Building and Installing the USRP Open Source Toolchain (UHD and GNU Radio) on Windows

Contents

AN-611

This Application Note provides a step-by-step guide for building, installing, and updating the open-source toolchain, specifically UHD and GNU Radio, for the USRP from source code on Windows.

UHD is fully supported on Windows 7, 8, 8.1, and 10 and can be compiled using Visual Studio 2013, 2015, or 2017. The future target is to support the latest three OS and Visual Studio releases.

The following dependencies are required for a regular build.

- Microsoft Windows (7, 8, 8.1, 10)
- Microsoft Visual Studio (2013, 2015, 2017)
- CMake (2.8.0 or later)
- Boost (1.53 or later)
- LibUSB (1.0 or later)
- Python (2.7.x)
- Mako (0.5.0 or later)
- Doxygen (1.8 or later, optional)
- NSIS (2.50 or later, optional)

Optional Tools:

- 7zip (http://www.7-zip.org/download.html)
- msysGit (https://gitforwindows.org/)

The free version of Visual Studio is sufficient for building UHD. This guide was tested using Windows 10 x64 and Microsoft Visual Studio Community 2017 (v15.9).

Users will need to install the "Desktop Development for C++" Workload for Visual Studio. This can be found in:

Tools >> Get Tools and Features... >> Workloads

https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads

https://www.visualstudio.com/downloads/

CMake is a cross-platform build system used to configure and generate the files necessary to compile and test UHD for a particular computer environment.

Made sure to download the ".msi" file, not the ".zip" file. During installation select the option to "Add CMake to the system PATH for the current user" or for "all users", depending on which is more applicable for your Windows usage.

CMake 3.13.2 (win64-x64) was used for the guide. UHD has been tested to work through CMake 3.17.2.

https://cmake.org/download/

Boost is a set of C++ libraries providing useful algorithms and data structures.

UHD builds with Boost through 1.72.0. From the link below, select the version of Boost you wish to build against. At the bottom of the directory listing will be a file "DEPENDENCY_VERSIONS.txt". Review this file to determine the correct version of the Boost installer to download for your specific MSVC version. The Boost binary installer must be selected to match the version of MSVC being used to compile UHD and architecture of Windows being run -- 32 or 64 bit. Watch out as Microsoft has done the version numbering of MSVC such that the year and version number do not match. Here are the recent entries in the noted file:

```
 Microsoft Visual Studio 2013 - msvc-12.0 - Update 5
 Microsoft Visual Studio 2015 - msvc-14.0 - Update 3
 Microsoft Visual Studio 2017 - msvc-14.1 - VS 15.9.17
 Microsoft Visual Studio 2019 - msvc-14.2 - VS 16.3.6
```

Note that VS2019 support (MSVC Toolset 14.2) starts unofficially with Boost 1.70.0, and then officially with 1.71.0.

After the Boost installer executable is downloaded, find it in the Windows File Explorer, right click on it and "Run as administrator". During the install, select the destination location of "C:\Program Files\Boost\" for the install; the installer will append the Boost directory and version info, such that the full destination will actually be, for example, "C:\Program Files\Boost\boost_1_68_0".

VS2017 uses the MSVC Toolset 14.1 so boost_1_68_0-msvc-14.1-64 was selected for this tutorial, and installed as "C:\Program Files\Boost\boost_1_68_0".

https://sourceforge.net/projects/boost/files/boost-binaries/

LibUSB is a cross-platform library providing access to USB devices. UHD is compatible through LibUSB 1.0.24 (the current release as of this update).

LibUSB releases are distributed as 7zip or tar.bz2 archives. There are plenty of free "unzip" programs available that can extract the files in either archive type; Windows does not provide this capability, so you'll need to download one of these programs. We are using the "7Zip" program as linked in the ?Setting Up the Environment? Section. After installing 7zip the LibUSB release archive can be extracted by right clicking on the downloaded file and selecting `7zip >> Extract files`.

LibUSB 1.0.22 was used for the guide and extracted to `C:\Users\username\libusb-1.0.22`.

https://sourceforge.net/projects/libusb/files/libusb-1.0/

Note: The directory to which you extract libusb must not contain spaces. This is to say that `C:\Users\user name\libusb-1.0.22` will cause compile issues moving forward.

Python is a widely-used general-purpose, high-level programming language. UHD includes several utilities written in Python and has several scripts which are part of the build process. UHD is compatible with Python 2.7, and 3.5 through 3.8. The default Python install will be 32 bit. To get the 64-bit install, you have to select it specifically -- "Download Windows x86-64 executable installer" for Python 3.0+ or "Download Windows x86-64 MSI installer" for Python 2.7 (any version). When installing Python make sure to add to the PATH so that "python.exe" is available from the commandline.

The Python 2.7.15 was used for this tutorial.

https://www.python.org/downloads/windows/

For building the LibUHD Python API the Python libraries NumPy and ruamel.yaml are required. NumPy adds support for large matrices and arrays and mathematical functions around them. ruamel.yaml is used for parsing/emitting YAML. To install them open a command line (Ctrl-X, select Run, type `cmd.exe` and click OK)

```
   cd C:\Python27\Scripts
   pip.exe install numpy
   pip.exe install ruamel.yaml
```

Mako is a python template library used to generate source files and is distributed using a Python package management system.

Open a command line (Ctrl-X, select Run, type `cmd.exe` and click OK)

```
   cd C:\Python27\Scripts
   pip.exe install mako
```

This installed version 1.0.7 at the time of writing this guide.

