

## How to build a working Cross-Compile Environment for Wago PFC Devices

- **This guide was only tested on Ubuntu 19.04 !**
- Follow the steps in the Official Guide “<https://github.com/WAGO/pfc-firmware-sdk>” until you reach step 6.)

→ You should now have a working environment to build Applications for the Wago PFC.

- Get the latest(desired) Qt-Version.  
Either by downloading the Source directly from the Website:

- “<https://download.qt.io/archive/qt/5.13/5.13.0/single/>”

Or by downloading a specific Qt Version with git:

- `git clone -b 5.13.0 git://code.qt.io/qt/qt5.git`  
Afterwards you need to call `./init_repository`, optionally with the Modules you would like otherwise all modules will be downloaded.

Info: For further information see: “[https://wiki.qt.io/Get the Source](https://wiki.qt.io/Get_the_Source)”

- Copy the Folder “linux-arm-PFCXXX-g++” into the downloaded Qt Folder:  
→ `qtbases → mkspecs → devices`
- Open a Terminal in the basefolder of the qt-source. You should see a file named `configure` in the folder.

Now we need to export some Variables for the Configuration, you should adjust the Paths to fit your paths.

ATTENTION: If you want to copy the Sysroot make sure to use `cp -r -L` to also follow the symbolic links

- “`export TOOLCHAIN=*PATH_TO_YOUR_TOOLCHAIN*`”  
For example the LINARO Toolchain used in the WAGO Tutorial  
→ `opt/wago/PFCXXX/toolchain/arm-linux-gnueabi/bin/arm-linux-gnueabi-`
- “`export HOST_SYSROOT=*PATH_TO_HOST_SYSROOT*`”  
The host sysroot is the folder of your Toolchain Sysroot, if you don’t have one follow either:
  - Download the precompiled sysroot from wago(“[PRECOMPILED\\_TOOLCHAIN](#)”)
  - Build it yourself(“[BUILD\\_TOOLCHAIN](#)”).→ `opt/wago/PFCXXX/toolchain/arm-linux-gnueabi/sysroot`
- “`export TARGET_SYSROOT=*PATH_TO_TARGET_SYSROOT*`”  
The Target sysroot is also the Folder we created in the Wago Tutorial  
You will find the Folder in `ptxproj → platform-wago-pfcXXX → sysroot-target`

- “export PREFIX=\*PATH\_YOU\_WANT\_THE\_QT\_VERSION\_TO\_INSTALL\_TO\*”  
The Path the newly build Qt-Version will install to.  
For example: /opt/Qt/5.13.0/wago\_pfcxxx

“export EXTPREFIX=\*PATH\_YOU\_WANT\_THE\_SYSROOT\_TO\_INSTALL\_TO\*”  
By default the sysroot will be installed into the Toolchain folder. If you want the Sysroot to install into a different folder you should set this Value, otherwise leave it out in the Configure.

- With all the Variables defined we can now build our Qt-Version for Cross-Compilation.

- Execute the command:  
./configure -device linux-arm-PFCXXXX-g++ -device-option  
CROSS\_COMPILE=\$TOOLCHAIN -device-option  
SYSROOT\_TARGET=\$TARGET\_SYSROOT -sysroot \$HOST\_SYSROOT -prefix  
\$PREFIX -extprefix \$EXTPREFIX -skip qtandroidextras -skip qtcharts -skip qtwinextras -  
skip qtlocation -skip qtwebengine -skip qtwebview -no-opengl -D WAGO\_PFC

