



Vehicle Solutions Group

Qt 5.12.2 User's Guide

Linux and Windows

VSUD2019-03

Revision A

Table of Contents

1.	Introduction.....	4
1.1	Purpose.....	4
1.2	Acronyms and Definitions	5
1.3	References.....	5
1.4	Revision History	5
2.	Requirements.....	6
2.1	Hardware	6
2.1.1	Supported Grayhill Display Hardware.....	6
2.1.2	Recommended Equipment.....	7
2.2	Software	7
2.2.1	Qt Installer (https://www.qt.io/download).....	7
2.2.2	Grayhill Qt Support Files (http://www.grayhill.com/qt43d)	7
2.2.3	Windows Utilities	7
3.	Installation.....	8
3.1	Install the Development Kit.....	8
3.2	Download and Install Qt Creator	9
3.3	Windows Utilities	22
3.3.1	Download and Install PuTTY	22
3.4	Configuring 3Dxx Display's IP Address	23
3.4.1	Linux	23
3.4.2	Windows	25
3.4.3	Verification of Established Session	27
3.4.4	Configure IP address.....	28
3.4.4.1	Linux	28
3.4.4.2	Windows	29
3.5	Download and Install Support Files.....	32
3.5.1	Linux.....	32
3.5.2	Windows	33
3.6	Build and Run 3Dxx Embedded Application.....	37
3.6.1	Launch Qt Creator.....	37
3.6.1.1	Linux	37
3.6.1.2	Windows	38
3.6.2	Open project.....	39
3.6.2.1	Linux	39

3.6.2.2	Windows	39
3.6.3	Build Project	41
Appendix A:	Configuring a Manual Qt Kit for Grayhill Displays	43
Appendix B:	Configuring a 3Dxx Project	60
Appendix C:	Debugging	72
Appendix D:	Setting up a 3Dxx Qt Program to Run at Boot Up	76
Appendix E:	Interfacing 3Dxx Hardware from QT Software	78
Appendix F:	Setting 3Dxx Flash File System R/W Mode	100
Appendix G:	Building Qt Library Source (optional)	101
Appendix H:	Dynamic IP Address.....	102
Appendix I:	Static IP Address	103

1. Introduction

1.1 Purpose

This document describes:

- Setup and usage of the Qt-based development environment for Grayhill 3Dxx display products
- Code development for a 3Dxx Display product in the Qt IDE
- Accessing various 3Dxx hardware features via this code
- Loading developed application code onto a 3Dxx Display product

The Qt cross-platform development environment runs under both Linux and Windows 10. The Linux platform is supported by a virtual machine using Oracle's VirtualBox (<https://www.virtualbox.org/wiki/VirtualBox>) software.

The virtual machine is Ubuntu 16.04 using gnome flashback for the desktop; additionally PuTTY (telnet client software - <http://www.putty.org>) is installed. 16.04 is a Long Term Support release, currently scheduled for end of life in April 2021. The VM also comes with Qt Creator and libraries installed.

For Virtual Machine installation, please reference "Virtual Machine Installation Using VirtualBox", which is available on the Grayhill web site (www.grayhill.com/qt43d)

This document is intended for use by software developers familiar programming in C/C++ using the Qt framework. Experience developing applications for Linux platforms is a definite plus.

Screen shots try to be as accurate as possible and are provided as reference.

N.B. Screen images are mixed between the Windows version of Qt Creator and Linux, but the steps are the same.

Note: Qt is licensed under the terms of LGPL and GPL. These are open-source licensing agreements. Please reference <https://www1.qt.io/qt-licensing-terms/> for a detailed explanation. Additional information is also located at <https://www.gnu.org/licenses/licenses.html>.

1.2 Acronyms and Definitions

3Dxx	Reference to any of the Grayhill 3D series displays (3D50, 3D70, 3D2104)
CAN	Controller Area Network
GB	Giga Byte
RAM	Random Access Memory
USB	Universal Serial Bus
VM	Virtual Machine

1.3 References

[1] VSTN2019-01	Linux - Upgrade existing Qt 5.9.3 Libraries to Qt 5.12.2
[2] VSTN2019-02	Windows 10 - Upgrade existing Qt 5.9.3 Libraries to Qt 5.12.2
[3] VSUD2019-06	Virtual Machine Installation Using VirtualBox

1.4 Revision History

Revision	Author	Date	Description
A	K. Struss	9/6/2019	Initial Release combining the previous independent Linux and Windows manuals

2. Requirements

2.1 Hardware

2.1.1 Supported Grayhill Display Hardware

The Qt-based development environment supports the following Grayhill 3Dxx Color Display Models:

- 3D50
- 3D70
- 3D2104

The table below summarizes the key features of each of these models. Note that the features of a specific product may vary depending on the purchased hardware configuration.

Model Number	3D50-x00	3D70-x00	3D2104-x00
Display Size (inches)	5	7	10.4
Pixel Count (w x h)	800 x 480	800 x 480	1024 x 768
Touch Screen Input	Yes	Yes	Yes
Real Time Clock	Yes	Yes	Yes
CAN Ports	2	2	3
Camera Inputs	2	3	4
USB ports	1 (maintenance only)	1 (maintenance only)	1 (maintenance only)
RS232	1 (maintenance only)	1 (maintenance only)	1 (maintenance only)
Built-in Ethernet	0	1	1
Digital Input (dedicated)	1	4	0
Digital Output (dedicated)	1	4	0
Digital Input / Output	3	0	4
Analog Input	0	2	0
Audio Output	No	1 channel	No
Buzzer	No	Yes	Yes

2.1.2 Recommended Equipment

It is strongly recommended the associated development kit be used for development.

- 3D50DEV-100 3D50 Development Kit
- 3D70DEV-100 3D70 Development Kit
- 3D2104DEV-100 3D2104 Development Kit

PC Running Windows 10 with the following minimum configuration:

- 4 GB RAM (minimum)
- 10/40 (VM) GB available hard drive space (minimum)
- Ethernet (RJ45) port (or USB adapter)¹
- RS232 Port (or USB to serial adapter)
- Internet Access

2.2 Software

The following software packages are available on-line

2.2.1 Qt Installer (<https://www.qt.io/download>)

2.2.2 Grayhill Qt Support Files (<http://www.grayhill.com/qt43d>)

- QtGhInstall5122Linux
- QtGhInstall5122Win10.exe
- Virtual Machine Appliance (optional)

2.2.3 Windows Utilities

- Notepad++ (<https://notepad-plus-plus.org/>)
- PuTTY (<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>)

¹ An Ethernet port connected to a DHCP server connected to the 3Dxx display. This port should be on the same network as the development PC.

3. Installation

This is a brief overview of the installation steps for the Qt-based development environment for Grayhill 3Dxx displays.

- Connect the 3Dxx Development Kit hardware to the PC
- Qt Creator for Windows is downloaded and installed on the development PC
- Optional third party utilities are downloaded, installed, and configured (Windows)
- The serial and Ethernet links to the target 3Dxx display hardware are established.
- Grayhill support files are downloaded and installed
- A script is run to configure the target 3Dxx display board
- Instructions on how to open and run a Qt demonstration project on the 3Dxx display target hardware or desktop environment. This demonstration project illustrates:
 - using touch screen “buttons”
 - using touch screen swipes
 - setting the 3Dxx backlight
 - operating the 3Dxx camera input
 - accessing and setting the real time clock

Instructions for using the desktop simulator are in **Error! Reference source not found.**

If a VM is going to be the development environment, it must be installed now. See [3] for complete instructions and configuration.

For VM installation, the procedure continues at 3.4 Configuring 3Dxx Display’s IP Address.

3.1 Install the Development Kit

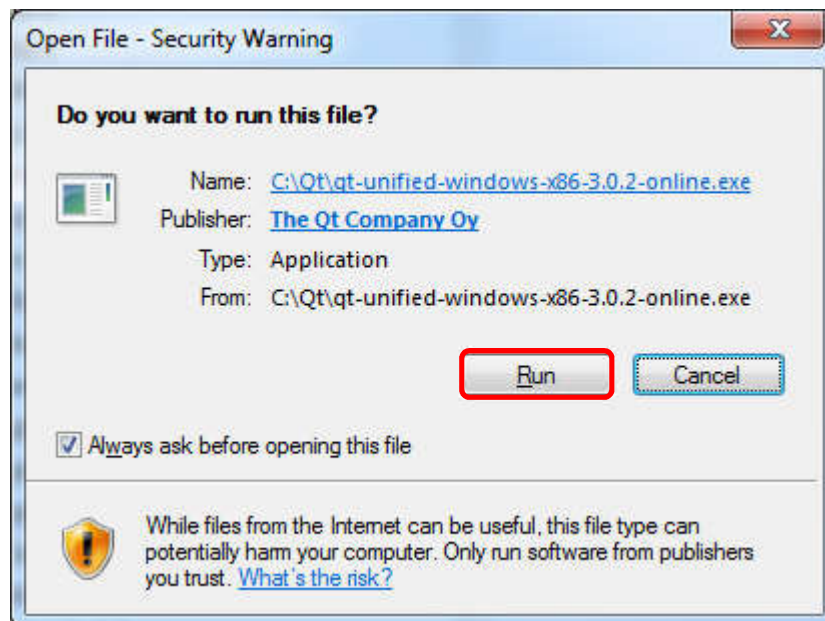
Connect the serial port and Ethernet port interfaces. The 3D50 display procedure is described in the document “3D50DEV Quick Start Guide.pdf” and the 3D70 in “3D70DEV Quick Start Guide.pdf”

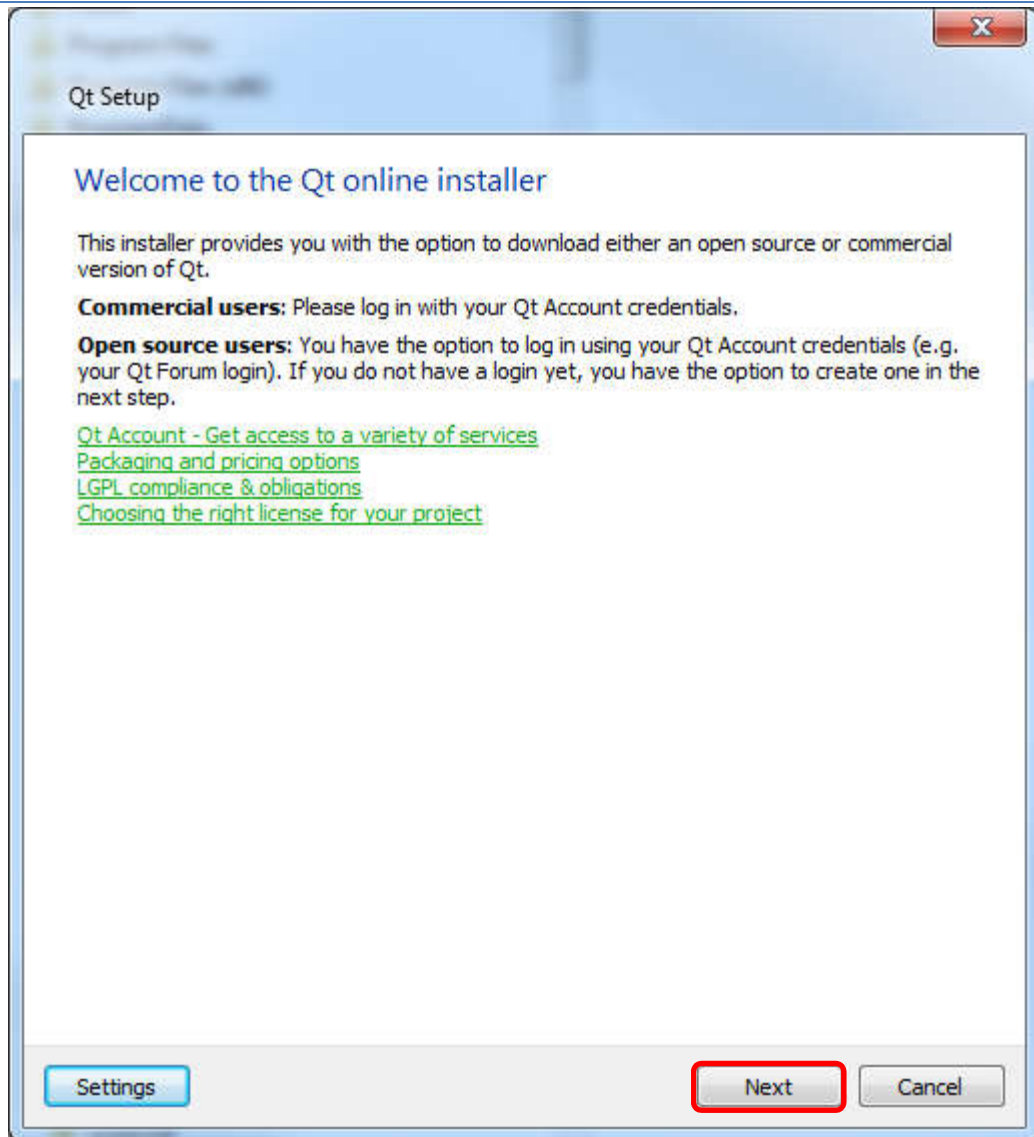
3.2 Download and Install Qt Creator

N.B. This section is **mandatory** for Windows users. The Linux VM comes with Qt Creator pre-installed.

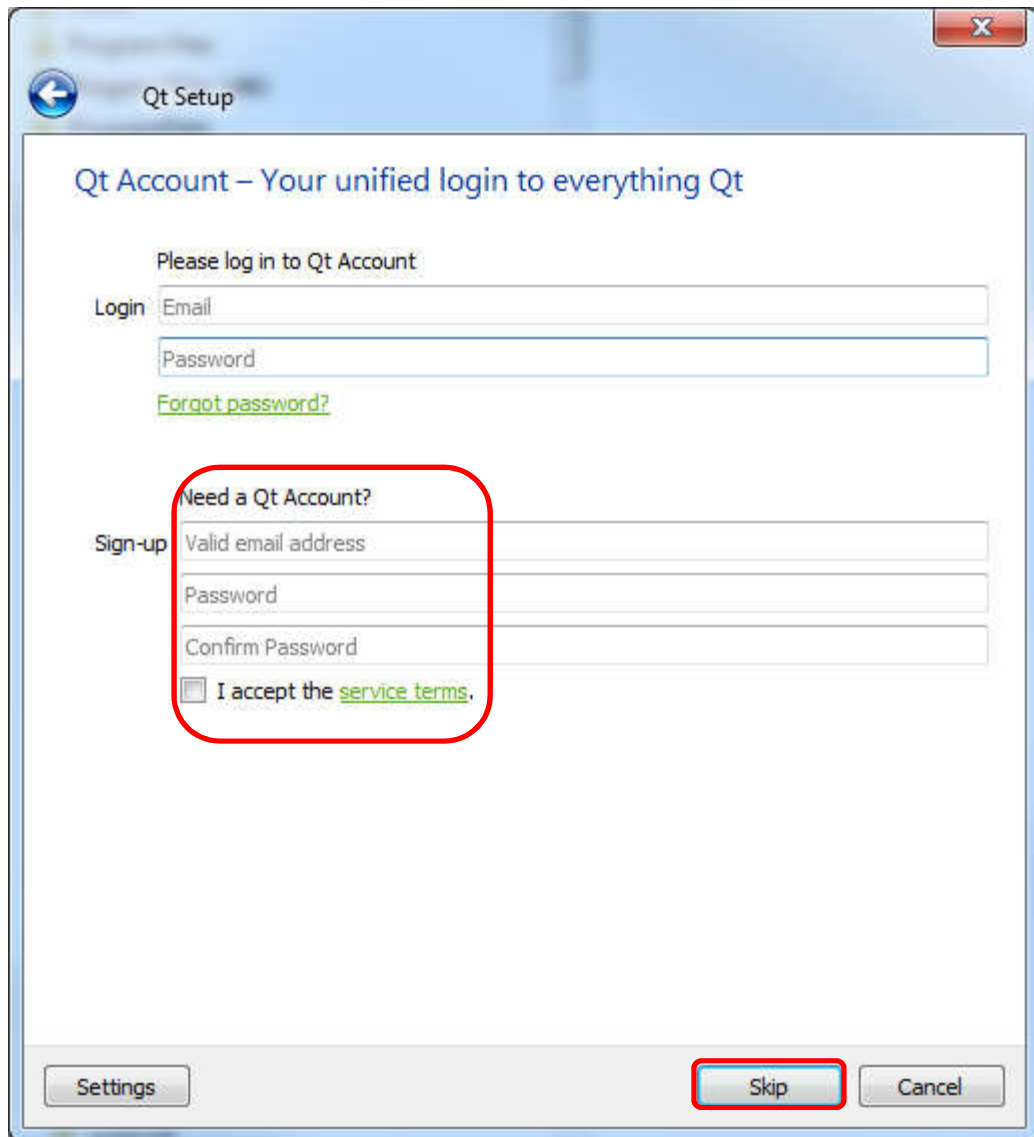
In this section, the Qt on-line installer will be downloaded and executed to download and install files from Qt. Once all the files are downloaded; Qt will be installed.

- Using your favorite web browser:
 - <https://www.qt.io/download>
 - scroll down and click “Go open Source”
 - scroll down and click “Download”
- After the file downloads, open the downloads folder and double click on the file to execute the installer. If a “Security Warning” similar to below appears; click “Run”

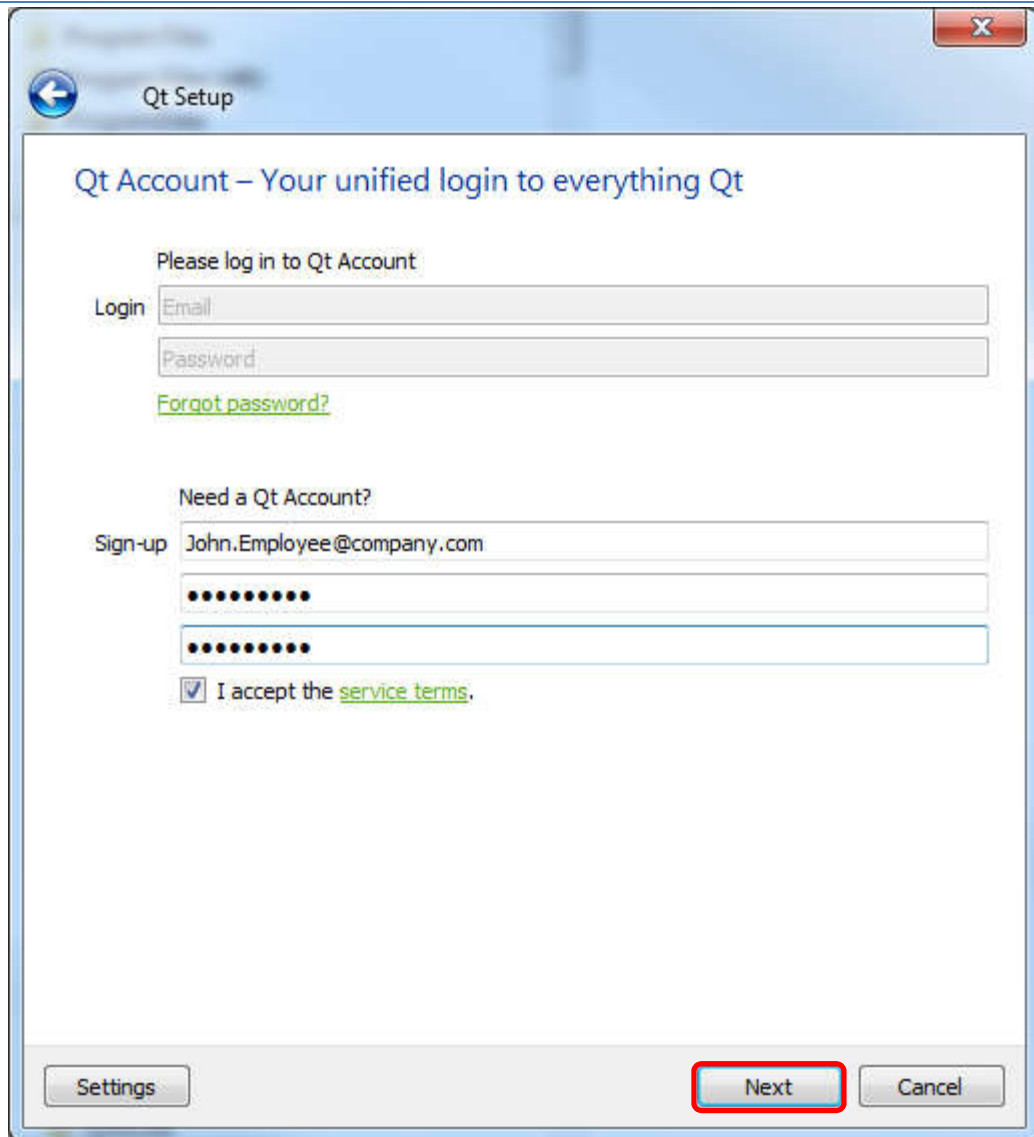




- Click Next

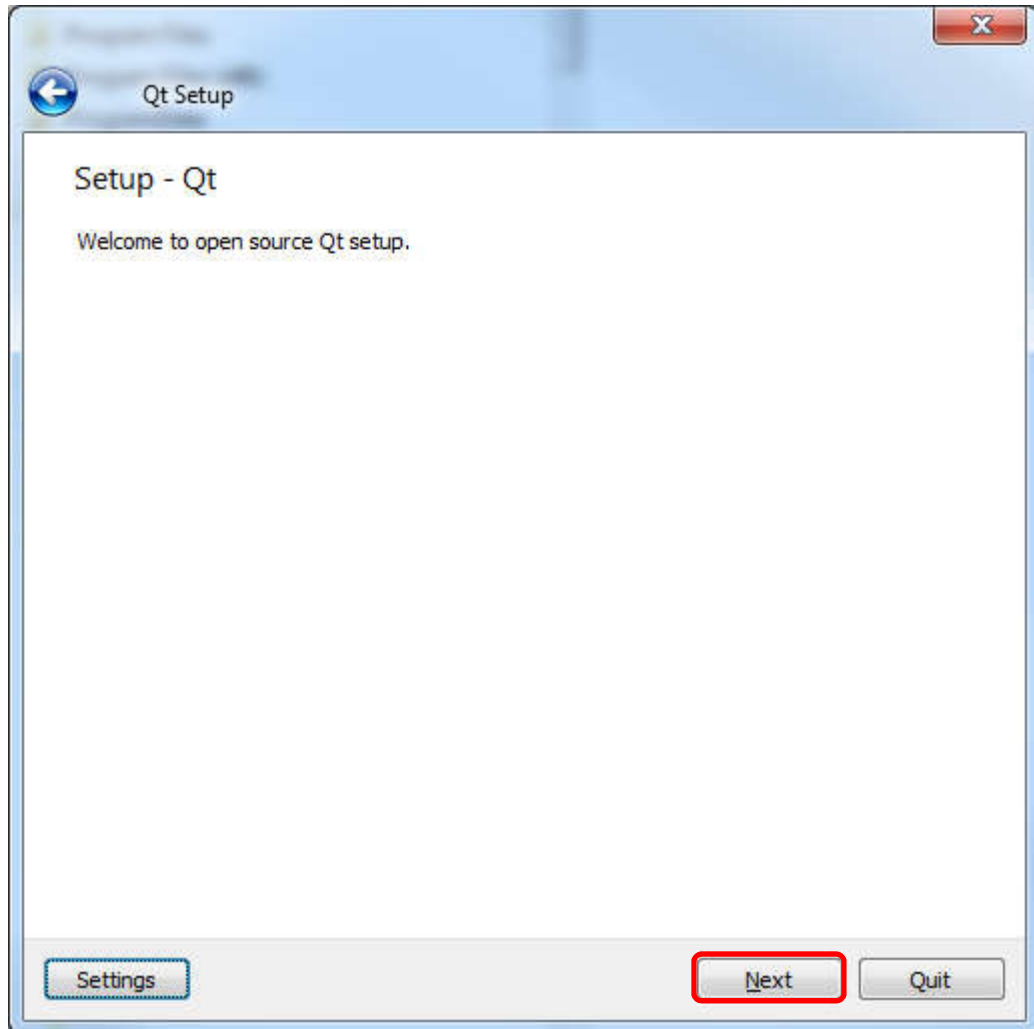


- Create an account, if desired – otherwise click “Skip”

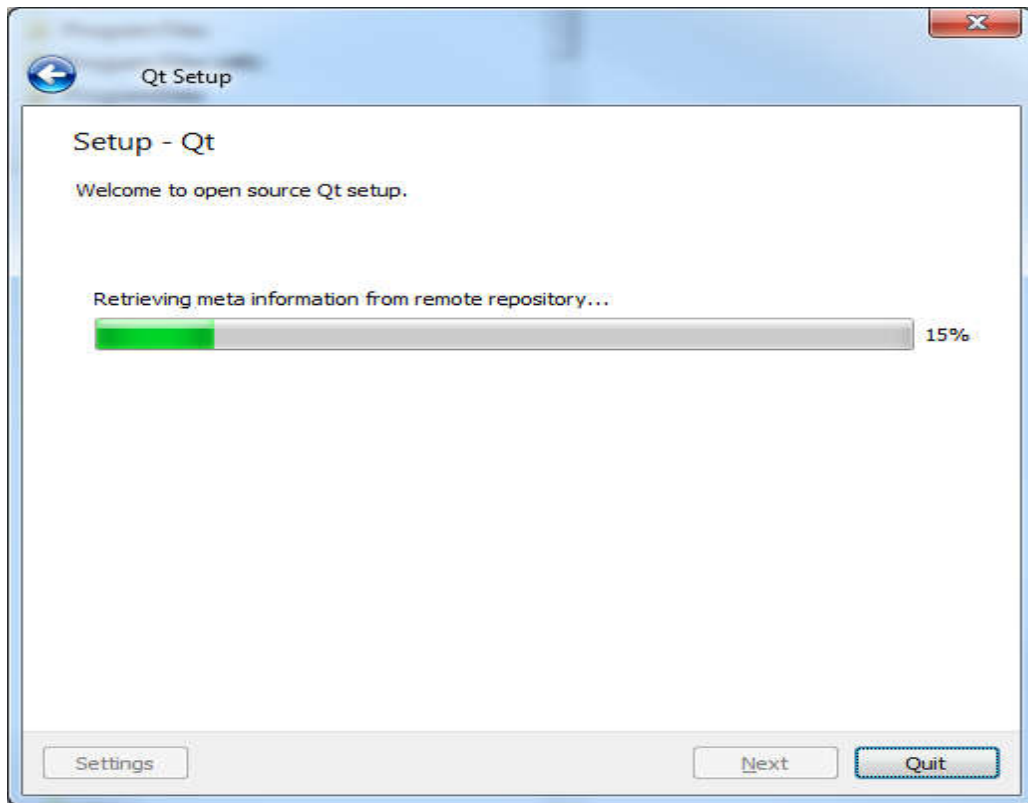
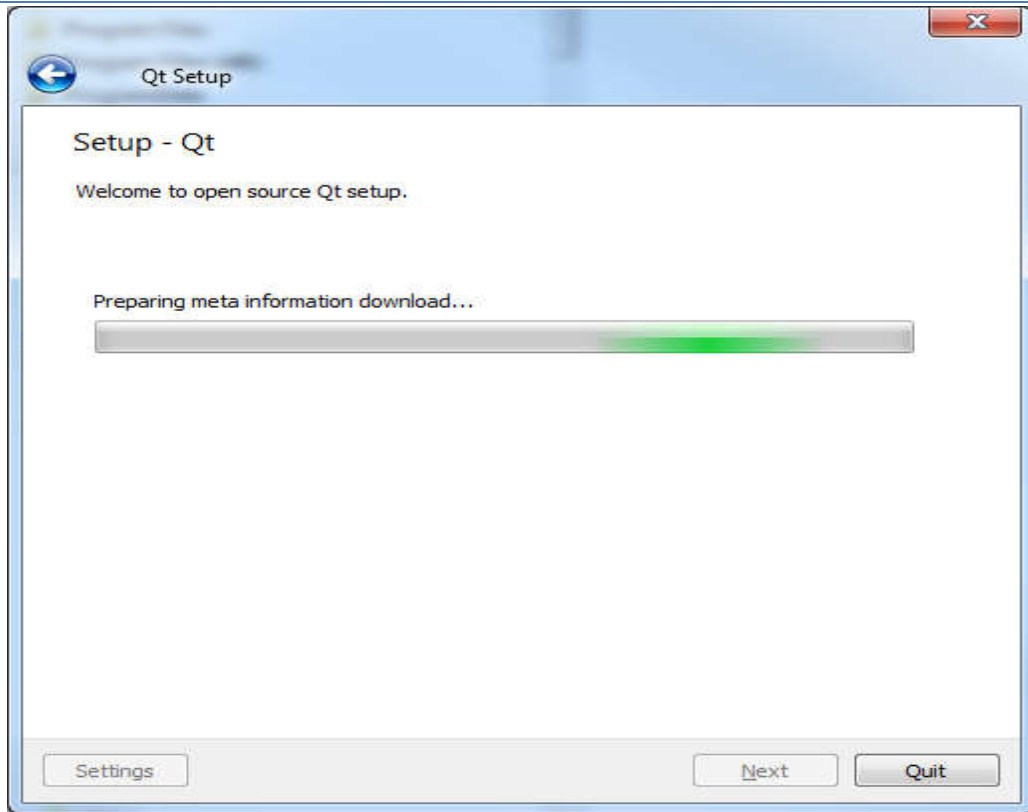


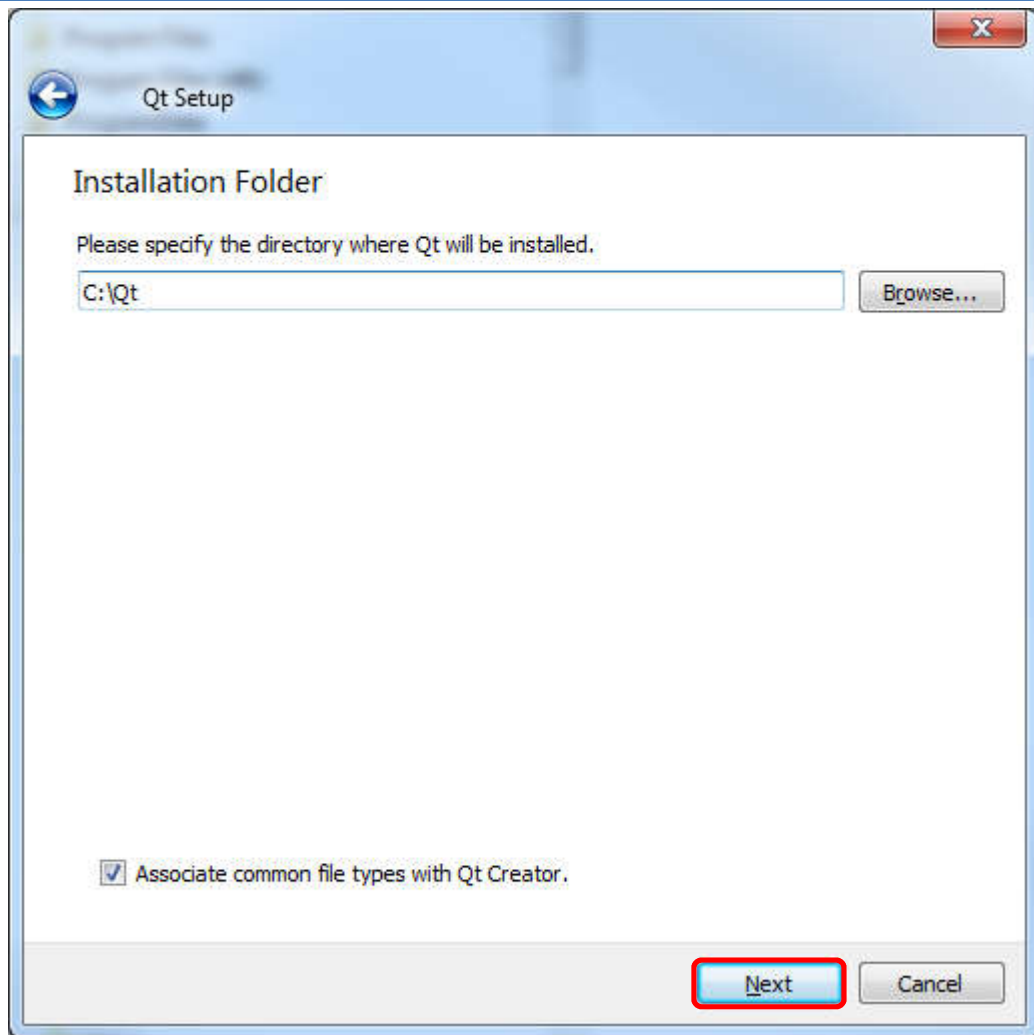
- If an account was created click “Next” – otherwise this screen will not appear