Doxygen is a documentation generator which creates the HTML manual from text in the source code. It optional for building UHD.

Version 1.8.14 was used for this guide.

https://sourceforge.net/projects/doxygen/files/rel-1.8.14/

NSIS is a toolkit for creating Windows installers. NSIS is used for creating binary packages of UHD enabling easy distribution and installation of UHD, associated utilities, and examples.

Version 3.03 was used for this guide.

https://sourceforge.net/projects/nsis/files/NSIS%203/

The UHD source code is stored in a GIT repository: https://www.github.com/EttusResearch/uhd . From this link, one can download an archive of the UHD source code for any release, the current public codebase itself, and the code state for some branches or tags. Which one you download is up to your needs. We generally recommend the latest release, which can be downloaded from our Github repository or from our website as noted below.

For the purposes of this guide, the archive was downloaded from our website, and extracted as `C:\Users\username\uhd-release\`. Please note that most UHD archives will extract to a directory name that corresponds to the UHD version; for example the one noted extracts to "uhd_3.15.0.0-release". After extraction, we change the directory name to that as noted ("uhd-release") and move it into the directory as noted. You can install the UHD source wherever you want to, of course; for this article we choose this location and use it later as such.

Archives of the source code for each stable release can be downloaded from the Ettus website.

http://files.ettus.com/binaries/uhd/

7zip can be used to extract the `tar.gz` archive to a location of your choosing.

This guide used the Git repository to obtain the newest release of UHD, read on for details.

The latest development code, as well as tagged releases, is available from the git repository hosted on GitHub

https://www.github.com/EttusResearch/uhd

For step-by-step instructions using a git client, see section Building and installing UHD from source code in the UHD Linux Installation Guide. Return to this document after you have successfully checked out your desired release of UHD, and note that some directory names moving forward may differ slightly.

All prerequisites have now been installed and downloaded. You are encouraged to restart your machine before continuing.

- Open the Cmake GUI
- Select source code directory
  `C:\Users\username\uhd-release\host\`
- Select binary build directory (this may require creating the folder \build\)
  `C:\Users\username\uhd-release\host\build\`
- Check the Advanced checkbox
- Click Configure
    - Set Visual Studio 15 2017 Win64 as the compiler
    - Click ?Finish? and allow CMake to Generate
- Change or add the following entries
  Boost_INCLUDE_DIR
  `C:\Program Files\Boost\boost_1_68_0\`
- Add the following entries with type PATH
  Boost_LIBRARY_DIRS
  `C:\Program Files\Boost\boost_1_68_0\lib64-msvc-14.1\`
  LIBUSB_INCLUDE_DIRS
  `C:\Users\username\libusb-1.0.22\include\libusb-1.0\`
- Add the following entry with type FILEPATH
  LIBUSB_LIBRARIES
  `C:\Users\username\libusb-1.0.22\MS64/dll\libusb-1.0.lib`
- Click Generate

Open Visual Studio 2017 and open the UHD project file generated by CMake.

```
File > Open Project
        C:\Users\username\uhd-release\host\build\UHD.sln
```

Change the build type from Debug to Release. The **ALL_BUILD** project should be selected in the Solution Explorer, select it if this is not the case. Run the build, `Build > Build` **`ALL_BUILD`**.

Select the **INSTALL** project in the Solution Explorer and run the build, `Build > Build` **`INSTALL`**.

Visual Studio must be run as Administrator for this to succeed as it needs write permission for the `C:\Program Files` directory.

Building the **PACKAGE** project will produce a binary installer if NSIS is installed. This installer with be for either 64 bit or 32 bit as chosen during the CMake step. Select the **PACKAGE** project in the Solution Explorer and run the build, `Build >` **`Package`**.

Running programs using UHD requires the Visual Studio C++ Runtime Redistributable to be installed. The version of the C++ Runtime Redistributable must match the version of Visual Studio used to compile the program.

- Visual Studio 2017
- Visual Studio 2015
- Visual Studio 2012 and 2013

If a USB connected USRP is used then the USB drivers must be installed. The drivers are located at http://files.ettus.com/binaries/misc/erllc_uhd_winusb_driver.zip.

There is a known issue with Windows 10 where an error message is shown at the end of driver installation. However resetting or power cycling the USRP enables full functionality.

It should be noted that this guide is intended to serve as an example; the software versions and dependencies listed above are not the only possible combinations. Be sure to review the Build Dependencies in the USRP Hardware Driver and USRP Manual for information regarding the minimum required driver versions for your build of UHD. To preserve the software stack from the previous iteration of this Application Note, here is an example driver stack for Windows 7 + VS 2013:

- Microsoft Windows 7 x64
- Microsoft Visual Studio 2013
- CMake 3.4.1 win32-x86
- Boost 1.58.0 msvc-12.0 x64
- LibUSB 1.0.20
- Python 2.7.11
- Mako 1.0.3
- Doxygen 1.8
- NSIS 2.50

Building GNU Radio from source on Windows is still an involved process due to the large number of dependencies. A set of scripts have been developed to automate the process by Geof Nieboer. The links below will detail the process to building GNU Radio + UHD.

Note: The linked instructions below will build GNU Radio along with UHD, which is separate from the instructions above within this Application Note. The scripts linked below are not maintained by Ettus Research, and are considered third-party binary packages, and are not directly supported by Ettus Research.

https://github.com/gnieboer/gnuradio_windows_build_scripts http://www.gcndevelopment.com/gnuradio/