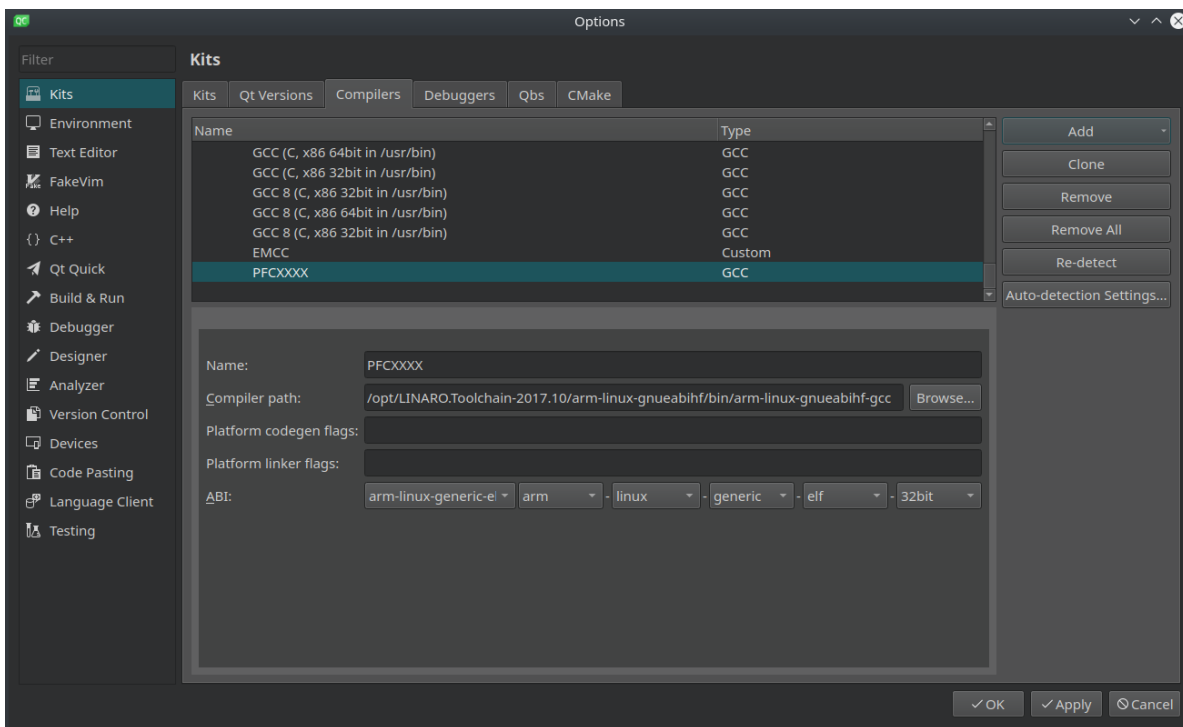
Note: For further options see “<https://doc.qt.io/archives/qttxtended4.4/buildsystem/over-configure-options-1.html>”

→ Follow the Prompts you will receive and check for a “success” message.

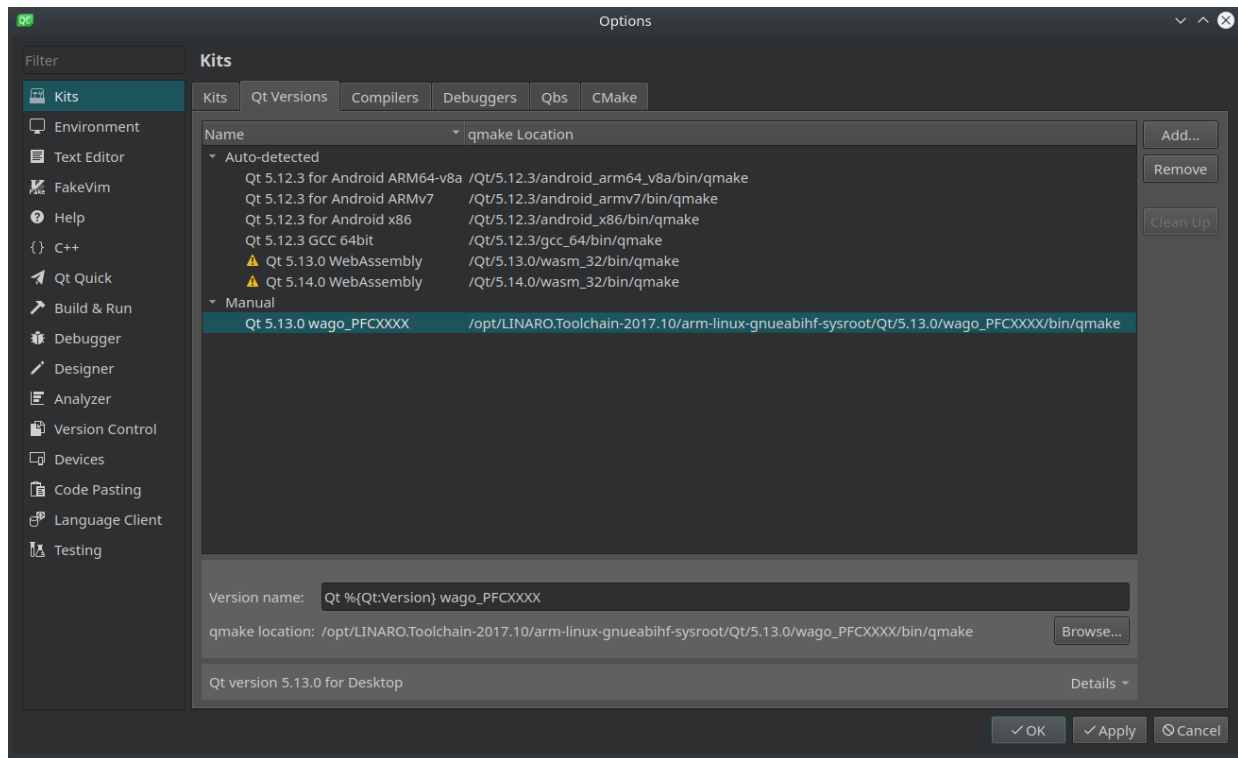
- -device: The Device for which we will configure the build. Contains information about the Architecture and so on.
- skip ... : Skips the Qt-Module so it will not be build, you can enter all the Qt-Modules you do not want to include into your build.
- No-opengl: The Wago-PFCXXXX does not support opengl and it will not work either. So we need to tell Qt not to use it.
- -D Add an explicit define to the build, so we can later determine our system in make/qmake
- After the configure, run make -jn, n is the amount of threads your PC has + 1
- Afterwards the Qt-Version will be build, this will take some time even on faster machines, expect it to take around 30-90 minutes.
- When the make is done, double check that no errors are listed in the last lines, if everything worked you are ready to install with: “make install”  
→ This will install the build into the Path provided by \$PREFIX
- That’s it we should now have a working Environment and can Cross Compile for the WagoPFC. If you need help configuring Qt to use the Build we just created, keep on reading.