-
- Whether “Skip” or an account was created; installation continues here



- Click “Next”

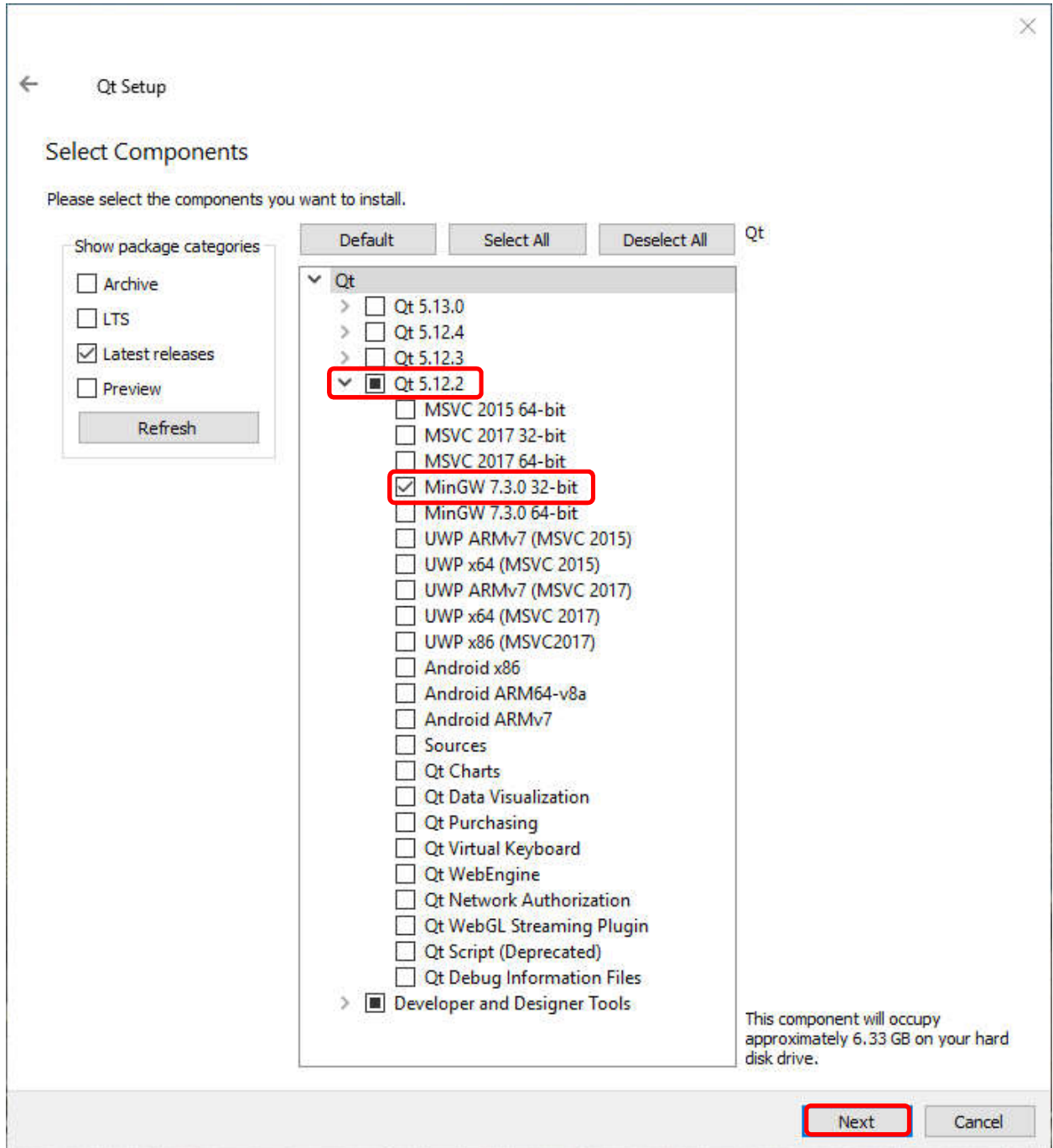




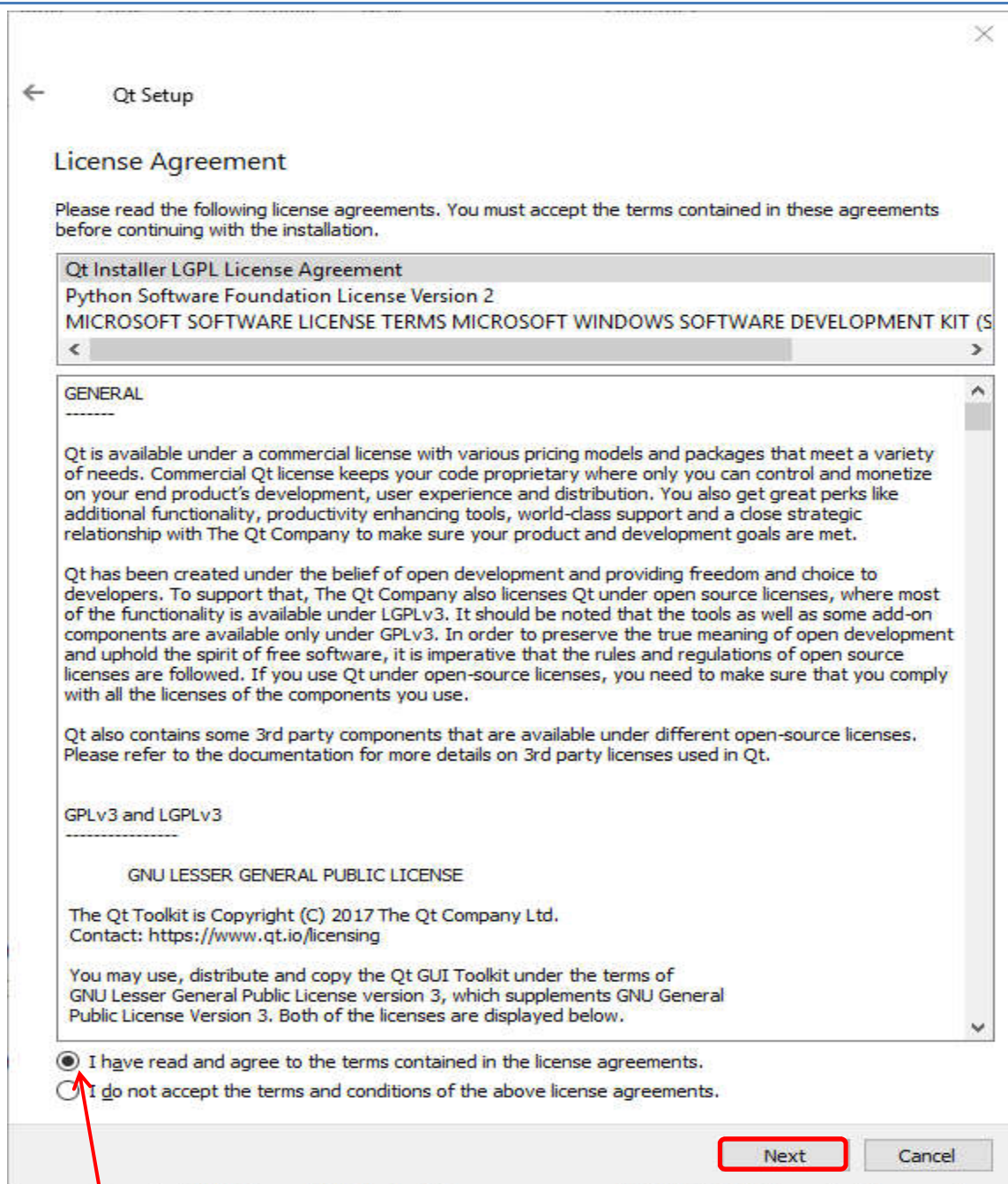
- Click "Next"

N.B. Due to the nature of Qt and the way it stores configuration information; Qt must be installed in C:\Qt.

- Expand Qt → Qt 5.12.2
- Select “MinGW 7.3.0 32bit”

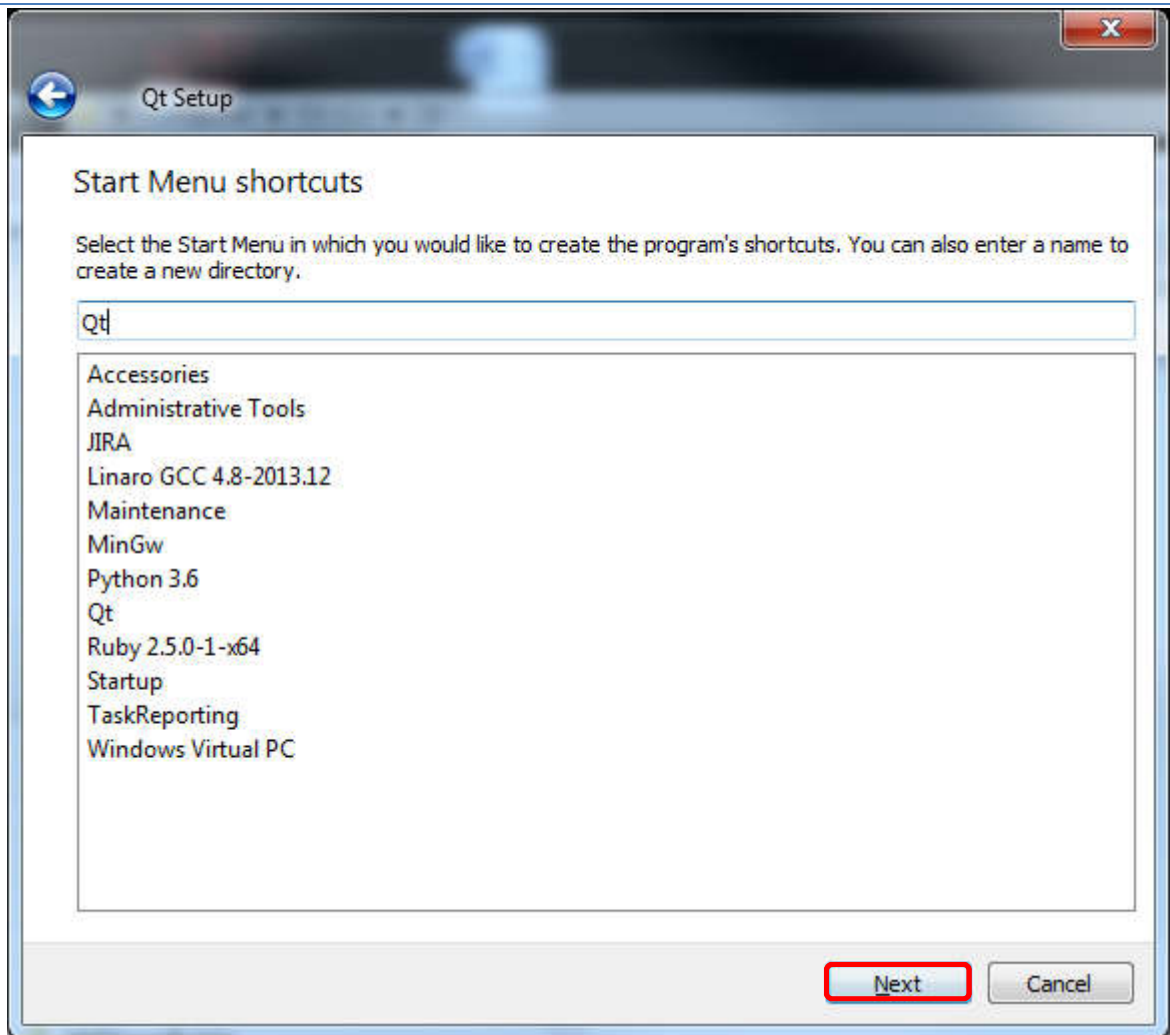


- Click “Next”

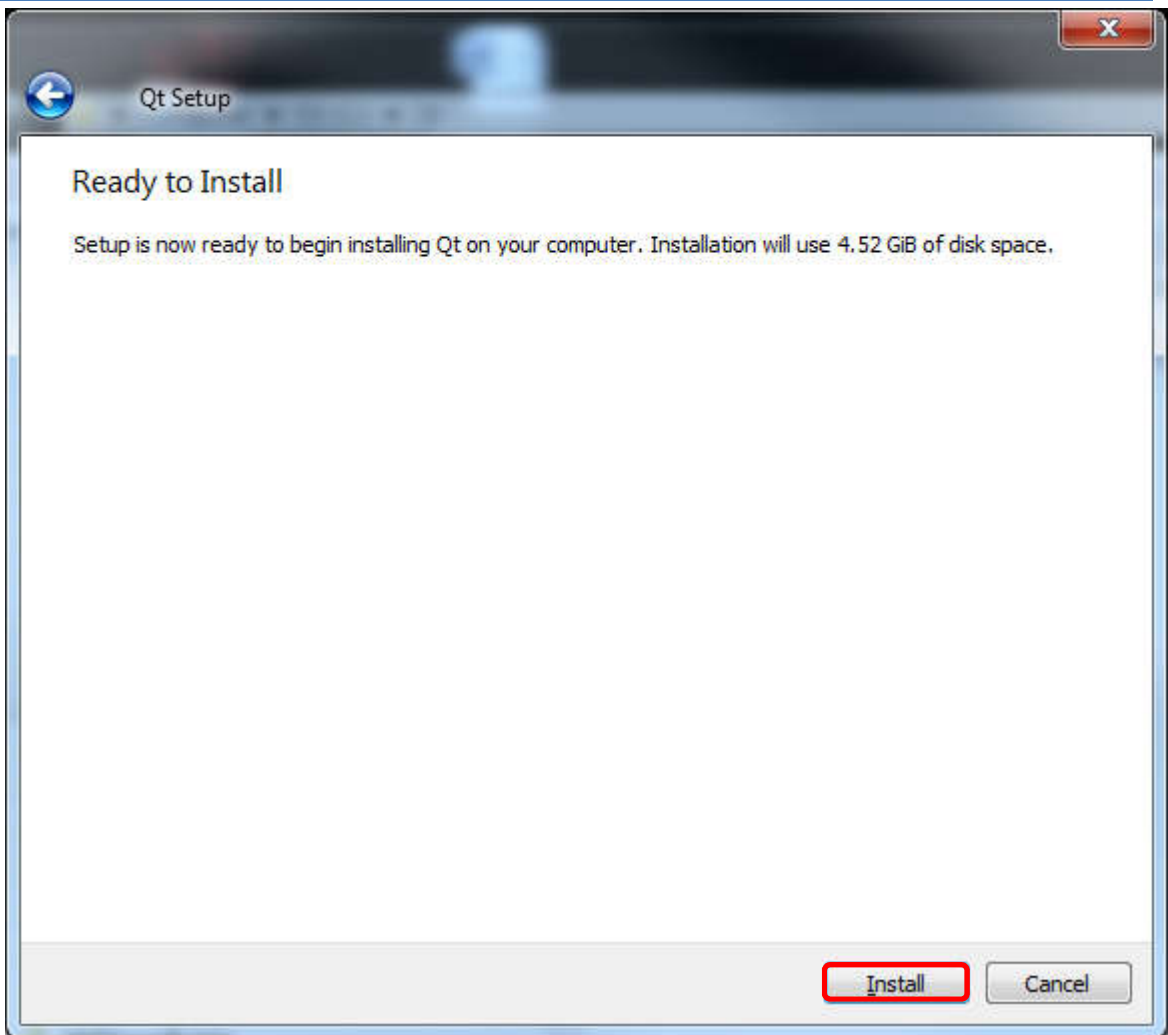


- If accepting of the license agreement select “I have read...”
- Click “Next”

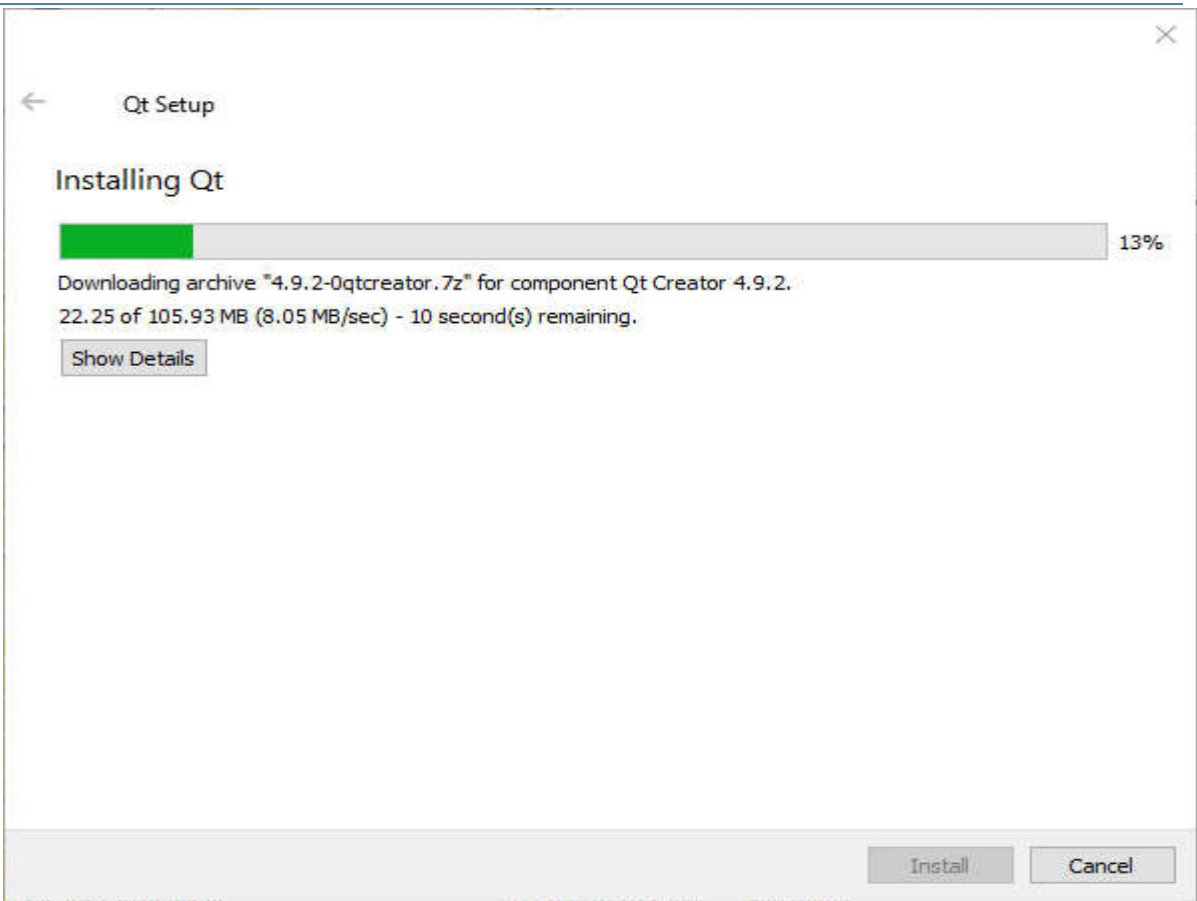
Note: Qt is licensed under the terms of LGPL and GPL; these are open-source licensing agreements. Please reference <https://www1.qt.io/qt/licensing-terms/> for a detailed explanation. Additional information is also located at <https://www.gnu.org/licenses/licenses.html>.

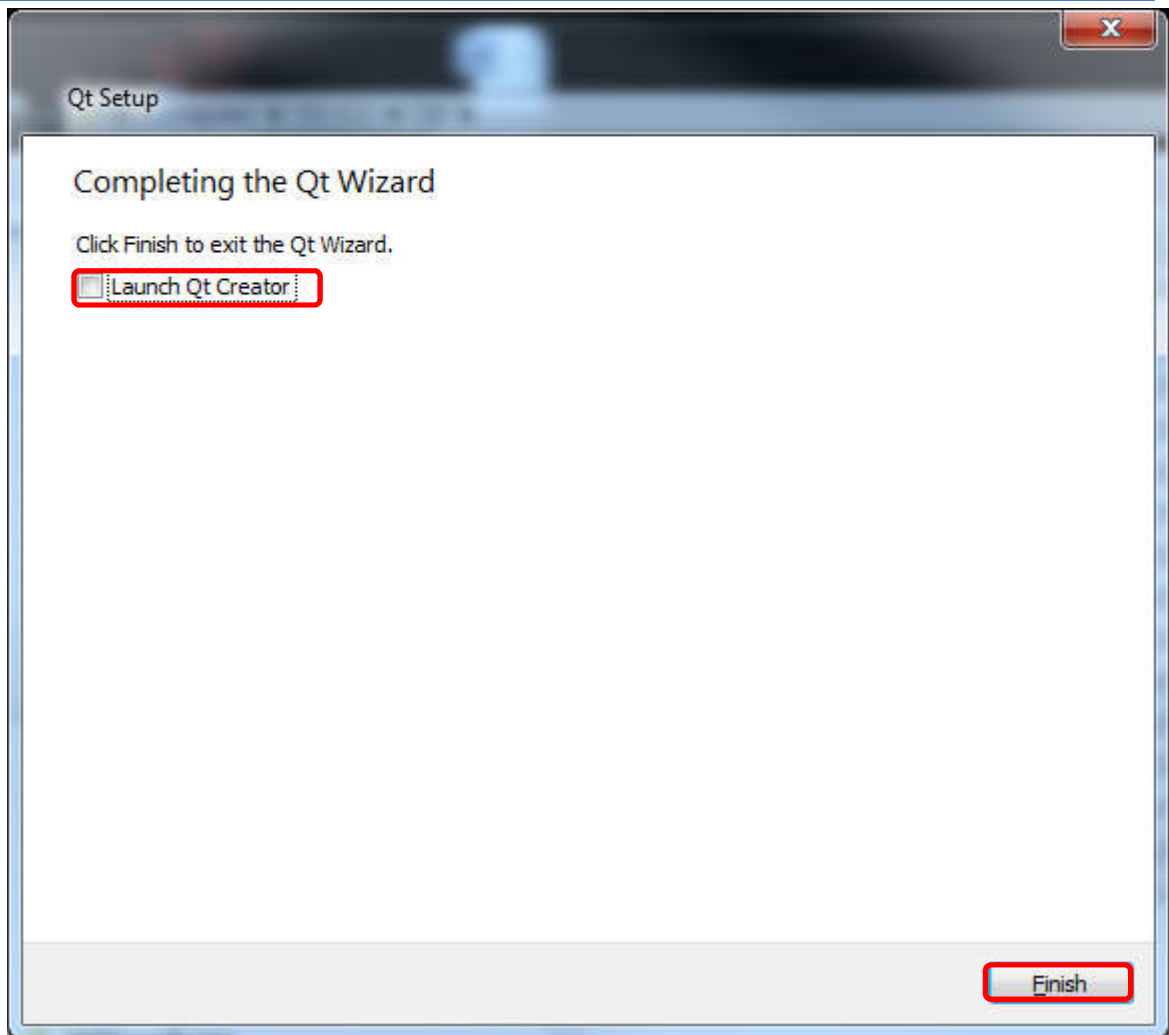


- Click "Next"



- Click "Install"





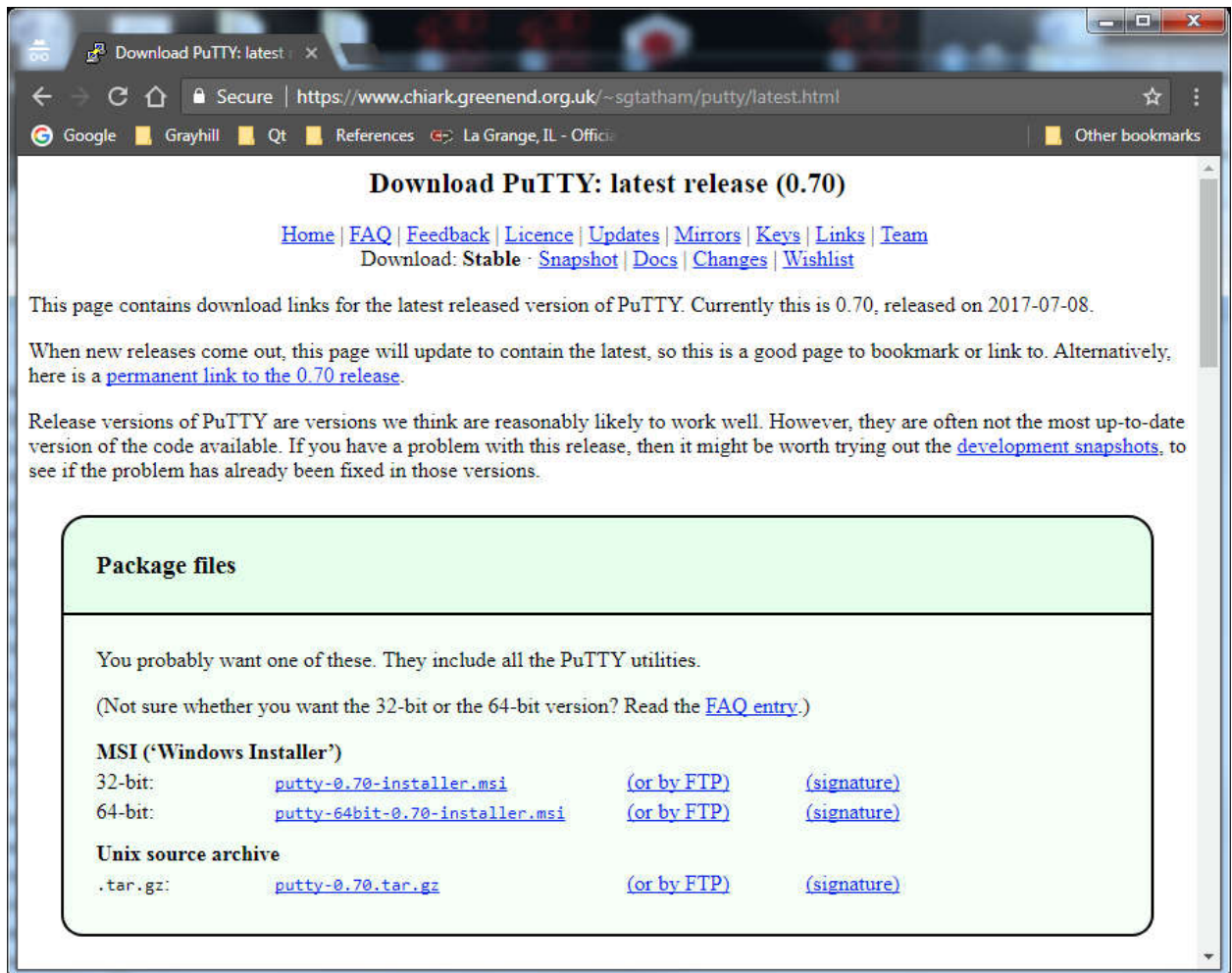
- Unselect "Launch Qt Creator"
- **N.B.** Qt Creator does **not** know the IP address of the target board at this time; the target board's IP address will be discovered and configured later. Any time the IP address of the display changes, Qt Creator must be re-launched if using the `/etc/hosts` file for IP address resolution.
- Click "Finish"

3.3 Windows Utilities

3.3.1 Download and Install PuTTY

The examples shown in this document reflect the use of PuTTY. Feel free to substitute a different client.

- Download Putty



- Open the downloads folder and double click to execute the PuTTY installer
- Follow the installation instructions – connection configuration is described later on in the document

3.4 Configuring 3Dxx Display's IP Address

In order to complete the setup of the Qt development environment for the 3Dxx Display hardware; the IP address assigned to the 3Dxx Display must be determined.

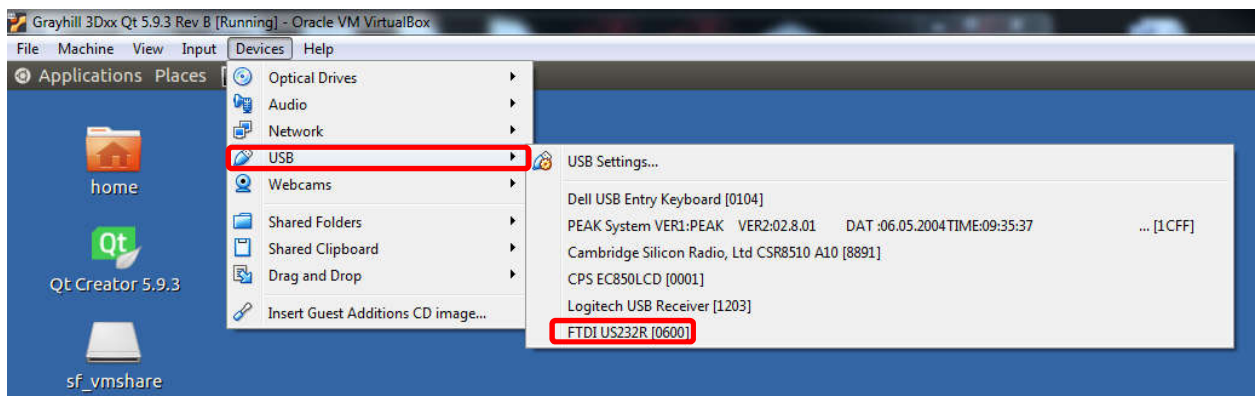
In order to perform these tasks, it is necessary to connect the 3Dxx Display to the same network as the development PC.

- Connect the 3Dxx display serial port to a serial port on the development PC
- Determine the serial port device name to use for PuTTY (serial communication between the physical PC and the target)

3.4.1 Linux

This depends on how the 3Dxx Display serial port is physically connected to the development PC. If using a built-in serial port on the development PC, the serial port device name is “/dev/ttyS0”. If using a USB to serial port adapter, the serial port device name is “/dev/ttyUSB0”.

