An ISO-9001 Company

Grayhill 3Dxx Display Products

Setup and Usage of Qt Development Software – Linux

Revision History

Revision	Date	Description
A	09/19/2014	Original Release
B	09/25/2014	Many Refinements.
C	12/11/2015	Added note about hardware needed to support Ethernet interface.
D	04/27/2016	• Added alpha blending and PAL support information to camera interface section • Changed instructions to utilize VirtualBox disk image with Qt pre-installed.
E	11/11/2016	Added support for Model 3D70 (Seven Inch Display)
F	04/04/2017	Added support for Qt-5.6.2
G	05/17/2017	Corrected some errors
H	08/30/2017	Added support for Model 3D2104 (10.4 Inch Display)

Table of Contents

Introduction

This Usage Guide describes how to setup and use a Qt-based development environment for a Grayhill 3Dxx Display product. This Qt cross-platform development environment runs under Linux, but the Linux system is hosted on a Windows 7 PC via the Oracle VirtualBox software. This document also describes how to develop code for a 3Dxx Display product in the Qt IDE, how to access various 3Dxx hardware features via this code, and how to load the developed application code onto a 3Dxx Display product.

Two versions of Qt are supported: Qt-4.8.6 which uses a frame buffer interface and Qt-5.6.2 which uses EGFLS and can support QtQuick and QML. The Qt-4.8.6 environment uses QWS command line parameters that can rotate the display by 0, 90, 180, or 270 degrees. The Qt-5.6.2 version does not use QWS and can only support display output in the native landscape mode.

Currently the following 3Dxx Display models are supported on this Qt-based development environment:
- Model 3D50 Five Inch Display
- Model 3D70 Seven Inch Display
- Model 3D2104 10.4 Inch Display

The different features of these displays will be described below and differences in the installation process for each model will be explained.

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

Supported Hardware Products

The Qt-based development environment is supported on the following Grayhill 3Dxx Color Display Models:

- 3D50 Five Inch Color Display (800 x 480 pixels and up to two CAN buses)
- 3D70 Seven Inch Color Display (800 x 480 pixels and up to two CAN buses)
- 3D2104 10.4 Inch Color Display (1024 x 768 pixels and up to three CAN buses)

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**	**3D70**	**3D2104**
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

Model Number	3D50	3D70	3D2104
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

Recommended Equipment from Grayhill

If using Model 3D50 Five Inch Display:

3D50DEV-100	3D50 Development Kit

If using Model 3D70 Seven Inch Display:

3D70DEV-100	3D70 Development Kit

If using Model 3D2104 Display:

3D2104DEV-100	3D2104 Development Kit

Other Recommended Equipment

Other Recommended Equipment (for all displays)

- o An Ethernet port connected to a DHCP server that can be connected to the 3Dxx Display. This port should be on the same network as the development PC.
- o PC Running Windows XP* or Windows 7 with the following features:
  - 3 GB RAM (minimum)
  - 250 GB available hard drive space (minimum)
  - Ethernet port
  - RS232 Port
  - Internet Access
    * Note: if using Windows XP make sure that exFAT file system support has been installed in order to read distribution media (http://www.microsoft.com/en-us/download/details.aspx?id=19364)

Software Required

3Dxx Qt Software Kit (that is applicable for both models) which includes:

- o Grayhill 3Dxx Linux Qt Rev G.ova
- o 3Dxx_Qt_Installation_RevH.zip which includes:
  - ▪ VMSharedFolder
  - ▪ 3Dxx_Qt_Usage_Guide.pdf (this document)
  - ▪ 3D50DEV Quick Start Guide.pdf
  - ▪ 3D70DEV Quick Start Guide.pdf
  - ▪ Qt User Notes.pdf

The files “Grayhill 3Dxx Linux Qt Rev G.ova” and “3Dxx_Qt_Installation_RevH.zip” can be downloaded from the Grayhill website at:

http://www.grayhill.com/qt43d/

Installation Overview

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

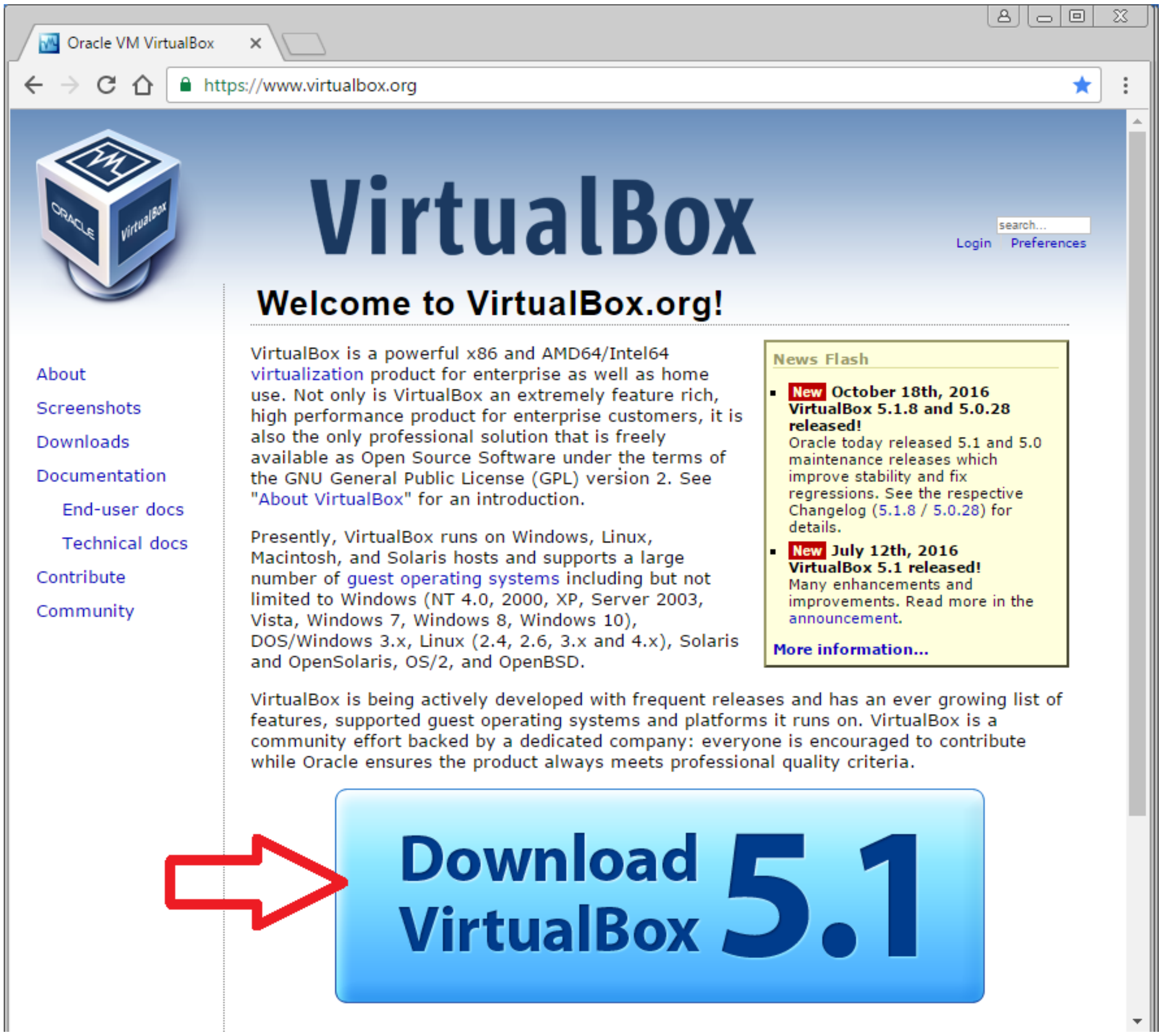
- First connect the 3Dxx Development Kit hardware to the PC being used. This includes connecting the serial port and Ethernet port interfaces. For the 3D50 Display this procedure is described in detail in the document “3D50DEV Quick Start Guide.pdf” and for the model 3D70 Display it is described in the document “3D70DEV Quick Start Guide.pdf”.
- Next the VirtualBox application will be downloaded from the internet and installed on the development PC. This application allows one to run a virtual computer system in a window on the PC. This means that all other Windows PC applications can be running along with this virtual computer application. This virtual computer will be used to run a version of Linux. All Qt-based development will be done under this Linux environment.
- Next the VirtualBox application will be configured. The only thing that the user must configure is the serial port interface, but this procedure will be explained. Grayhill provides a pre-configured image of Linux that has the Qt development environment already installed. (There is information in the document *Qt User Notes.pdf* that explains how the Qt development software was downloaded and installed but this information is included for reference only.)
- Then a Linux script will be run to update some items in the Qt development environment. This script will only need to be run once.
- The serial and Ethernet links to the target 3Dxx Display hardware will be established. Another Linux script will be run to configure the actual 3Dxx Display product to operate with Qt instead of VUI Builder©. This second script will need to be run on each 3Dxx Display product that will be operated with Qt.
- Finally instructions are provided on how to open and run a Qt demonstration project in either the Linux desktop environment or on the 3Dxx Display target hardware. This demonstration project shows how to use touch screen “buttons”, how to use touch screen swipes, how to set the 3Dxx backlight, how to operate the 3Dxx camera input, and how to access and set the real time clock. For the 3D70 Display there are also samples of using the audio output, the analog input, and the internal buzzer.

Installation of VirtualBox

This section shows how to download and install VirtualBox Version 5.1.8. If a newer version is available, it will probably work just fine, but the screen shots shown below may be different.

- Navigate to this web page: https://www.virtualbox.org/

- Click on “Download VirtualBox …” selection.



Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

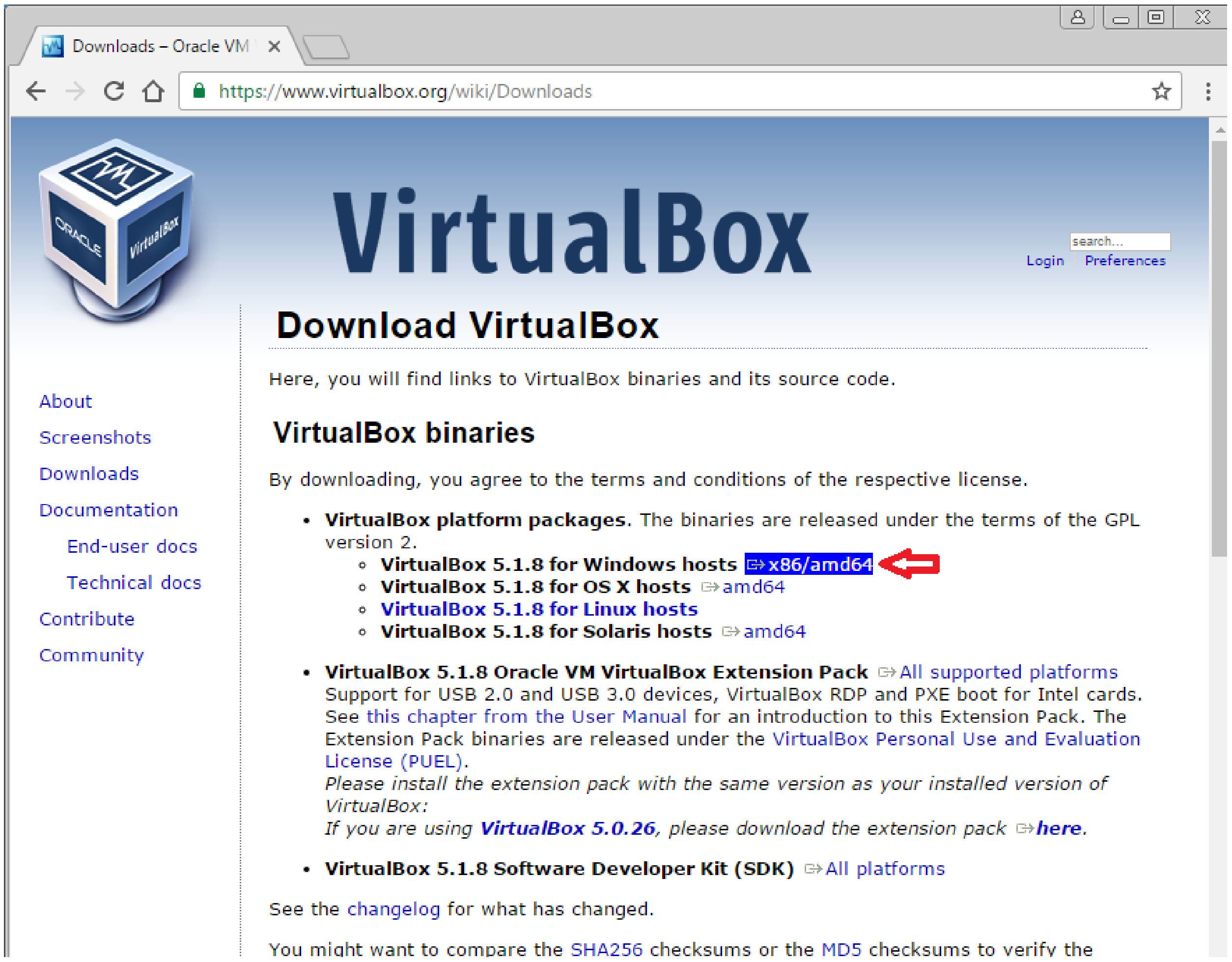
VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

News Flash

- New October 18th, 2016 **VirtualBox 5.1.8 and 5.0.28 released!** Oracle today released 5.1 and 5.0 maintenance releases which improve stability and fix regressions. See the respective Changelog (5.1.8 / 5.0.28) for details.
- New July 12th, 2016 **VirtualBox 5.1 released!** Many enhancements and improvements. Read more in the announcement.

More information...

- Click on download for VirtualBox 5.1.8 for Windows as shown below:



Download VirtualBox

Here, you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

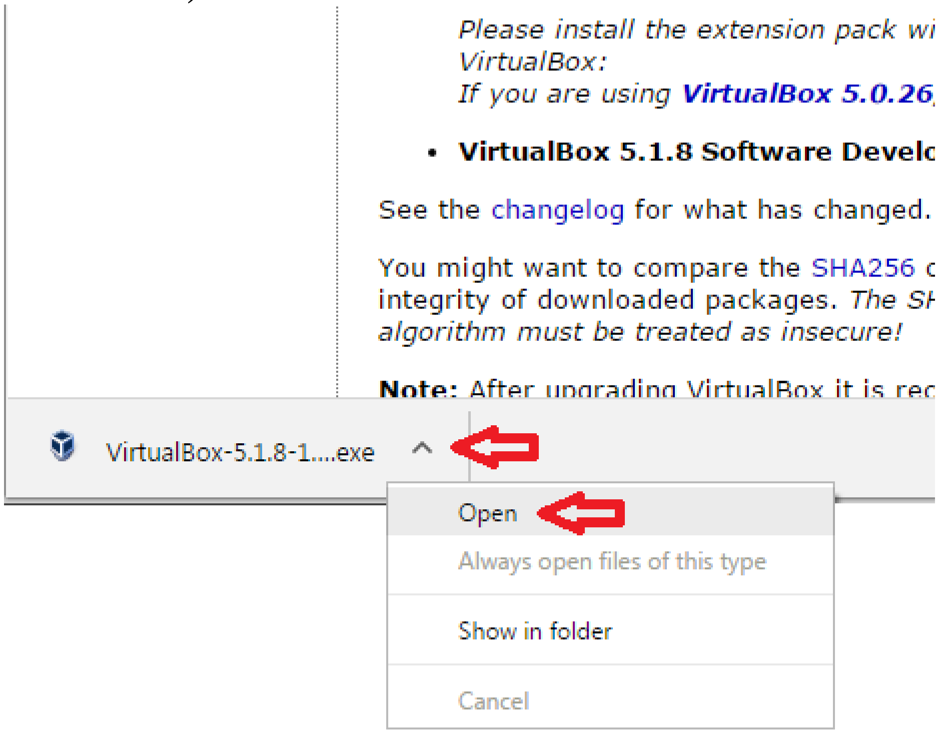
By downloading, you agree to the terms and conditions of the respective license.

- **VirtualBox platform packages**. The binaries are released under the terms of the GPL version 2.
  - **VirtualBox 5.1.8 for Windows hosts** ⇨x86/amd64
  - **VirtualBox 5.1.8 for OS X hosts** ⇨amd64
  - **VirtualBox 5.1.8 for Linux hosts**
  - **VirtualBox 5.1.8 for Solaris hosts** ⇨amd64
- **VirtualBox 5.1.8 Oracle VM VirtualBox Extension Pack** ⇨All supported platforms
  Support for USB 2.0 and USB 3.0 devices, VirtualBox RDP and PXE boot for Intel cards. See this chapter from the User Manual for an introduction to this Extension Pack. The Extension Pack binaries are released under the VirtualBox Personal Use and Evaluation License (PUEL).
  Please install the extension pack with the same version as your installed version of VirtualBox:
  *If you are using **VirtualBox 5.0.26**, please download the extension pack ⇨**here**.*
- **VirtualBox 5.1.8 Software Developer Kit (SDK)** ⇨All platforms

See the changelog for what has changed.

You might want to compare the SHA256 checksums or the MD5 checksums to verify the

- When the download completes, click on “Open” option. (Note that this download example was done using a Chrome browser. If a different web browser is used then this open operation may look different.)



Please install the extension pack wi
VirtualBox:
*If you are using **VirtualBox 5.0.26***

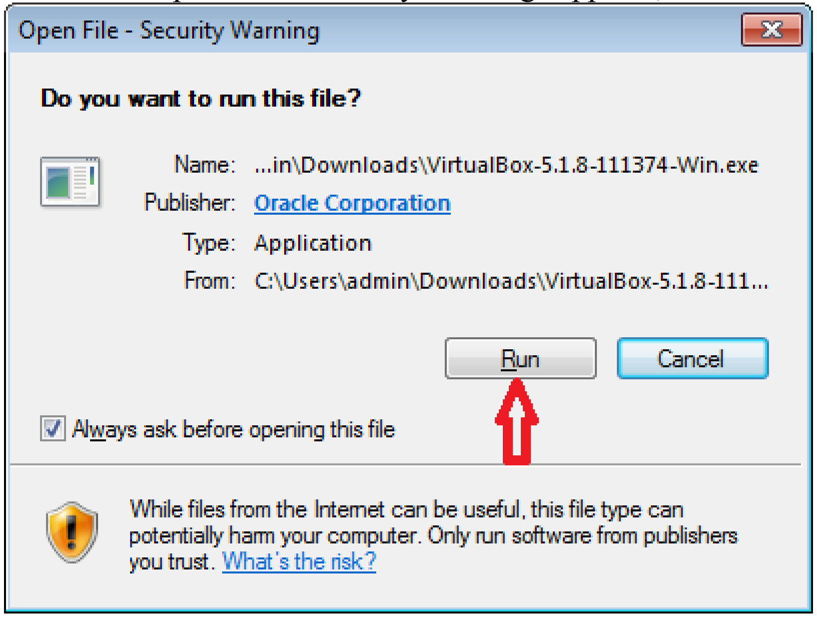
- **VirtualBox 5.1.8 Software Develc**

See the changelog for what has changed.

