

RHX

Application Note: Building RHX Source Code

28 February 2022

The RHX code is built with a C++ framework called Qt, and while the initial download and installation of Qt can take some time, it is necessary for developers who wish to modify the Intan RHX software (or any of the Intan legacy programs that similarly are built with Qt). Qt is a free framework that allows for cross-platform (Windows, Mac, and Linux) programs with useful features like UI, multi-threading, and much more. This document aims to assist developers in setting up (1) the Qt Creator IDE, (2) the install of the Qt framework itself (and an accompanying C++ compiler), and (3) the steps to properly build the Intan RHX project with Qt.

While the examples in this document use Windows, Qt is cross-platform and analogous steps can be taken on Mac and Linux. The end of this document includes a section on platform-specific concerns for non-Windows builds.

Qt and Qt Creator can be downloaded from the Qt website: <https://www.qt.io/product/development-tools>. By selecting 'Download Qt', 'Go open source', and 'Download the Qt Online Installer' you can download Qt's installer program that will then allow you to install the specific Qt features you need. This will require you to create a free Qt account.

1. **Qt Creator.** While it is possible to compile through other IDEs, we recommend you use the (free) IDE, Qt Creator, available from the Qt Online Installer. The exact version of Qt Creator is not that important; you should probably use the most up-to-date release available. At the time of writing this document, we are using version 4.14.0. Note that the version of Qt Creator is not related to the version of Qt that a project is built with, which can be much more relevant. With a single version of Qt Creator, developers can select several different versions of Qt to build a project with. Once Qt Creator has been downloaded and installed, you can double-check the version you have with 'Help' -> 'About Qt Creator...'.
Qt/C++. You'll need the Qt framework itself, which can also be downloaded through the Qt Online Installer. Qt has a lot of different versions, some with significant changes, so it's important to get a version close to what we have used to develop the Intan RHX software. We currently use Qt 5.12.8 for RHX compilation, and we compile RHX as a 64-bit binary. While you could likely use different Qt versions successfully, we recommend you install the 64-bit 5.12.8 version. Note that in the Qt installer, you can click the 'Archive' checkbox and re-run the filter to get a more complete list of available Qt versions.

You'll also need a C++ compiler to link your Qt version to make up a 'kit'. For Windows, we use MSVC 2017 64-bit. The full name is Microsoft Visual C++ Compiler 16.5.30011.22 (x86_amd64). You can probably use a different compiler with some minor changes, but for consistency we recommend using the same compiler if possible.

3. **Build Intan RHX Software.** Finally, using Qt Creator, we link the Qt version with our C++ compiler to create a kit, and then use that kit to build the IntanRHX project. The first step is to open the IntanRHX.pro file, with 'File' -> 'Open File or Project...'. Click the 'Projects' tab on the left, then click on 'Manage Kits' to bring up a dialog to set up your configuration. If the auto-detect feature works with no problems, you should see 'Desktop Qt 5.12.8 MSVC2017 64bit' listed under the auto-detect section of Kits. (Don't worry if the other listed Qt kits vary from the picture, as this will change depending on what versions of Qt and C++ compilers you have installed.)