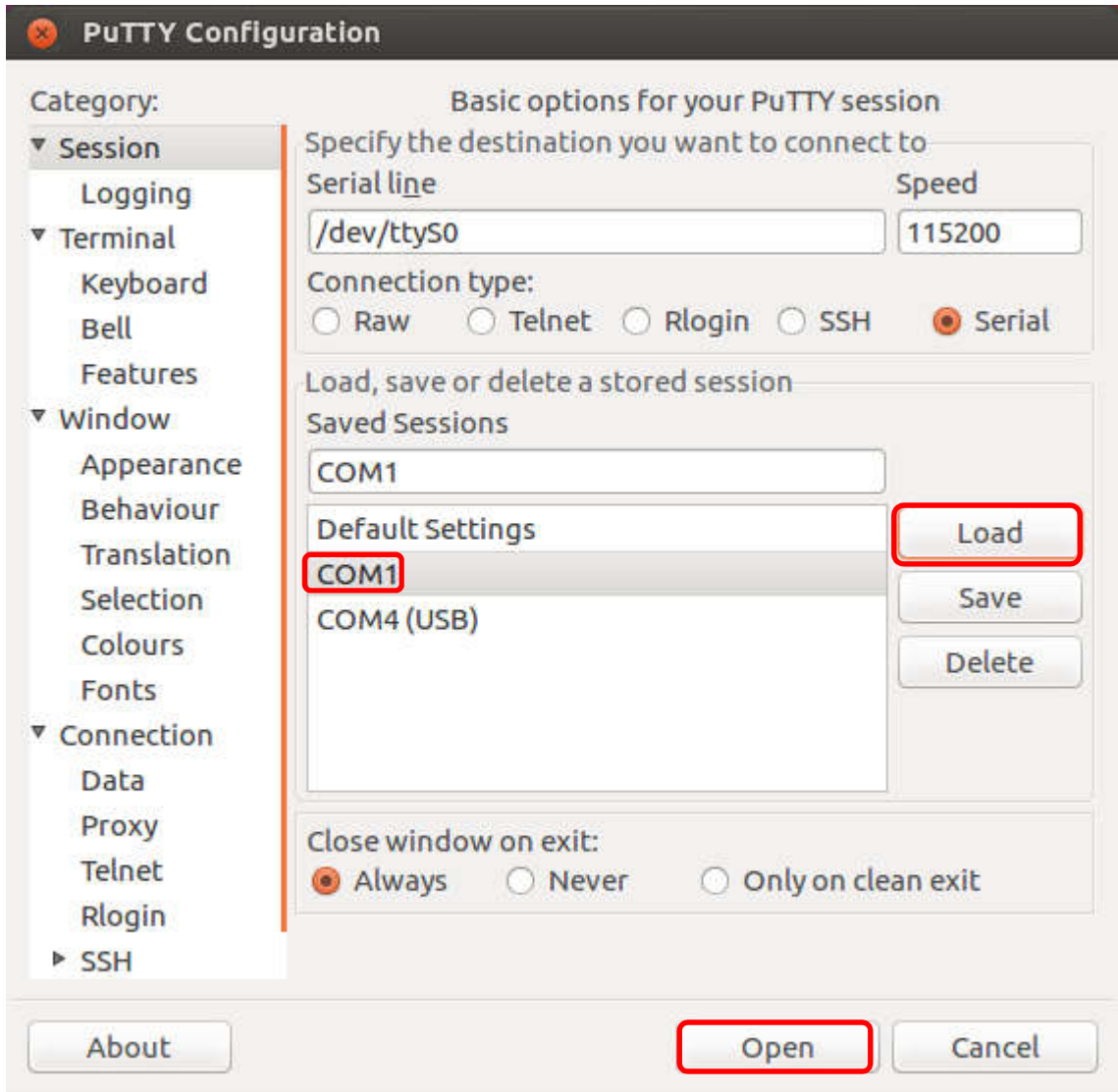
Only if using the USB to serial port adapter: it must be activated at this time by clicking on the “Devices” menu option at the top of the VirtualBox screen. Select “USB” and click on the USB to serial port adapter device name in order to select it. A sample is shown here (the USB device name may be different than shown):



The Linux VM comes with PuTTY pre-installed, so minimal configuration is required. The “Serial line” value may need to be updated based on the above connector.

- Launch PuTTY, the “PuTTY Configuration” screen appears

- Select COM1
- Click “Load”

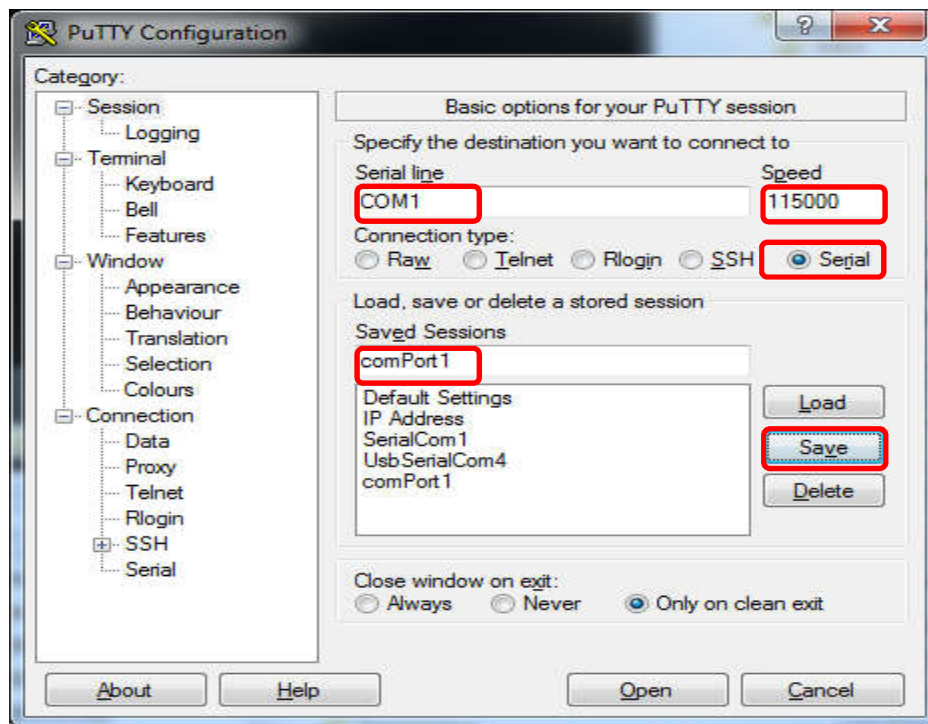


- Click “Open”

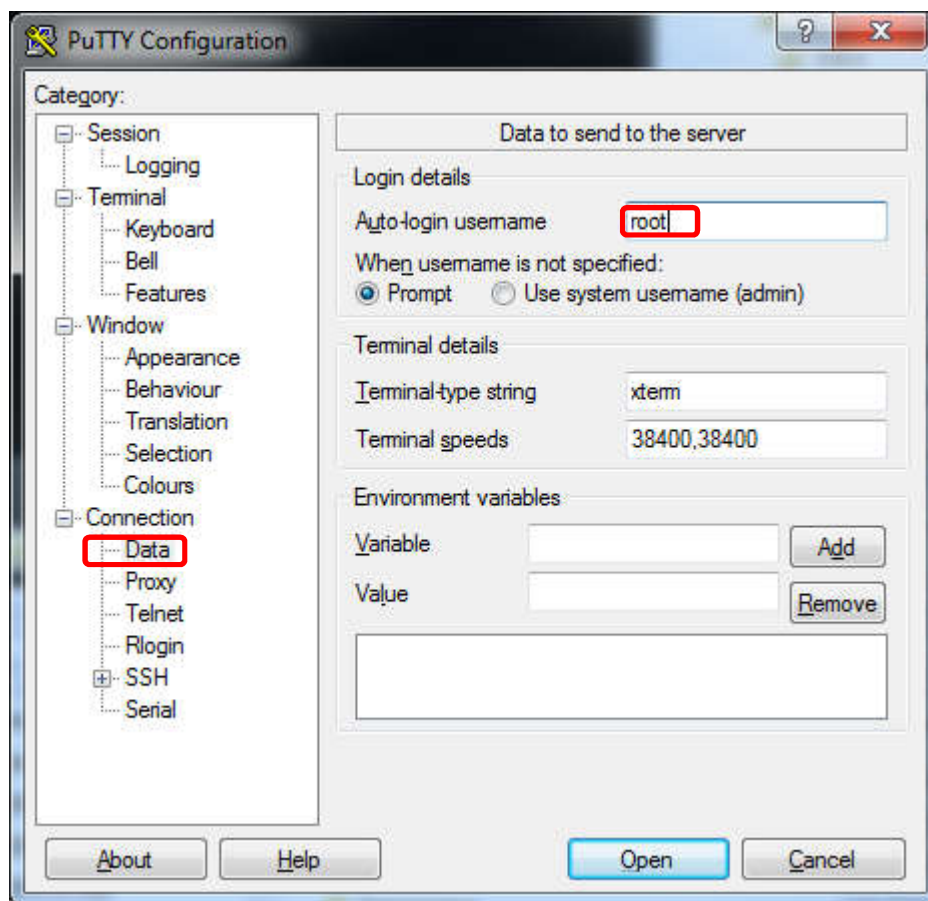
3.4.2 Windows

- Launch PuTTY, the “PuTTY Configuration” screen appears configure as follows:
 - Select the “Serial” button
 - Set “Serial line” to appropriate COM Port
 - Change the “Speed” to 115000
 - Enter a name in “Saved Sessions” (e.g. comPort1)
 - Click “Save”

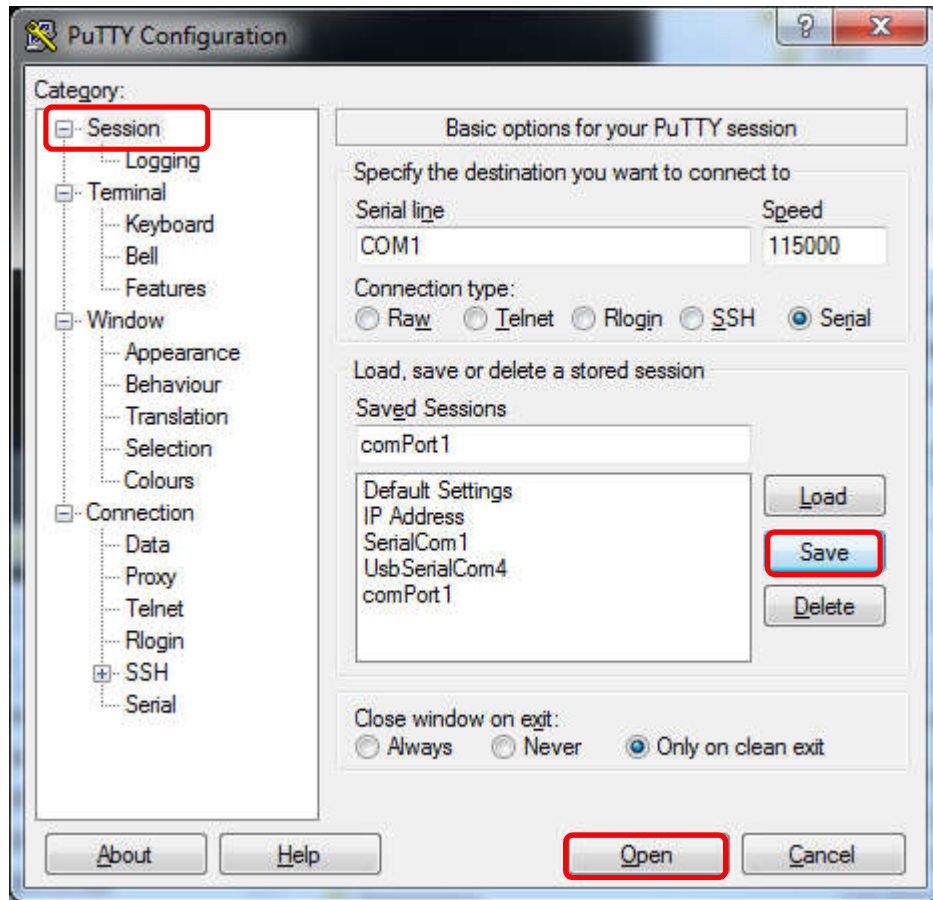
N.B. If “Open” is clicked any unsaved configuration modifications are **lost!**



- Click on “Data”
- Set “Auto-login username” to “root”



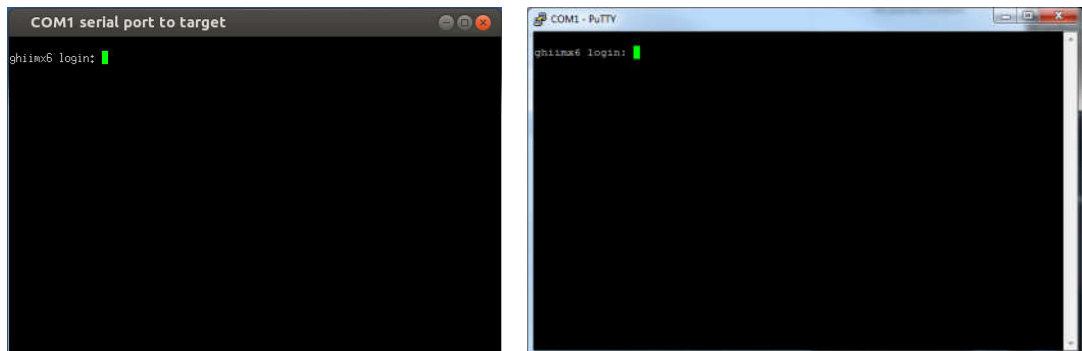
- Click back on “Session”, then click “Save” again



- Lastly, click “Open” to establish a connection

3.4.3 Verification of Established Session

Make sure that the 3Dxx display is powered up and press the “Enter” key (Linux on left, Windows on right)



-
- A “ghiimx6 login:” prompt should appear. If the 3Dxx display was just powered up; startup messages may appear as well, but when they are done, pressing the “Enter” key should produce a “ghiimx6 login:” prompt as shown.
 - At the “ghiimx6 login:” prompt enter “root” (no password is required)
 - Depending on the IP address type, refer to the appropriate appendix:
 - Dynamic Appendix H: Dynamic IP Address
 - Static Appendix I: Static IP Address

3.4.4 Configure IP address

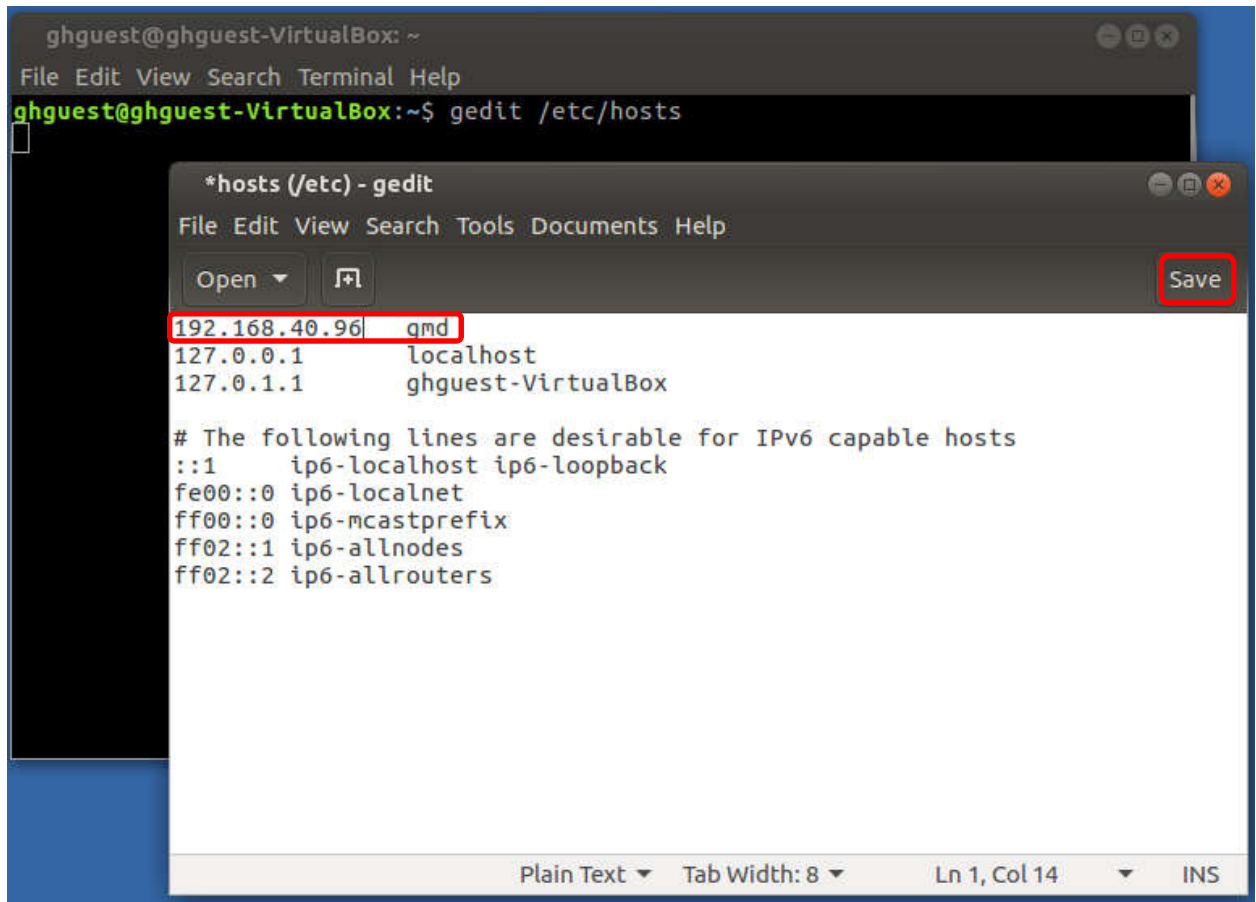
Create an alias for the display’s IP address to be referenced by the host computer.

N.B. If the IP address of the display changes; *hosts* must be updated and Qt Creator re-launched.

Linux

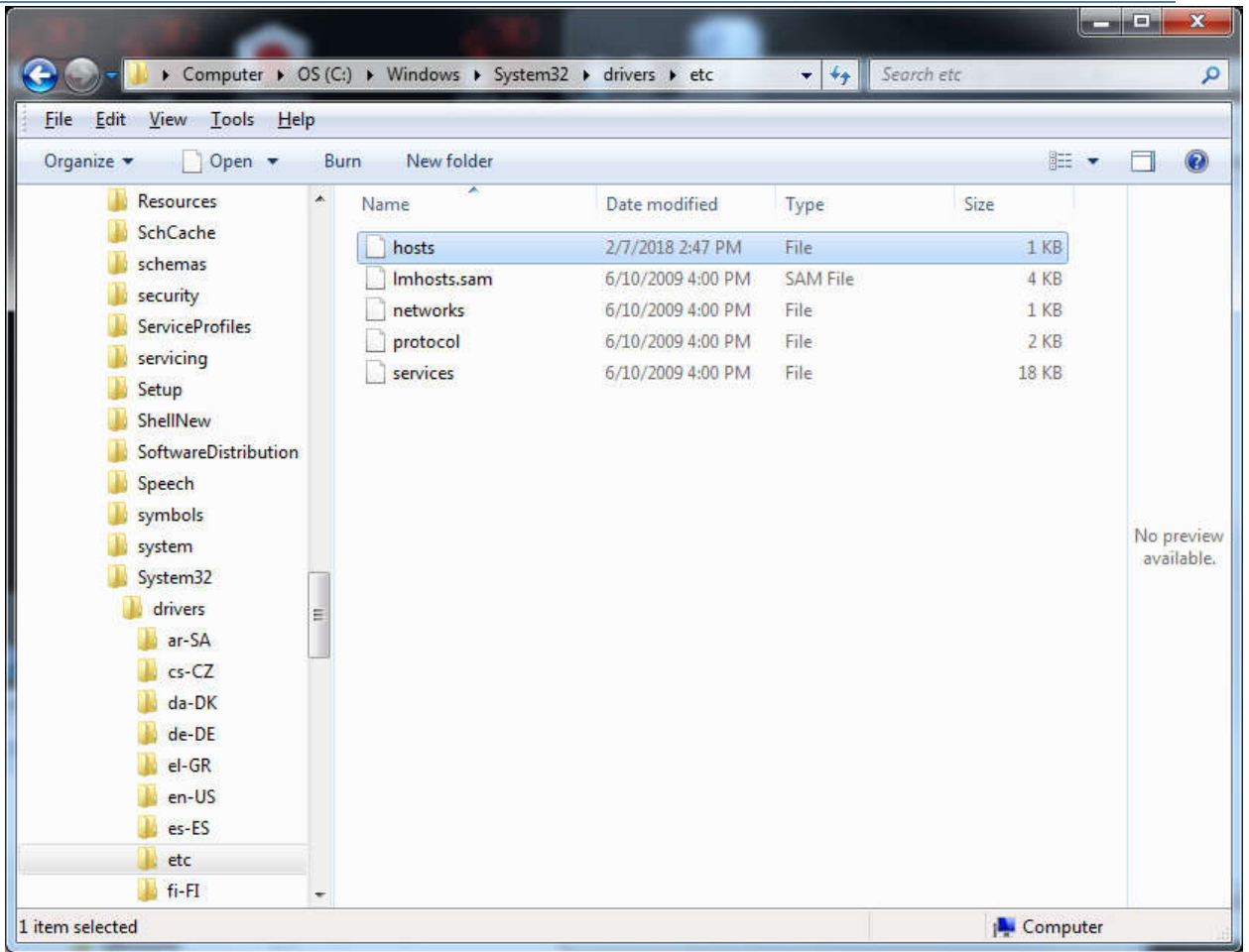
- Launch a Terminal Command Window
- In the terminal window type the following command:
 - `gedit2 /etc/hosts`
- Update the IP address associated with “gmd”
- Click “Save”; then close the editing session

² vi is also available as a text editor if preferred

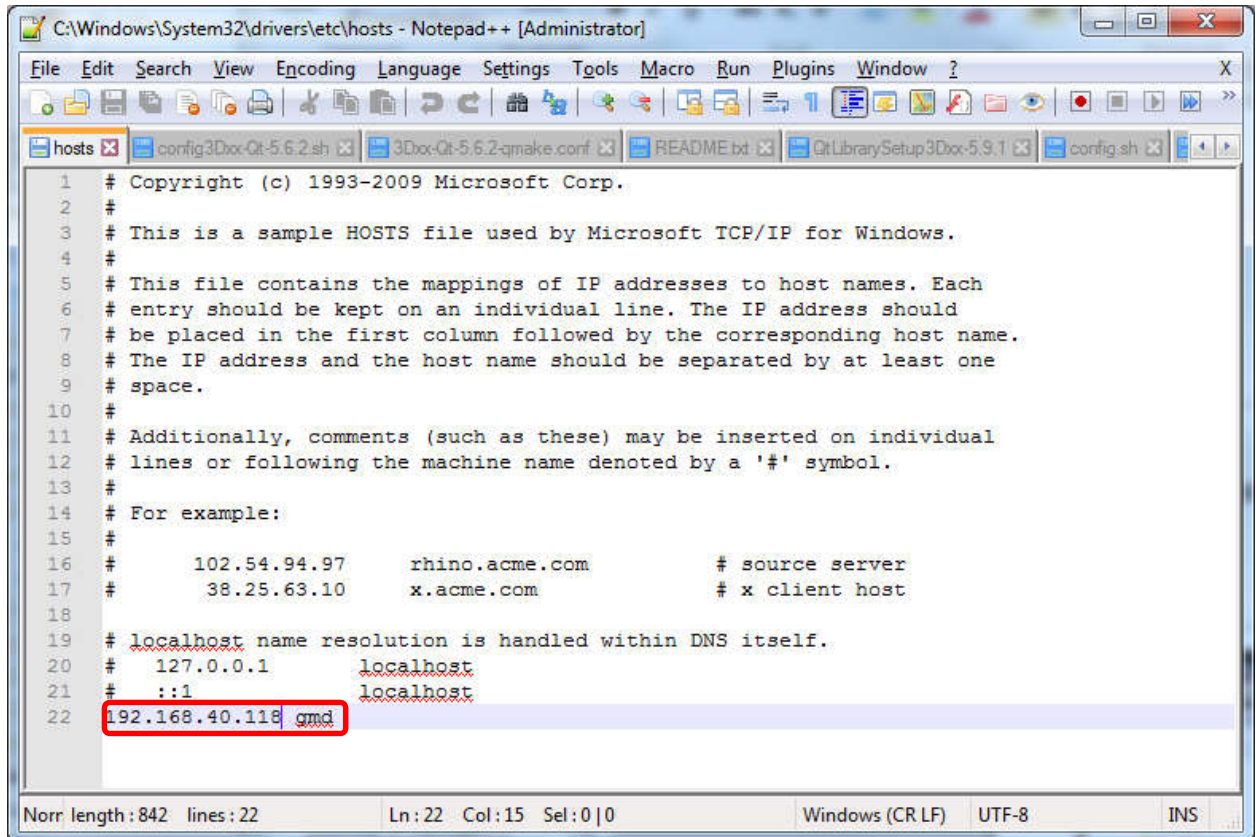


Windows

- Open Windows Explorer window (<Window>-e)
- Navigate to C: → Windows → System32 → drivers → etc and select “hosts”



- Right click to edit the file using your favorite flavor of editor (Screenshot illustrates Notepad++)
- After the editor is launched, Windows Explorer can be closed
- Add the IP address and “gmd” as illustrated below:



```
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #       102.54.94.97       rhino.acme.com       # source server
17 #       38.25.63.10      x.acme.com         # x client host
18
19 # localhost name resolution is handled within DNS itself.
20 #   127.0.0.1       localhost
21 #       ::1         localhost
22 192.168.40.118 gmd
```

- Save the file

N.B. The editor may ask to restart in admin mode; allow it to continue as *hosts* is a system file

3.5 Download and Install Support Files

This section details downloading and installation of the necessary Qt support files. It also describes configuration of the host machine and 3Dxx display for operation with the Qt development environment. The scripts work for all display models 3D50, 3D70, and 3D2104.

- Launch an internet browser
- Navigate to www.grayhill.com/qt43d

3.5.1 Linux

N.B. Firefox can be launched from Applications → Internet

- Download QtGhInstall5122Linux
- Copy/move the downloaded file to /home/ghguest
- Open a terminal window and cd to home (**cd**)
- Make QtGhInstall5122Linux executable (it should already be executable)
 - `chmod 755 QtGhInstall5122Linux`
- Unarchive the files (self-extracting archive)
 - `./QtGhInstall5122Linux`
- Make installation script executable
 - `chmod 755 QtGhInstallLinuxInstall`
- Install Qt support files on display
 - `./QtGhInstallLinuxInstall`

The above script without any arguments defaults to updating both the VM and the display. To update additional displays, connect the 3Dxx and update **gmd** (see previous section) with the IP address then re-run the installation script to configure the display.

- `./QtGhInstallLinuxInstall 3dxx`

The script reboots the display and requires a few minutes to complete execution. If everything works correctly these are the last few lines:

```
setup3Dxx completed successfully... rebooting
```

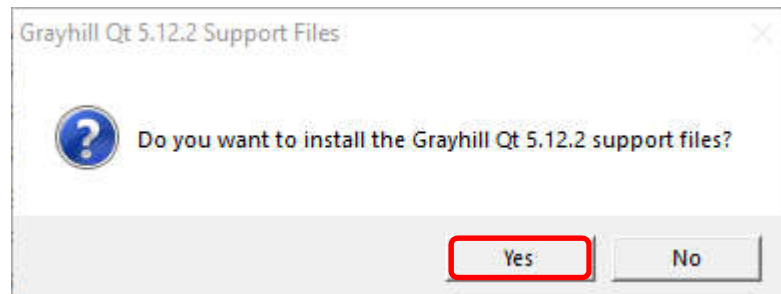

If a message similar to this does not appear, the problem(s) **must** be corrected before continuing.

The following files/directories are created on the VM:

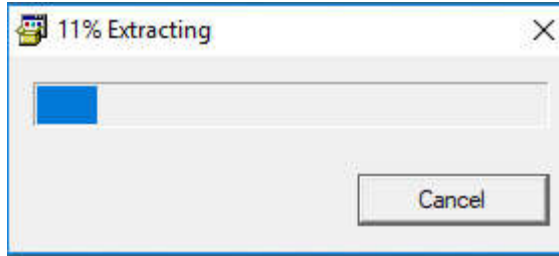
- GrayhillDisplayPlatform <dir> sysroot for cross-compiling
- GrayhillExamples <dir> sample projects
- QtGhInstallLinuxInstall <file> installation script (re-run for additional displays)
- targetFiles <dir> files copied/installed to the 3Dxx

3.5.2 Windows

- Download “Qt Creator Windows Support Files” from the Grayhill website
- Open the download folder and double click on “QtGhInstall5122Win10.exe”
- A User Account Control window may pop-up
 - Click “Yes” to allow the self-extracting zip file to proceed
- The following window appears



- Click “Yes”



```
C:\WINDOWS\system32\cmd.exe

Installing Grayhill Qt support files in C:\QtGhSupport ...
(patience please this takes a few minutes)
40192 File(s) copied

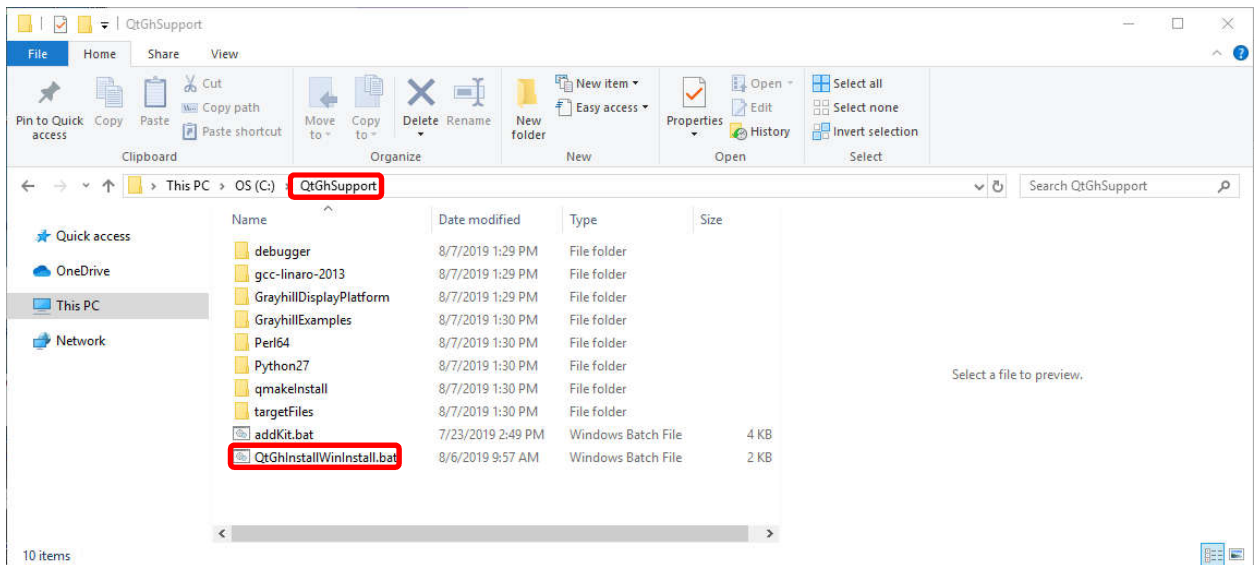
Creating Qt kit for Grayhill Display

Adding toolchain
Adding debugger
Adding qmake (Qt Version)
Adding display (target device)
Creating kit

Remember to install the files on the display:
  cd C:\QtGhSupport
  QtGhInstallWinInstall.bat

Press any key to close...
```

- Using Windows Explorer; navigate to “C: QtGhSupport” and verify the folder was installed



- Double click on “QtGhInstallWinInstall.bat” to configure the display

```
C:\Windows\system32\cmd.exe
```

```
Ensuring display file system is writable...  
The authenticity of host 'gmd (192.168.5.70)' can't be established.  
RSA key fingerprint is SHA256:DqmL/DR2Jk4/Klvji+cGNGs2J88o//GH8Lp1HzR1Ad8.  
Are you sure you want to continue connecting (yes/no)?
```

```
C:\WINDOWS\system32\cmd.exe
Ensuring display file system is writable...

Transferring files to display...
(patience please this takes a few minutes)

Configuring the display...

Configuring Grayhill 3Dxx Display for Qt-5.12.2
Wed Aug 7 13:57:02 CDT 2019
Disabling VUI Builder and Codesys applications on 3Dxx
Installing IPKs...
Updating GCC Libraries on 3Dxx Display
Upgrading gcclibs on root from 4.7.3 to 4.8.3...
Removing obsolete file /usr/lib/libstdc++.so.6.0.17.
Configuring gcclibs.
Updating Grayhill I/O Libraries on 3Dxx Display
Installing ghdrv-lib (1.1) to root...
Configuring ghdrv-lib.
Updating glibc Libraries on 3Dxx Display
Upgrading glibc on root from 2.16.0 to 2.18.0...
Removing obsolete file /lib/libnss_compat-2.16.so.
Removing obsolete file /lib/libc-2.16.so.
Removing obsolete file /lib/libdl-2.16.so.
Removing obsolete file /lib/libutil-2.16.so.
Removing obsolete file /lib/libpthread-2.16.so.
Removing obsolete file /lib/libnsl-2.16.so.
Removing obsolete file /lib/ld-2.16.so.
Removing obsolete file /lib/libnss_files-2.16.so.
Removing obsolete file /lib/libcrypt-2.16.so.
Removing obsolete file /lib/libm-2.16.so.
Removing obsolete file /lib/libnss_dns-2.16.so.
Removing obsolete file /lib/libresolv-2.16.so.
Removing obsolete file /lib/librt-2.16.so.
Configuring glibc.
Updating GPU Libraries on 3Dxx Display
Installing gpu-viv2 (3.0.101+4.1.1) to root...
Configuring gpu-viv2.
Installing Qt5 Libraries on 3Dxx Display
No packages removed.
No packages removed.
Installing qt5122 (5.12.2) to root...
Configuring qt5122.
bootangs already set to console=ttyMXC0,115200 lpj=7905280 rootfstype=ext4 root=/dev/mmcblk0p1 no rootwait board-ghi_imx
6.pn=3D70VT-100 quiet
Updating /etc/profile.local script
Updating /usr/lib/fonts
setup3Dxx completed successfully... rebooting
Wed Aug 7 13:58:47 CDT 2019
Making 3Dxx Flash File system writeable and adding writeablefs script
Rebooting to make following steps run faster

Press any key to close...
```

- Restore any custom modifications. The setup script preserves files by appending a timestamp

3.6 Build and Run 3Dxx Embedded Application

This section details how to build and run a demo application.