## Setup QtCreator to use the Cross Compile Environment

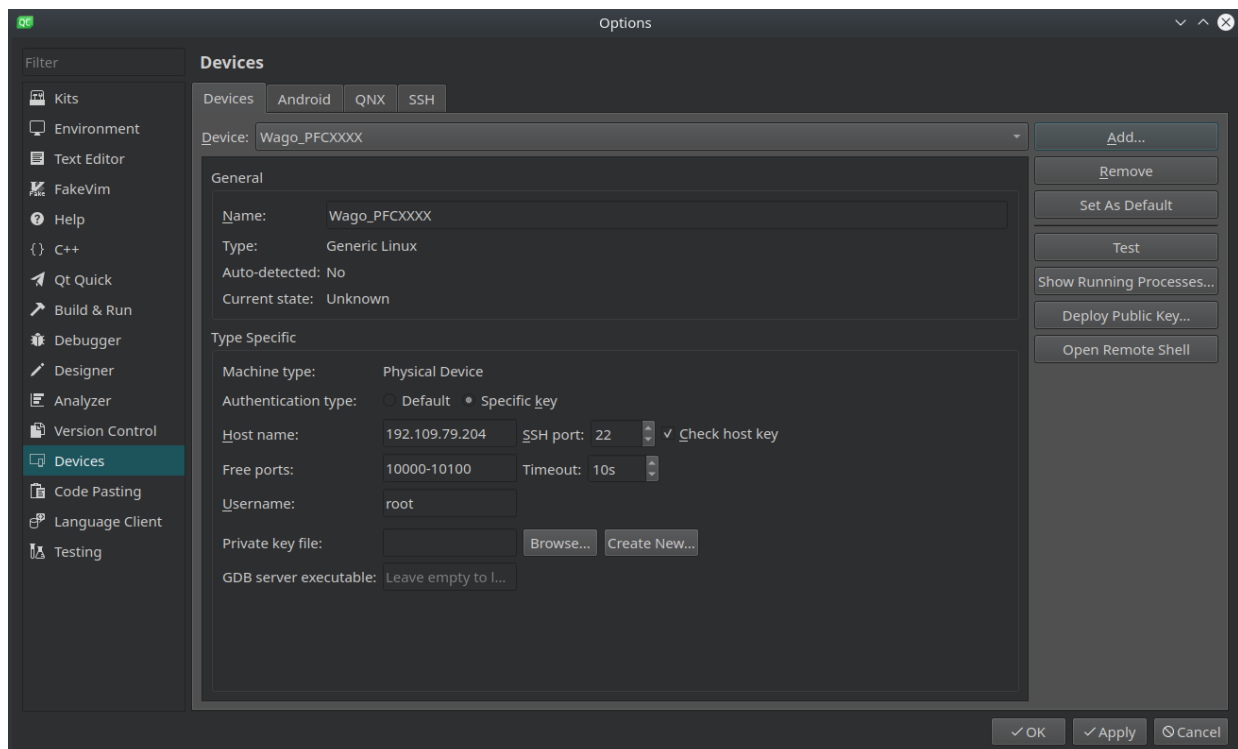
- If not already installed, download and install QtCreator → “<https://www.qt.io/download>”
- Open QtCreator. Navigate to Tools → Options → Kits
  - First setup the Compiler by clicking on the Tab Compilers:



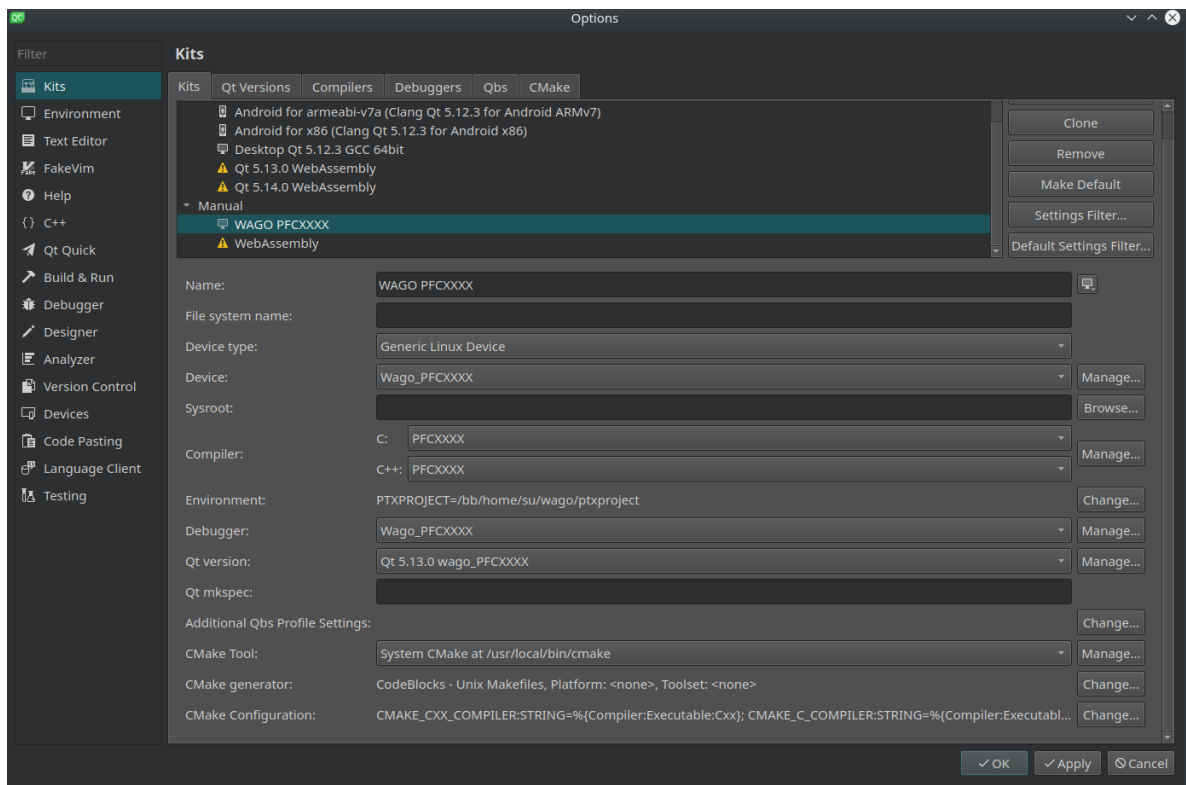
- Click on Add → GCC → C and enter a name and compiler path as seen in the Picture above, the Toolchain should be the same as used in the configure above.
- Repeat the step for the C++ Compiler: Add → GCC → C++, now use g++ instead of gcc
- Configure the Debugger the same way we configured the Compilers but with the ending gdb.



- Now we need to add our Qt-Version:  
→ If you added the Variable “EXTPREFIX” you will find the sysroot in that Path instead. Otherwise it will be in the Toolchain folder.



- We should now add the Device to our Options, so we can remotely debug and deploy. Navigate to the Tab “Devices”. Click on Add → Generic Linux Device → Start Wizard and follow the prompts by entering the IP of your Wago PFC Device with the “sd.hdimg” Image.



- Finally we can add the Kit to Qt, navigate back to the Tab Kits:  
Click on Add and enter a name for your Kit:
  - Choose from the Device List our Wago PFC we just added. Make sure Device type is set to Generic Linux Device.
  - Add our C / C++ Compiler in the compiler section
  - Add our debugger in the Debugger section
  - Add the Qt Version in the Qt Version section.
- We are finished now and can start programming for the Wago PFCXXXX.
- Note: For an example, on how to work with Qt (qmake) see the Demo Program included.