You might want to compare the SHA256 c integrity of downloaded packages. *The SH algorithm must be treated as insecure!*

Note: After upgrading VirtualBox it is rec

- When the “Open File – Security Warning” appears, click on “Run”.



Open File - Security Warning

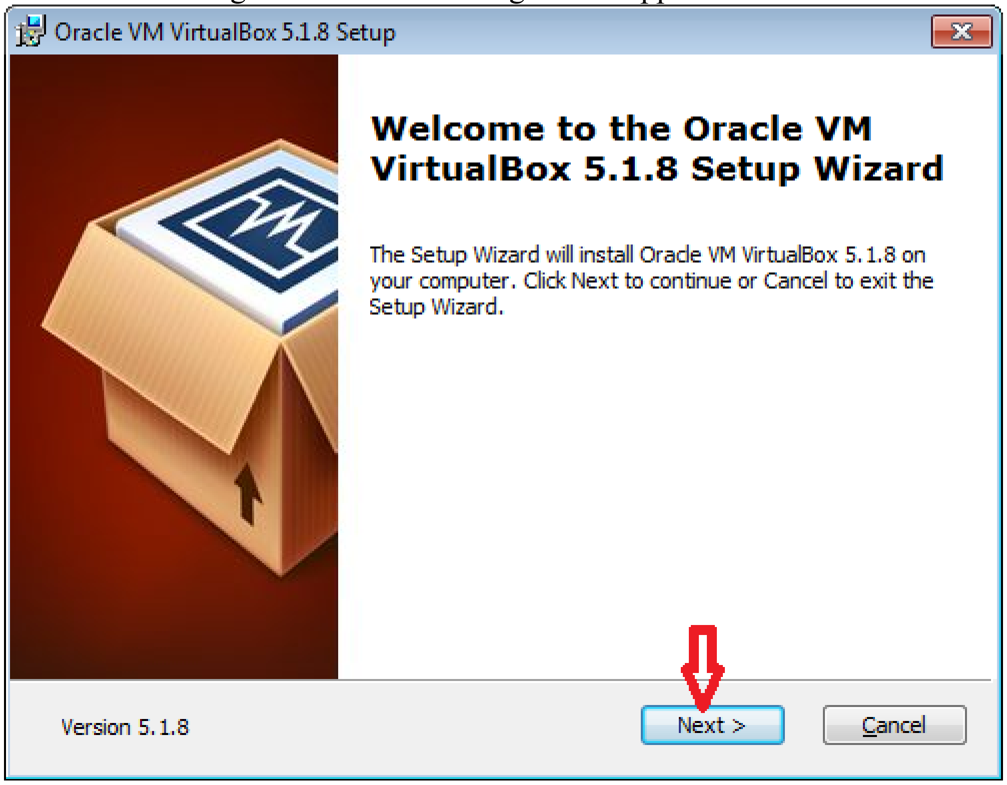
Do you want to run this file?

Name: ...in\Downloads\VirtualBox-5.1.8-111374-Win.exe
Publisher: Oracle Corporation
Type: Application
From: C:\Users\admin\Downloads\VirtualBox-5.1.8-111...

☑ Always ask before opening this file

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

- Next the following “Welcome…” dialog should appear. Click on “Next” button.


Oracle VM VirtualBox 5.1.8 Setup

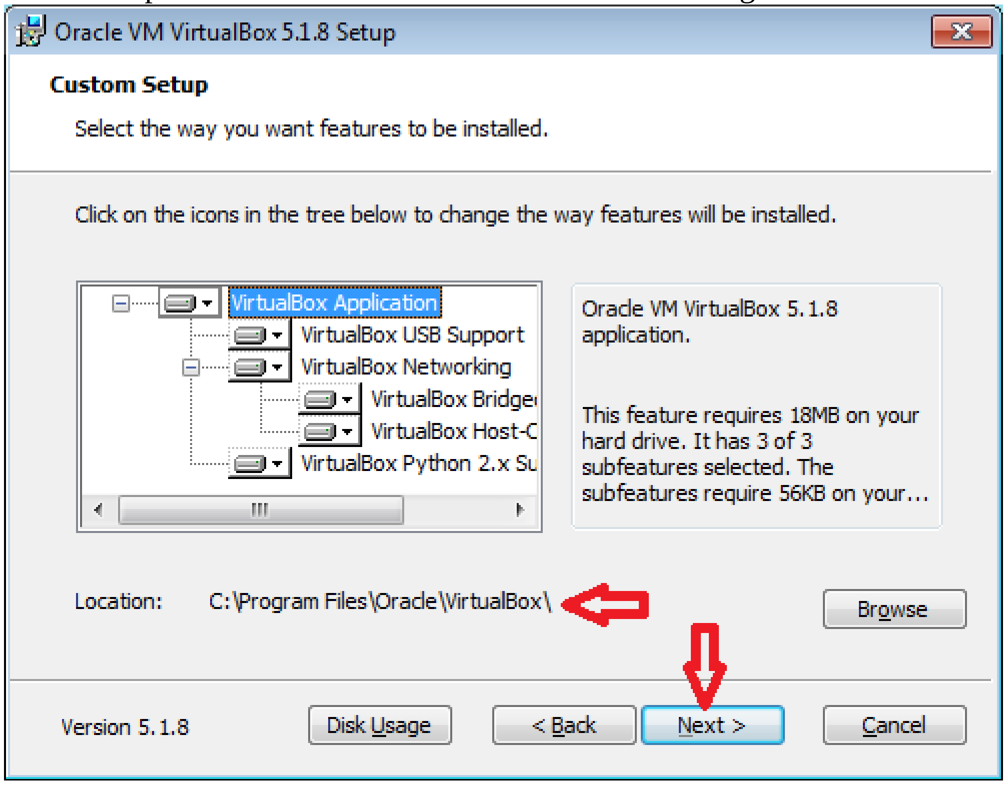
Welcome to the Oracle VM VirtualBox 5.1.8 Setup Wizard

The Setup Wizard will install Oracle VM VirtualBox 5.1.8 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

- Next will appear this “Custom Setup” dialog. Do not make any changes, just click “Next” button.

CAUTION!

Make sure that the installation location is on a local disk drive, not a network storage unit! This is because during installation the network connections will be disconnected so that the VirtualBox can install network adapter software and this will make network storage units inaccessible.



Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

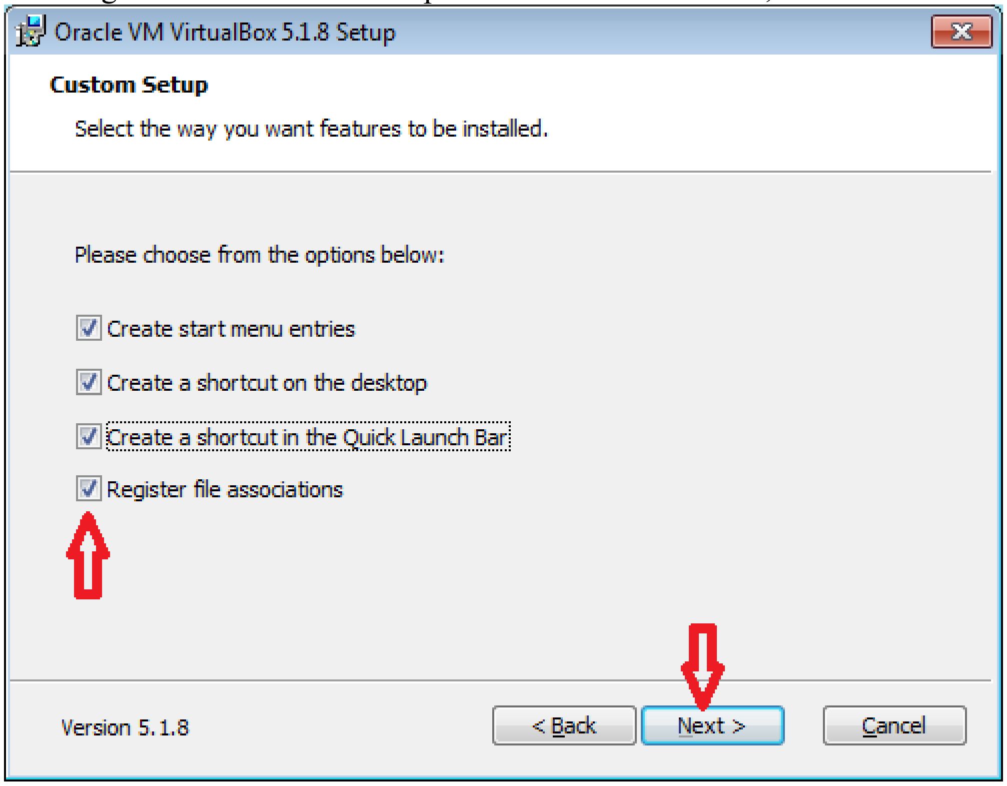
Oracle VM VirtualBox 5.1.8 application.

This feature requires 18MB on your hard drive. It has 3 of 3 subfeatures selected. The subfeatures require 56KB on your...

Location: C:\Program Files\Oracle\VirtualBox\

Version 5.1.8

- On the second "Custom Setup" screen the first three options may be adjusted as desired, but leave the "Register file associations" option checked. When done, click "Next" button.



Oracle VM VirtualBox 5.1.8 Setup

Custom Setup

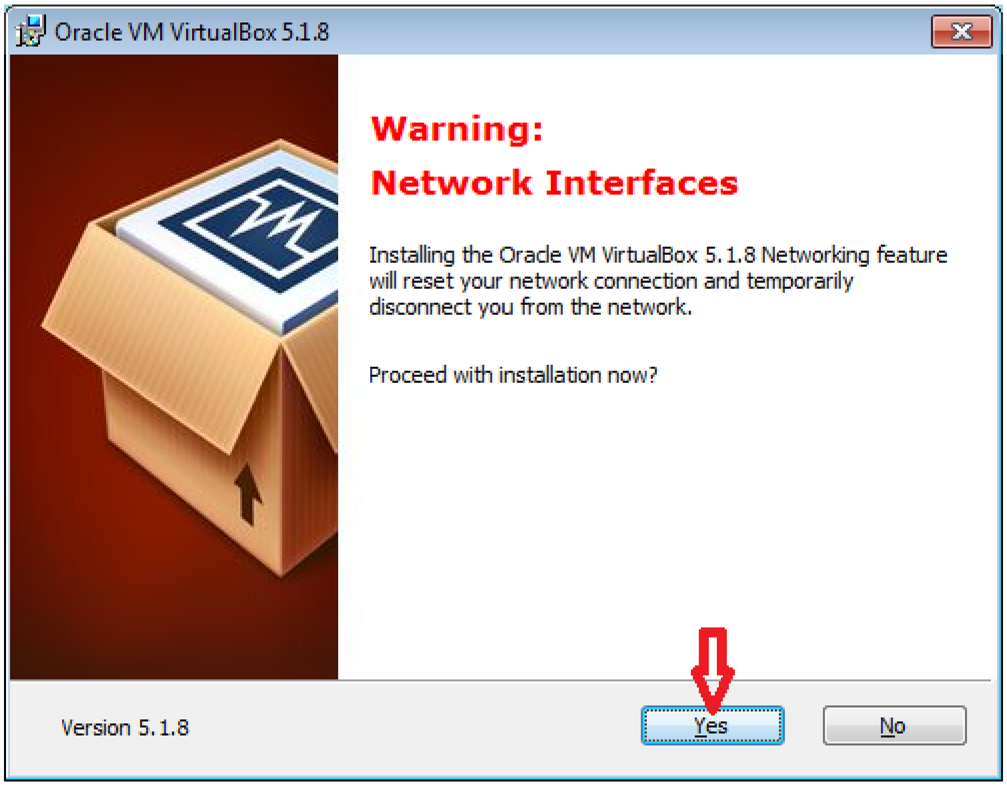
Select the way you want features to be installed.

Please choose from the options below:

- [✓] Create start menu entries
- [✓] Create a shortcut on the desktop
- [✓] Create a shortcut in the Quick Launch Bar
- [✓] Register file associations

Version 5.1.8

- The following “Warning: Network Interface” dialog should appear next. Make sure that there are no network accessing programs running (i.e. email) on the computer. Exit any such programs and then click “Yes” button.



Oracle VM VirtualBox 5.1.8

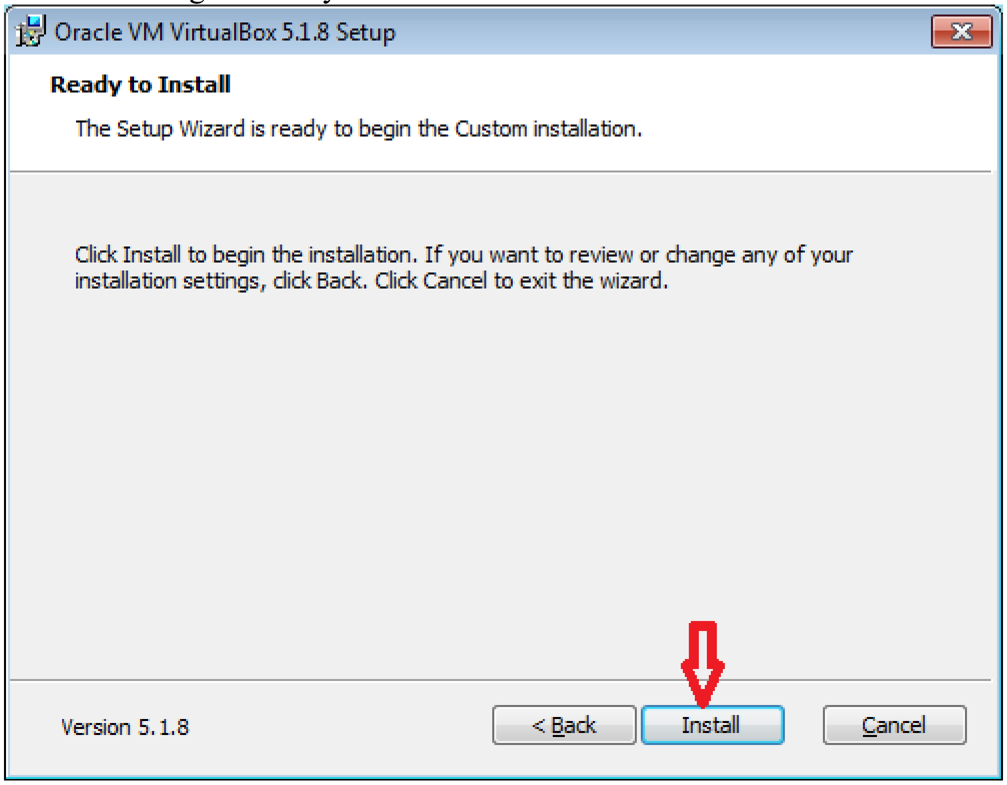
Warning:
Network Interfaces

Installing the Oracle VM VirtualBox 5.1.8 Networking feature will reset your network connection and temporarily disconnect you from the network.

Proceed with installation now?

Version 5.1.8

- The next dialog is "Ready to Install" as shown here. Click on the "Install" button.



Oracle VM VirtualBox 5.1.8 Setup

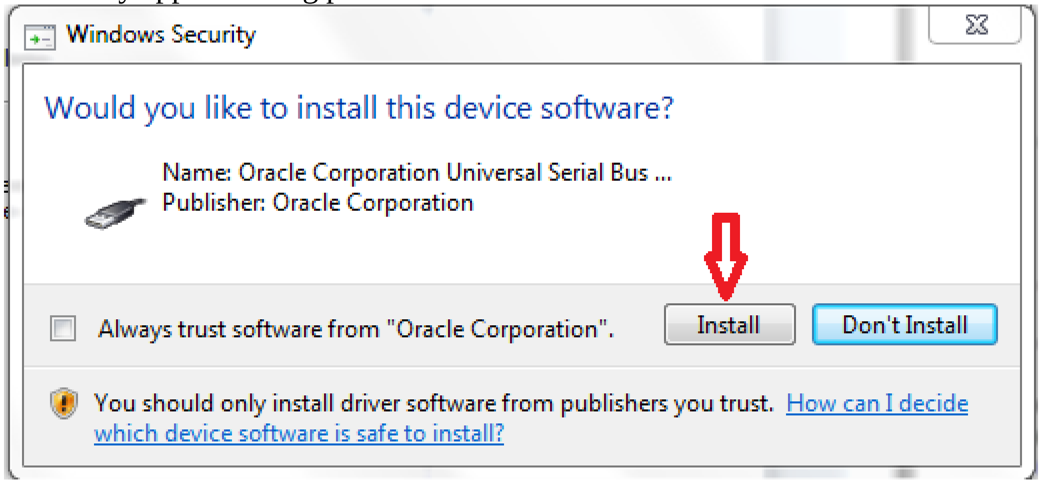
Ready to Install

The Setup Wizard is ready to begin the Custom installation.

Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Version 5.1.8

- If a “User Account Control” window appears, click “Yes” button.
- If any “Windows Security” windows appear as shown below, click on “Install” button. Several such windows may appear asking permission to install various driver software modules.



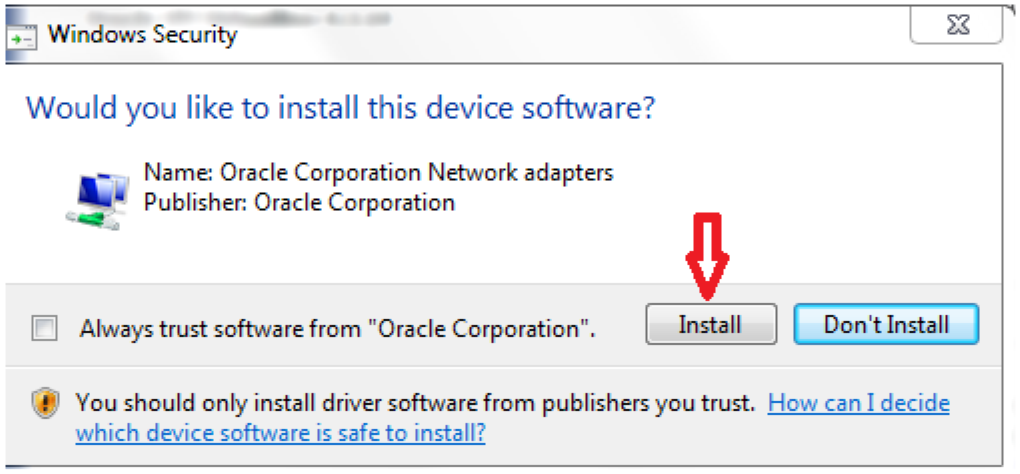
Windows Security

Would you like to install this device software?