A Qt QML demonstration project is provided which runs (configured as necessary) on each of the 3Dxx displays as well as the host machine.

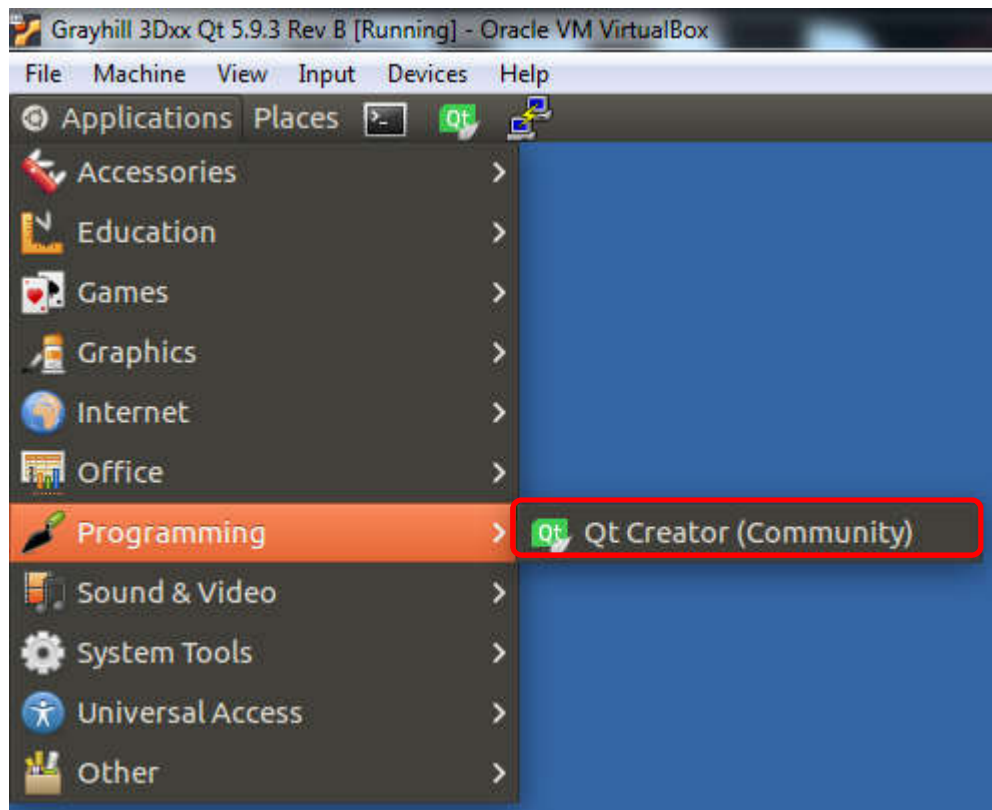
Complete configuration instructions are in the appendices.

3.6.1 Launch Qt Creator

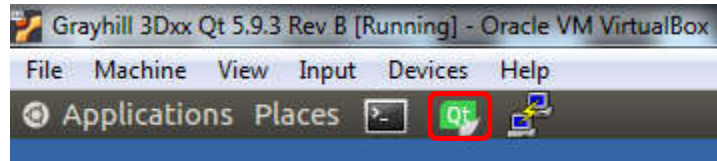
Linux

Launch Qt Creator using one of the following methods:

- Select “Applications” (upper left-hand corner of the Linux window), then navigate through “Programming” and click on “Qt Creator ...”



- Click on the Qt icon in the panel

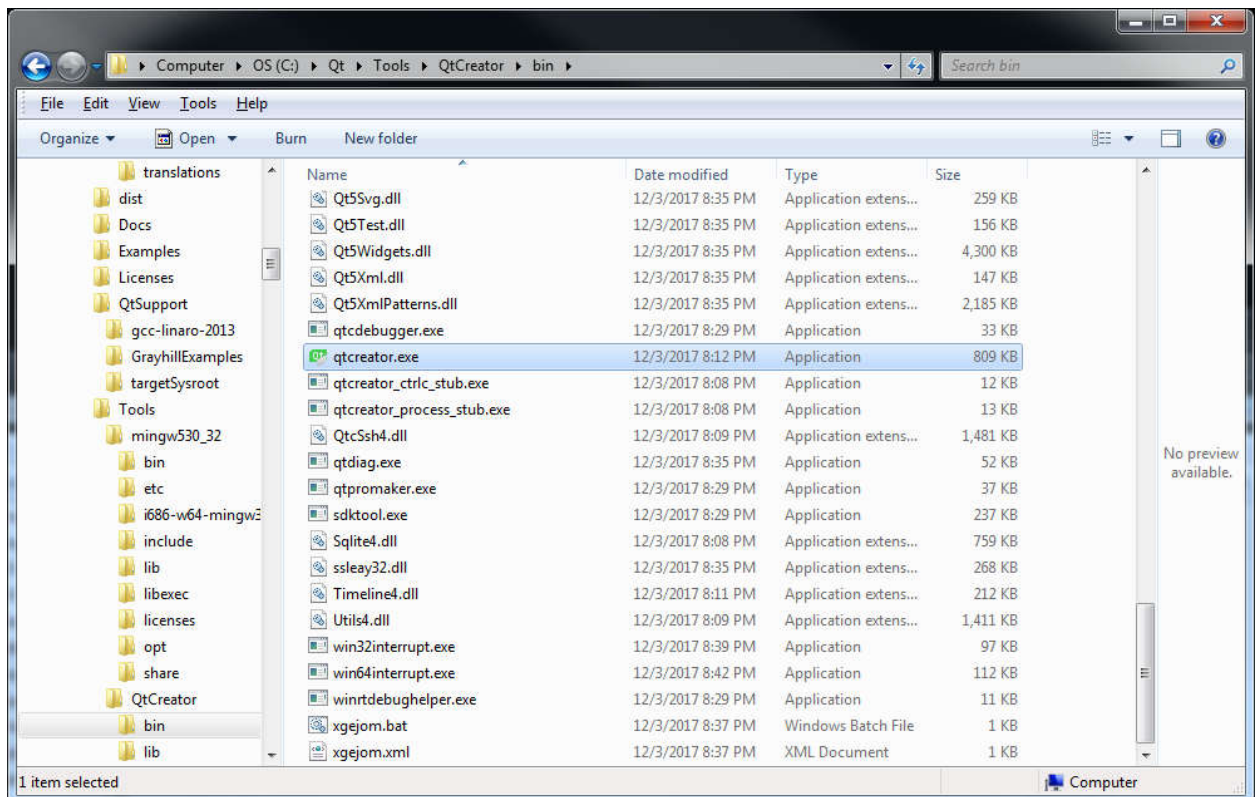


- Double click on the Qt icon on the desktop



Windows

- Launch Windows Explorer (<Windows>-e)
- Navigate to C: → Qt → Tools → QtCreator → bin → qtcreeator.exe

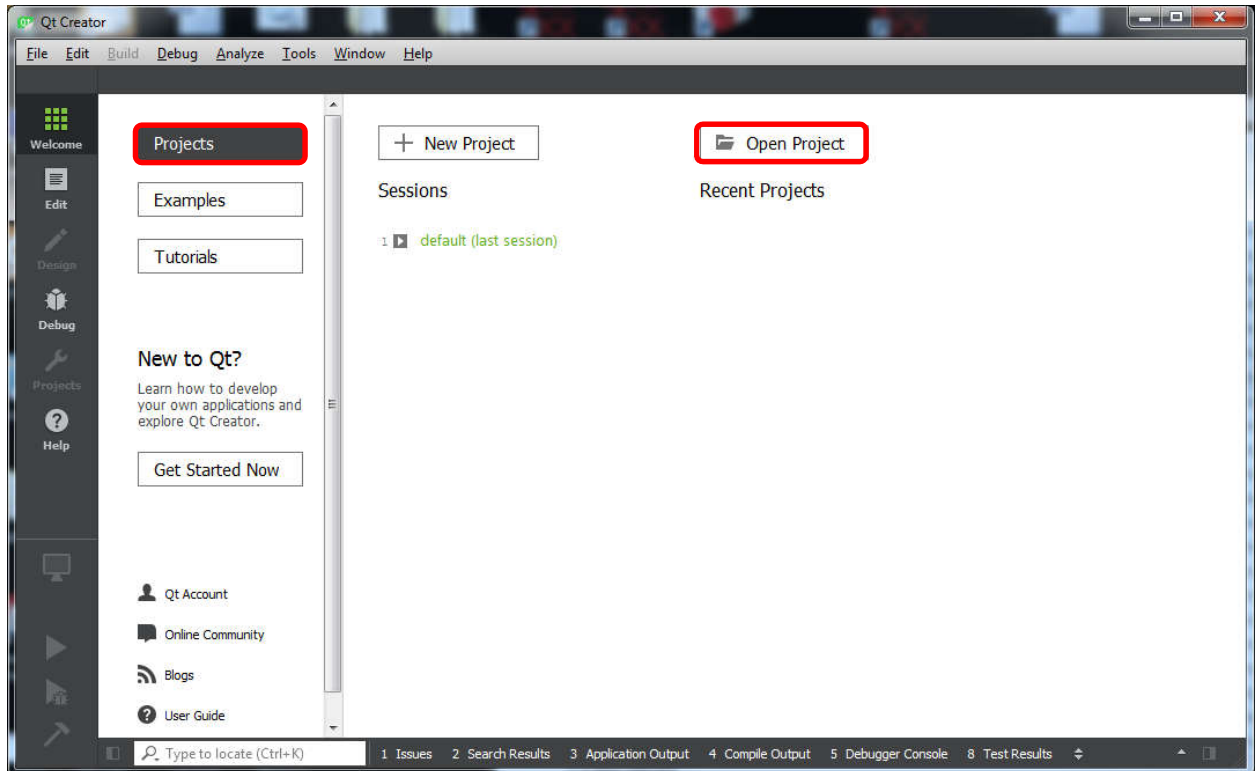


- Right click to select options like
 - "Pin to Taskbar"

- “Send to” → Desktop (create shortcut)
- Double click to launch Qt Creator

3.6.2 Open project

- Select “Projects”



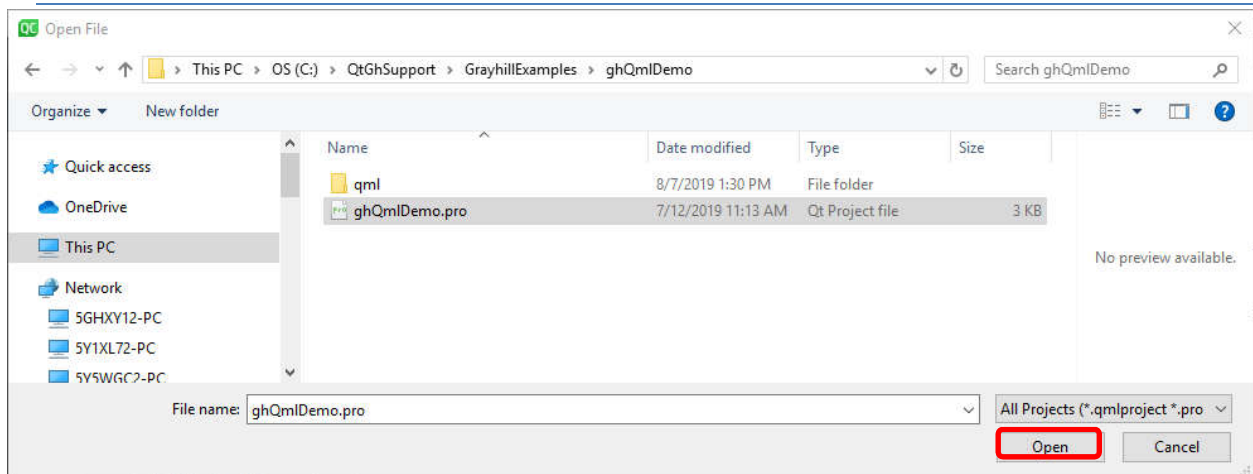
- Click on “Open Project” (“Welcome” should be automatically selected on launch)
- Navigate to the desired project

Linux

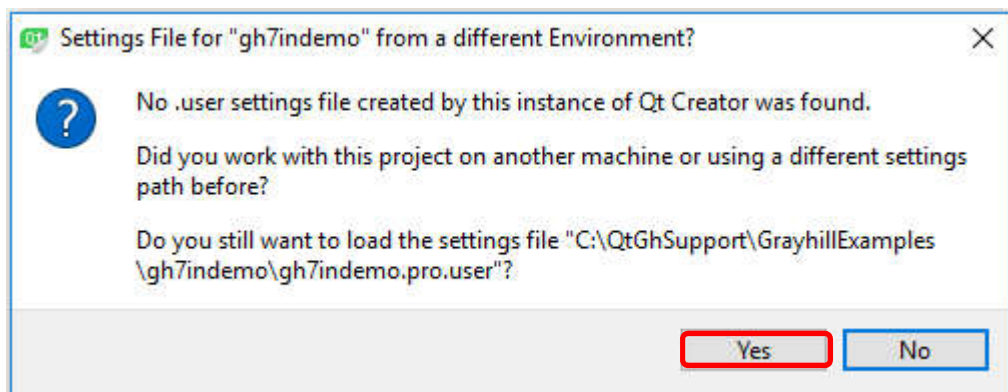
/home → GrayhillExamples → ghQmlDemo

Windows

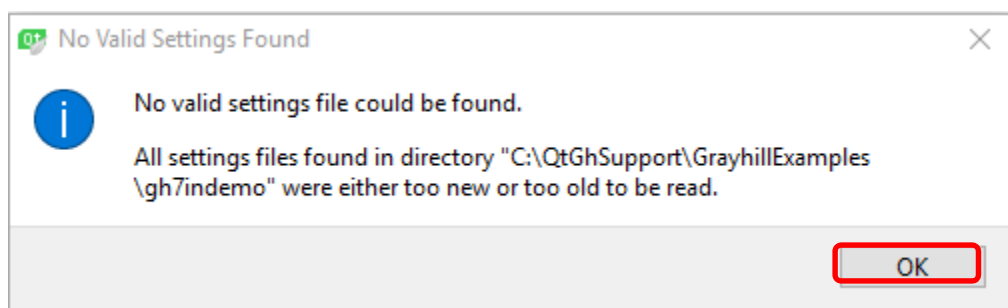
C: → QtGhSupport → GrayhillExamples → ghQmlDemo



- Select ghQmlDemo.pro
- Click “Open”
- If a similar box appears, click “Yes”



- If a similar box appears, click “OK”. **Refer to Appendix B: Configuring a 3Dxx Project before continuing.** The current project configuration file is not compatible with the current version of Qt Creator and the project’s settings need to be re-configured.





Configure Project

The following kits can be used for project **ghQmlDemo**:

The project **ghQmlDemo** is not yet configured.

Qt Creator uses the kit **Desktop Qt 5.12.2 MinGW 32-bit** to parse the project.

 Select all kits

<input checked="" type="checkbox"/> Desktop Qt 5.12.2 MinGW 32-bit	Details ▼
<input checked="" type="checkbox"/> Qt-5.12.2-3Dxx	Details ▼
Import Build From...	Details ▼

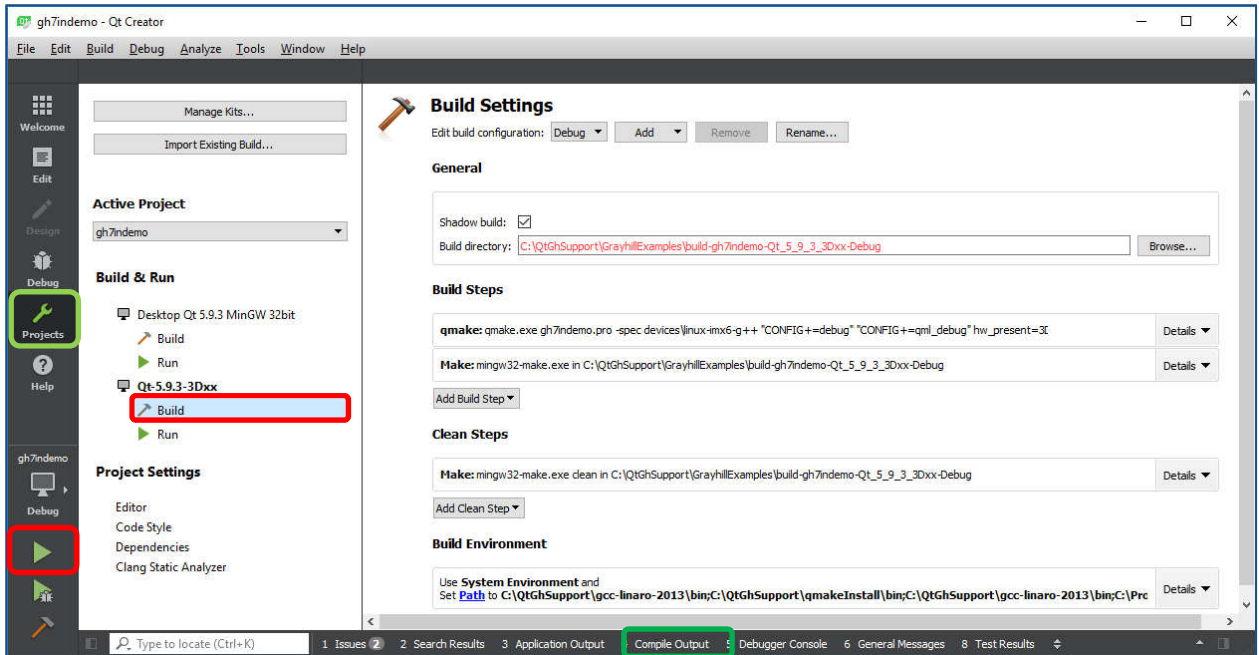
Configure Project

3.6.3 Build Project

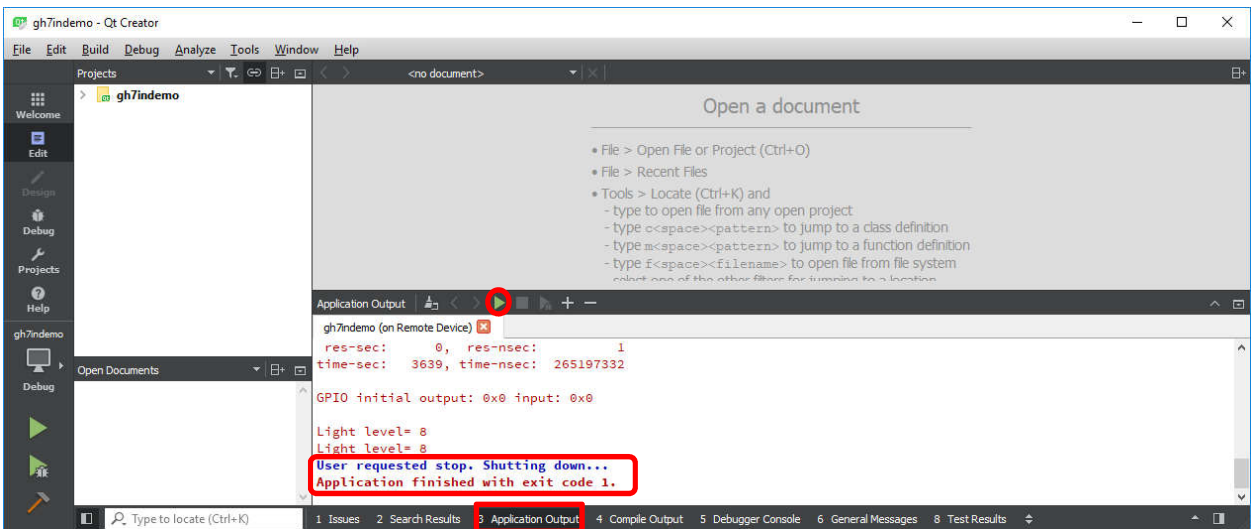
- Select “**Projects**” view
- Select “Build” under “**Qt-5.12.2-3Dxx**”³
- Expand qmake
- Verify qmake “Additional arguments:” is set to “**hw_present=yes**”⁴
target=3D70. **N.B.** use 3D50 or 3D2104 based on actual display.

³ To build for the desktop select “Build” under “Desktop” Certain features are not supported (e.g. Camera)

⁴ hw_present **must** not be present for desktop builds and windowsOnly=true needs to be set for Window builds of the demo



- Click on the green arrow to run (a check to see if the executable is up to date is performed; if compilation is necessary the output can be viewed by clicking on the “Compile Output” tab)

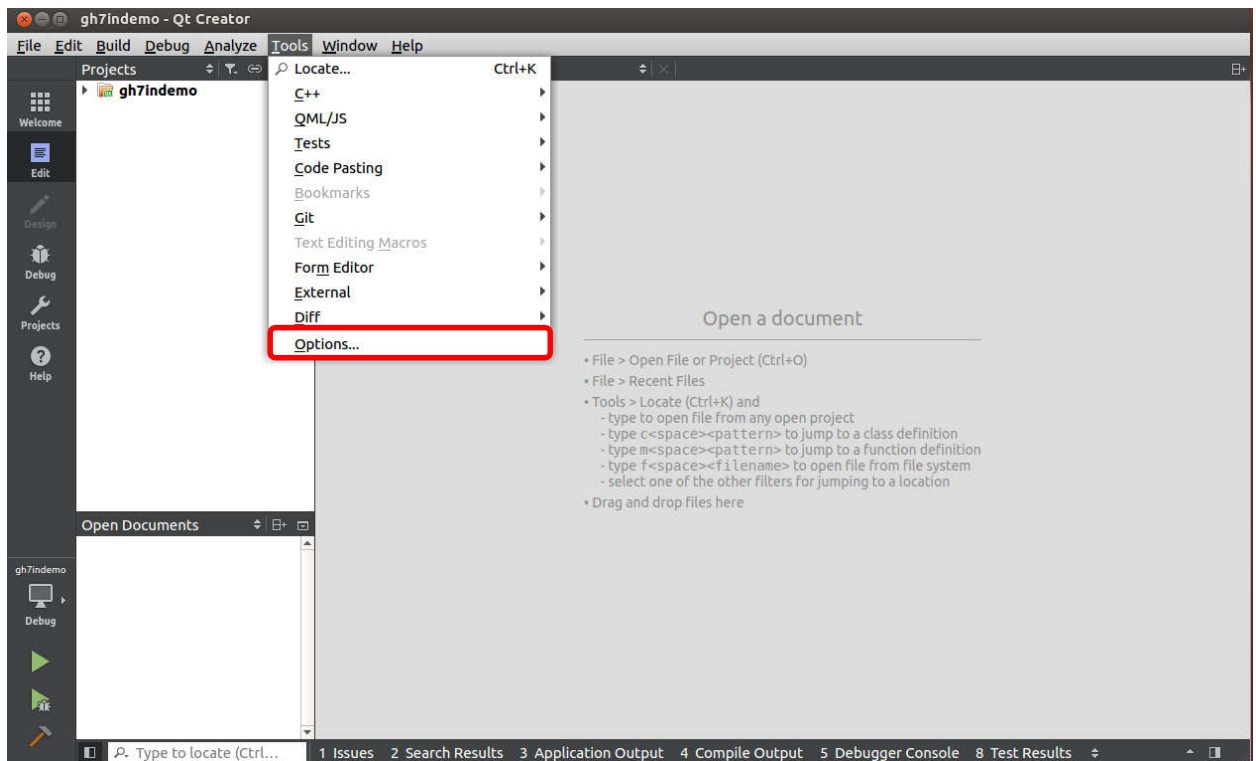


- Select the “Application Output” tab
- Click the red (when application is running on target) square to terminate the target session

Appendix A: Configuring a Manual Qt Kit for Grayhill Displays

N.B. This appendix is included for reference and is not a required installation step. Grayhill automatically installs the kit configuration as part of the support file installation. A kit is a collection of utilities (qmake, compilers, debugger, etc...) used to build a project.

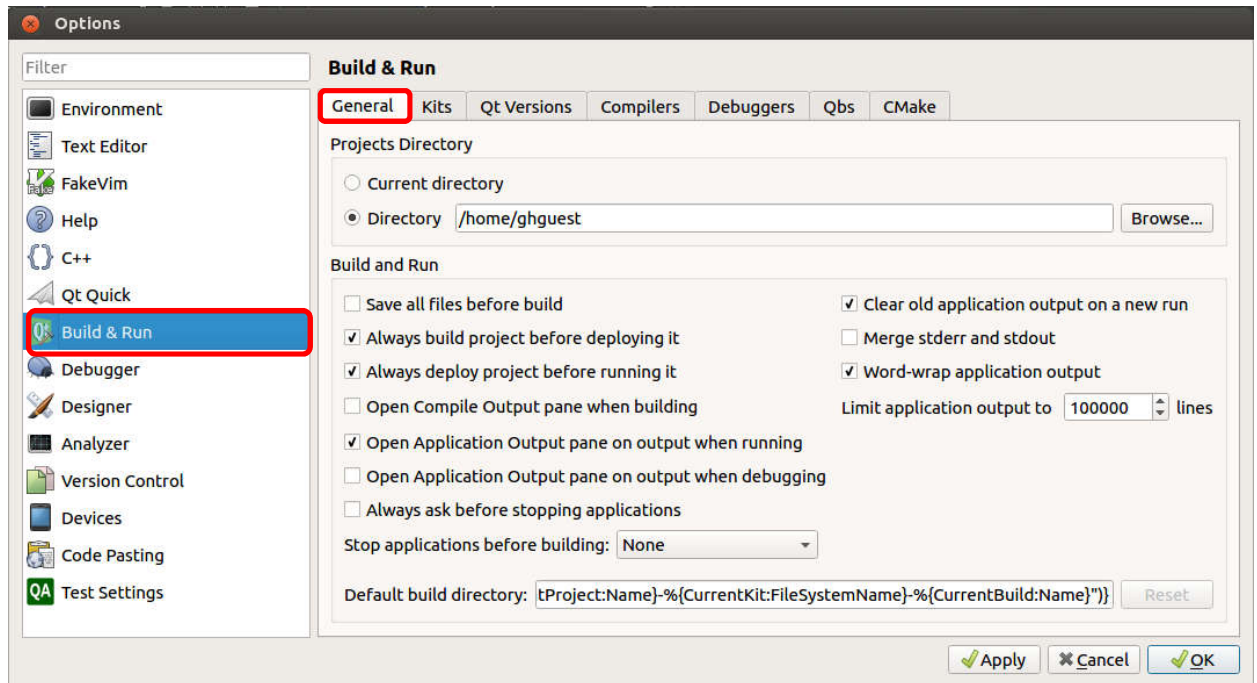
- Launch Qt Creator
- Select Tools → Options



Alternatively, “Manage Kits” can be selected from the “Projects” view.

General

The “General” tab is where project wide customization is done. Review and select the desired configuration.

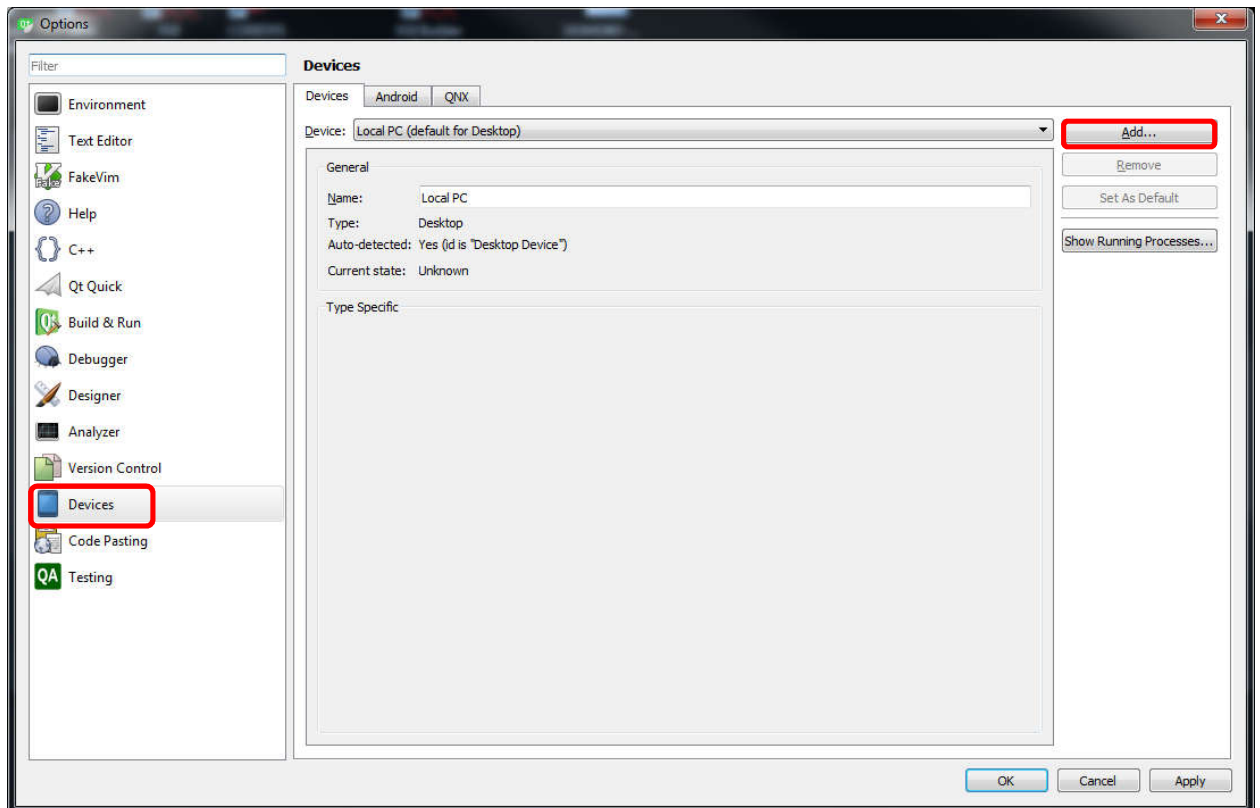


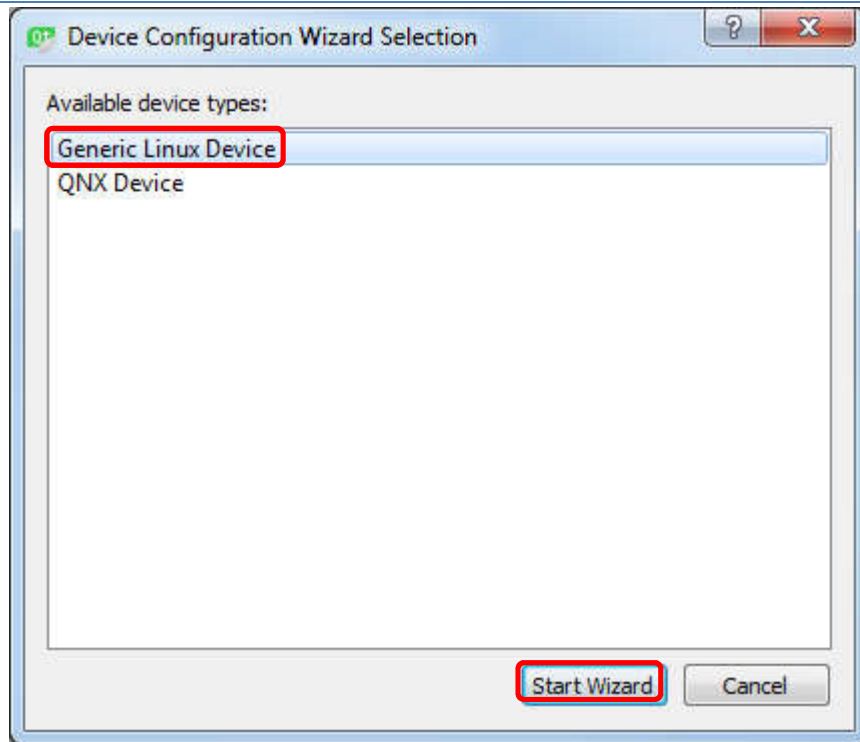
- Select “Build & Run”
- Select “General” tab
- Click “Apply” to continue and select other tabs, “OK” if finished

Device

The section describes how to establish an Ethernet based connection to the display.

