake
- 27 Compile and Install
- 28 Running the Application
- 29 Additional Example Programs

AN-204

Date	Author	Details
2016-05-01	Neel Pandeya Nate Temple	Initial creation

This AN explains how to write and build C++ programs that use the UHD API.

This Application Note will walk through building a basic C++ program with UHD. This program will initialize, configure the USRP device, set the sample rate, frequency, gain, bandwidth, and select the antenna.

The UHD Manual is hosted at http://files.ettus.com/manual/index.html and provides information on how to use the USRP devices and how to use the UHD API to connect to them through your own software. The manual is split into two parts: The device manual, and the UHD/API manual. The first part describes details of Ettus Research devices, motherboards and daughterboards, as well as aspects of using UHD. The second is meant for developers writing UHD-based applications, and includes descriptions of the API, sorted by namespaces, classes, and files.

Defines a safe wrapper that places a catch-all around main. If an exception is thrown, it prints to stderr and returns.

Set the scheduling priority on the current thread. Same as set_thread_priority but does not throw on failure.

Make a new multi usrp from the device address.

Set the clock source for the usrp device. This sets the source for a 10 MHz reference clock. Typical options for source: internal, external, MIMO.

Set the RX frontend specification. The subdev spec maps a physical part of a daughter-board to a channel number. Set the subdev spec before calling into any methods with a channel number. The subdev spec must be the same size across all motherboards. For more details on selecting the Sub Device, see the Specifying the Subdevice section of the UHD Manual.

Get a printable summary for this USRP configuration.

Set the RX sample rate. The rate is in Samples Per Second.

Gets the RX sample rate. Returns the rate is in Samples Per Second.

Create a tune request, with the RF frequency in Hz. Set the RX center frequency.

Get the RX center frequency. Returns the frequency in Hz.

Set the RX gain value for the specified gain element. For an empty name, distribute across all gain elements. Sets the gain in dB.

Get the RX gain value for the specified gain element. For an empty name, sum across all gain elements. Returns the gain in dB.

Set the RX bandwidth on the frontend. Sets the bandwidth in Hz.

Get the RX bandwidth on the frontend. Returns the bandwidth in Hz.

Select the RX antenna on the frontend.

Get the selected RX antenna on the frontend. Returns the antenna name.

Use the `uhd/host/examples/init_usrp/CMakeLists.txt` file as template

- Add the names of your C++ source files to the add_executable(...) section
- Put both modified CMakeLists.txt file and C++ file into an empty folder

- Create a ?build? folder and invoke CMake the usual way:

```
mkdir build
cd build
cmake ../
make
```

```
$ ./usrp_basic
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.git-202-g9e0861e1


Creating the usrp device with: addr=192.168.10.2...
-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes
Lock mboard clocks: internal
subdev set to: A:0
Using Device: Single USRP:
  Device: USRP2 / N-Series Device
  Mboard 0: N210r4
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: WBXv2 RX+GDB
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: WBXv2 TX+GDB

Setting RX Rate: 1.000000 Msps...
Actual RX Rate: 1.000000 Msps...

Setting RX Freq: 915.000000 MHz...
Actual RX Freq: 915.000000 MHz...

Setting RX Gain: 10.000000 dB...
Actual RX Gain: 10.000000 dB...

Setting RX Bandwidth: 1.000000 MHz...
```

```
Actual RX Bandwidth: 1.000000 MHz...

Setting RX Antenna: TX/RX
Actual RX Antenna: TX/RX
```

Additional C++ example programs using the UHD API are provided within the Ettus Research Github Repository, located in the `host/examples/` directory. These examples are installed with UHD and will be located at `$prefix/lib/uhd/examples` directory of your system.