RHX Application Note

Qt Options - Qt Creator

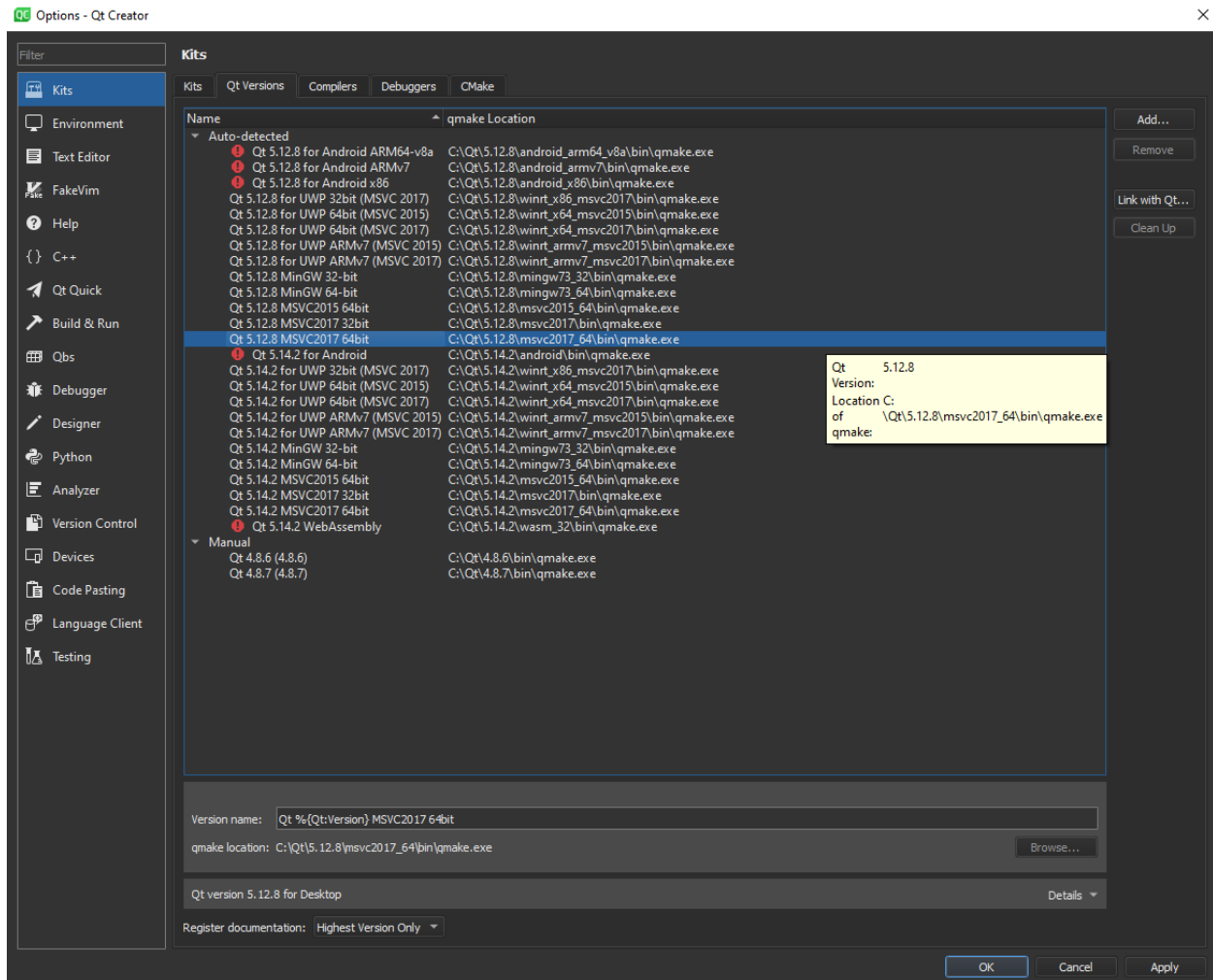
Kits

Desktop Qt 5.12.8 MSVC2017 64bit

Device type: Desktop
Device: Local PC
Sys Root:
Compiler: Microsoft Visual C++ Compiler 16.5.30011.22 (x86_amd64)
Environment:
Debugger: CDB Engine using "C:\Program Files (x86)\Windows Kits\10\Debuggers\x64\cdb.exe"
Qt version: Qt 5.12.8 MSVC2017 64bit
mkspec:
Additional Qbs Profile Settings:
CMake: System CMake at C:\Program Files\CMake\bin\cmake.exe
CMake Generator: Generator: Ninja
Extra generator:
CMake Configuration: CMAKE_CXX_COMPILER:STRING=% {Compiler:Executable:Cxx}
{Compiler:Executable:Cxx}
CMAKE_C_COMPILER:STRING=%{Compiler:Executable:C}
CMAKE_PREFIX_PATH:STRING=%{Qt:QT_INSTALL_PREFIX}
QT_QMAKE_EXECUTABLE:STRING=% {Qt:qmakeExecutable}