- Select “Devices”
- Click “Add...”





- Select “Generic Linux Device”
- Click “Start Wizard”

New Generic Linux Device Configuration Setup

Connection

> Connection
Summary

The name to identify this configuration: 3Dxx Target

The device's host name or IP address: gmd

The username to log into the device: root

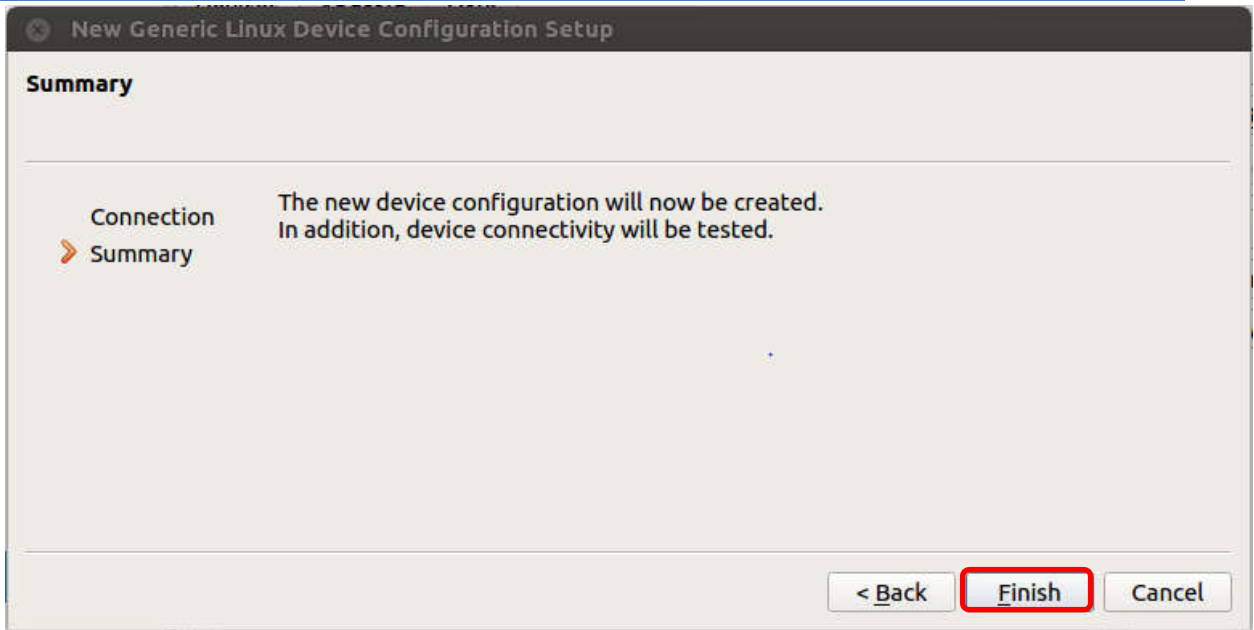
The authentication type: Password Key Agent

The user's password:

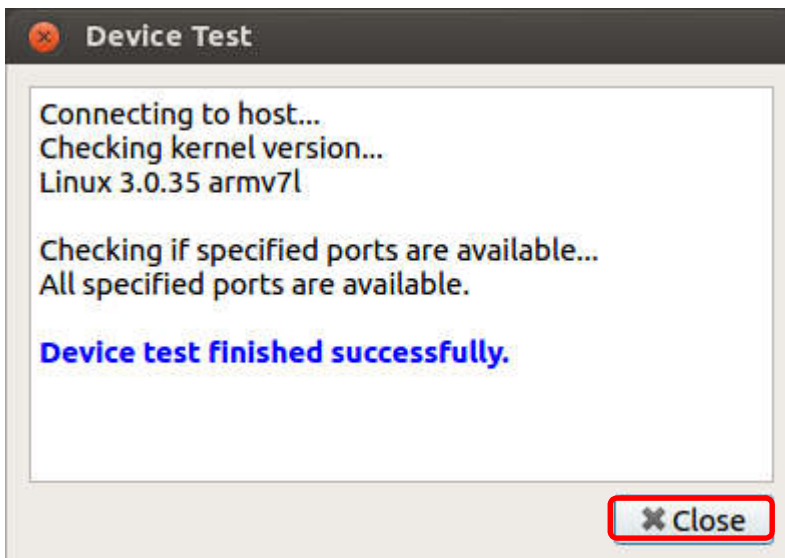
The file containing the user's private key: /ghguest/.ssh/id_rsa Browse...

Next > Cancel

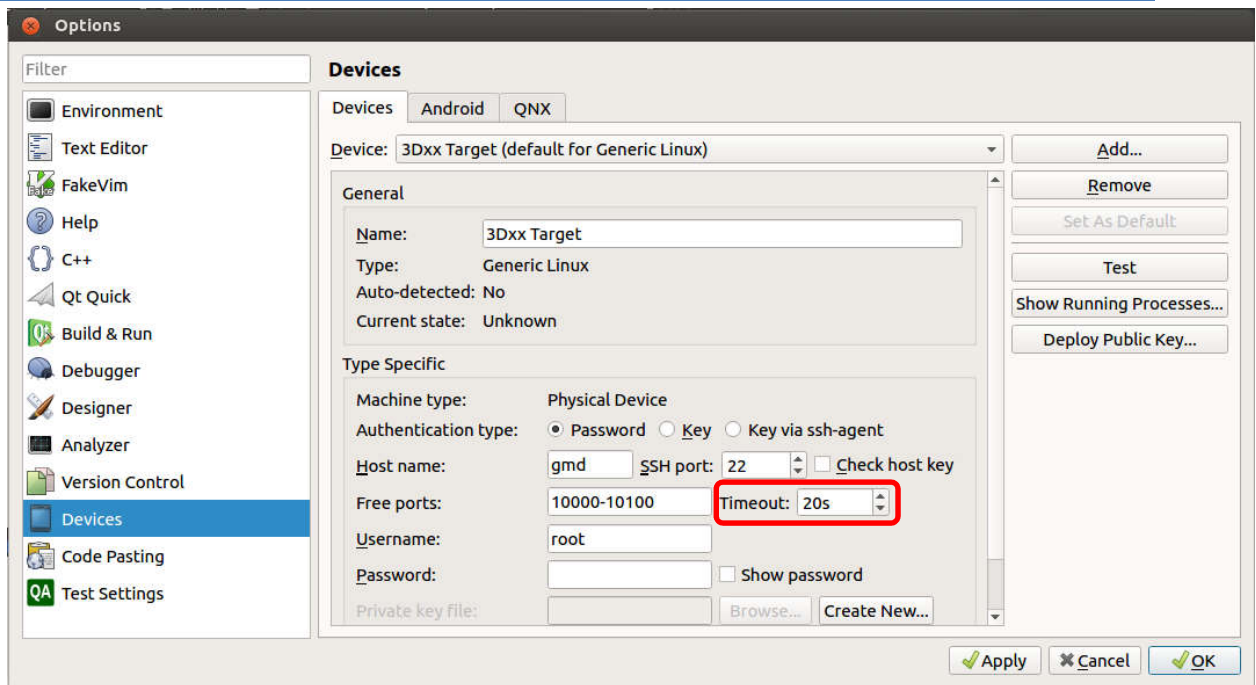
- Populate the fields as illustrated above
- **N.B.** The IP address associated with **gmd** is located in **/etc/hosts** (Linux) and **C:\Windows\System32\drivers\etc** (Windows)
- Click “Next”



- Verify the 3Dxx Display is still powered up
- Click “Finish” – The Ethernet link to the 3Dxx Display will be tested and if successful the following result screen appears

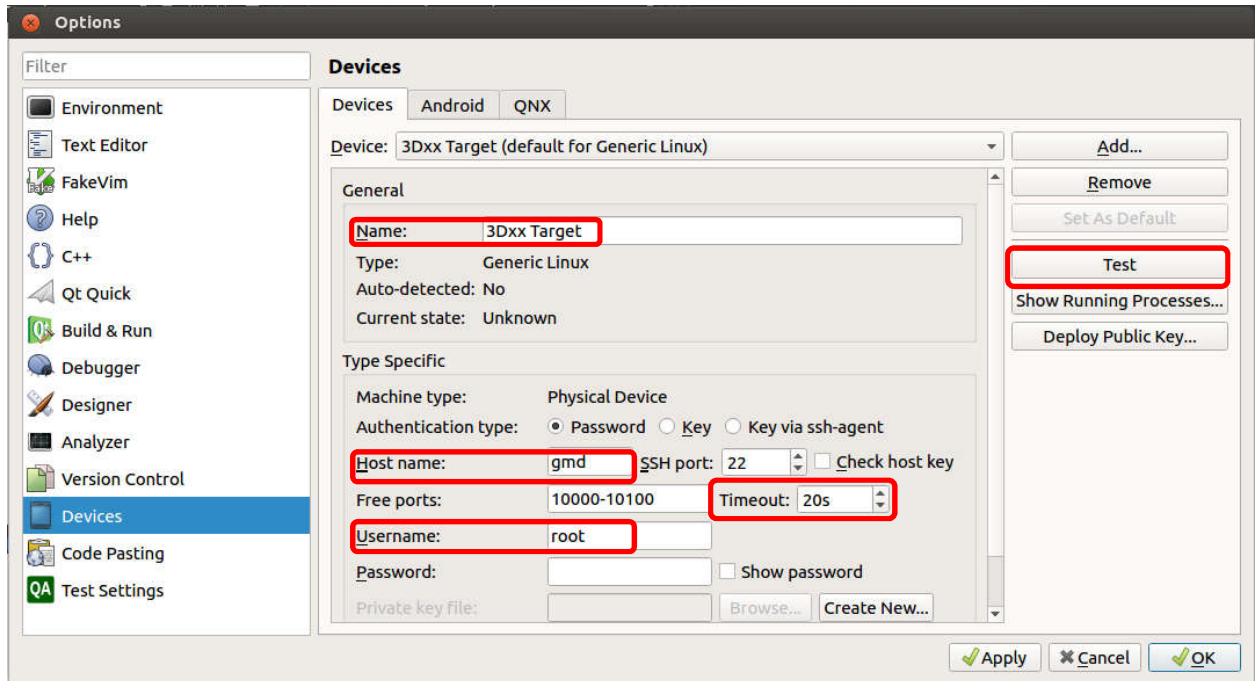


- Click “Close”



- Click the upper arrow on the right side of the “Timeout:” box to increase timeout value to “20s”

Devices Summary



- Name name of the device
- Host name ***gmd*** alias -- specified in ***hosts***
- Timeout 20s
- Username root

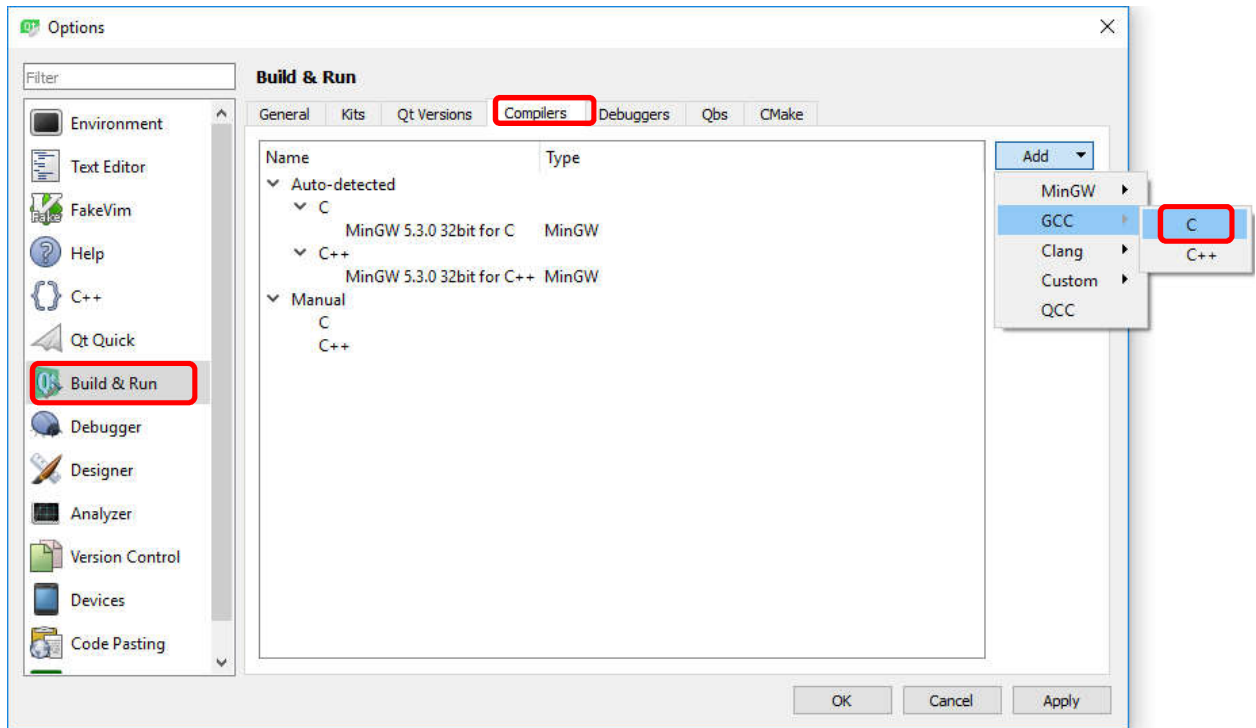
N.B. Remember verify connectivity using “Test”

Compiler

Select “Build & Run”

Select “Compilers” tab

Click “Add”; then select GCC → C

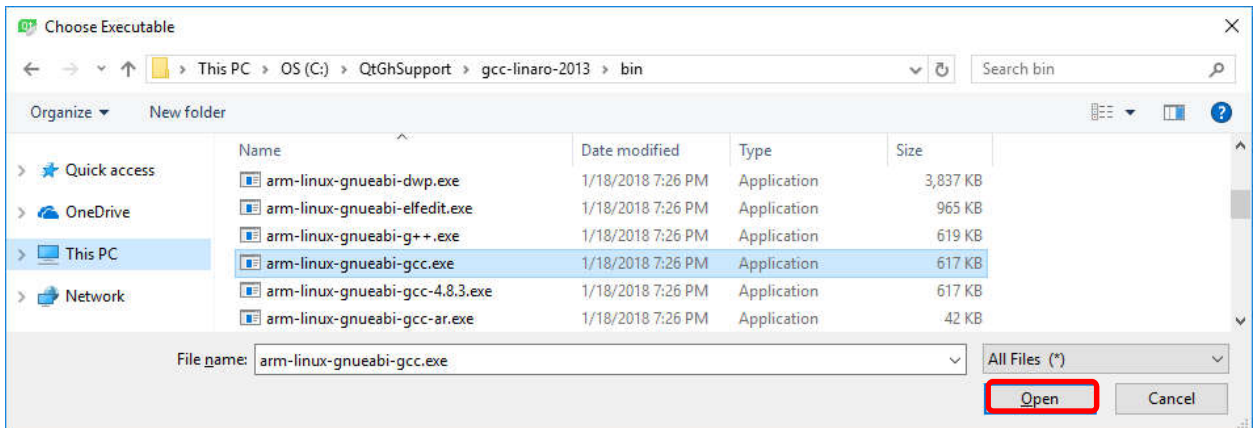


Populate the fields as illustrated

- “Name:” ARM-GCC
- “Compiler path:” Click “Browse...” and navigate to the desired file
 - `/opt/OSELAS.Toolchain-2013.12.3/arm-cortexa9-linux-gnueabi/gcc-4.8.3-glibc-2.18-binutils-2.24-kernel-3.12-sanitized/bin/arm-cortexa9-linux-gnueabi-gcc`



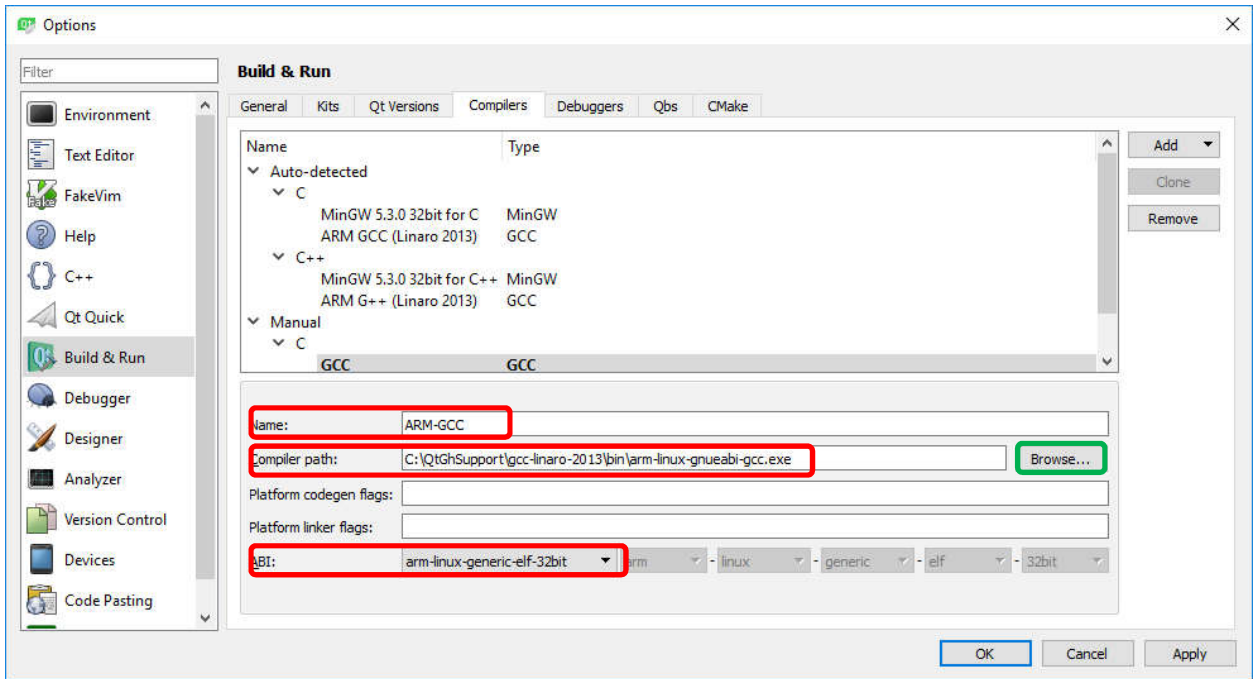
- C:\QtGhSupport\gcc-linaro-2013\bin\ arm-linux-gnueabi-gcc.exe



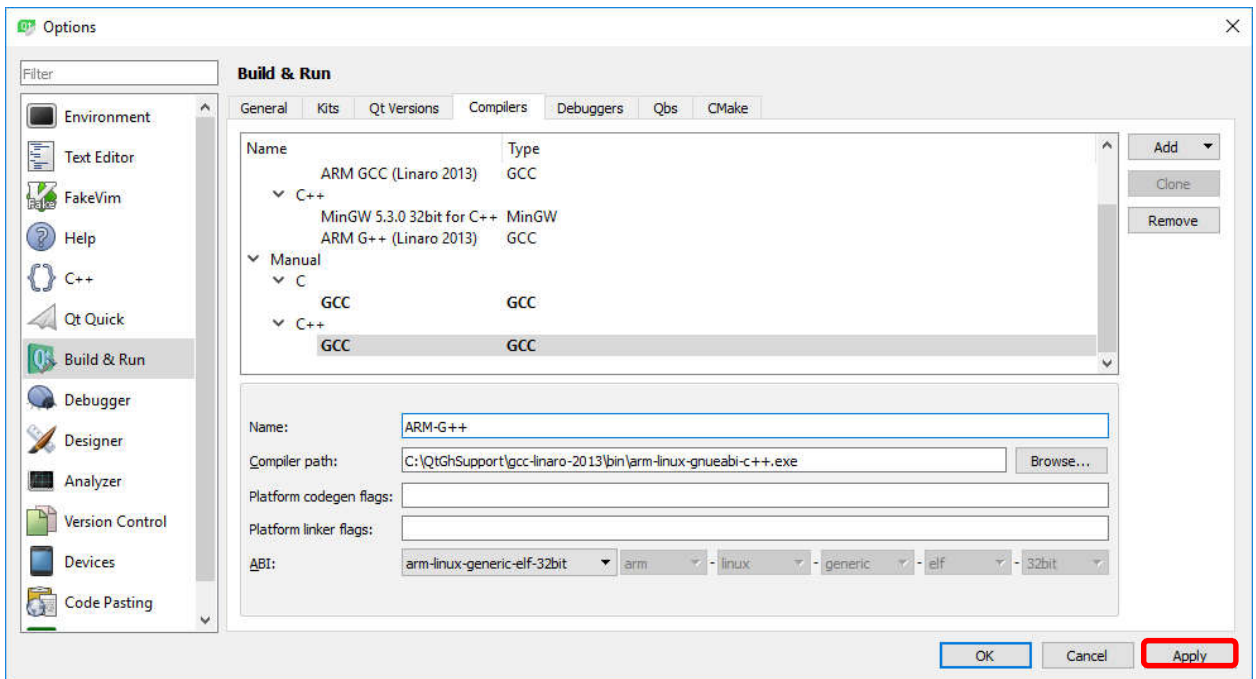
Click “Open”

“ABI:” Select “arm-linux-generic-elf-32bit”

The configuration portion of the screen should look similar to:



Repeat the above steps for GCC→C++



- Click “Apply”

Debugger

Select the “Debuggers” tab

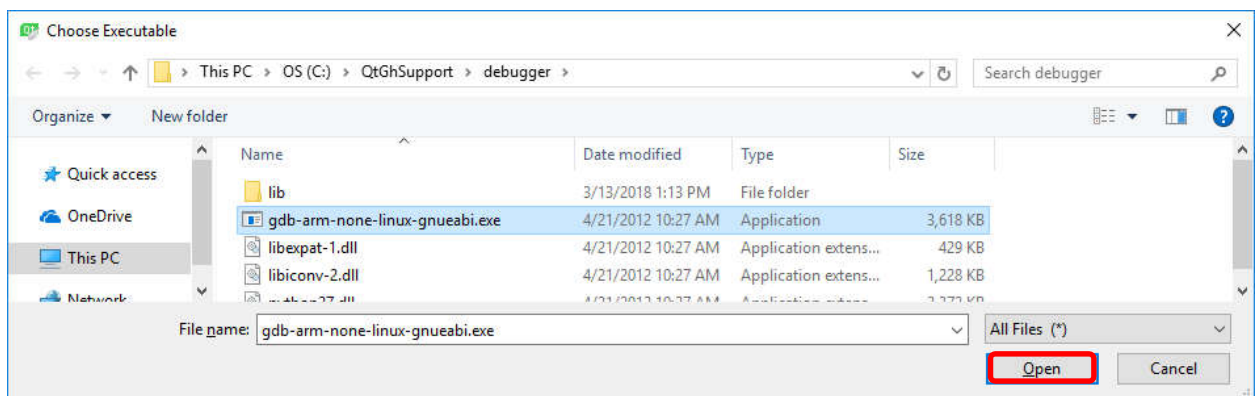
Click “Add”

Populate the fields as illustrated

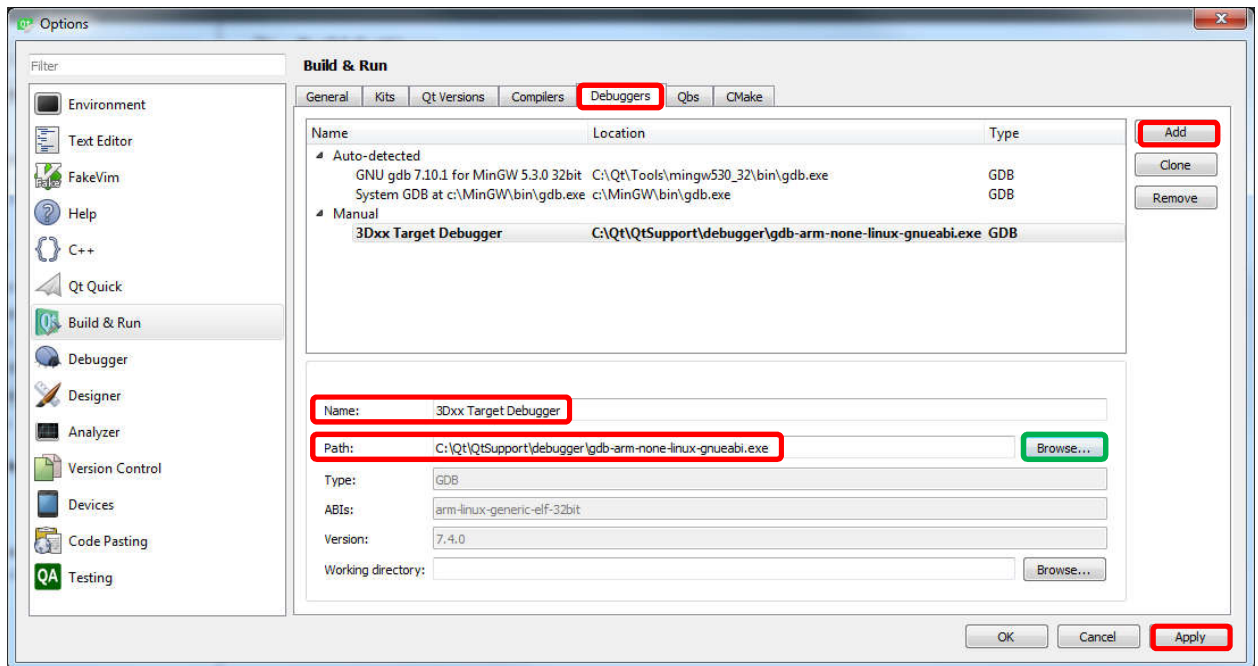
- “Name:” 3Dxx Target Debugger
- “Path:” Click “Browse...” and navigate to the desired file (should be previous directory)
 - /opt/OSELAS.Toolchain-2013.12.3/arm-cortexa9-linux-gnueabi/gcc-4.8.3-glibc-2.18-binutils-2.24-kernel-3.12-sanitized/bin/arm-cortexa9-linux-gnueabi-gcc



- C:\QtGhSupport\debugger\arm-linux-gnueabi-gcc.exe



- Click “Open”; the configuration portion of the screen should look similar to



- Click “Apply”

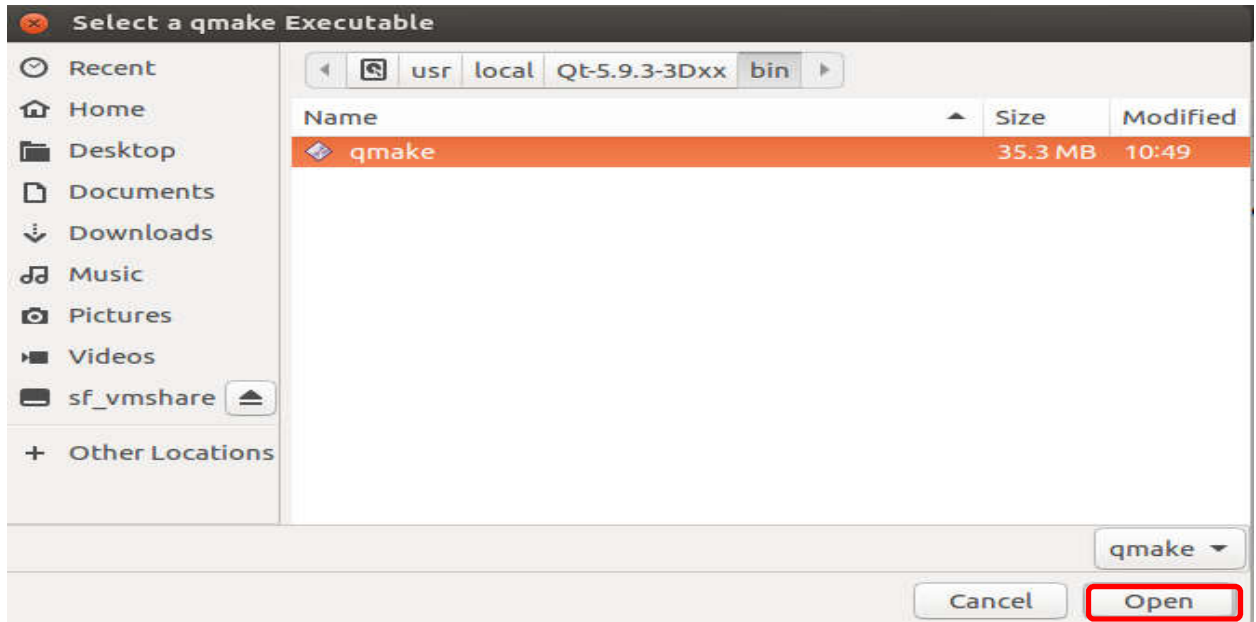
qmake

Select the “Qt Versions” tab

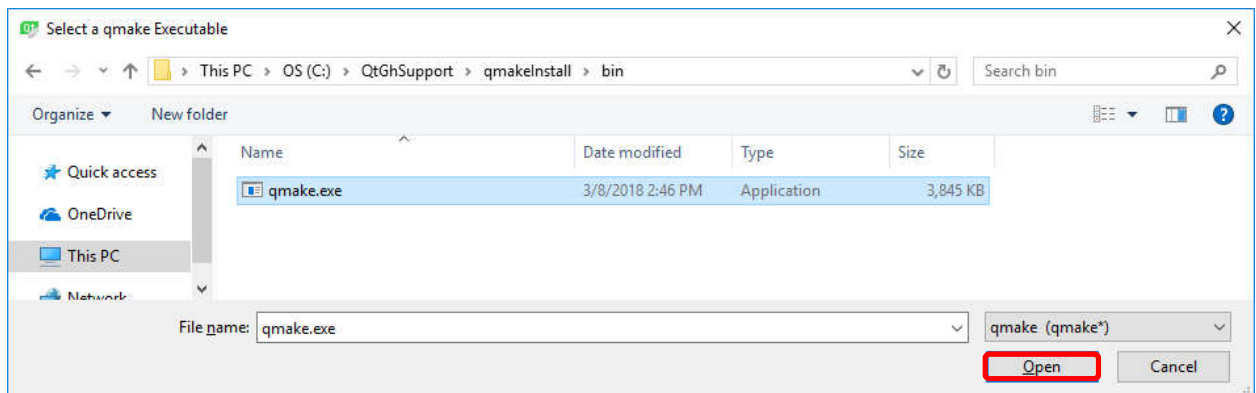
Click “Add” (Select a qmake Executable dialog box appears, still referencing the last path)