Name: Oracle Corporation Universal Serial Bus ...
Publisher: Oracle Corporation

Always trust software from "Oracle Corporation".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?



Windows Security

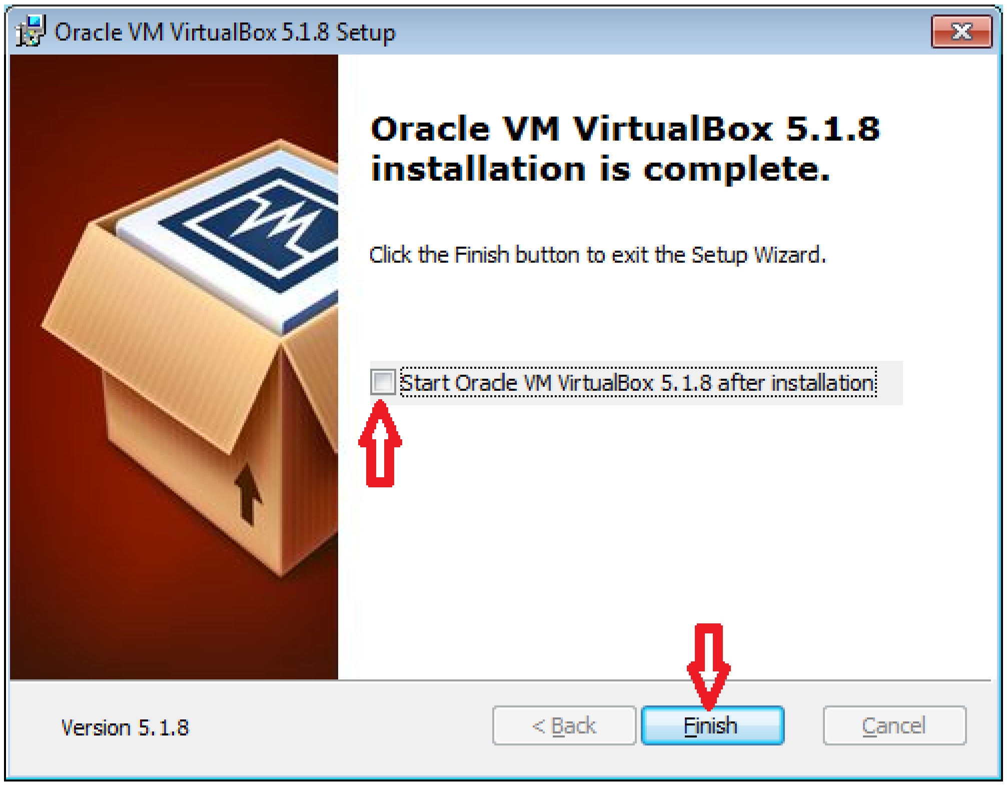
Would you like to install this device software?

Name: Oracle Corporation Network adapters
Publisher: Oracle Corporation

Always trust software from "Oracle Corporation".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

- When installation is finished, this window should appear. Uncheck the “Start Oracle VM VirtualBox…” selection and click the “Finish” button.



Oracle VM VirtualBox 5.1.8 Setup

Oracle VM VirtualBox 5.1.8 installation is complete.

Click the Finish button to exit the Setup Wizard.

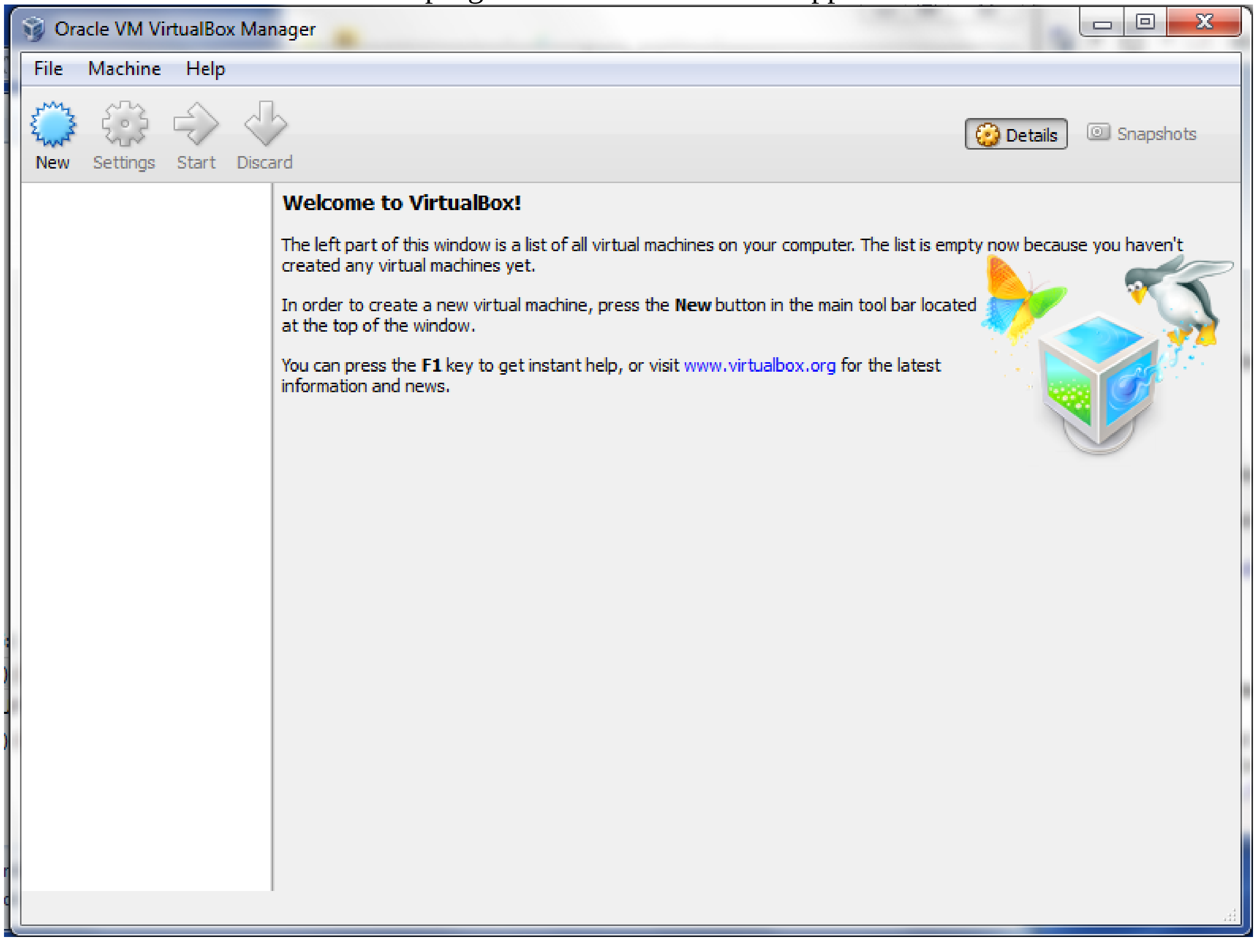
Start Oracle VM VirtualBox 5.1.8 after installation

Version 5.1.8

- Reboot computer at this time to re-establish network connections.

Setup VirtualBox Linux Environment

- After computer reboots, copy the folder "VMSharedFolder" and all of its contents from the "3Dxx_Qt_Installation_RevG.zip" file to a newly created file folder called "C:\VMSharedFolder". The files in this folder will be used later during Qt configuration, but this folder can be used after that to transfer files between the Windows and Linux environments.
- Copy the file "Grayhill 3Dxx Linux Qt Rev G.ova" to any place on the development PC. Remember the location of this copied .ova file as it will be needed later.
- Start the Oracle VM VirtualBox program. This screen should appear:



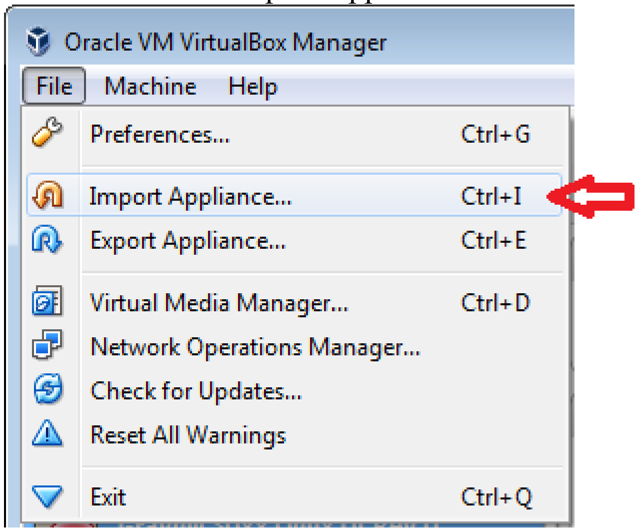
Welcome to VirtualBox!

The left part of this window is a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.

In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for the latest information and news.

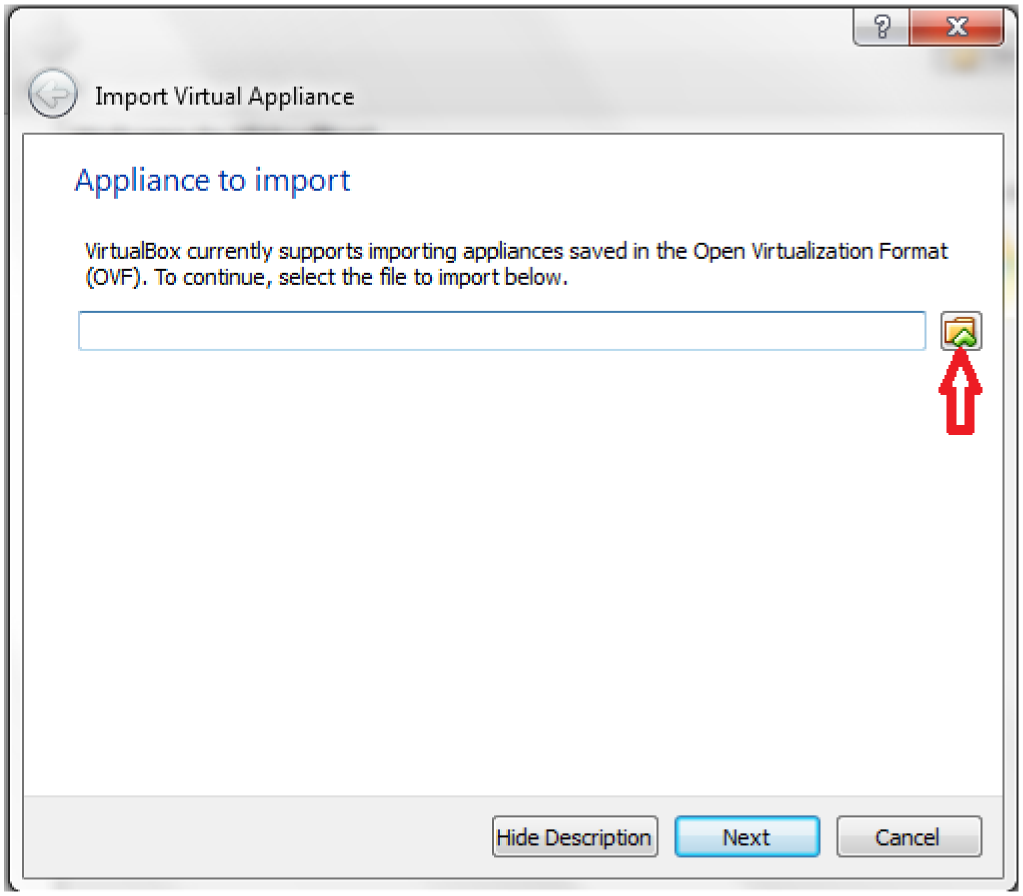
- Click on “File -> Import Appliance” as shown below:


Oracle VM VirtualBox Manager
File
Machine
Help
Preferences... Ctrl+G
Import Appliance... Ctrl+I
Export Appliance... Ctrl+E
Virtual Media Manager... Ctrl+D
Network Operations Manager...
Check for Updates...
Reset All Warnings
Exit Ctrl+Q

- Click on the open folder icon as shown below and navigate to the previously copied file: “Grayhill 3Dxx Linux Qt Rev G.ova” and open it.

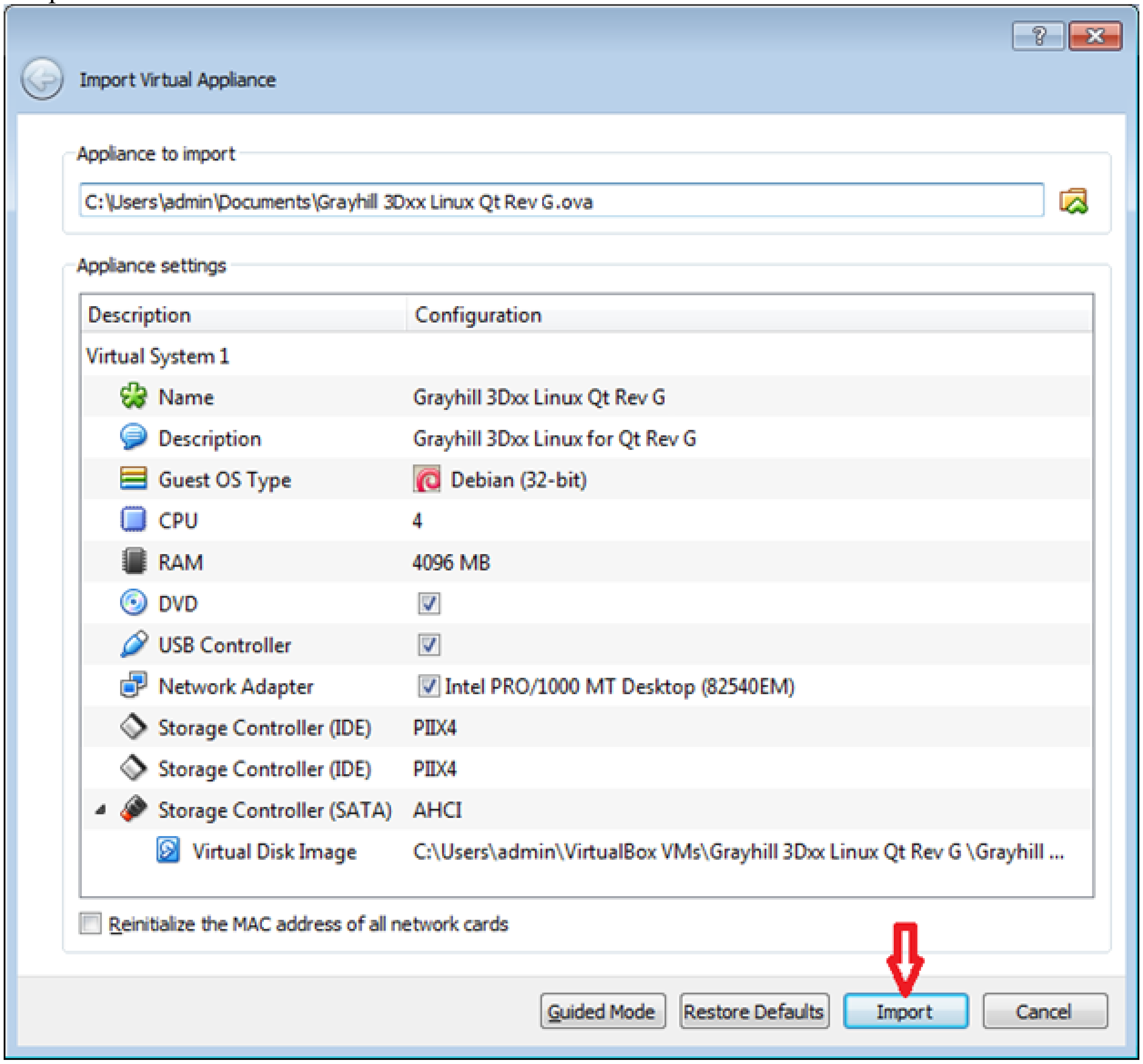
NOTE

The above .ova file is quite large and if distributed on a USB memory stick, that memory stick must use exFAT format. If operating under Windows XP, then the exFAT file system support package must be installed in order to access this media (see note under **Other Recommended Equipment** for more details).



VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

- Adjust any settings as appropriate, such as number of processors or RAM size and then click "Import" button.



Import Virtual Appliance

Appliance to import

C:\Users\admin\Documents\Grayhill 3Dxx Linux Qt Rev G.ova

Appliance settings

Description	Configuration
Virtual System 1	
Name	Grayhill 3Dxx Linux Qt Rev G
Description	Grayhill 3Dxx Linux for Qt Rev G
Guest OS Type	Debian (32-bit)
CPU	4
RAM	4096 MB
DVD	✓
USB Controller	✓
Network Adapter	✓ Intel PRO/1000 MT Desktop (82540EM)
Storage Controller (IDE)	PIIX4
Storage Controller (IDE)	PIIX4
Storage Controller (SATA)	AHCI
Virtual Disk Image	C:\Users\admin\VirtualBox VMs\Grayhill 3Dxx Linux Qt Rev G \Grayhill ...