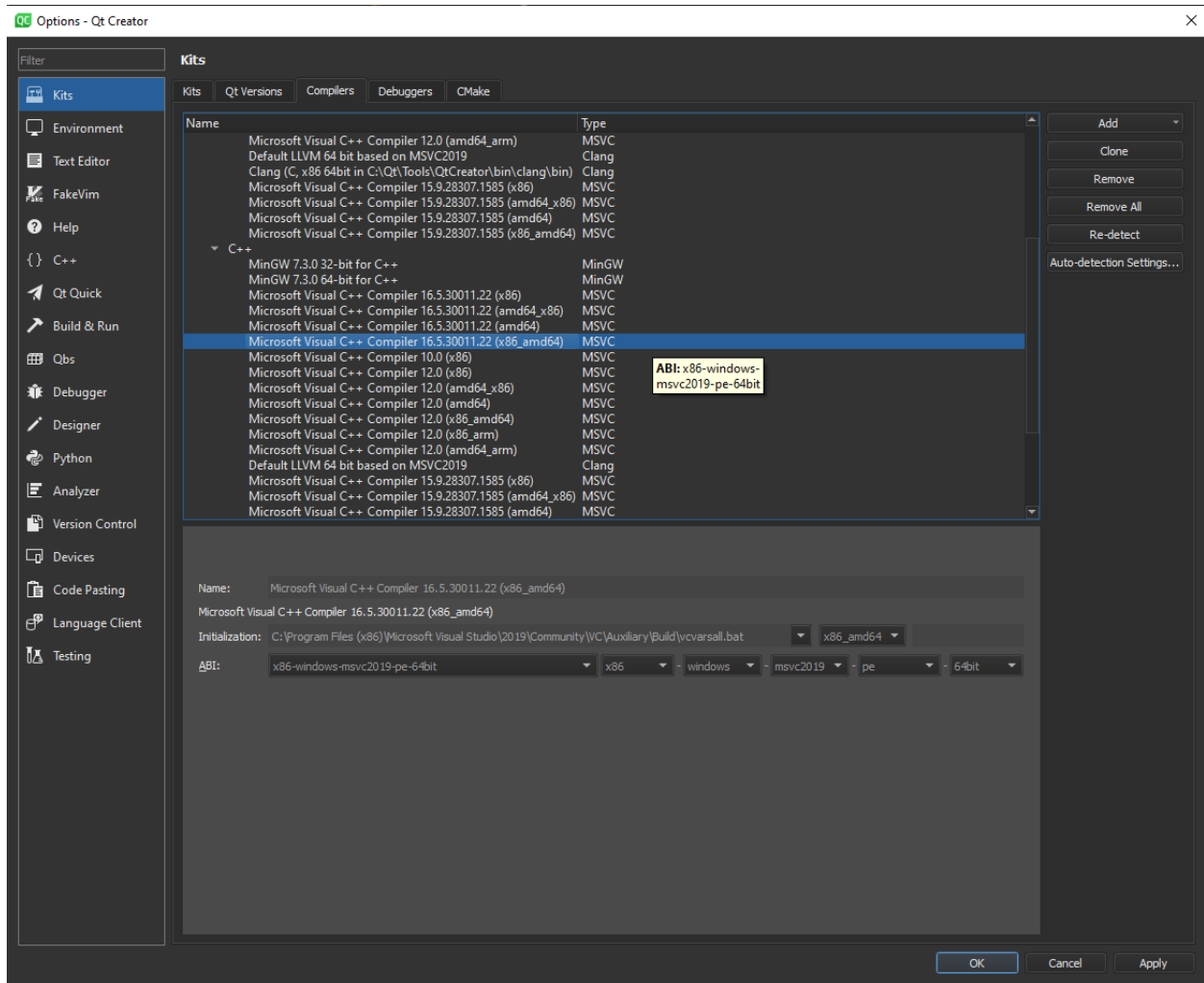
RHX Application Note

If you don't see this yet, make sure your Qt Versions looks something like this:



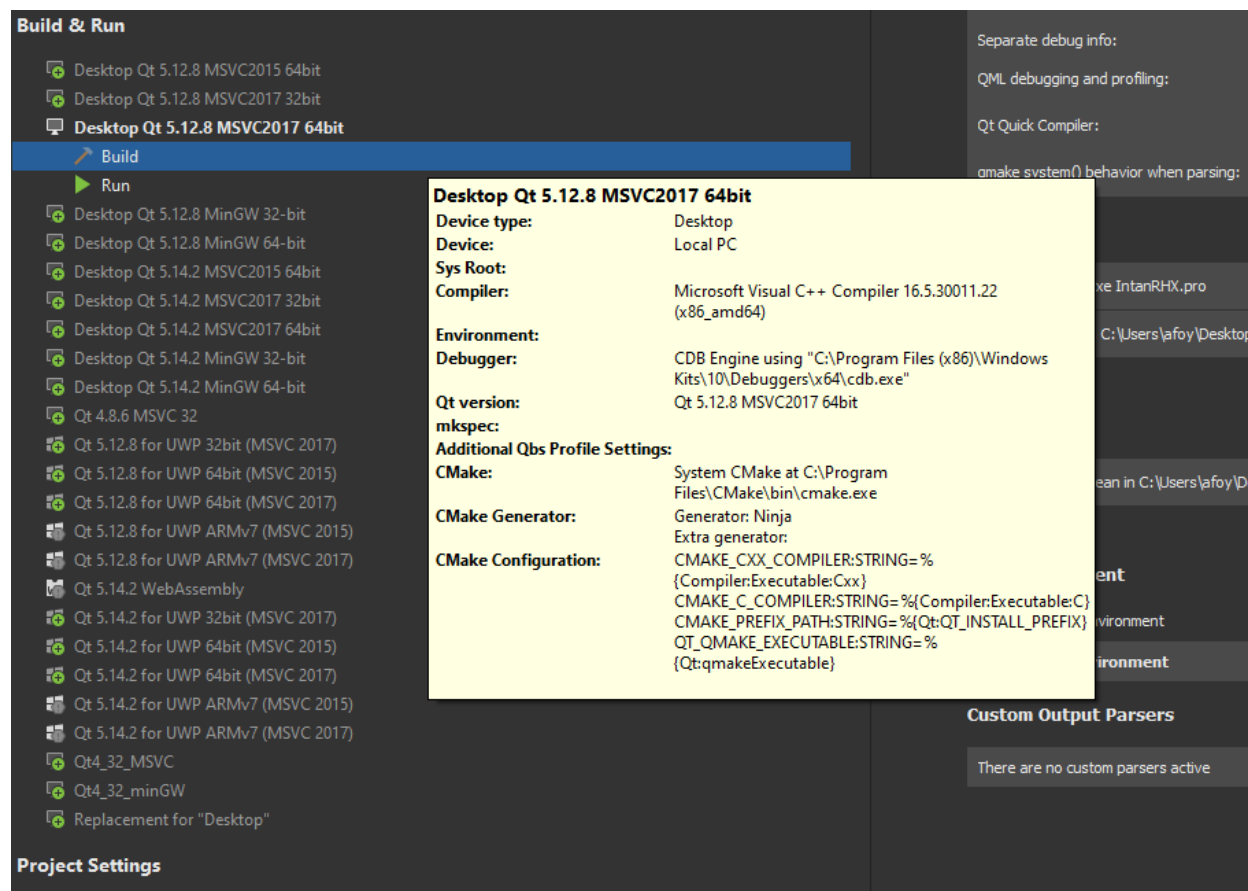
RHX Application Note

And your Compilers looks something like this:



RHX Application Note

While there are options to manually add your own Qt versions and compilers, we've only had to do that rarely. If your installed compilers don't show up from the auto-detect, then something is probably wrong. You might need to point Qt Creator to where the correct 'qmake.exe' file for your given Qt version is located. Once you have a valid kit set up, just make sure that it is active in the 'Projects' tab:



The screenshot shows the 'Build & Run' tab in Qt Creator. On the left, a list of kits is shown under the 'Build' and 'Run' sections. The kit 'Desktop Qt 5.12.8 MSVC2017 64bit' is selected. A tooltip window is open over this kit, displaying the following details:

| Desktop Qt 5.12.8 MSVC2017 64bit | |
|----------------------------------|--|
| Device type: | Desktop |
| Device: | Local PC |
| Sys Root: | |
| Compiler: | Microsoft Visual C++ Compiler 16.5.30011.22 (x86_amd64) |
| Environment: | |
| Debugger: | CDB Engine using "C:\Program Files (x86)\Windows Kits\10\Debuggers\x64\cdb.exe" |
| Qt version: | Qt 5.12.8 MSVC2017 64bit |
| mkspec: | |
| Additional Qbs Profile Settings: | |
| CMake: | System CMake at C:\Program Files\CMake\bin\cmake.exe |
| CMake Generator: | Generator: Ninja Extra generator: |
| CMake Configuration: | CMAKE_CXX_COMPILER:STRING= % {Compiler:Executable:Cxx} CMAKE_C_COMPILER:STRING= % {Compiler:Executable:C} CMAKE_PREFIX_PATH:STRING= % {Qt:QT_INSTALL_PREFIX} QT_QMAKE_EXECUTABLE:STRING= % {Qt:qmakeExecutable} |

As long as that is active, you should be able to click 'Edit' and begin to navigate the IntanRHX project and edit C++ files at will, and then select either 'Debug' or 'Release' above the green Run button at the bottom left. You should then be able to click the green Run button to start the code building and running. Note that the first build is likely to take a long time.

Note that at runtime, the IntanRHX software looks in the executable directory for various files, including 'kernel.cl', 'ConfigRHDController.bit', 'ConfigRHDInterfaceBoard.bit', 'ConfigRHController.bit', 'ConfigXEM6010Tester.bit', 'USBEvaluationBoard.bit', and 'okFrontPanel.dll'.

RHX Application Note

Platform-Specific Concerns

Mac: Instead of using a Qt 5.12.8 MSVC2017 64bit kit, we've used Qt 5.12.10 with the 'Clang (C++, x86 64bit in /usr/bin)' compiler. Also, instead of looking for 'okFrontPanel.dll', the file 'libokFrontPanel.dylib' is needed. Depending on which stage of the build/deploy process you're on, you may need this file to be alongside the IntanRHX executable (inside the IntanRHX.app -> Contents -> MacOS directory), or in Frameworks (inside the IntanRHX.app -> Contents -> Frameworks directory).

Linux: We've had success only when building Qt statically. The details of how to build Qt statically on Linux is beyond the scope of this document, but ultimately the command that worked for us was:

```
./configure -static -prefix ~/qt-build-dir -platform linux-g++ -xcb-xlib  
-fontconfig -system-freetype
```

Then,

```
make
```

followed by

```
make install
```

Once Qt is built, build IntanRHX statically from the directory that contains the .pro file:

```
PATH=~/qt-build-dir/bin:$PATH
```

```
export PATH
```

```
qmake -config release
```

```
make
```

After this, an IntanRHX binary should be built. Make sure to run this with libokFrontPanel.so and rules.d in the same directory, and you should be able to run it.