Navigate to the qmake version associated with the library

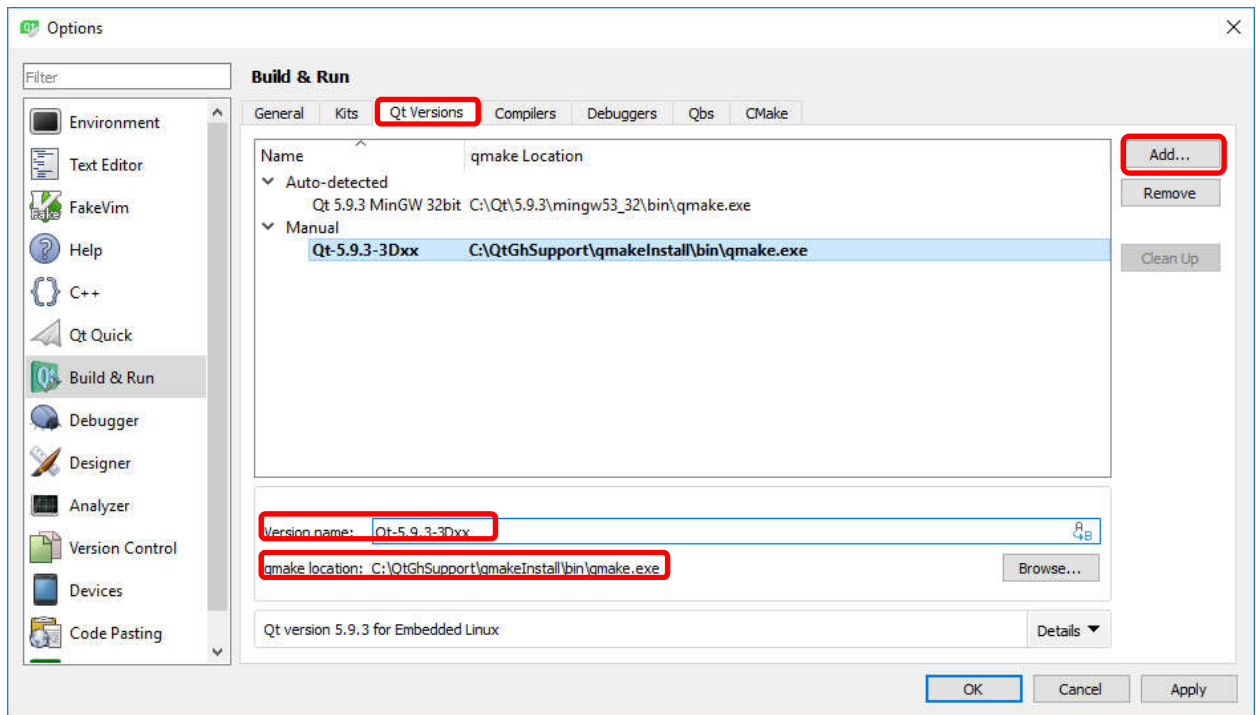
- /usr/local/Qt-5.12.2-3Dxx/bin/qmake



- C:\QtGhSupport\qmake\insatll\bin\qmake.exe



- Click “Open”
- Update “Version name:” to “Qt-5.12.2-3Dxx”



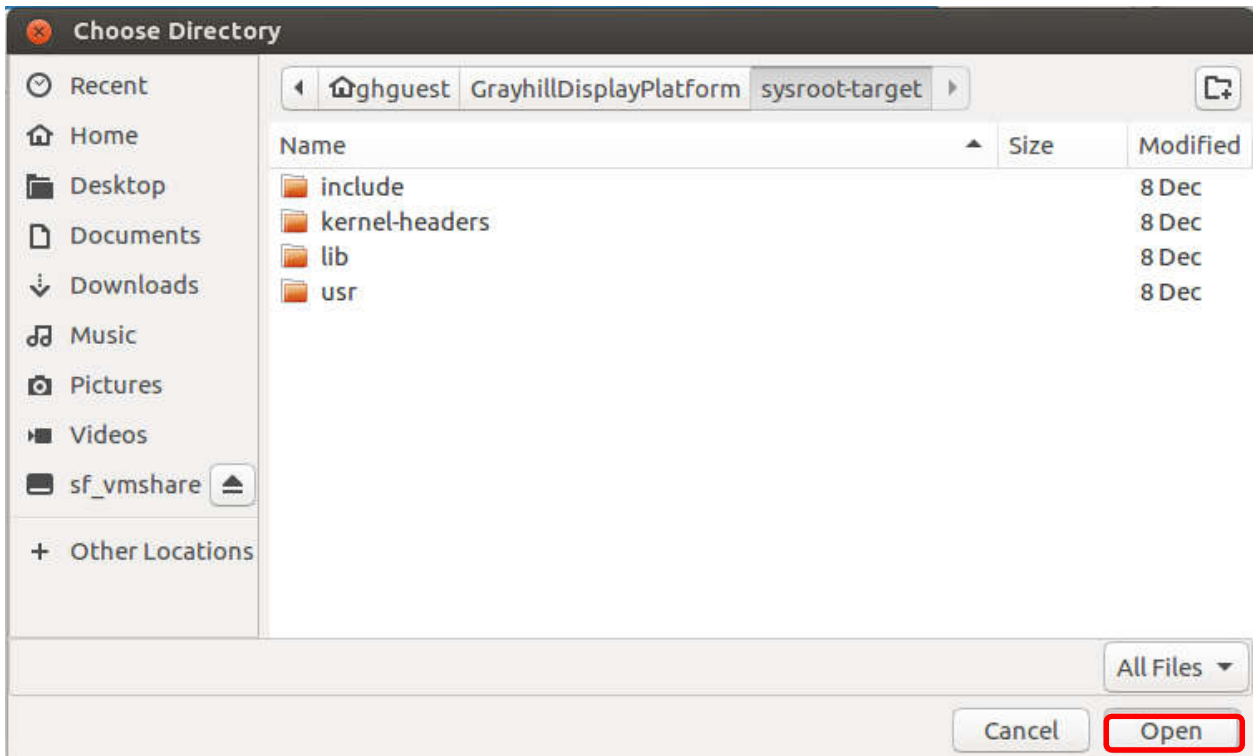
Kit

Select the “Kits” tab

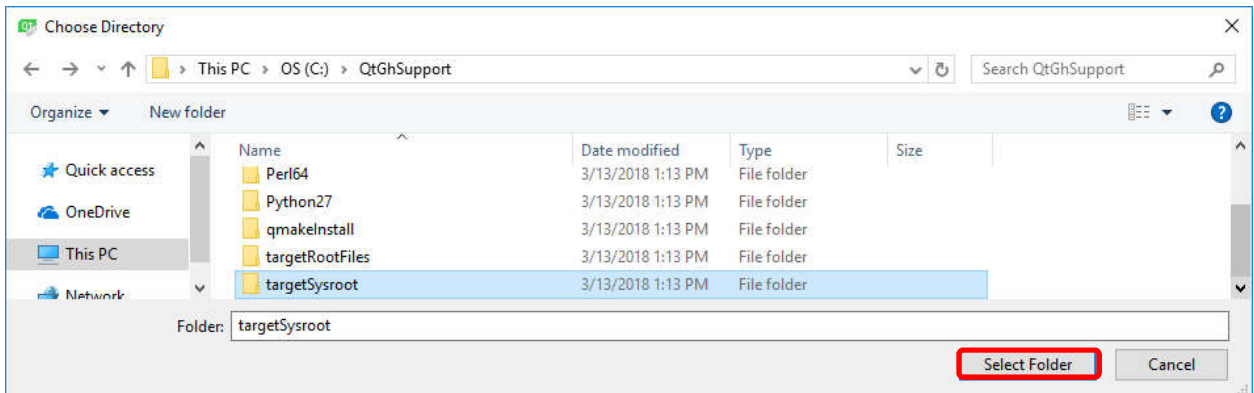
Click “Add”

Populate the fields as illustrated

- “Name:” **Qt-5.12.2-3Dxx**
- “Device type:” Select “Generic Linux Device” from the pick list
N.B. Automatically updates Device
- “Sysroot”: Click “Browse...” and navigate to desired path
 - /home/ghguest/GrayhillDisplayPlatform/sysroot-target



- Click “Open”
 - C:\QtGhSupport\GrayhillDisplayPlatform\sysroot-target

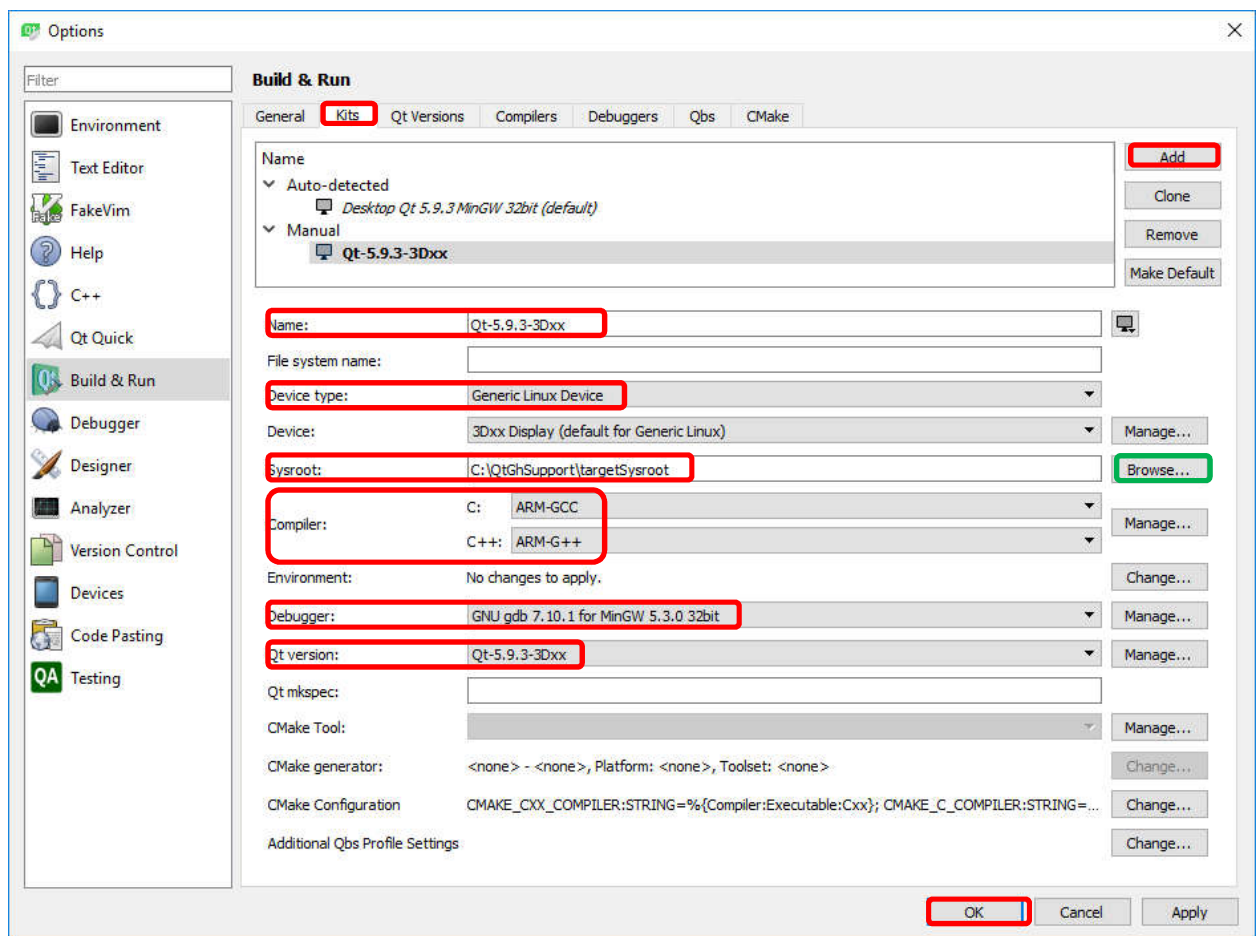


- Click “Select Folder”
 - “Compiler: C:” Select “ARM-GCC” from the pick list
 - “Compiler: C++:” Select “ARM-G++” from the pick list
 - “Debugger:” Select “3Dxx Target Debugger” from the pick list

“Qt version:” Select “Qt-5.12.2-3Dxx” from the pick list

N.B. The selected names must match those used when creating the various kit sub-components

Summary



- Verify contents are correct
- Click “OK”

Now that a Qt kit is configured; it is possible to develop, build, test, debug, run and enjoy Qt applications.

Appendix B: Configuring a 3Dxx Project

N.B. This appendix is included for reference and is not a required installation step; Grayhill automatically configures the project as part of the support file installation.

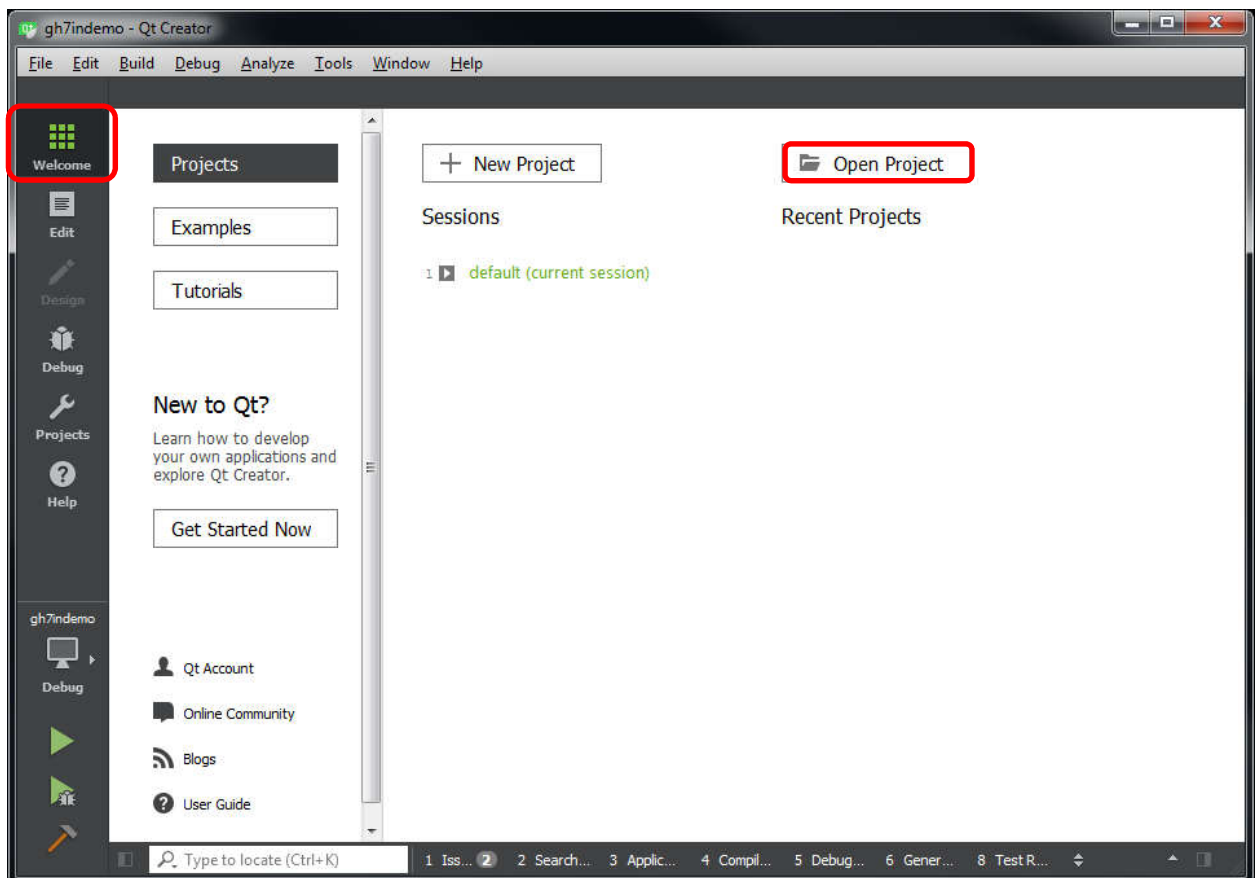
This section details how to setup and configure a project for the 3Dxx Display.

If not already running, launch Qt Creator. (See

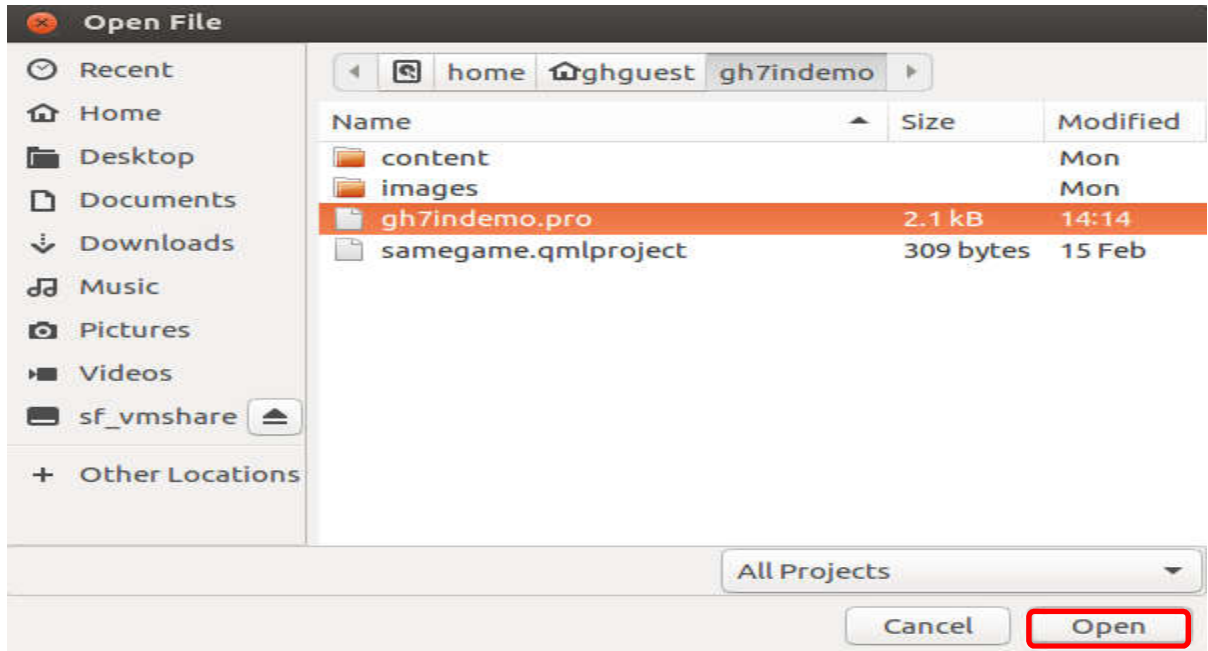
Build and Run 3Dxx Embedded Application)

Open a project from “Qt Creator” main window click on “Open Project” button.

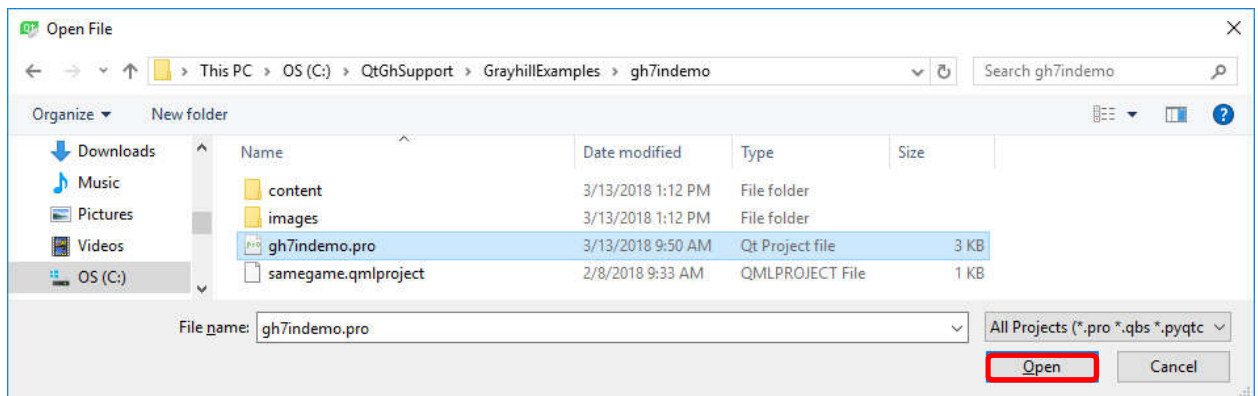
N.B. If present, a previous project can be opened by clicking on the project name listed below “Recent Projects”.



- An “Open File” dialog window will appear
- Navigate to the 3Dxx Demo project’s “.pro” file for either Linux or Windows as illustrated below
 - /home/GrayhillExamples/ghQmlDemo



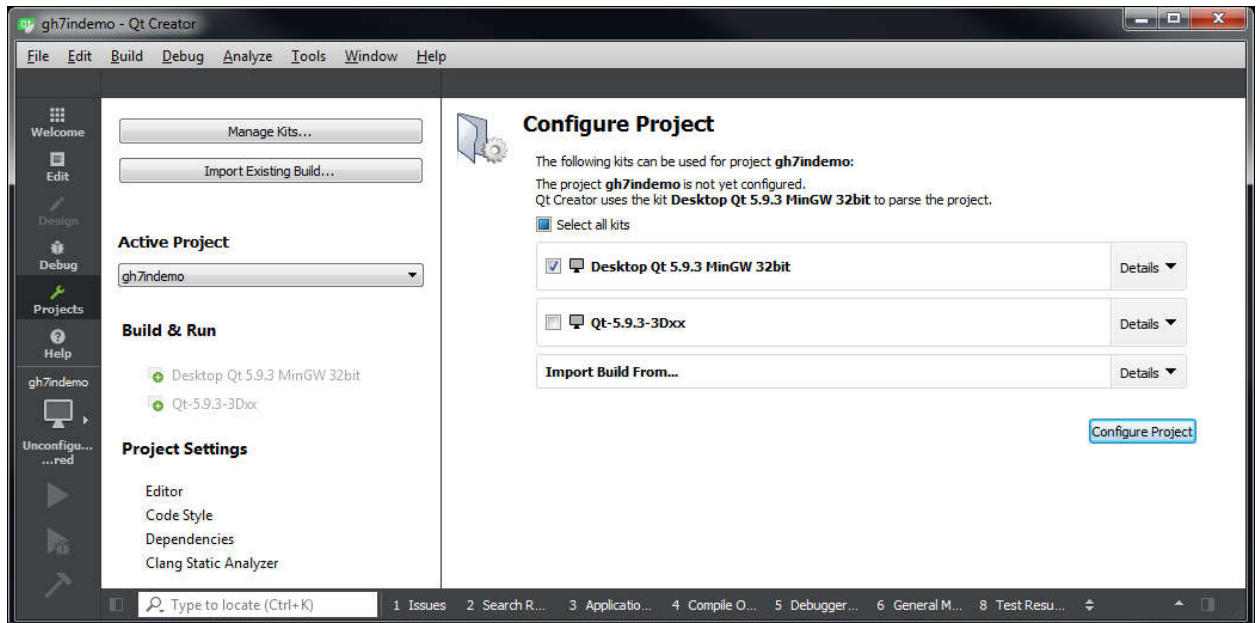
- C:\QtGhSupport\GrayhillExamples\gh7indemo\gh7indemo.pro



- Click “Open”

If the “**project.pro.user**” file is missing, which is normal if the project has never been opened before, a “Configure Project” dialog appears. If this dialog doesn’t appear, proceed to where the “Projects” icon is selected.

If the “Configure Project” dialog appears (remember screen shot illustrations are for reference purposes and may not reflect current observations)

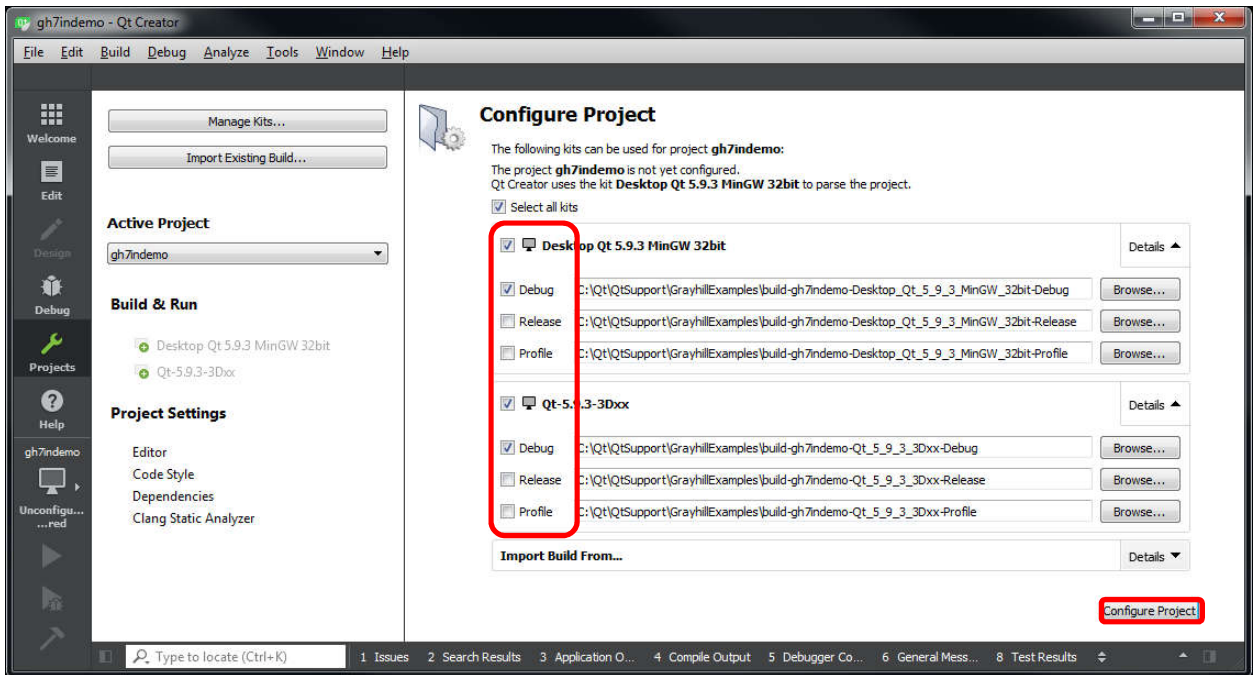


“Desktop Qt 5.9.3 MinGW 32bit”

- Expand by clicking on “Details”
 - Unselect “Release”
 - Unselect “Profile”

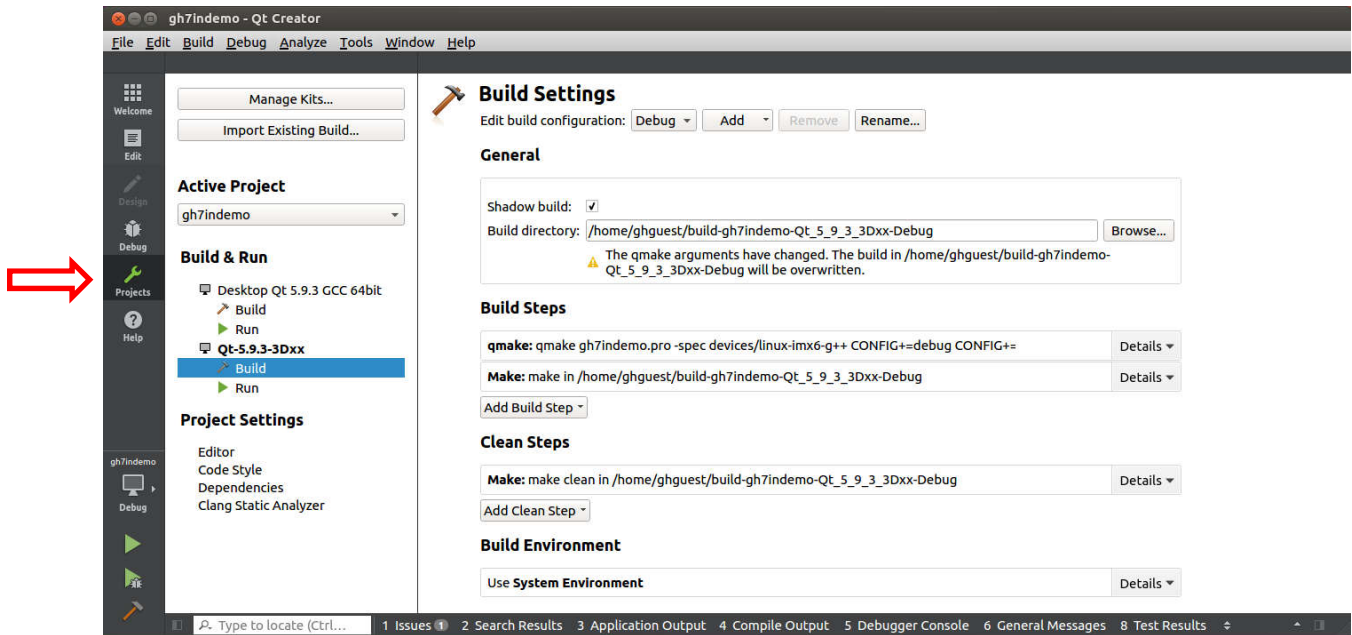
“Qt-5.9.3-3Dxx”

- Expand by clicking on “Details”
- Select “Qt-5.9.3-3Dxx” (this selection will select the three boxes below)
 - Unselect “Release”
 - Unselect “Profile”



- Click "Configure Project"

- On the main “Qt Creator” window select “Projects”



- If the desired kit is not shown see

Appendix A: Configuring a Manual Qt Kit for Grayhill Displays

N.B. Clicking “Manage Kits” is the same as selecting “Tools → Options”

“Active Project” is a drop down pick list with the active project shown.

“Build & Run” lists the available kits.

N.B. The selected kit is emphasized in **bold**. A kit (set of utilities) is how the project will be built, e.g. the main kit difference between desktop and target is the compiler as the **Qt-5.12.2-3Dxx** kit uses a cross compiler for the display.