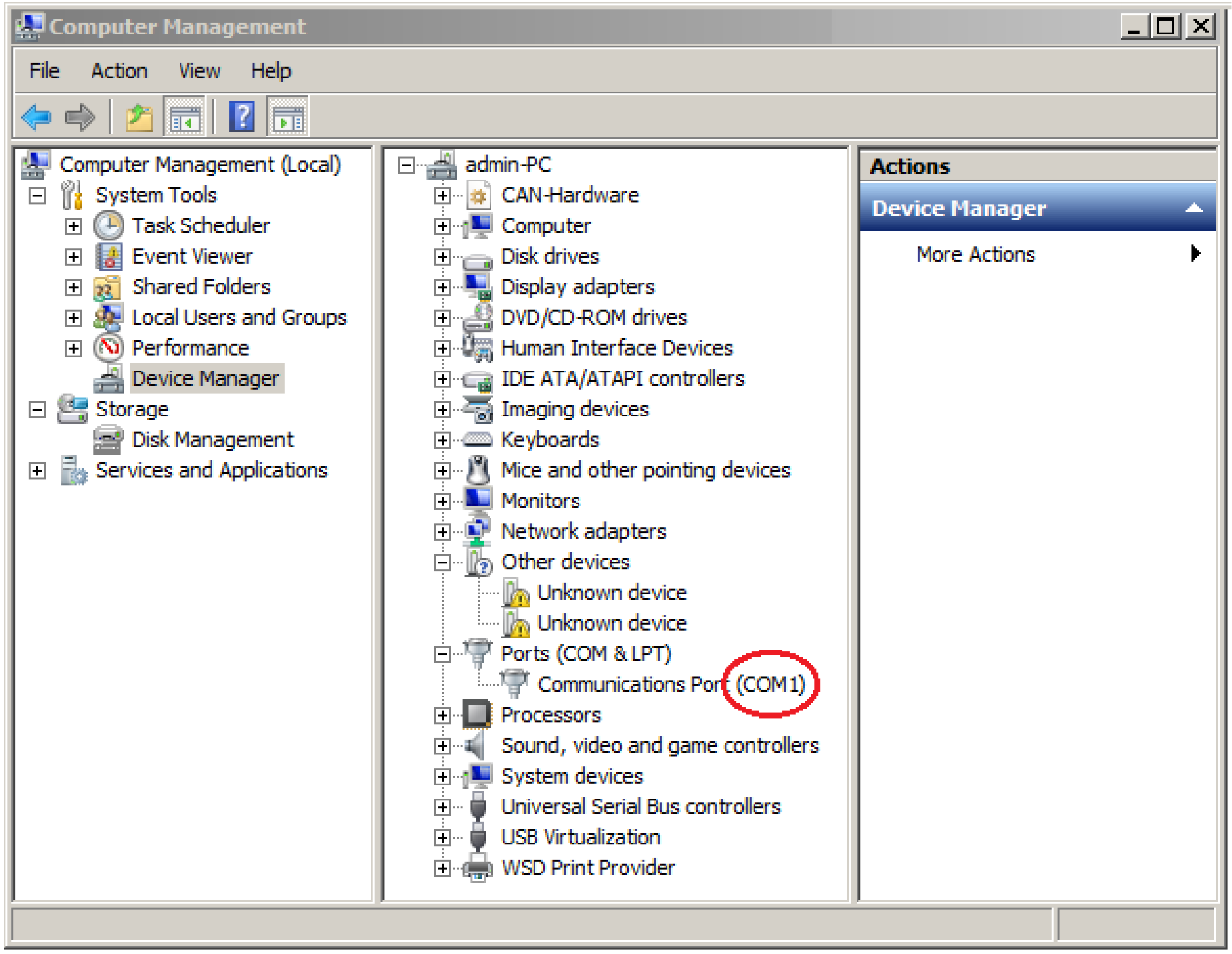
☐ Reinitialize the MAC address of all network cards

Setup VirtualBox Serial Port

In order to access the 3Dxx Display Linux console, a serial port that operates at 115200 baud is required. If the development PC has a built-in serial port that is going to be used for this purpose, then proceed with the setup instructions below to configure the VirtualBox Serial Port. If a USB to serial port converter is going to be used, then skip the VirtualBox Serial Port setup and continue with the step: **Starting Linux Development Environment from VirtualBox**.

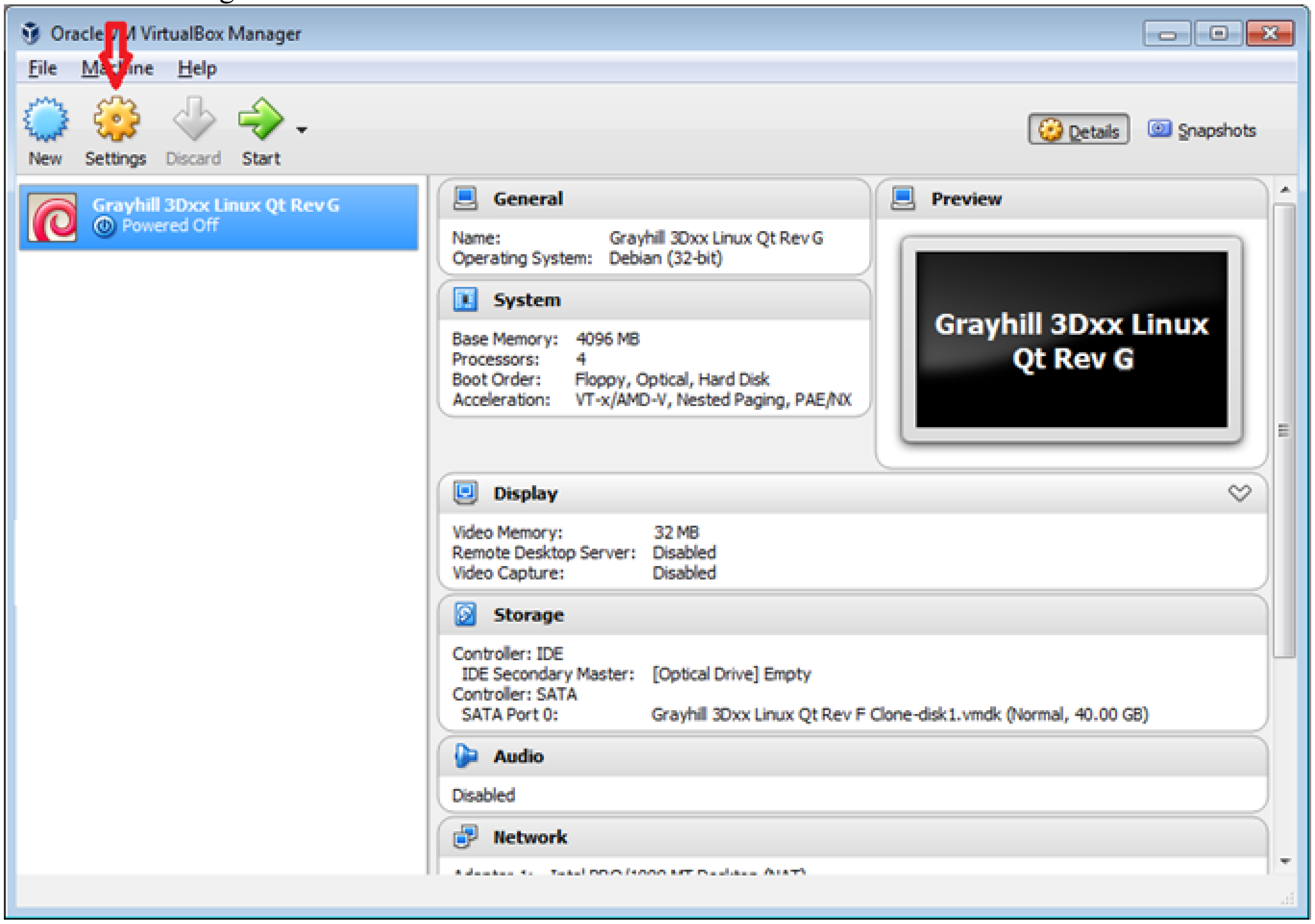
- Determine what COM port is assigned to the serial port that is going to be used. This can be determined by accessing the Device Manager window and looking under the “Ports (COM & LPT)” entry. In the example shown below the serial port is assigned to “COM1”. Port settings for this port do not need to be setup here; they will be setup later via the Linux “Minicom” utility.

WARNING!
Using a COM port greater than 9 will not work!



- Close the “Device Manager” dialog box, but remember the COM port number used.

- After the previous VirtualBox "Import" operation was performed, this screen should appear. Click on the "Settings" icon as shown here:



Name: Grayhill 3Dxx Linux Qt Rev G
Operating System: Debian (32-bit)

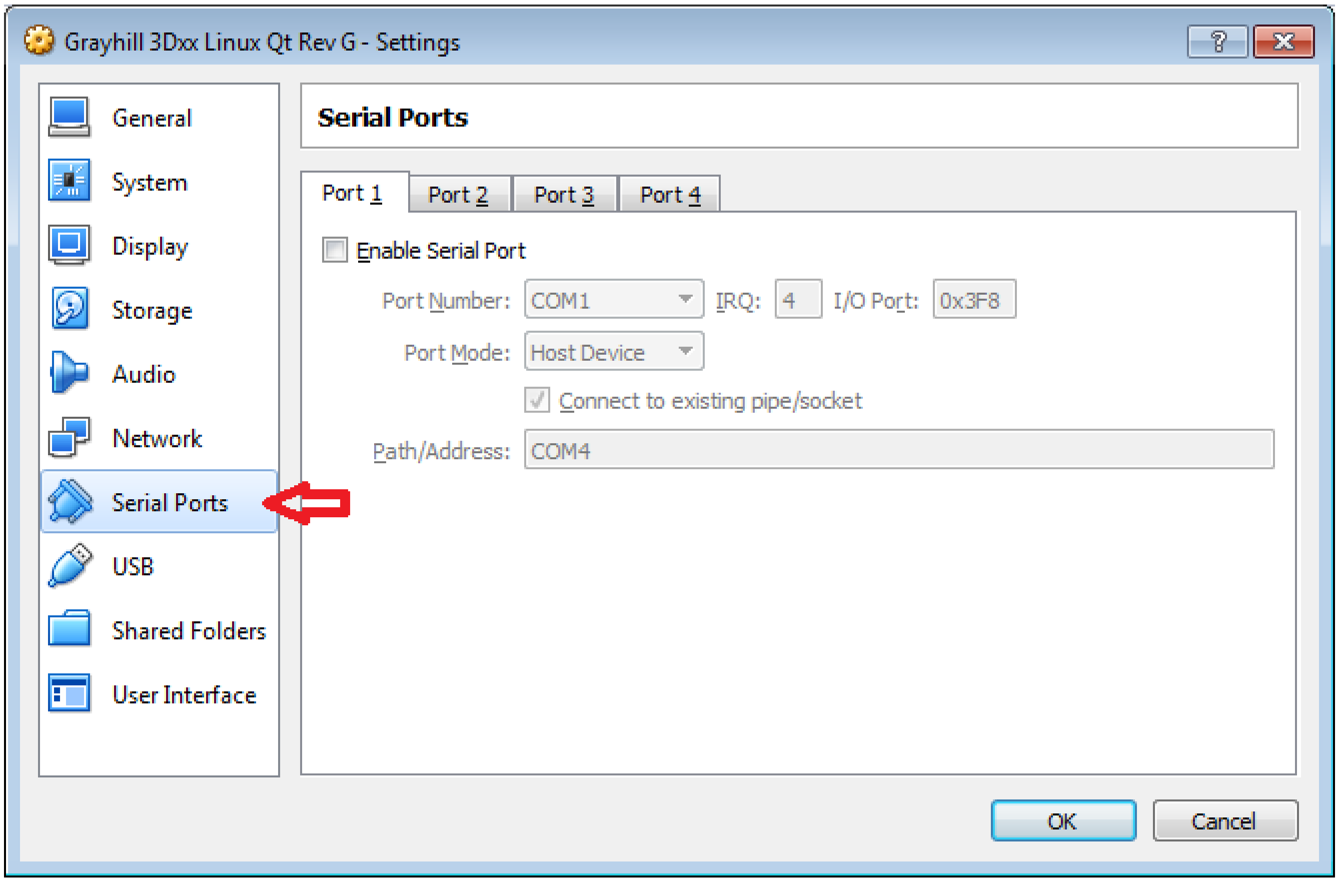
Base Memory: 4096 MB
Processors: 4
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX

Video Memory: 32 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: Grayhill 3Dxx Linux Qt Rev F Clone-disk1.vmdk (Normal, 40.00 GB)

Disabled

- When the “Grayhill 3Dxx Linux Qt … -Settings” dialog box appears, click on the “Serial Ports” item on the left column as shown:



Grayhill 3Dxx Linux Qt Rev G - Settings

Serial Ports

Port 1 | Port 2 | Port 3 | Port 4

- [ ] Enable Serial Port

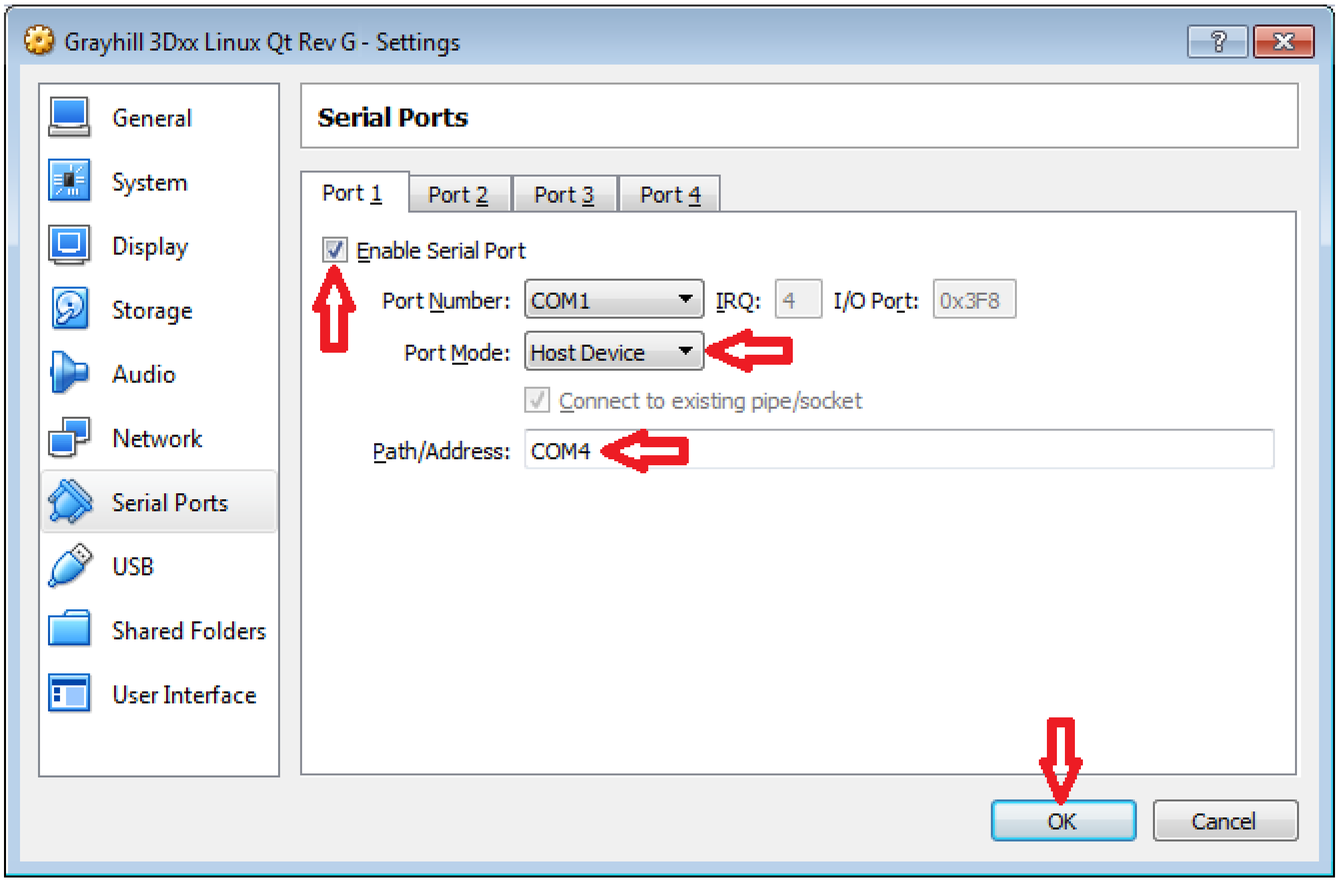
Port Number: COM1 IRQ: 4 I/O Port: 0x3F8

Port Mode: Host Device

- [✓] Connect to existing pipe/socket

Path/Address: COM4

- On the "Port 1" tab, make sure that the "Enable Serial Port" box is checked and leave the "Port Number:" box set to "COM1" no matter what COM port is being used. Set the "Port Mode:" to "Host Device" and in the "Port/File Path:" enter the COM port name and number from the previous "Device Manager" step. An example is shown below. When these settings are correct click on the "OK" button.


Grayhill 3Dxx Linux Qt Rev G - Settings

Serial Ports

Port 1

[✓] Enable Serial Port

Port Number: COM1 IRQ: 4 I/O Port: 0x3F8

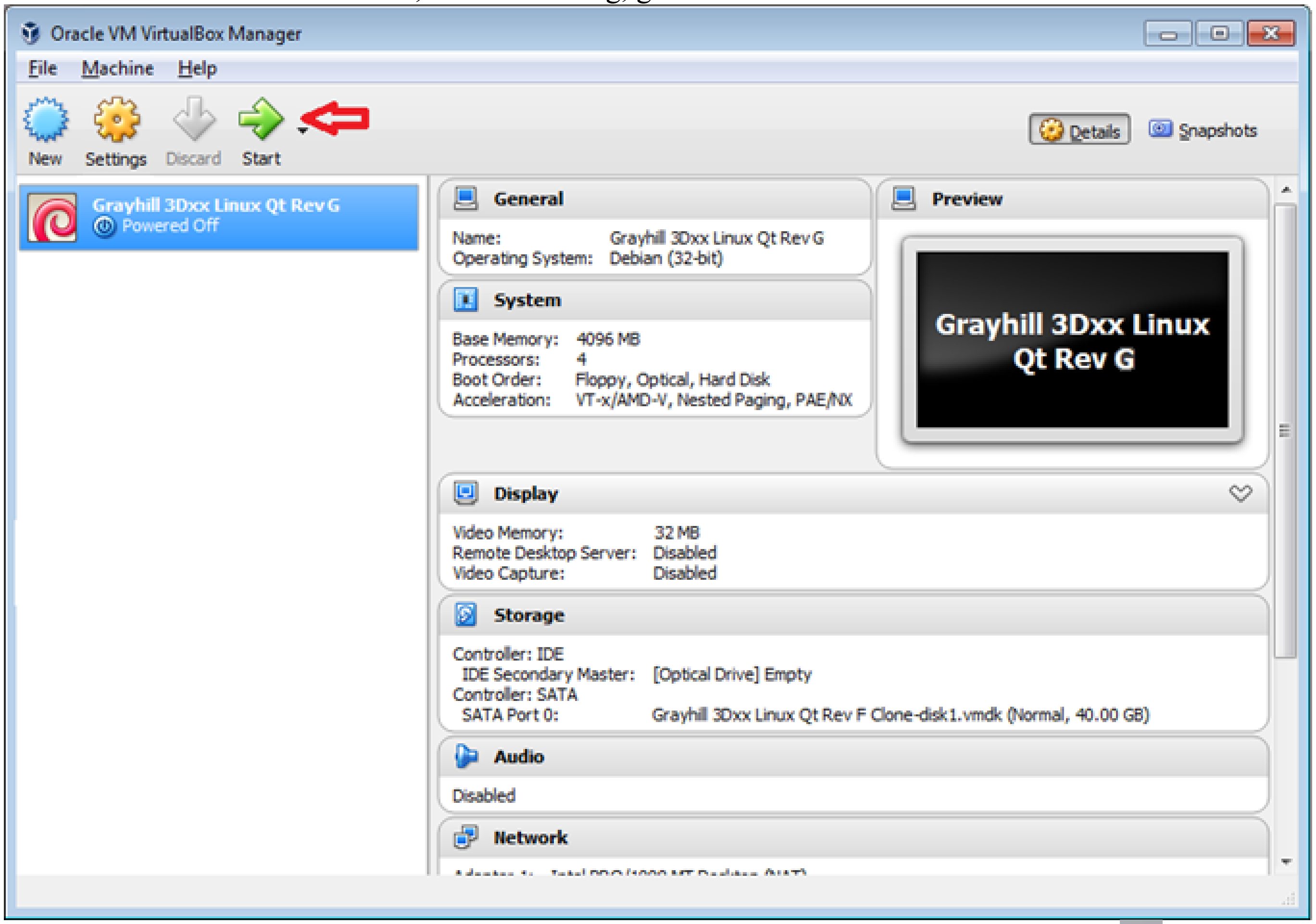
Port Mode: Host Device

[✓] Connect to existing pipe/socket

Path/Address: COM4

Starting Linux Development Environment from VirtualBox

- When focus returns to this screen, click on the big, green "Start: arrow.



Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Start
Details Snapshots
Grayhill 3Dxx Linux Qt Rev G
Powered Off
Preview
Grayhill 3Dxx Linux Qt Rev G

General
Name: Grayhill 3Dxx Linux Qt Rev G
Operating System: Debian (32-bit)

System
Base Memory: 4096 MB
Processors: 4
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX

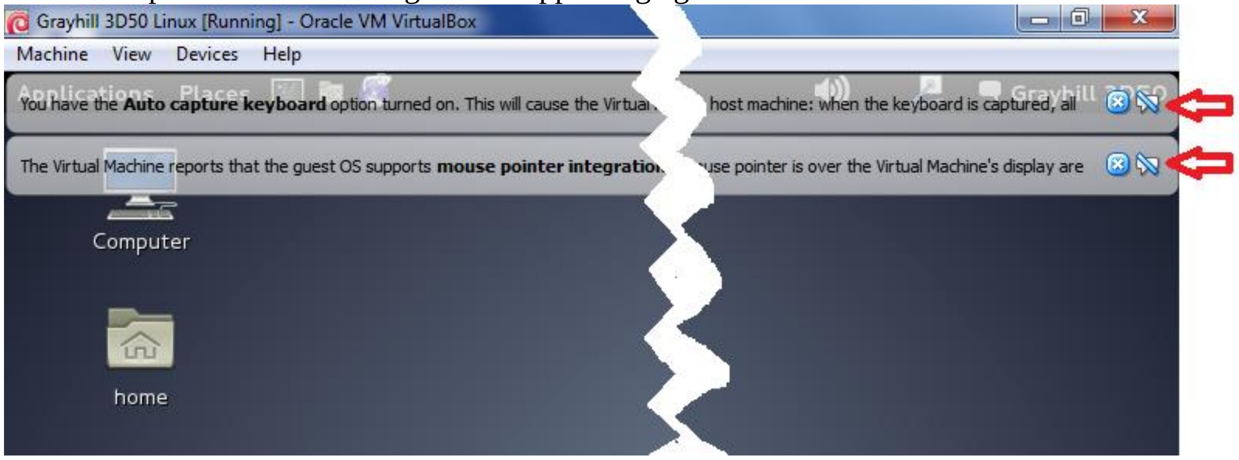
Display
Video Memory: 32 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: Grayhill 3Dxx Linux Qt Rev F Clone-disk1.vmdk (Normal, 40.00 GB)

Audio
Disabled

Network

- If any messages such as the ones shown below appear when Linux first starts, click on the icon to prevent these messages from appearing again.



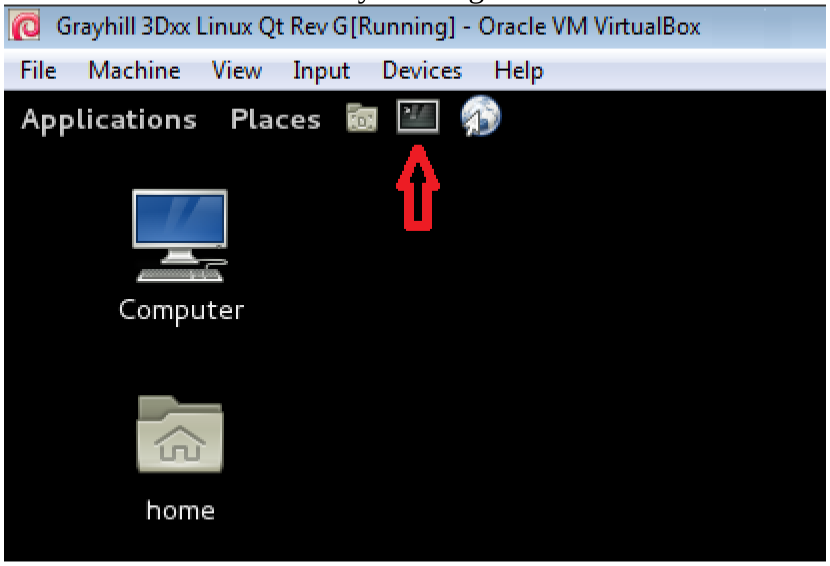
Grayhill 3D50 Linux [Running] - Oracle VM VirtualBox
Machine View Devices Help
Computer
home

You have the **Auto capture keyboard** option turned on. This will cause the Virtual host machine: when the keyboard is captured, all

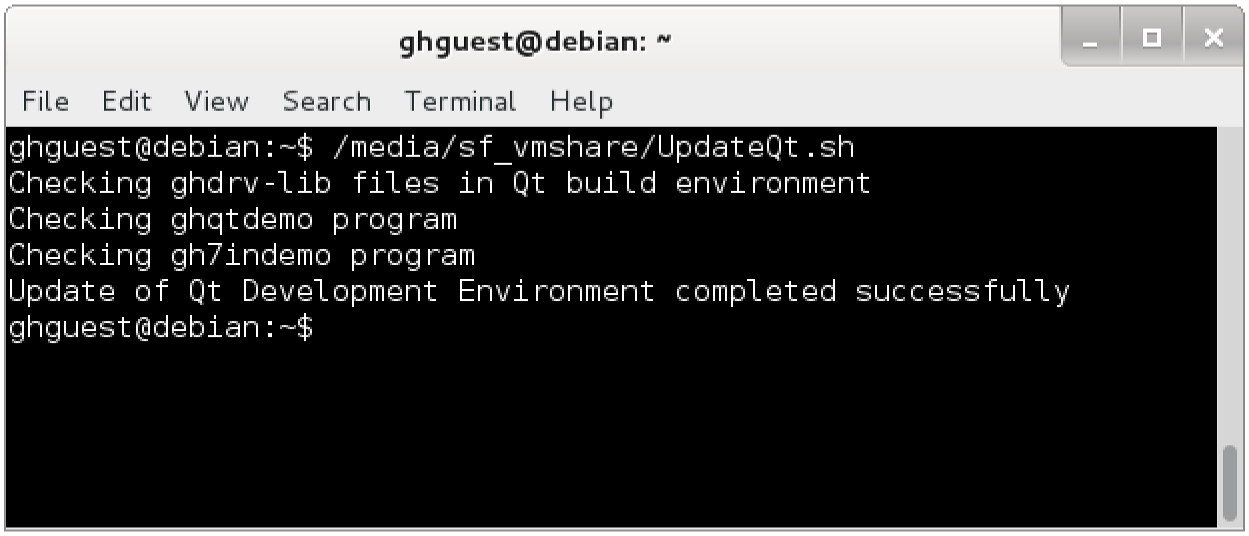
The Virtual Machine reports that the guest OS supports **mouse pointer integration** use pointer is over the Virtual Machine's display are

Updating Qt Development Environment

- Launch a Terminal Command Window by clicking once on the icon shown here:


Grayhill 3Dxx Linux Qt Rev G [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places
Computer
home

- In the terminal window type the following command:
  - `/media/sf_vmshare/UpdateQt.sh`
- This will run a script that will update some support libraries and include files and will update the two Qt demonstration programs. A sample is shown below, but more messages may be output as files are updated. Note that this script only updates a file if it is newer than the existing copy. The "Update... completely successfully" output shows that the update script worked.

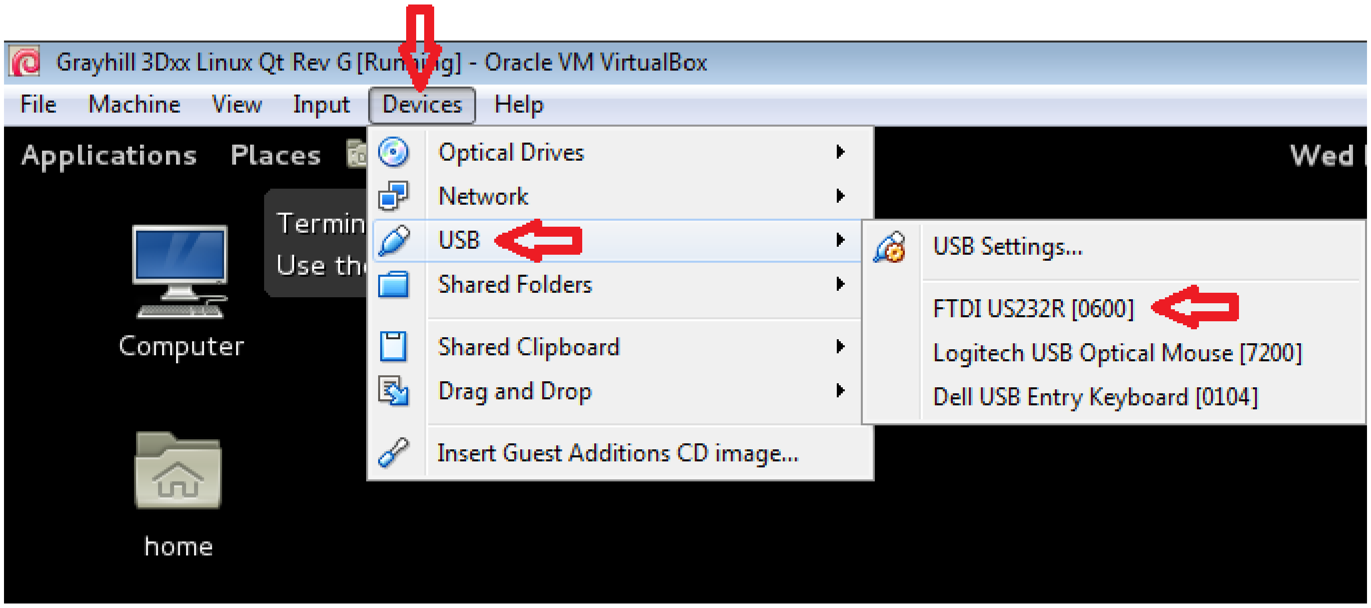


```
ghguest@debian:~$ /media/sf_vmshare/UpdateQt.sh
Checking ghdrv-lib files in Qt build environment
Checking ghqtdemo program
Checking gh7indemo program
Update of Qt Development Environment completed successfully
ghguest@debian:~$
```

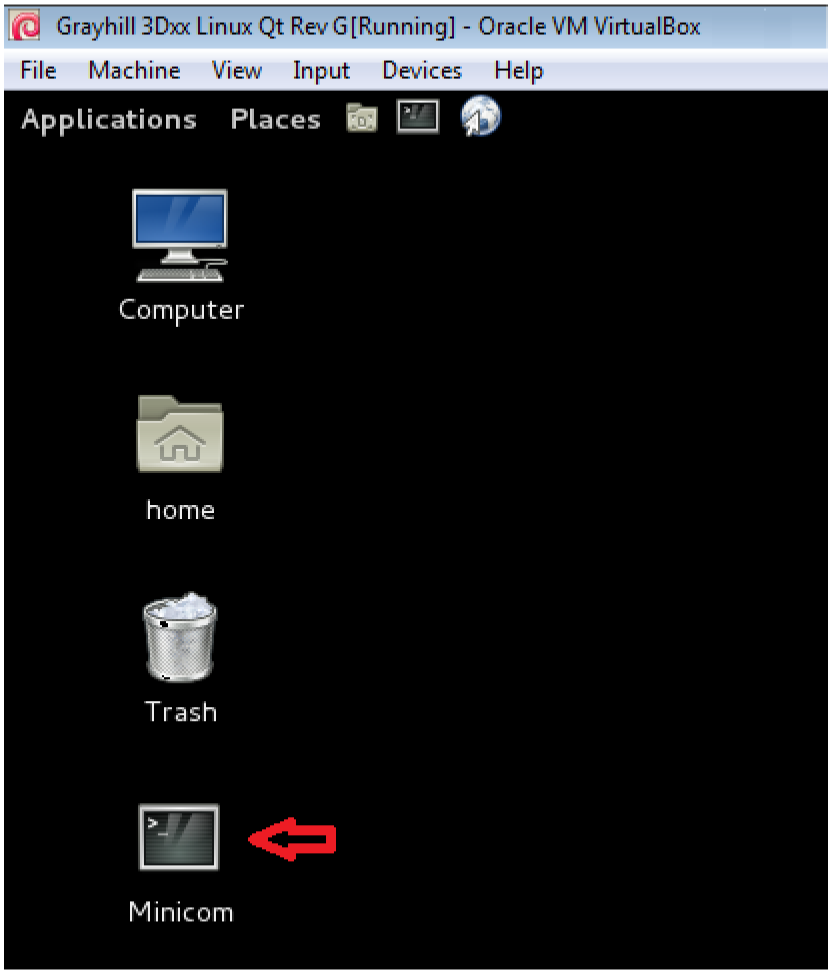
- Enter "exit" command to close terminal window.

Configuring 3Dxx Display for Qt Development

- 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. Also, some other items on the 3Dxx Display must be configured so that it can operate properly with Qt. In order to perform these tasks, it will be necessary to connect the Ethernet on the 3Dxx Display to the same network as the development PC. (Refer to the first item in the section **Installation Overview** for more details on performing the hardware connection). The Ethernet network the 3Dxx Display is connected to must provide a DHCP server that will automatically assign an IP address to the 3Dxx Display.
- The 3Dxx Display serial port must also be connected to a serial port on the development PC. (Refer to section **Installation Overview** for more details).
- Determine the serial port device name to use for the Linux Minicom serial communications program depending on how the 3Dxx Display serial port is connected to the development PC. If using a built-in serial port on the development PC, then remember that the serial port device name will be “/dev/ttyS0”. If using a USB to serial port adapter, then remember that the serial port device name will be “/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. Then 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):

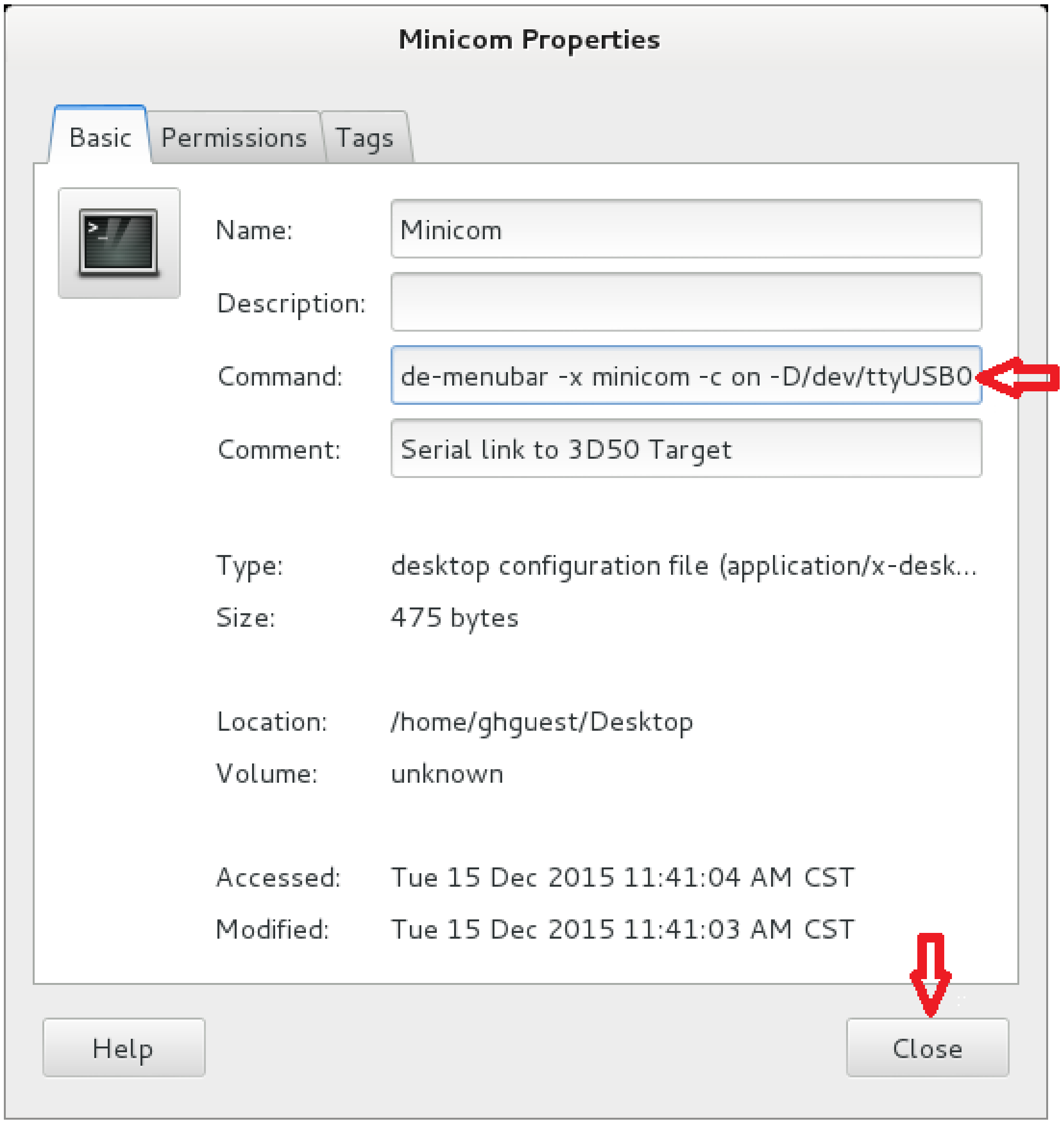


- Right-click on the “Minicom” icon as shown below and then left-click on “Properties”.