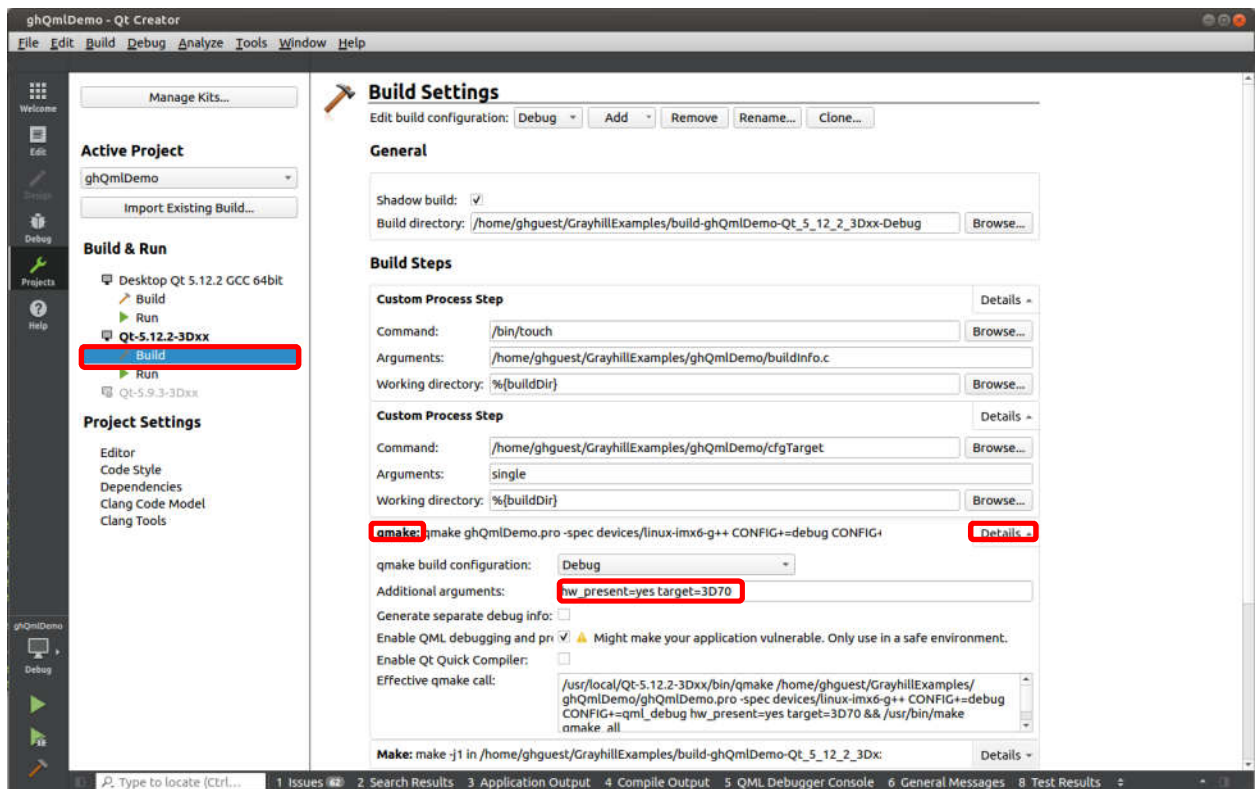
N.B. Clicking on an actual kit name selects either Build or Run (depending on which one was previously selected)

Build

This section describes how to configure the example project for the target (3Dxx).

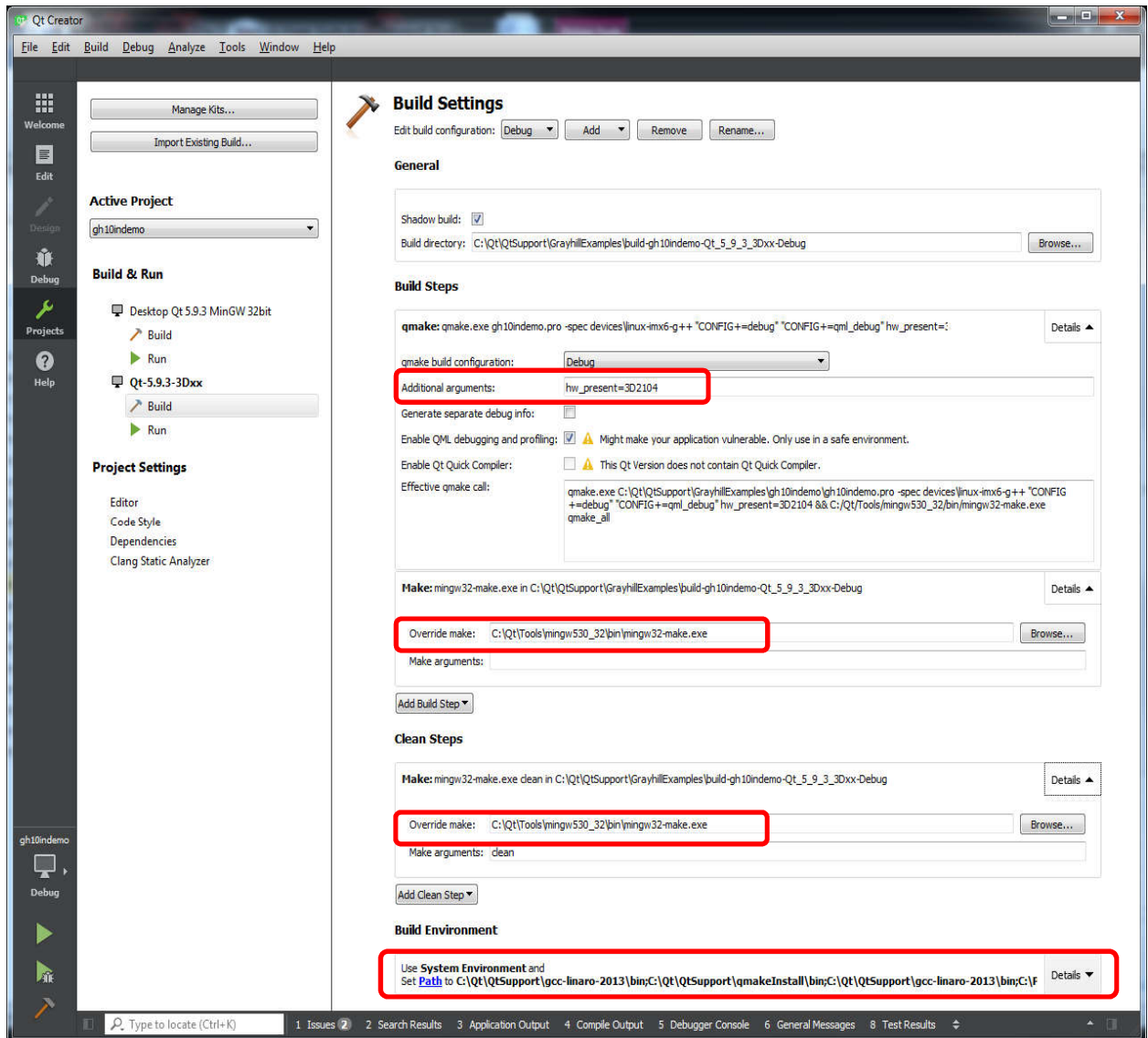
- Select “Build”
- Expand the Details tab associated with qmake (under Build Steps)
- “Additional arguments”
 - Enter “**hw_present=yes target=3D70**” – N.B. This is a **case sensitive** field.

N.B. Parameters are automatically added to the “effective qmake call” command syntax. This field is configured based on the actual target hardware display size. The processing of these arguments is in the .pro file for the project.



The above image also shows two custom steps, the first used for versioning and the second for camera configuration. Kernel 4.1.15 will support multiple camera views on the 3D70 and 3D2104. These are only supported on the Linux VM.

Windows Reference



Build Steps

- Additional arguments (see Linux screen capture above)
- Override make → C:\Qt\Tools\mingw73_32\bin\mingw32-make.exe

Clean Steps

- Override make → C:\Qt\Tools\mingw73_32\bin\mingw32-make.exe

Build Environment

- Path Append ;C:\Qt\5.12.2\mingw73_32\bin

Run/Deployment

This section describes how to compile and deploy the example project to the target (3Dxx).

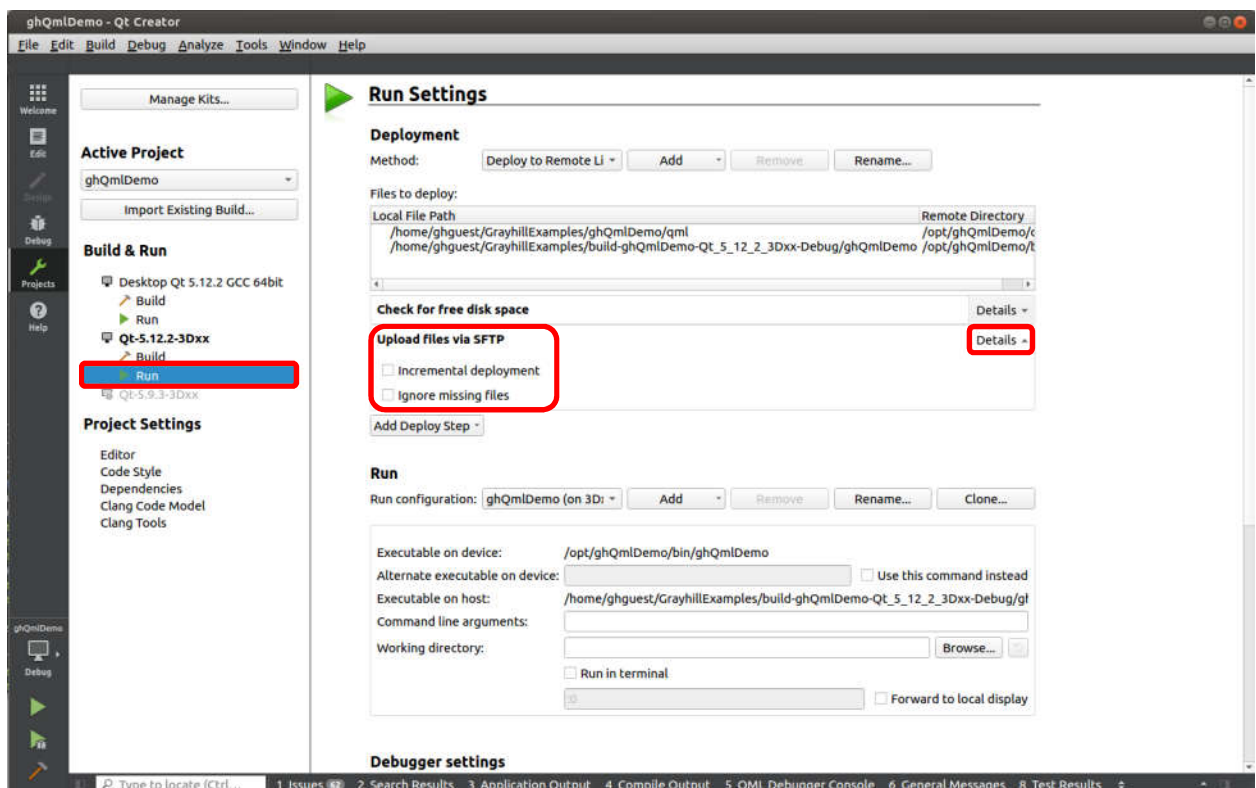
- Select “Run”
- Deployment
 - Method: Deploy to Remote Linux Host (default)
 - Files to deploy:

Local File Path	location on host	(auto-populated)
Remote Directory	location on target	(auto-populated)

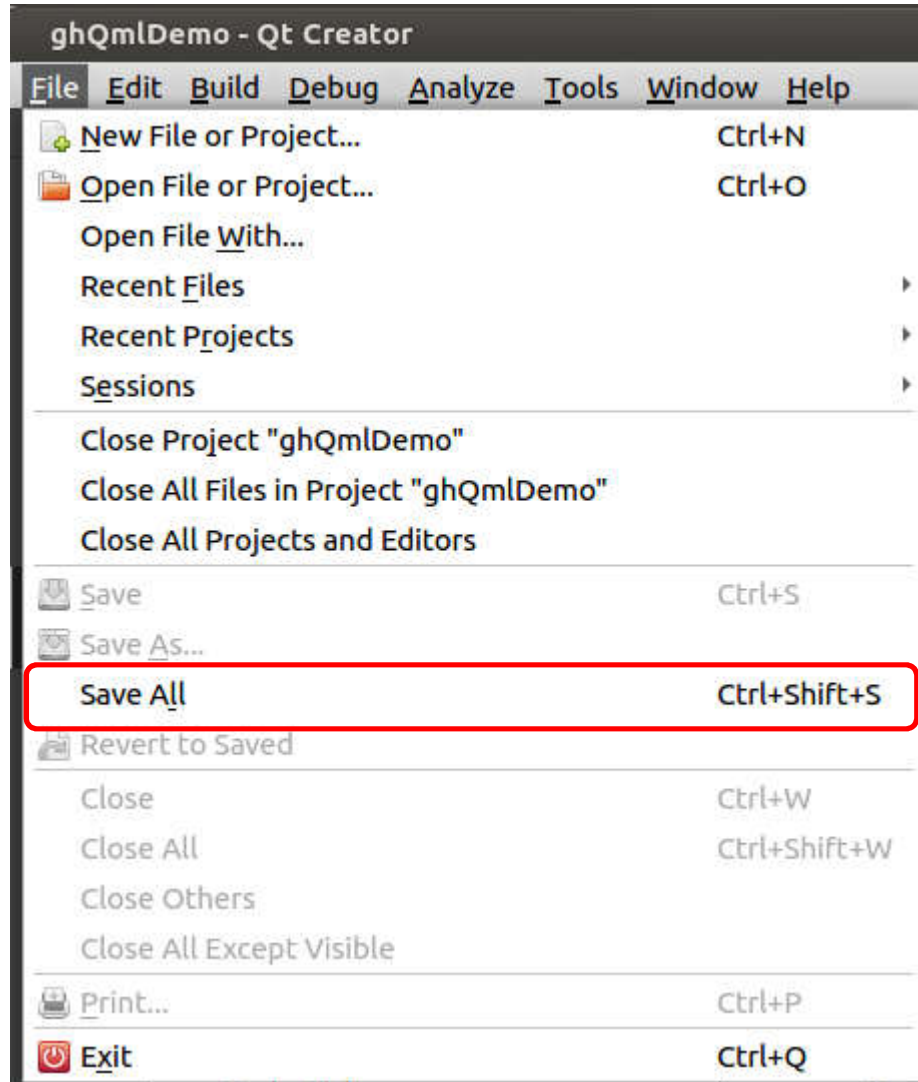
N.B. File information may not populate until after a build is done.

- Expand “Details” for “Upload files via SFTP”
- Make sure neither box is selected

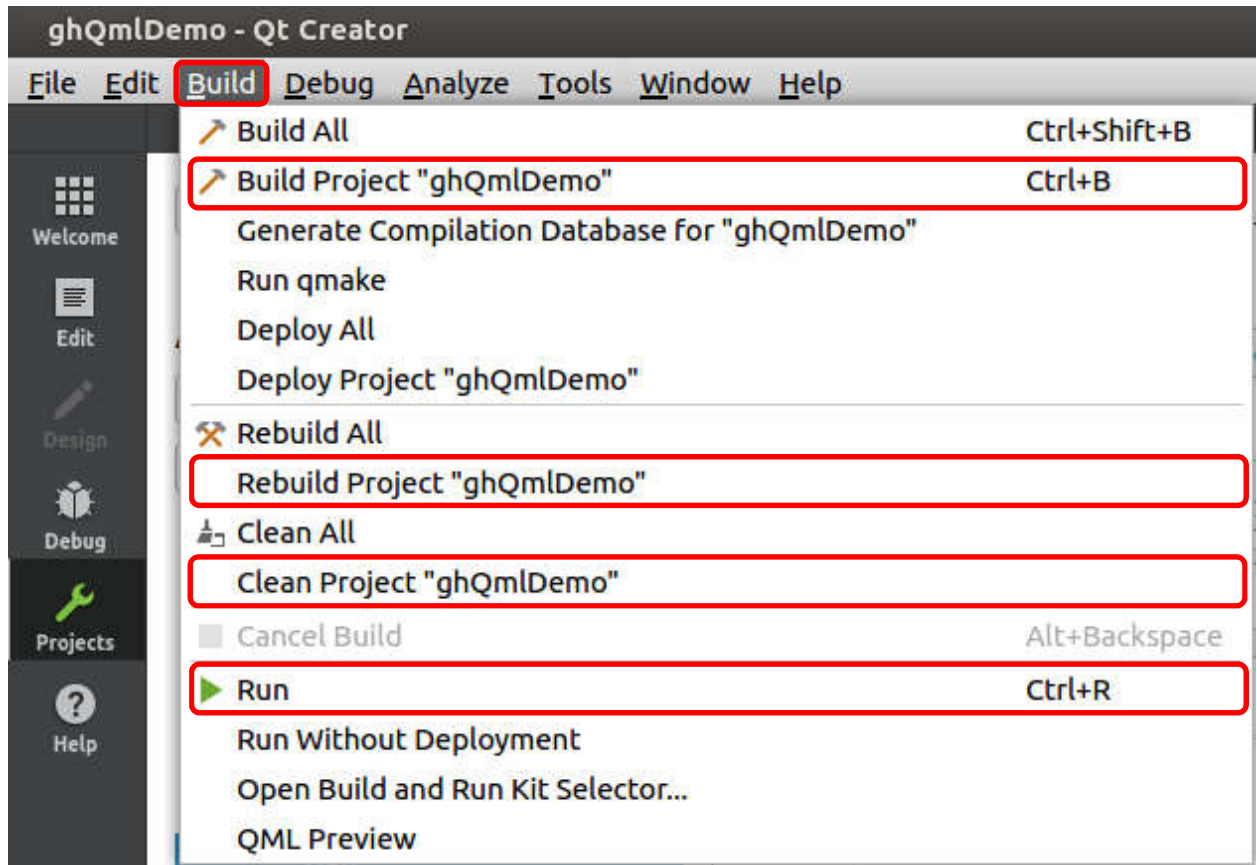
N.B. On rare occasions, Qt Creator thinks the files have already been deployed and will not re-send the files to the target; disabling this functionality avoids the situation.



- **Save!** File → Save All

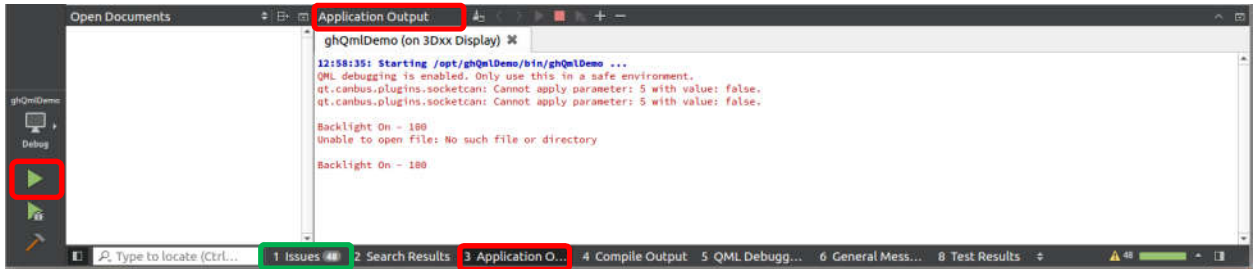


- Build options



- Build** Let Qt Creator decide what is out of date
- Rebuild** Force Qt creator to re-compile everything
- Clean** Remove existing artifacts generated by previous builds
- Run** Deploy the executable to the target and execute the image

- Build the image for the target by clicking the green triangle



The bottom ribbon of Qt Creator has various panes (views) that can be examined. “Application Output” is shown; this pane is also where qDebug messages will be output.

Click the paintbrush icon to clear the contents.

Click the red square to terminate the target session.

N.B. Errors and issues are summarized in the “Issues” tab.

Appendix C: Debugging

Let's face it; code never initially does what it is *supposed* to do; but rather what it was **told** to do! Luckily, Qt Creator has a built-in debugger!

N.B. In order to debug QML, then file(s) must be list in the QML folder. If they are not, then check to make sure qtquickcompiler is not set in the .pro file. Also verify that "Enable Qt Quick Compiler" is not checked in Build for qmake step.

Additional debugging information can be found by Googling "qt debugging" which includes the following link.

<https://doc.qt.io/qtcreator/creator-debugging.html>

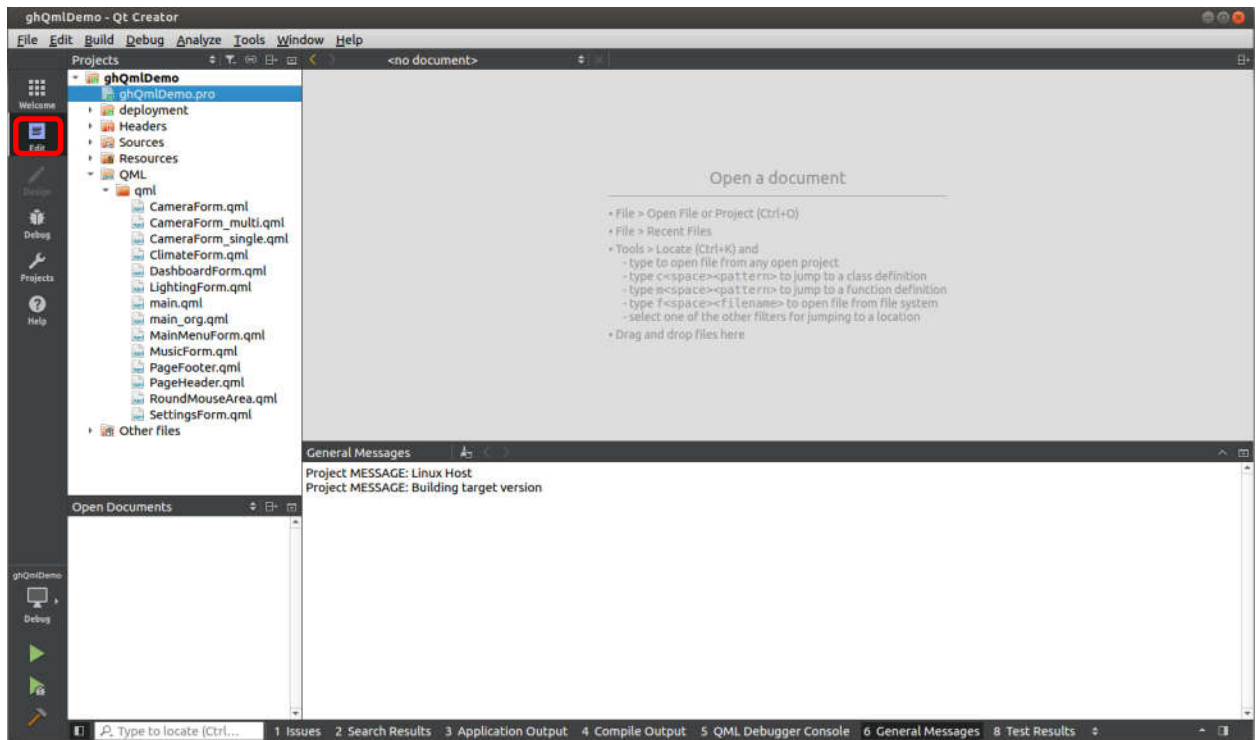
Debugger stepping option icons (Mouse over the icons for a description)



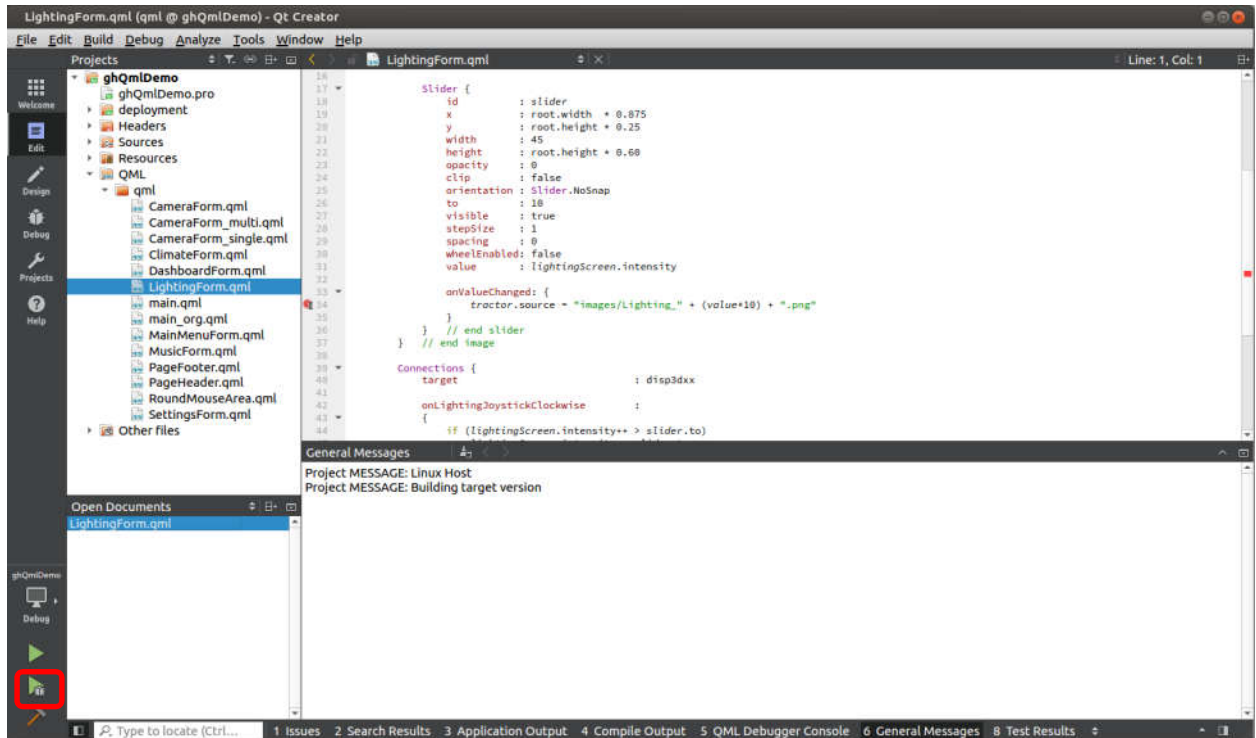
- Continue
- Stop
- Step Over <F10>
- Step In <F11>
- Step Out <Shift>+<F11>

The debugger tool bar icon's functionality are also available under the Debug drop down menu.

- Load ghQmlDemo
- Select the “Edit” view
- Expand contents of ghQmlDemo → QML → qml



- Select “LightingForm.qml”
- Add a breakpoint by left clicking in the gutter at line 34 (tractor.source = ...)

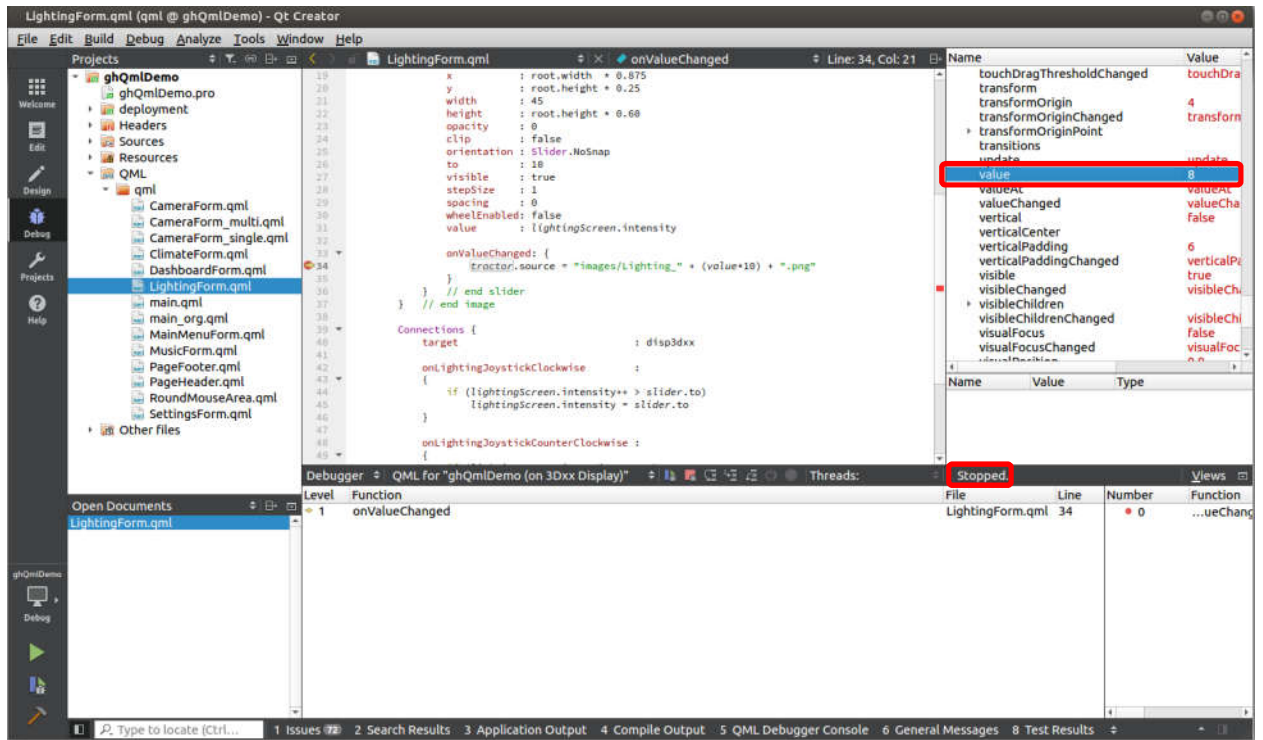


- Verify building for target and configuration parameters are set
- Click Green triangle with cute little bug!

The code begins to execute once compiled (the project may re-compile) and then hits the breakpoint. This happens during the initial loading of the form.

Press <F5> to Continue (or click on continue icon in the Debugger bar).

- Select the lighting screen
- Click on the 8th lighting level



- Execution will stop (the display will not update, as the breakpoint is at the point where the lighting level image is loaded)
- Expand “this” and scroll down to value to see the new value

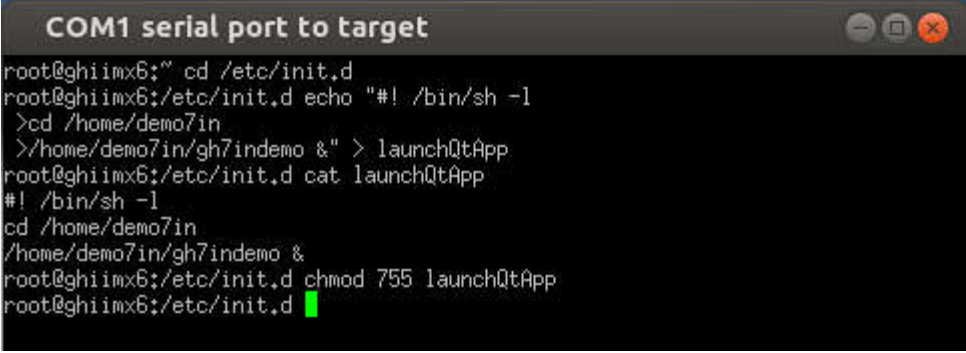
Appendix D: Setting up a 3Dxx Qt Program to Run at Boot Up

This section describes how to configure a program to automatically execute at boot up.

- Open a terminal window on the 3Dxx display (Error! Reference source not found. describes how to launch “PuTTY”)
- Create⁵ a launch script for the desired application

Explanation

<code>cd /etc/init.d</code>	<i>change into proper directory</i>
<code>echo "#! /bin/sh -l</code>	<i>treat as login (runs profile)</i>
<code>/opt/ghQmlDemo/bin/ghQmlDemo &"/> > launchQtApp</code>	<i>spawn application process</i>
<code>cat launchQtApp</code>	<i>verify contents</i>
<code>chmod 755 launchQtApp</code>	<i>make script executable</i>



```
COM1 serial port to target
root@ghiiix6:" cd /etc/init.d
root@ghiiix6:/etc/init.d echo "#! /bin/sh -l
>cd /home/demo7in
>/home/demo7in/gh7indemo &" > launchQtApp
root@ghiiix6:/etc/init.d cat launchQtApp
#! /bin/sh -l
cd /home/demo7in
/home/demo7in/gh7indemo &
root@ghiiix6:/etc/init.d chmod 755 launchQtApp
root@ghiiix6:/etc/init.d █
```

⁵ vi (text editor) can also be used for those familiar with vi, instead of the command line

-
- Create a link to the launch script created above

Explanation

cd /etc/rc.d

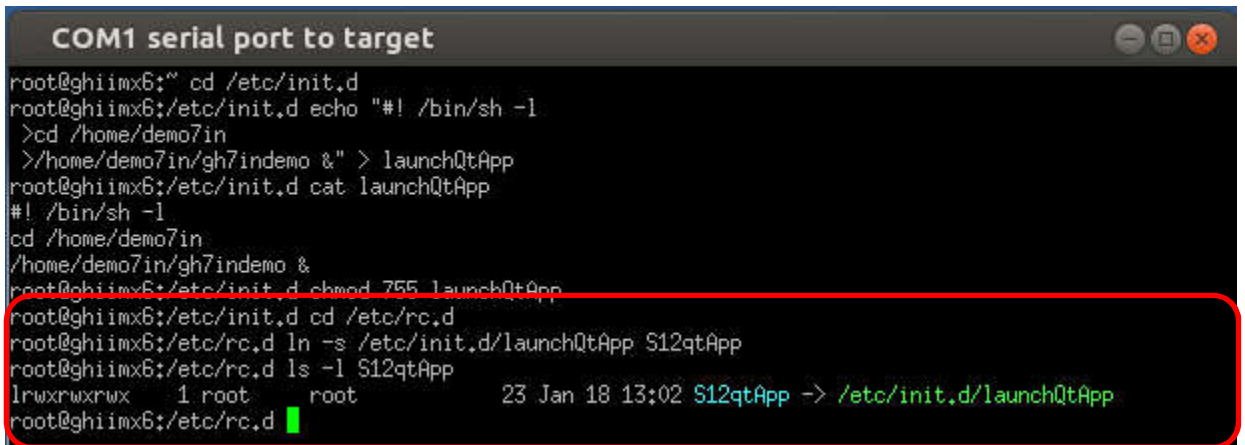
ln -s /etc/init.d/launchQtApp S12qtApp

ls -l S12qtApp

set into proper directory

create soft link to executable file

verify link creation



```
COM1 serial port to target
root@ghiiimx6:~# cd /etc/init.d
root@ghiiimx6:/etc/init.d echo "#! /bin/sh -l"
>cd /home/demo7in
>/home/demo7in/gh7indemo &" > launchQtApp
root@ghiiimx6:/etc/init.d cat launchQtApp
#!/bin/sh -l
cd /home/demo7in
/home/demo7in/gh7indemo &
root@ghiiimx6:/etc/init.d chmod 755 launchQtApp
root@ghiiimx6:/etc/init.d cd /etc/rc.d
root@ghiiimx6:/etc/rc.d ln -s /etc/init.d/launchQtApp S12qtApp
root@ghiiimx6:/etc/rc.d ls -l S12qtApp
lrwxrwxrwx  1 root  root    23 Jan 18 13:02 S12qtApp -> /etc/init.d/launchQtApp
root@ghiiimx6:/etc/rc.d █
```

N.B. Do not try to launch multiple Qt applications at boot up or try to launch the ghvehicleapp application along with a Qt application as they conflict with one another.

N.B. When switching from running one application to another, even between Qt applications, it is a good idea to do a reboot of the 3Dxx Display in between to make sure that the hardware is properly reset. This can be done by entering the “reboot” command on the 3Dxx Display Linux console.

Appendix E: Interfacing 3Dxx Hardware from QT Software

This section explains how to access the functionality of these components. The programming interfaces and provided API functions are covered, with the syntax and parameters defined. Sample code is also provided where appropriate.

The 3Dxx Display contains the following custom component interfaces:

- Analog Input driver (Model 3D70 only)
- Audio Output (Model 3D70 only)
- Buzzer (Models 3D70, 3D2104)
- Camera driver
- CAN driver
- Digital I/O driver
- LCD
- LCD Backlight

Analog Inputs (Model 3D70 only)

The Model 3D70 Display has two analog inputs. Analog Input 1 is connected to Pin 4 on Connector B and Analog Input 2 is connected to Pin 5 on Connector B. The Analog Inputs can be used to read resistance, voltage, or current with respect to the analog return pin (pin 7 on Connector B).

The following Grayhill files are required:

- ghiolib.h (header)
- libghiodrv.so (linker)

Interface

A Qt application may configure or read an analog input pin by calling the appropriate C library function as described below.

```
#define GHIOLIB_CH1 (0x01)
#define GHIOLIB_CH2 (0x02)

#define GHIOLIB_MAX_ANALOG_IN (2)

#define GHIOLIB_ANALOG_5V (0)
#define GHIOLIB_ANALOG_1500OHM (1)
#define GHIOLIB_ANALOG_10V (2)
#define GHIOLIB_ANALOG_5000OHM (3)
#define GHIOLIB_ANALOG_20MA (4)

#define GHIOLIB_RET_OK 0
#define GHIOLIB_RET_ERROR 1
#define GHIOLIB_RET_NOTSUPPORTED 2

typedef struct _ADCVALUES
{
    uint16_t adcch;
    uint16_t adcvref;
    uint16_t adcstatus;
    uint16_t adcconfig;
} ADCVALUES, *PADCVALUES;
```

ghiolib_setADCcfg

This function configures an analog input for one of five different reading modes.

Syntax

```
int ghiolib_setADCcfg(int ch, uint8_t config);
```

Type	Name	in/out	Description	Values
int	ch	in	Channel to configure	GHIOLIB_CH1 GHIOLIB_CH2
uint8_t	config	in	Channel configuration	GHIOLIB_ANALOG_5V GHIOLIB_ANALOG_10V GHIOLIB_ANALOG_1500OHM GHIOLIB_ANALOG_5000OHM GHIOLIB_ANALOG_20MA
int			Return value	GHIOLIB_RET_OK GHIOLIB_RET_ERROR GHIOLIB_RET_NOTSUPPORTED

ghiolib_getADCin

This function reads a value from an analog input pin.

Syntax

```
int ghiolib_getADCin(int ch, PADCVALUES p);
```

Type	Name	in/out	Description	Values
int	ch	in	Channel to read	GHIOLIB_CH1 GHIOLIB_CH2
PADCVALUES	P	out	Read value is returned in member "adcch" of this structure. Other items in this structure can be ignored.	
int			Return value	GHIOLIB_RET_OK GHIOLIB_RET_ERROR GHIOLIB_RET_NOTSUPPORTED

Sample Qt Code

```
#include <QDebug>

// For access to ghiolib
typedef u_int16_t uint16_t;
typedef u_int8_t uint8_t;
#ifdef __cplusplus
extern "C" {
#endif

#include "ghiolib.h"

#ifdef __cplusplus
}
#endif

int         channel = 0;
ADCVALUES  analogData;
int         gpioStatus;

// Set analog input 1 to read 0 to 10 volts
gpioStatus = ghiolib_setADCcfg(channel + 1, GHIOLIB_ANALOG_10V);
if (GHIOLIB_RET_OK != gpioStatus)
{
    qDebug("ERROR (%d) doing ghiolib_setADCcfg on channel: %d\n",
           gpioStatus, channel + 1);
}

// Get current reading
gpioStatus = ghiolib_getADCin(channel + 1, &analogData);
if (GHIOLIB_RET_OK != gpioStatus)
{
    qDebug("ERROR (%d) doing ghiolib_getDigOut on channel: %d\n",
           gpioStatus, channel + 1);
}
qDebug("Reading from channel %d is %d millivolts\n", channel + 1,
       analogData.adcch);
```

Audio Output (Model 3D70 only)

The Model 3D70 Display has the ability to play an mp3 audio file and send the audio output to a monaural line out (pins 1, AUDIO OUT, and 2, AUDIO RET, on the B connector).

There are no required header or linker files, but the mpg123 application must be installed on the display and its location be in the search PATH.

Interface

A Qt application can start playing an mp3 audio file and can stop the playing of the audio file using a Linux utility called mpg123.

Sample Qt Code

```
//  
// To play mp3 file "sounds.mp3"  
//  
// Note that by placing mp3 file in "images" folder, Qt will  
// automatically download the mp3 file to the target with the  
// other image files being used.  
//  
// Command shown to play mp3 file will first stop playing any mp3  
// file that may already be playing.  
//  
system("test `pidof mpg123` && kill `pidof mpg123` ;"  
       "mpg123 -q images/sounds.mp3 &");  
  
// To stop playing mp3 file (if any)  
system("test `pidof mpg123` && kill `pidof mpg123`");
```

Buzzer (Models 3D70, 3D2104)

The Model 3D70 and 3D2104 Displays have an internal buzzer that can be sounded on command.

There are no additional required files.

Interface

A Qt application can turn the internal buzzer on or off by sending the proper number to the buzzer control file.

Sample Qt Code

```
#include <QString>
#include <QDebug>

QFile      buzzerFile;
bool       buzzerFileOpen;

buzzerFile.setFileName("/sys/class/backlight/pwm-
backlight.3/brightness");
buzzerFileOpen = buzzerFile.open(QIODevice::WriteOnly |
QIODevice::Text);

if (false == buzzerFileOpen)
{
    qDebug("Error opening buzzer file\n");
}

// To turn buzzer ON
if (true == buzzerFileOpen)
{
    QTextStream buzzerOut (&buzzerFile);
    buzzerOut << 10;
}

// . . .

// To turn buzzer OFF
if (true == buzzerFileOpen)
{
    QTextStream buzzerOut (&buzzerFile);
    buzzerOut << 0;
}
```

Camera Driver Interface

The Grayhill 3Dxx Display device can contain multiple camera inputs. NTSC and PAL format video inputs are supported by modifying the camera input sensor parameters. The camera output can be displayed on the LCD. The following camera display parameters can be modified:

- Window parameters – window size and window position
- Color parameters – brightness, contrast, saturation and hue
- Rotation
- Input sensor parameters – provides support for NTSC and PAL formats
- Camera output to LCD foreground or background with color key

Camera output is 30 frames per second (fps).

The following Grayhill files are required:

- libghiodrv.so (linker)

N.B. Only one camera input can be active at a time with kernel 3.0.35. Kernel 4.1.15 will support multiple camera views on the 3D2104 and 3D70.

Data Types

```
typedef struct _SENSORPARAMS // Must be set according to camera
input type
{
    // NTSC    PAL
    unsigned int top;        // 4    5
    unsigned int left;       // 0    4
    unsigned int height;     // 480  567
    unsigned int width;      // 640  640
} SENSORPARAMS, *PSENSORPARAMS;

#define FOREGROUND (1)
#define BACKGROUND (0)

#define FB_DEV_0 (0) // GRAPHICS being sent to /dev/fb0
#define FB_DEV_1 (1) // GRAPHICS being sent to /dev/fb1

//
// These are the only allowed values for VIDEO_COLOR_KEY_xxx
//
#define VIDEO_COLOR_KEY_BLACK (0x00000000)
#define VIDEO_COLOR_KEY_RED (0x00FF0000)
#define VIDEO_COLOR_KEY_GREEN (0x0000FF00)
#define VIDEO_COLOR_KEY_BLUE (0x000000FF)
#define VIDEO_COLOR_KEY_YELLOW (0x00FFFF00)
#define VIDEO_COLOR_KEY_CYAN (0x0000FFFF)
#define VIDEO_COLOR_KEY_MAGENTA (0x00FF00FF)
#define VIDEO_COLOR_KEY_WHITE (0x00FFFFFF)

typedef struct _DISPLAYPARAMS
{
    unsigned int top; // top left window y-coordinate
    unsigned int left; // top left window x-coordinate
    // (must be divisible by 4)
    unsigned int height; // window vertical size
    unsigned int width; // window horizontal size
    // NOTE: top + height must not exceed
    // height of display
    // and left + width must not exceed
    // display width
    unsigned int rotate; // 0-7, see below
    unsigned int fg; // FOREGROUND or BACKGROUND +
    // VIDEO_COLOR_KEY_xxx
} DISPLAYPARAMS, *PDISPLAYPARAMS;
```

The camera output always operates in native landscape mode. Use the following rotation values to support other display and camera orientations:

Value	Rotation
0	No rotation
1	Vertical flip
2	Horizontal flip
3	180
4	90 right
5	90 right with vertical flip
6	90 right with horizontal flip
7	90 left

```
#define HUE_CODE_00      (0x00)
#define HUE_CODE_7F     (0x7F)
#define HUE_CODE_80     (0x80)

typedef struct _COLORPARAMS
{
    unsigned int brightness; // 0-255
    unsigned int saturation; // 0-255
    unsigned int hue;        // HUE_CODE_00, HUE_CODE_7F, or
                            // HUE_CODE_80
    unsigned int contrast;  // 0-255
} COLORPARAMS, *PCOLORPARAMS;
```

Interface

The Qt application interfaces with the Camera driver using the Camera class.

Camera::Camera

Camera class constructor

Syntax

```
Camera::Camera (int camnum, int fbdev = FB_DEV_0);
```

Type	Name	in/out	Description	Values
int	camnum	in	Camera Number	3D50 : 1-2 3D70 : 1-3 3D2104: 1-4
int	fbdev	in	Frame buffer device	See below

The "fbdev" value indicates whether the GRAPHICS are being sent to fb0 or fb1.

When GRAPHICS are being sent to fb0, then video will be sent to fb1 and only foreground mode is allowed. This is the default assumed if "fbdev" is missing.

If GRAPHICS are being sent to fb1, then video will be sent to fb0 and both foreground and background modes are supported. In order to send GRAPHICS to fb1, add this parameter to the command line that launches Qt: `-display LinuxFb:/dev/fb1`

Camera::setdisplayparams

Sets the following display window parameters

- origin
- window size
- rotation
- foreground or background with color key (When using background mode the camera video only shows through where the graphics data is set to the color that matches the specified color key. Graphics of any other color will appear on top of the camera video image.)

Syntax

```
int Camera::setdisplayparams(PDISPLAYPARAMS p);
```

Type	Name	in/out	Description	Values
PDISPLAYPARAMS	p	in	refer to DISPLAYPARAMS structure	
int			Return value	0 – success -1 - failure

Camera::setcolorparams

Sets the following camera color parameters

- Brightness
- Saturation
- Contrast
- Hue

Syntax

```
int Camera::setcolorparams(PCOLORPARAMS p);
```

Type	Name	in/out	Description	Values
PCOLORPARAMS	p	in	refer to COLORPARAMS structure	
int			Return value	0 – success -1 - failure

Camera::setsensorparams

Sets the camera sensor parameters

Syntax

```
int Camera::setsensorparams(PSENSORPARAMS psensor);
```

Type	Name	in/out	Description	Values
PSENSORPARAMS	psensor	in	refer to SENSORPARAMS structure	
int			Return value	0

Camera::show

Enables or disables the camera

Syntax

```
int Camera::show(int enable);
```

Type	Name	in/out	Description	Values
int	enable	in	Turn on/off camera	0 – disable 1 - enable
int			Return value	0 – success -1 - failure

Sample Code

```
#include "camera.h"

COLORPARAMS color;
DISPLAYPARAMS disp;
int cameranum = 1; // camera input 1

Camera cam(cameranum);

disp.top      = 0;
disp.left     = 80;
disp.height   = 480;
disp.width    = 640;
disp.rotate   = 4; // rotate 90 degree right
disp.fg       = FOREGROUND;
// configure display parameters
cam.setdisplayparams(&disp);

// start camera
cam.show(1);

// change color parameters
color.brightness = 50;
color.saturation = 128;
color.contrast   = 128;
color.hue        = 0;
// configure color parameters
cam.setcolorparams(&color);

....

// stop 1+camera
cam.show(0);
```

CAN Driver Interface

The 3D50 and 3D70 displays includes two CAN controller modules. Available CAN ports are CAN1 and CAN2. The 3D2104 display includes three CAN controller modules. Available CAN ports are CAN1, CAN2, and CAN3. The CAN controller supports both standard and extended frames.

The following Grayhill files are required:

- libghiodrv.so (linker)

Data Types

```
/*
 * special flag bits for the CAN_ID
 */
#define CAN_EFF_FLAG 0x80000000U /* EFF flag (add to ID to
                                activate 29-bit ID) */
#define CAN_RTR_FLAG 0x40000000U /* remote transmission request */
#define CAN_ERR_FLAG 0x20000000U /* error frame */

struct _CANMSG
{
    unsigned int ID;
    unsigned int Length; // Data Length Code of the Msg (0..8)
    unsigned char Data[8];
};
typedef struct _CANMSG CANMSG, *PCANMSG;
```

Interface

The Qt application interfaces with the CAN bus driver using the CAN class.

CAN::CAN

CAN class constructor

Syntax

```
CAN::CAN(int num);
```

Type	Name	in/out	Description	Values
int	num	in	CAN port number	3D50 : 1-2 3D70 : 1-2 3D2104: 1-4

CAN::OpenPort

Opens the CAN socket

Syntax

```
int CAN::OpenPort(void);
```

Type	Name	in/out	Description	Values
int			Return value	0 – success -1 - failure

CAN::WritePort

Writes a single CAN frame to the CAN port.

Syntax

```
int CAN::WritePort(PCANMSG TxMsg);
```

Type	Name	in/out	Description	Values
PCANMSG	TxMsg	in	CAN frame to be written	
int			Return value	0 – success -1 - failure

CAN::ReadPort

Attempts to read a single CAN frame from the CAN port. Note that the CAN socket is configured to be non-blocking, so calls to ReadPort will return even if there is no data.

Syntax

```
int CAN::ReadPort(PCANMSG RxMsg);
```

Type	Name	in/out	Description	Values
PCANMSG	RxMsg	in	CAN frame received	
int			Return value	Bytes read or (-1) for failure

CAN::ClosePort

Closes the CAN socket

Syntax

```
void CAN::ClosePort(void);
```

Sample Code

```
#include "can.h"

CANMSG TxMsg;
CANMSG RxMsg;
int bytesread = 0;
int cannum = 1;    // CAN1

/* Init TX and RX message */
TxMsg.ID = 0x23;
TxMsg.Length = 8;
for (int i=0; i<8; i++)
    TxMsg.Data[i] = (0x11 * (i+1));    // fill random data
memset((void *)&RxMsg, 0, sizeof(CANMSG));

// CAN1
CAN can(cannum);
can.OpenPort();
can.WritePort(&TxMsg);
do
{
    bytesread = can.ReadPort(&RxMsg);
    // add delay
} while (bytesread != sizeof(CANMSG));
can.ClosePort();
```

Digital I/O Driver Interface

The Model 3D50 Display, Model 3D70 Display, and Model 3D2104 Display each have four digital inputs and four digital outputs, but they are configured differently and these differences will be explained. Each device uses the same library calls to read the digital inputs and set the digital outputs.

On the 3D50 Five Inch Display Pin 4 on its connector is a dedicated input only pin. Pin 5 is a dedicated output only pin. Pins 6, 7, and 8 are shared I/O pins that can be used to output a signal or input a signal.

On the Model 3D70 Seven Inch Display each of the four inputs are dedicated and so operate independently of any output pins.

On the Model 3D2104 10.4 Inch Display all digital output pins are shared I/O pins that can be used to output a signal or input a signal.

For a shared I/O pin to function as an input, the corresponding output must be set low.

The following table summarizes all of the digital I/O pins for each model:

Model 3D50 Pins	Model 3D70 Pins	Model 3D2104 Pins
Input 1 (Pin 4)	Input 1 (Pin 4 Connector A)	Input 1 or Output 1 (Pin 10)
Input 2 or Output 2 (Pin 6)	Input 2 (Pin 8 Connector B)	Input 2 or Output 2 (Pin 21)
Input 3 or Output 3 (Pin 7)	Input 3 (Pin 9 Connector B)	Input 3 or Output 3 (Pin 32)
Input 4 or Output 4 (Pin 8)	Input 4 (Pin 10 Connector B)	Input 4 or Output 4 (Pin 9)
Output 1 (Pin 5)	Output 1 (Pin11 Connector B)	
	Output 2 (Pin12 Connector B)	
	Output 3 (Pin13 Connector B)	
	Output 4 (Pin14 Connector B)	

The following Grayhill files are required:

- ghillib.h (header)
- libghiodrv.so (linker)

Interface

A Qt application may set or get the digital I/O pin states by calling the appropriate C library function as described below.

```
#define GHIOLIB_CH1 (0x01)
#define GHIOLIB_CH2 (0x02)
#define GHIOLIB_CH3 (0x03)
#define GHIOLIB_CH4 (0x04)

#define GHIOLIB_MAX_DIGITAL_IO (4)
#define GHIOLIB_DIG_IN_FLOAT (0)
#define GHIOLIB_DIG_IN_PULL_DN (1)
#define GHIOLIB_DIG_IN_PULL_UP (2)

#define GHIOLIB_RET_OK 0
#define GHIOLIB_RET_ERROR 1
#define GHIOLIB_RET_NOTSUPPORTED 2
```

ghiolib_setDigIncfg (Model 3D70 only)

Sets input pin pull-up/pull-down configuration.

Syntax

```
int ghiolib_setDigIncfg(int ch, uint8_t config);
```

Type	Name	in/out	Description	Values
int	ch	in	Input pin to configure	GHIOLIB_CH1 GHIOLIB_CH2 GHIOLIB_CH3 GHIOLIB_CH4
uint8_t	config	in	pin configuration	GHIOLIB_DIG_IN_FLOAT GHIOLIB_DIG_IN_PULL_DN GHIOLIB_DIG_IN_PULL_UP
int			Return value	GHIOLIB_RET_OK GHIOLIB_RET_ERROR GHIOLIB_RET_NOTSUPPORT ED

ghiolib_getDigIn

This function reads the state of an input pin.

Syntax

```
int ghiolib_getDigIn(int ch, uint8_t *value);
```

Type	Name	in/out	Description	Values
int	ch	in	Input pin to read	GHIOLIB_CH1 GHIOLIB_CH2 GHIOLIB_CH3 GHIOLIB_CH4
uint8_t	value	out	pin status	0 – input low 1 – input not low
int			Return value	GHIOLIB_RET_OK GHIOLIB_RET_ERROR GHIOLIB_RET_NOTSUPPORT ED

ghiolib_getDigOut

Reads the current state of an output pin.

Syntax

```
int ghiolib_getDigOut(int ch, uint8_t *value);
```

Type	Name	in/out	Description	Values
int	ch	in	Output pin to read	GHIOLIB_CH1 GHIOLIB_CH2 GHIOLIB_CH3 GHIOLIB_CH4
uint8_t	value	out	pin status	0 – output low 1 – output not low
int			Return value	GHIOLIB_RET_OK GHIOLIB_RET_ERROR GHIOLIB_RET_NOTSUPPORT ED

ghiolib_setDigOut

This function sets the current state of an output pin.

Syntax

```
int ghiolib_setDigOut(int ch, uint8_t value);
```

Type	Name	in/out	Description	Values
int	ch	in	Output pin to set	GHIOLIB_CH1 GHIOLIB_CH2 GHIOLIB_CH3 GHIOLIB_CH4
uint8_t	value	in	pin status	0 – set pin output low !0 – set pin output high
int			Return value	GHIOLIB_RET_OK GHIOLIB_RET_ERROR GHIOLIB_RET_NOTSUPPORT ED

Sample Qt Code

```
#include <QDebug>

// For access to ghiolib
typedef u_int16_t uint16_t;
typedef u_int8_t uint8_t;

#ifdef __cplusplus
extern "C" {
#endif

#include "ghiolib.h"

#ifdef __cplusplus
}
#endif

int channel;
uint8_t digValue;
int gpioOutput;
int gpioInput;
int gpioStatus;

// Set inputs to pull down mode and read current inputs and outputs
// for each channel
gpioOutput = 0;
gpioInput = 0;
for (channel = 0; channel < GHIOLIB_MAX_DIGITAL_IO; channel++)
{
    // Set input to pull down mode
```

```

        gpioStatus = ghiolib_setDigIncfg(channel + 1,
GHIOLIB_DIG_IN_PULL_DN);
        if ((GHIOLIB_RET_OK != gpioStatus) && (GHIOLIB_RET_NOTSUPPORTED
!= gpioStatus))
        {
            qDebug("ERROR (%d) doing ghiolib_setDigIncfg on channel:
%d\n",
                gpioStatus, channel + 1);
        }

        // Read current output setting
        digValue = 0;
        gpioStatus = ghiolib_getDigOut(channel + 1, &digValue);
        if (GHIOLIB_RET_OK != gpioStatus)
        {
            qDebug("ERROR (%d) doing ghiolib_getDigOut on channel:
%d\n",
                gpioStatus, channel + 1);
        }
        else
        {
            if (1 == digValue)
            {
                gpioOutput |= (1 << channel);
            }
        }

        // Read current input
        digValue = 0;
        gpioStatus = ghiolib_getDigIn(channel + 1, &digValue);
        if (GHIOLIB_RET_OK != gpioStatus)
        {
            qDebug("ERROR (%d) doing ghiolib_getDigIn on channel:
%d\n",
                gpioStatus, channel + 1);
        }
        else
        {
            if (1 == digValue)
            {
                gpioInput |= (1 << channel);
            }
        }
    }
    qDebug("GPIO initial output: 0x%x input: 0x%x\n", gpioOutput,
gpioInput);

```

LCD

The Grayhill 3Dxx Series Display uses a 16 bit per pixel LCD screen. The pixel dimensions of various 3Dxx Display products are shown in the section [Error! Reference source not found.](#) The default orientation of the frame buffer is landscape mode (wider pixel dimension is in horizontal direction).

LCD Backlight

The LCD Backlight setting is a value between 0 (minimum) and 100 (maximum) inclusive. The brightness value can be set in the file ***/sys/class/backlight/pwm-backlight.0/brightness***

Sample Code

```
int value = 80;
QFile file("/sys/class/backlight/pwm-backlight.0/brightness");
if (file.open(QIODevice::WriteOnly | QIODevice::Text))
{
    QTextStream out(&file);
    out << value;
    file.close();
}
```

Appendix F: Setting 3Dxx Flash File System R/W Mode

To immediately configure the 3Dxx Display file system for read-write mode enter the following command in a terminal window on the display:

- `mount -o remount,rw /`

The above command only remains in effect until the next reboot.

The 3Dxx installation script utilizes the following technique to configure the 3Dxx Display file system to be in read-write mode to make Qt development more convenient.

The above command is placed in a script file which is invoked during display initialization. This file `/home/writeablefs` is called via the following link:

- `ln -s /home/writeablefs /etc/rc.d/S03writeablefs`

To leave the display in read-only mode:

- `mv /home/writeablefs /home/writeablefs.bak`

Alternatively, to set the 3Dxx display file system to read-write mode on boot-up, edit the file `/etc/init.d/rc-once` and add the above command to the end of this file just before the final “exit” command like this:

```
...
case "$1" in
    start)
        do_start >&2
        ;;
    *)
        echo "Usage: $0 {start}" >&2
        exit 1
        ;;
esac

mount -o remount,rw /

exit 0
```

To leave the 3Dxx Display file system set to read-only mode on boot-up, edit the file `/etc/init.d/rc-once` and remove the “`mount -o remount,rw /`” line near the end of the file (or comment it out by putting a “`#`” in column one of that line).

Appendix G: Building Qt Library Source (optional)

N.B. This appendix is included for reference and is not a required step.

N.B. Library building is supported only under Linux.

This section describes the procedure to download and build the Qt library source code.

Please reference <http://doc.qt.io/qt-5/windows-requirements.html> for additional information.

This procedure relies on Qt Creator and the Grayhill support files being installed.

Grayhill provides an archive script with the following files:

- mkLibs script used to build the libraries
- qmake.conf Grayhill modified configuration file
- qt_configure.prf Grayhill modified rule file
- qtLibSrcInstall Qt library installation script
- updInc fix incorrectly generated makefile
- Download the Qt library installation archive from Grayhill
 - qtLibSrc5122.tgz
- Copy/move the downloaded file to /home/ghguest
- Un archive
 - tar xf qtLibSrc5122.tgz
- Run installer (this fetches the library source from Qt)
 - ./qtLibSrcInstall
- Build Libs
 - Follow instructions provided by installation script; mkLibs provides the root password as needed

N.B. To reduce build time, increase the VM's processor count. This needs to be done in a VM powered off state.

Appendix H: Dynamic IP Address

To find the 3Dxx display's Ethernet IP address, issue the following command in a terminal window on the display

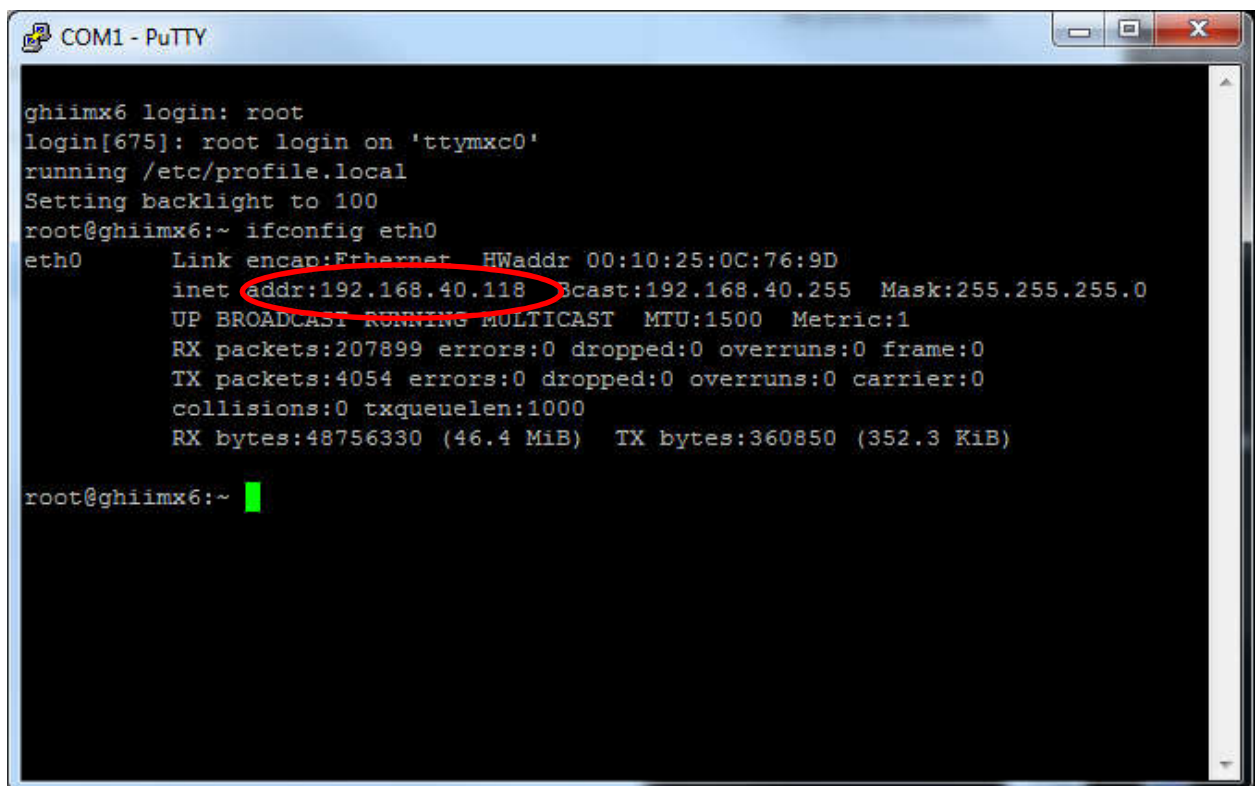
- **ifconfig eth0**

The IP address of the 3Dxx Display is displayed after the tag "inet addr:" and is circled in **red** in the example output shown below.

If the tag "inet addr:" is not present; enter these commands and try the "ifconfig eth0" command again

- **ifdown eth0**
- **ifup eth0**

Please make a note of the IP address, in this example the IP address is 192.168.40.118



```
COM1 - PuTTY
ghiimx6 login: root
login[675]: root login on 'ttyMXC0'
running /etc/profile.local
Setting backlight to 100
root@ghiimx6:~ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:10:25:0C:76:9D
          inet addr:192.168.40.118  Bcast:192.168.40.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:207899 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4054 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:48756330 (46.4 MiB)  TX bytes:360850 (352.3 KiB)

root@ghiimx6:~ █
```

Appendix I: Static IP Address

If using a **static** IP address for the display, once the address is determined:

- `cp /etc/network/interfaces /etc/network/interfaces.bak`
- `vi /etc/network/interfaces`
- replace lines

```
iface eth0 inet dhcp
    udhcpc_opts -t 5 -T 3 -A 20 -S &
```

- with

```
iface eth0 inet static
    address 192.168.40.118
    netmask 255.255.255.0
```

Google “linux interface file” for additional information.