- Click in the box labeled “Command:” and use arrow or end keys to move to the end of the command line. Make sure that the device name as determined above is entered at the end of the string. Then click on the “Close” button. This sample shows setting the device name to “/dev/ttyUSB0”: Do not change any other parameters on the command line. The command line is typically set to:

```
gnome-terminal -t Minicom --hide-menubar -x minicom -c on -D /dev/ttyS0
```



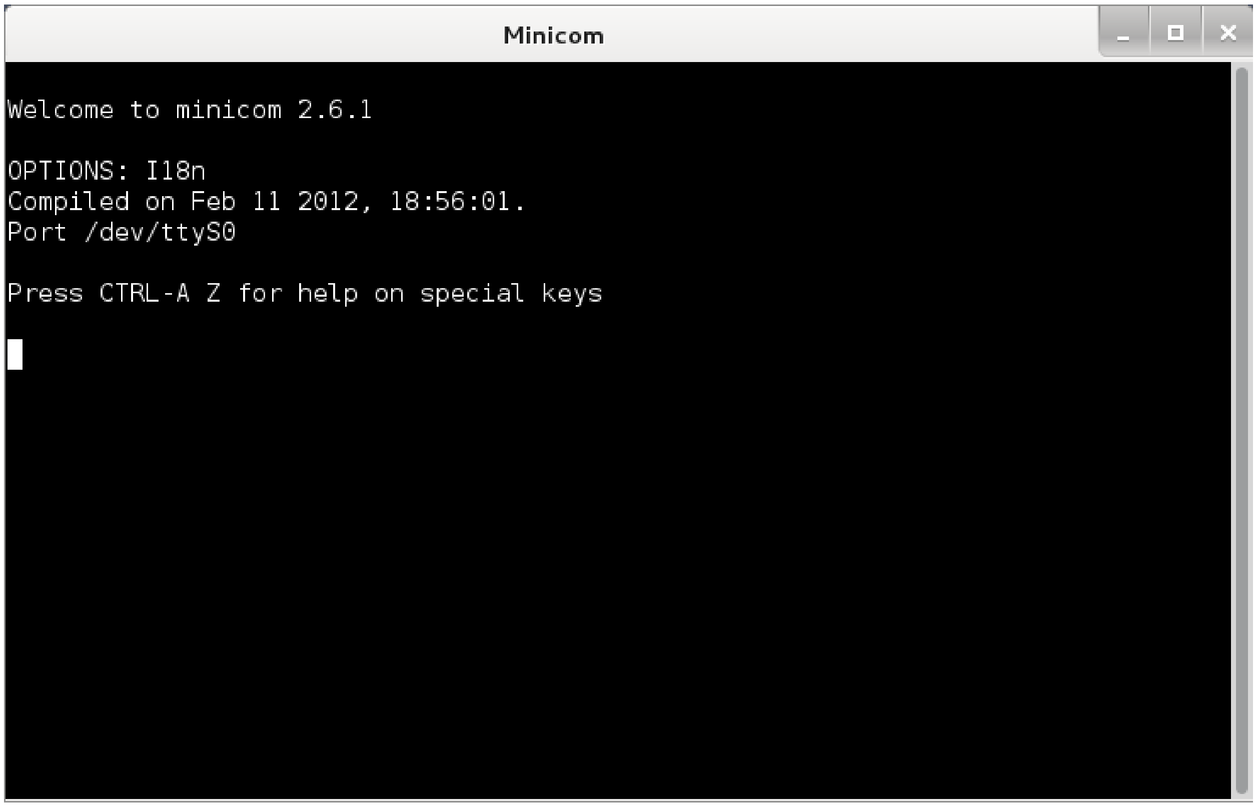
Name: Minicom
Description:
Command: de-menubar -x minicom -c on -D/dev/ttyUSB0
Comment: Serial link to 3D50 Target

Type: desktop configuration file (application/x-desk...
Size: 475 bytes

Location: /home/ghguest/Desktop
Volume: unknown

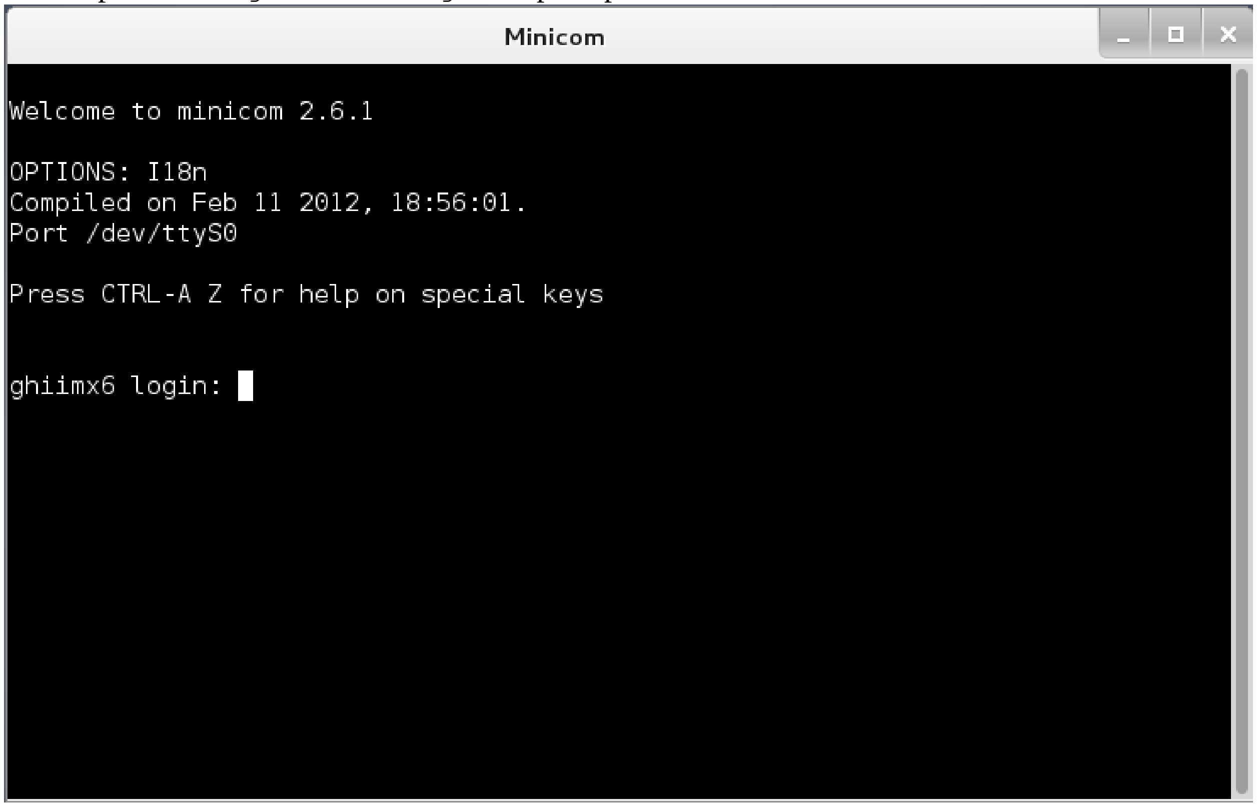
Accessed: Tue 15 Dec 2015 11:41:04 AM CST
Modified: Tue 15 Dec 2015 11:41:03 AM CST

- Double-click on the Minicom desktop icon to start the program. The Minicom window should look like this when it first starts:



```
Welcome to minicom 2.6.1

OPTIONS: I18n
Compiled on Feb 11 2012, 18:56:01.
Port /dev/ttyS0

Press CTRL-A Z for help on special keys
```

- Make sure that the 3Dxx Display is powered up and press the "Enter" key. A "`ghiimx6 login:`" prompt should appear as shown below. If the 3Dxx Display was just powered up, then many startup messages may appear as well, but when they are done, pressing the "Enter" key should produce a "`ghiimx6 login:`" prompt as shown.



```
Welcome to minicom 2.6.1

OPTIONS: I18n
Compiled on Feb 11 2012, 18:56:01.
Port /dev/ttyS0

Press CTRL-A Z for help on special keys


ghiimx6 login: 
```

- At the "`ghiimx6 login:`" prompt enter "`root`" (no password is required).
- The 3Dxx Display internal flash file system is configured to be read-only so that the system boots up faster and to protect the file system from corruption. In order to download and test files from the Qt development environment, this internal flash file system must be set to read-write mode. This must be done using this command:
  - mount -o remount,rw /

  This setting only remains in effect until the next time the 3Dxx Display is rebooted. There are instructions at the end of this document that describe how to set the 3Dxx Display flash file system to read-write mode on boot-up. The special 3Dxx configuration script that will be run in a few steps will use this technique to configure the 3Dxx Display file system to be in read-write mode to make Qt development more convenient.

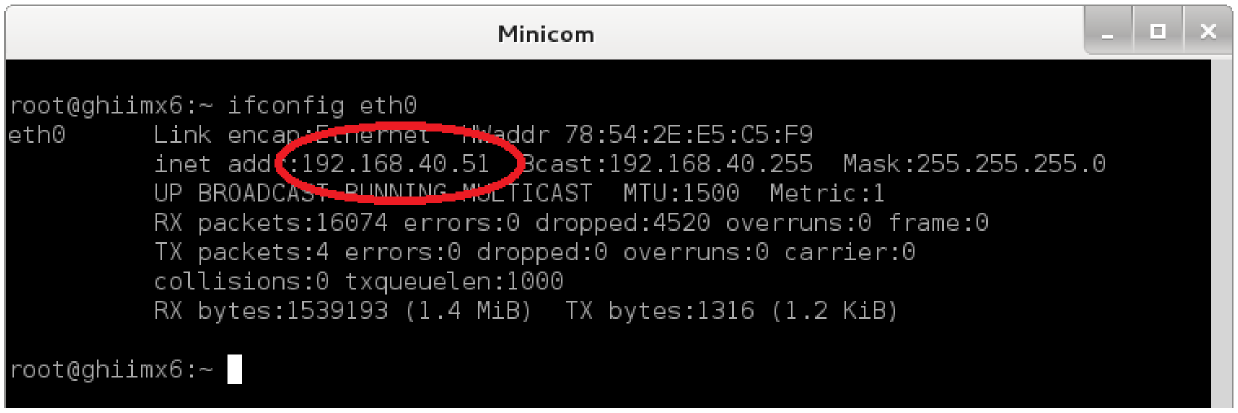
CAUTION!

It is not recommended to leave the 3Dxx Display flash file system in read-write mode on fielded units because this may result in file system corruption and slower boot-up times.

- Enter this command to find the 3Dxx Display Ethernet IP address:
  - `ifconfig eth0`

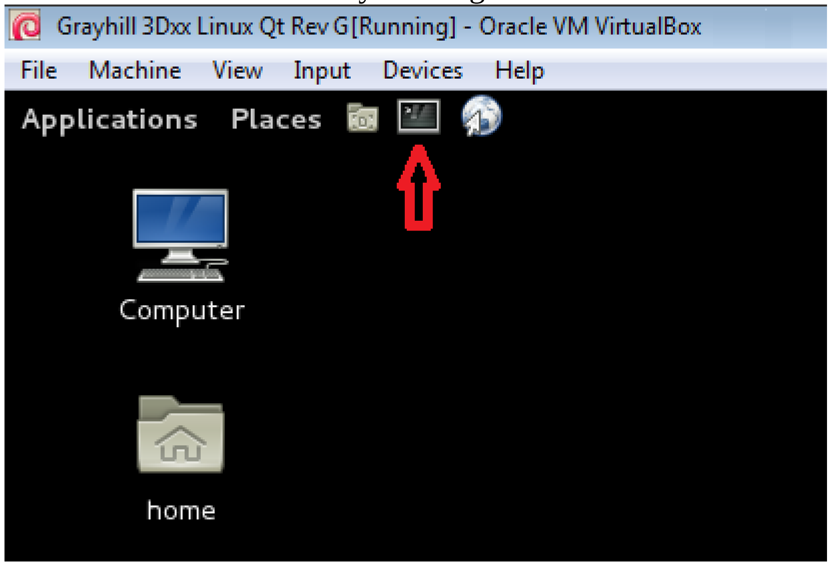
  The IP address of the 3Dxx Display is after the tag "`inet addr:`" and it is circled in red in the example output shown below.
- If the tag "`inet addr:`" is not present, then enter these two commands and try the "ifconfig eth0" command again.
  - ifdown eth0
  - ifup eth0

In this example the IP address is 192.168.40.51. Make note of this IP address for the next setup step.



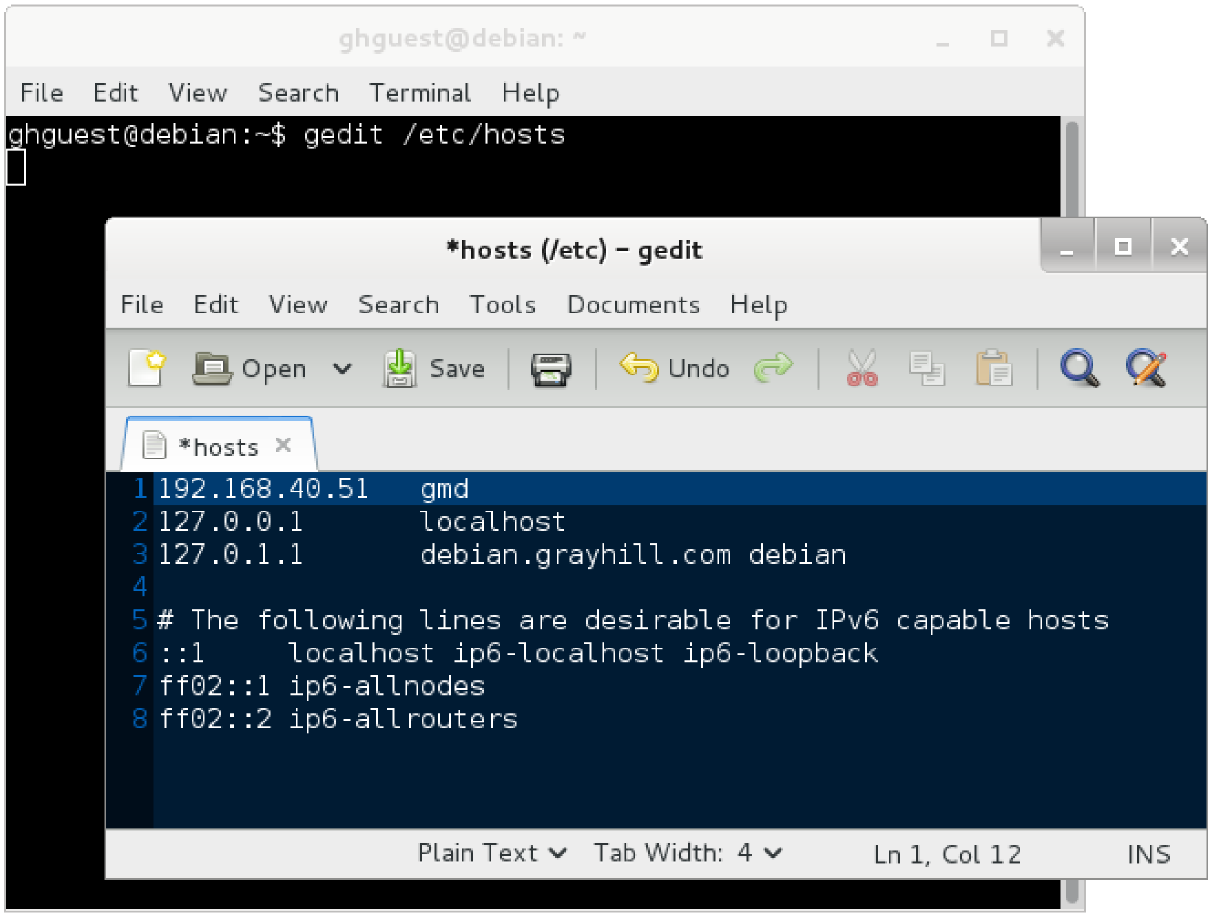
```
root@ghiimx6:~ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 78:54:2E:E5:C5:F9
          inet addr:192.168.40.51  Bcast:192.168.40.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16074 errors:0 dropped:4520 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1539193 (1.4 MiB)  TX bytes:1316 (1.2 KiB)

root@ghiimx6:~
```

- Launch a Terminal Command Window by clicking once on the icon shown here:


Grayhill 3Dxx Linux Qt Rev G [Running] - Oracle VM VirtualBox
File
Machine
View
Input
Devices
Help
Applications
Places
Computer
home

- In the terminal window type the following command:
    - `gedit /etc/hosts`

  This will open the IP address configuration file for the Linux system in a text editor program so that the IP address of the 3Dxx Display can be configured. Here is a sample:



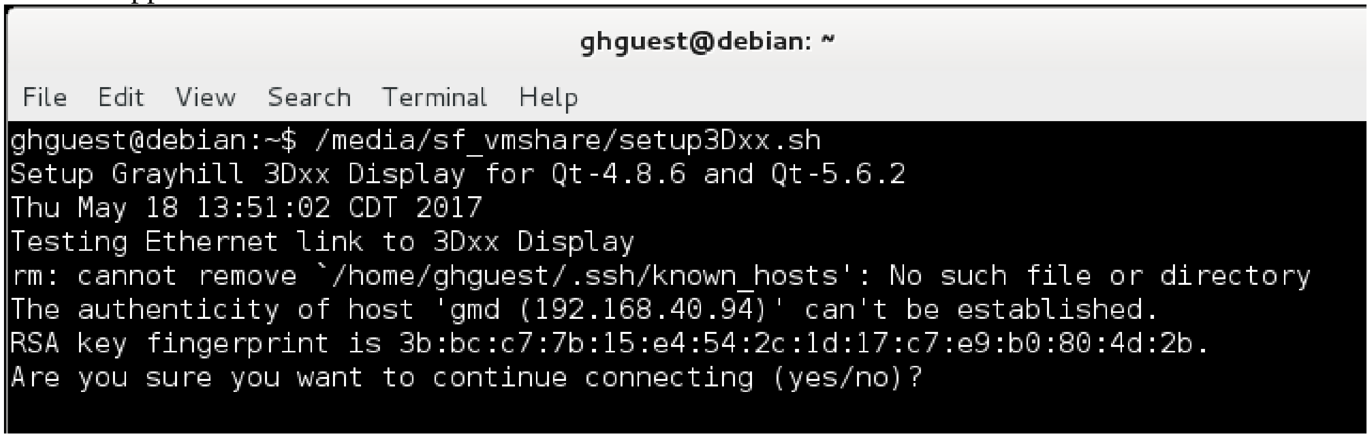
```
ghguest@debian:~$ gedit /etc/hosts
```

```
192.168.40.51   gmd
127.0.0.1       localhost
127.0.1.1       debian.grayhill.com debian

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

- Enter the 3Dxx IP address on the line marked “gmd”. This will enable Ethernet access to the 3Dxx Display by using the hostname “gmd”.
- Click on the “Save” button on the editor to the save the changes made to the IP address. Then close the editor window, but leave the terminal window open for the next step.

- In the terminal window type the following command:
  - `/media/sf_vmshare/setup3Dxx.sh`

  This will configure the 3Dxx Display for operation with the Qt development environment. This script will work for Models 3D50, 3D70, and 3D2104. When this script starts a message similar to this will appear:

```
ghguest@debian:~$ /media/sf_vmshare/setup3Dxx.sh
Setup Grayhill 3Dxx Display for Qt-4.8.6 and Qt-5.6.2
Thu May 18 13:51:02 CDT 2017
Testing Ethernet link to 3Dxx Display
rm: cannot remove `/home/ghguest/.ssh/known_hosts': No such file or directory
The authenticity of host 'gmd (192.168.40.94)' can't be established.
RSA key fingerprint is 3b:bc:c7:7b:15:e4:54:2c:1d:17:c7:e9:b0:80:4d:2b.
Are you sure you want to continue connecting (yes/no)?
```

  Just enter "yes" to continue.
- The script will do a reboot of the display and will require about five minutes to run. It will output this message if everything works correctly:

```
    setup3Dxx completed successfully
```

  If this message does not appear, the problem must be corrected before continuing.
- Close the terminal window after the script finishes successfully by entering "exit" command.

Choosing a 3Dxx Qt Widget Demo Project

Qt Widget demonstration projects are provided for each of the 3Dxx Displays. There is a file in each demonstration program called “ghwrapper.cpp”. This file is a focal point for the demonstration program’s operation and in the very beginning of this file are comments explaining how the demonstration program works.

This table compares the features of the demonstration programs:

Program Name	ghqtdemo	gh7indemo	gh10indemo
Target Display	Model 3D50	Model 3D70	Model 3D2104
Orientation	Portrait	Landscape	Landscape
Real Time Clock setting	Yes	Yes	Yes
CAN input	Yes	Yes	Yes
CAN output	No	Yes	Yes
Touch Screen tap input	Yes	Yes	Yes
Touch Screen Swipes	Yes	Yes	Yes
Digital Inputs shown	4	4	4
Digital Outputs shown	4	4	4
Video inputs shown	2	3	3
Buzzer demo	No	Yes	Yes
Audio Output demo	No	Yes	No
Analog Input demo	No	Yes	No

Note that Qt-4.8.6 can run in portrait or landscape mode and so either the “ghqtdemo” or the “gh7indemo” demonstration program can run under Qt-4.8.6.

For now programs running under Qt-5.6.2 can only operate in native landscape mode, so only the “gh7indemo” and “gh10indemo” programs can be run under Qt-5.6.2.

There will be one set of instructions for explaining how to configure and run either the “ghqtdemo” or the “gh7indemo”demonstration programs under Qt-4.8.6 and the desktop environment (which uses Qt-5.5.1).

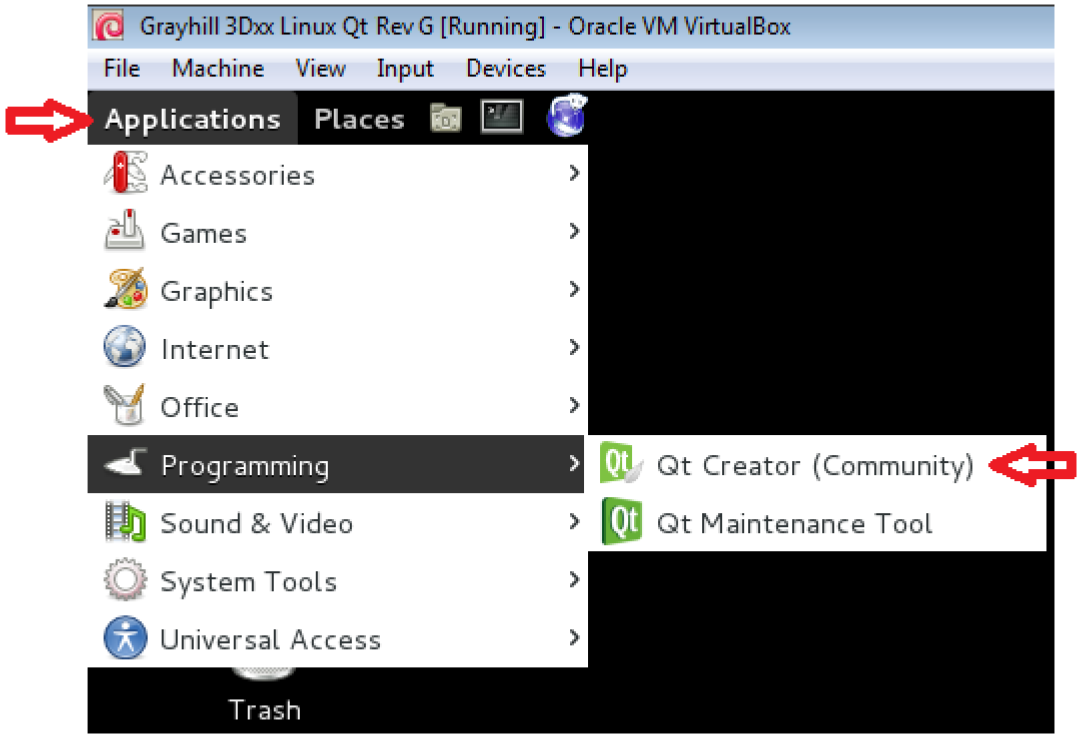
There will be a separate set of instructions that will explain how to configure and run the “gh7indemo” program using Qt-5.6.2.

Configure a 3Dxx Demo Project to Build and Run (Qt-4.8.6)

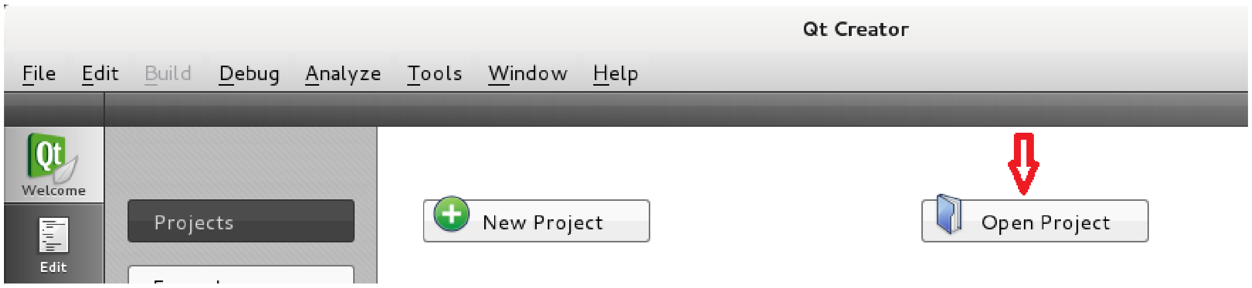
These instructions explain how to configure one of the demonstrations programs to run on a 3Dxx Display using Qt-4.8.6 or on the development environment desktop using Qt-5.5.1. These instructions are applicable to demonstration programs “ghqtdemo” and “gh7indemo”. The screen shots shown below are for the “ghqtdemo” demonstration program, but in certain places things will need to be done differently for the “gh7indemo” demonstration program as will be explained.

These kit configuration instructions have already been performed on the “ghqtdemo” and “gh7indemo” programs provided in the preconfigured Qt development environment. These instructions are provided here to aid in understanding how to setup a new project.

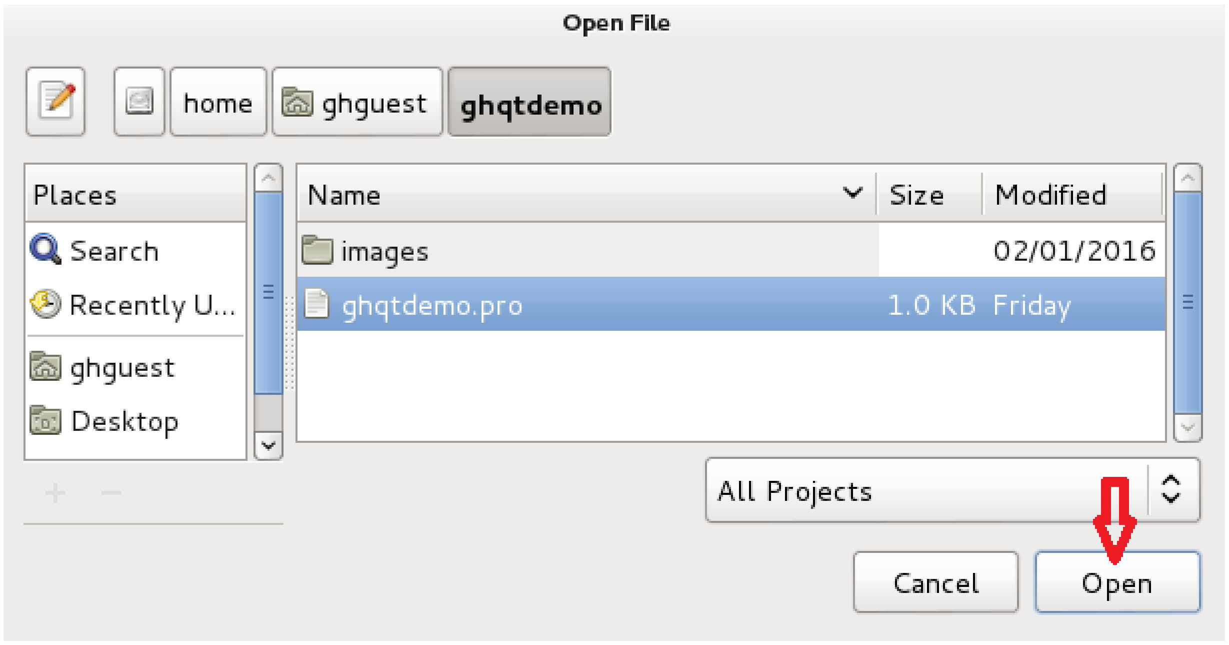
- If not already started, start the Qt Creator by clicking on the “Applications” word in the upper left-hand corner of the Linux window and then navigating through “Programming” and then clicking on the “Qt Creator …” item as shown here:



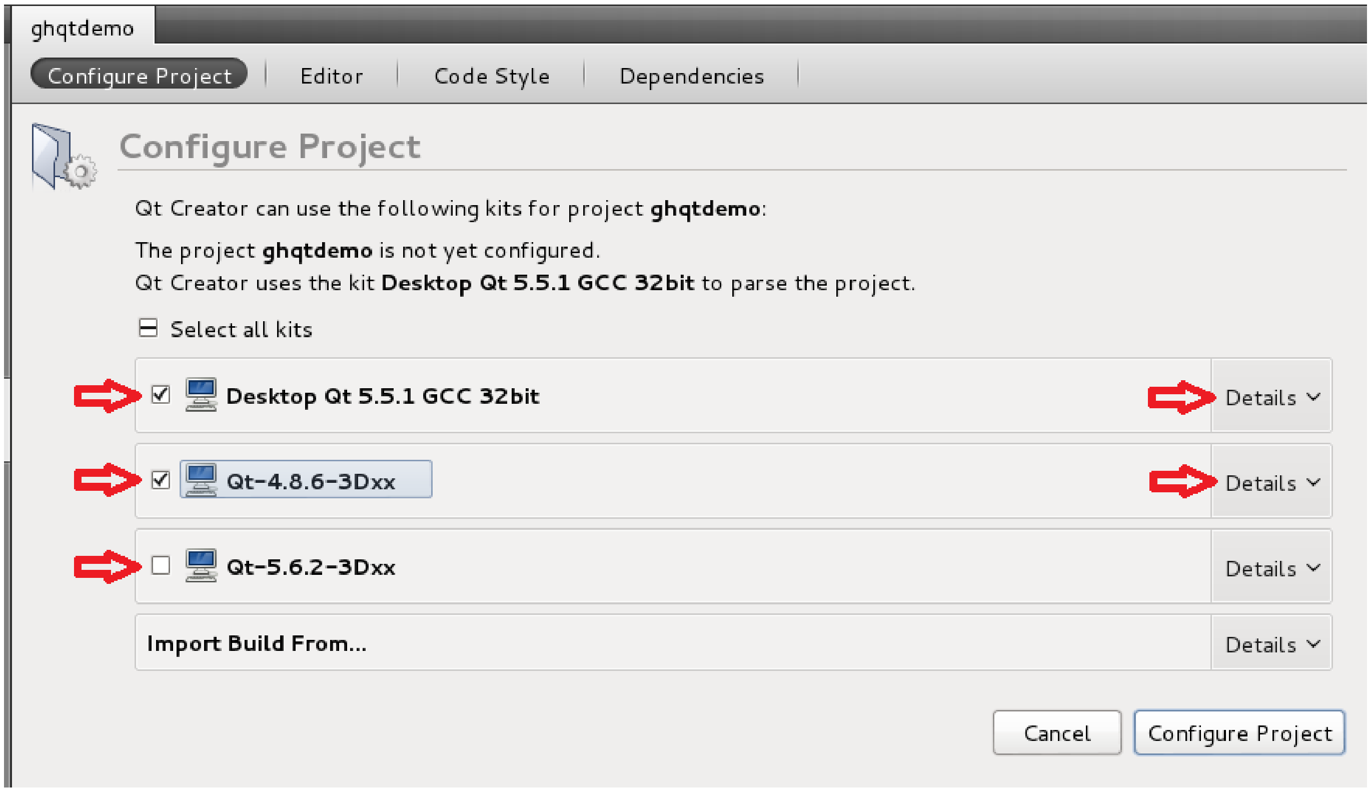
- From "Qt Creator" main window click on "Open Project" button as shown here:



- An "Open File" dialog window will appear. Navigate to the 3Dxx Demo project's ".pro" file as shown in the example below and then click on "Open" button. The example shown below is for "ghqtdemo"; for "gh7indemo", navigate to folder "gh7indemo" and open file "gh7indemo.pro".



- If the "ghqtdemo.pro.user" file is missing, which is normal if the project has never been opened before, then a "Configure Project" dialog will appear. If this dialog doesn't appear, then jump ahead to the step where the "Projects" icon is selected.
- If the "Configure Project" dialog does appear, then check boxes next to "Desktop Qt 5.5.1 GCC 32-bit" and "Qt-4.8.6-3Dxx" and uncheck box next to any other kits such as "Qt-5.6.2-3Dxx". Then click on "Details" button to the right of the "Desktop Qt 5.5.1 GCC 32-bit" and "Qt-4.8.6-3Dxx" kits.



Configure Project

Qt Creator can use the following kits for project **ghqtdemo**:

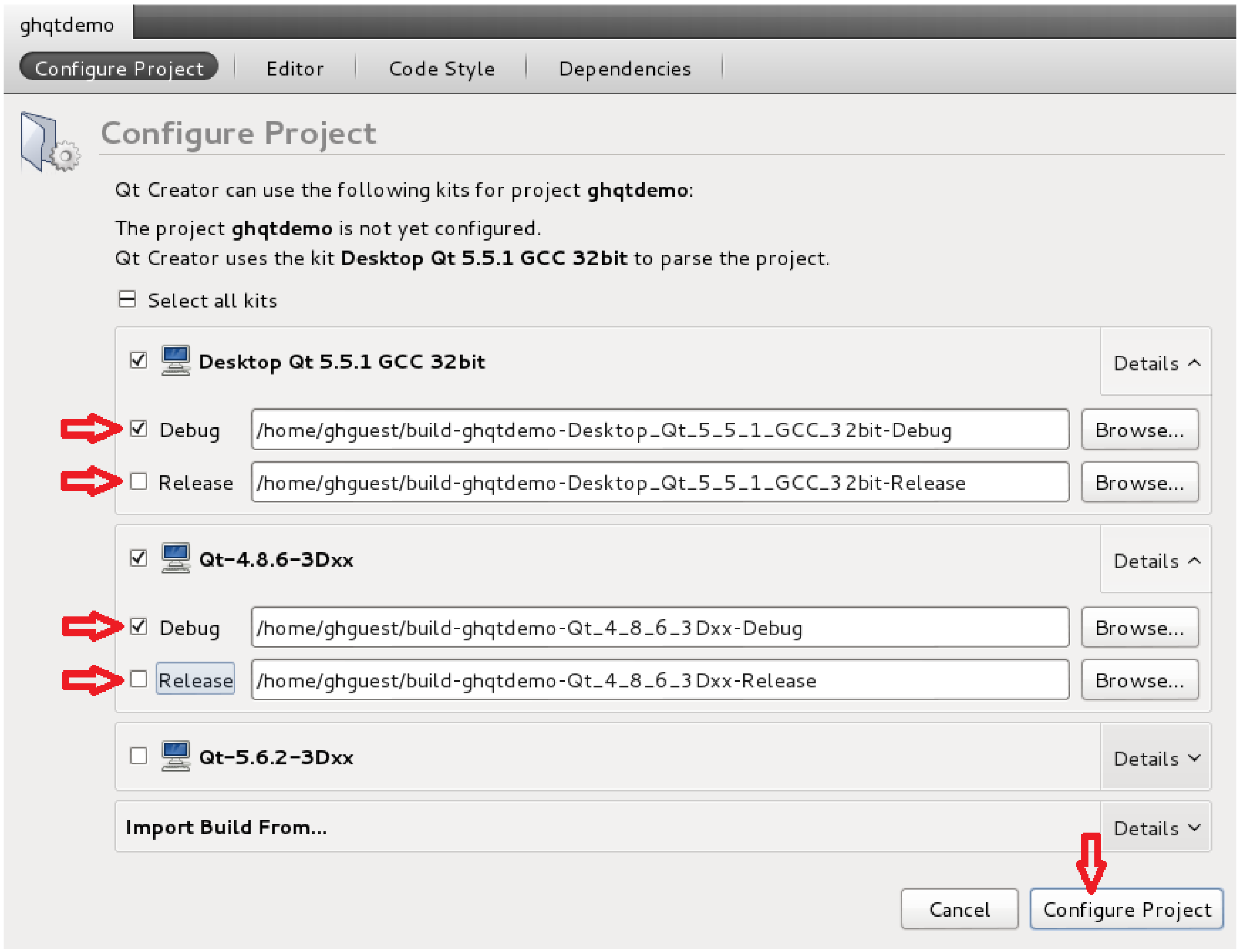
The project **ghqtdemo** is not yet configured.
Qt Creator uses the kit **Desktop Qt 5.5.1 GCC 32bit** to parse the project.

☐ Select all kits

- [✓] **Desktop Qt 5.5.1 GCC 32bit** — Details
- [✓] **Qt-4.8.6-3Dxx** — Details
- [ ] **Qt-5.6.2-3Dxx** — Details

Import Build From... — Details

- With the "Details" showing for the "Desktop Qt 5.5.1 GCC 32-bit" and "Qt-4.8.6-3Dxx" kits, make sure that the boxes next to "Debug" are checked and that the boxes next to "Release" are unchecked as shown below.
- When all is correct click on "Configure Project" button.



Qt Creator can use the following kits for project **ghqtdemo**:

The project **ghqtdemo** is not yet configured.
Qt Creator uses the kit **Desktop Qt 5.5.1 GCC 32bit** to parse the project.

⊟ Select all kits

[✓] **Desktop Qt 5.5.1 GCC 32bit** — Details ^

[✓] Debug — /home/ghguest/build-ghqtdemo-Desktop_Qt_5_5_1_GCC_32bit-Debug — Browse...

[ ] Release — /home/ghguest/build-ghqtdemo-Desktop_Qt_5_5_1_GCC_32bit-Release — Browse...

[✓] **Qt-4.8.6-3Dxx** — Details ^

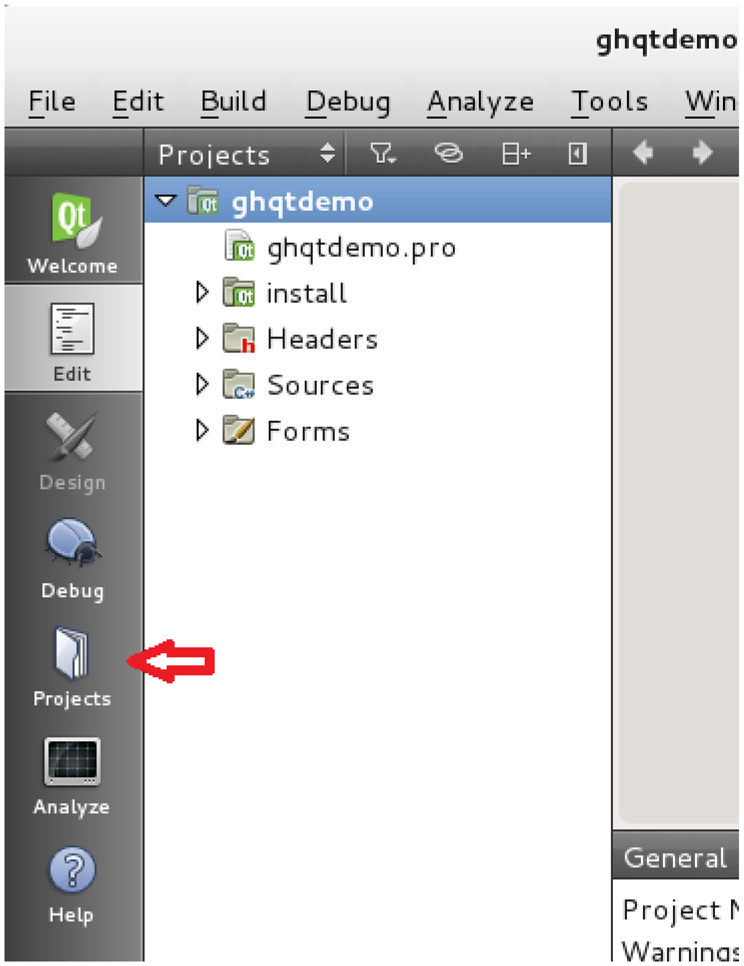
[✓] Debug — /home/ghguest/build-ghqtdemo-Qt_4_8_6_3Dxx-Debug — Browse...

[ ] Release — /home/ghguest/build-ghqtdemo-Qt_4_8_6_3Dxx-Release — Browse...

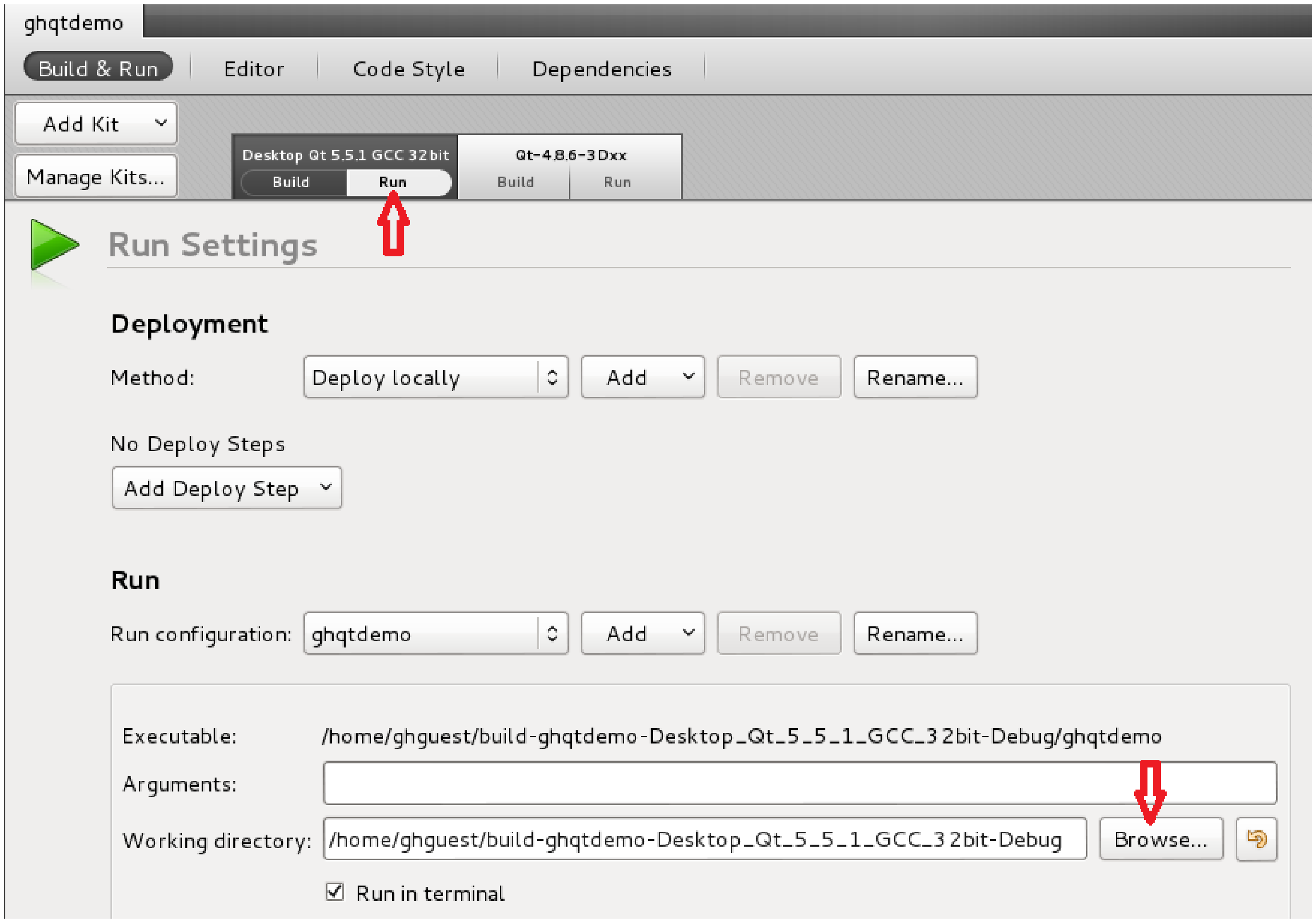
[ ] **Qt-5.6.2-3Dxx** — Details ⌄

Import Build From... — Details ⌄

- On the main "Qt Creator" window click on the "Projects" icon on the left side of the window as shown here:



- If the kits shown below do not appear, click on the “Add Kit” button to add the kit that is missing.
- On the “Projects” screen click on the “Run” option under the “Desktop Qt 5.5.1 GCC 32-bit” kit.
- Click on the “Browse” button to the right of the “Working Directory” box.



Method: Deploy locally

No Deploy Steps

Run configuration: ghqtdemo

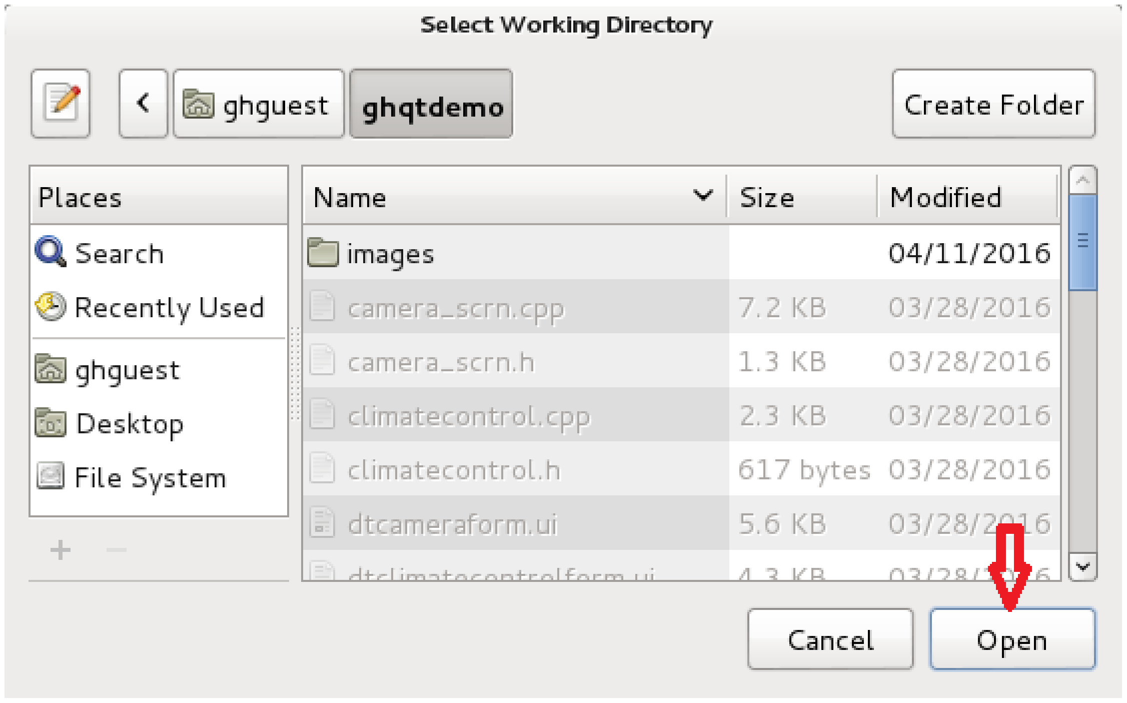
Executable: /home/ghguest/build-ghqtdemo-Desktop_Qt_5_5_1_GCC_32bit-Debug/ghqtdemo

Arguments:

Working directory: /home/ghguest/build-ghqtdemo-Desktop_Qt_5_5_1_GCC_32bit-Debug

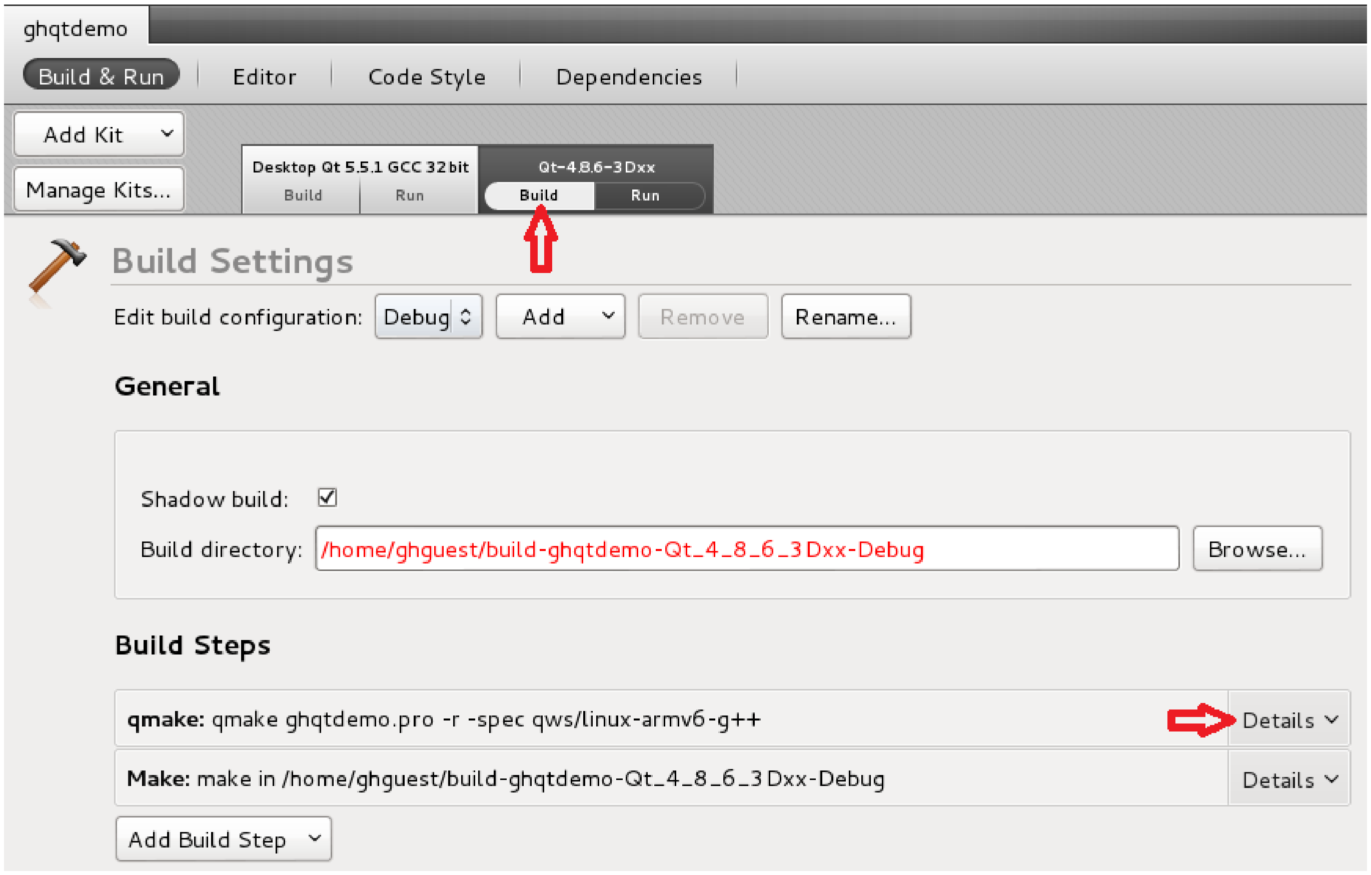
☑ Run in terminal

- When the “Select Working Directory” dialog appears, navigate to the folder “/home/ghguest/ghqtdemo” and then click on the “Open” button. For the “gh7indemo” program navigate to the folder “/home/ghguest/gh7indemo” instead.



Name	Size	Modified
images		04/11/2016
camera_scrn.cpp	7.2 KB	03/28/2016
camera_scrn.h	1.3 KB	03/28/2016
climatecontrol.cpp	2.3 KB	03/28/2016
climatecontrol.h	617 bytes	03/28/2016
dtcameraform.ui	5.6 KB	03/28/2016

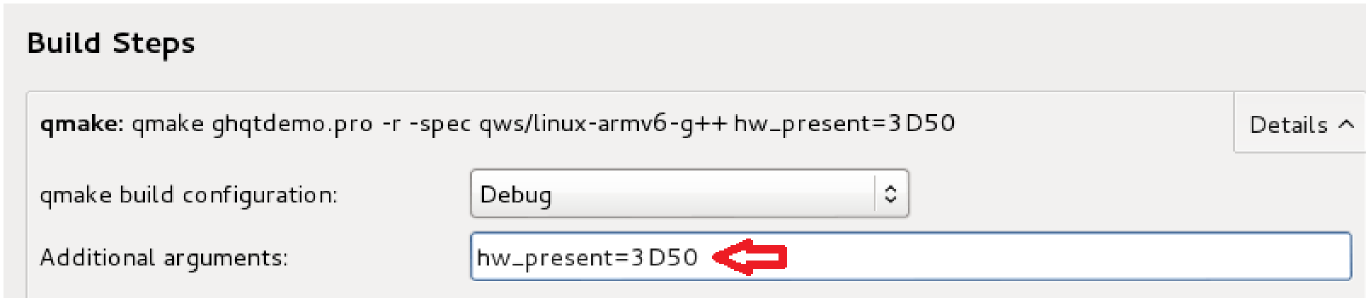
- Next click on the "Build" option under the "Qt-4.8.6-3Dxx" kit.
- Click on the "Details" button to the right of the "qmake:" box under the "Build Steps" group.



Shadow build: ✓
Build directory: /home/ghguest/build-ghqtdemo-Qt_4_8_6_3Dxx-Debug

qmake: qmake ghqtdemo.pro -r -spec qws/linux-armv6-g++
Make: make in /home/ghguest/build-ghqtdemo-Qt_4_8_6_3Dxx-Debug

- In the "Additional arguments:" box type "hw_present=3D50" as shown below. For the "gh7indemo" program enter "hw_present=3D70" instead.



qmake: qmake ghqtdemo.pro -r -spec qws/linux-armv6-g++ hw_present=3D50
qmake build configuration: Debug
Additional arguments: hw_present=3D50

- Next click on the "Run" option under the "Qt-4.8.6-3Dxx" kit. Then scroll down to the "Run" area.
- In the box labeled "Arguments:" type the appropriate entry for the chosen demo program:

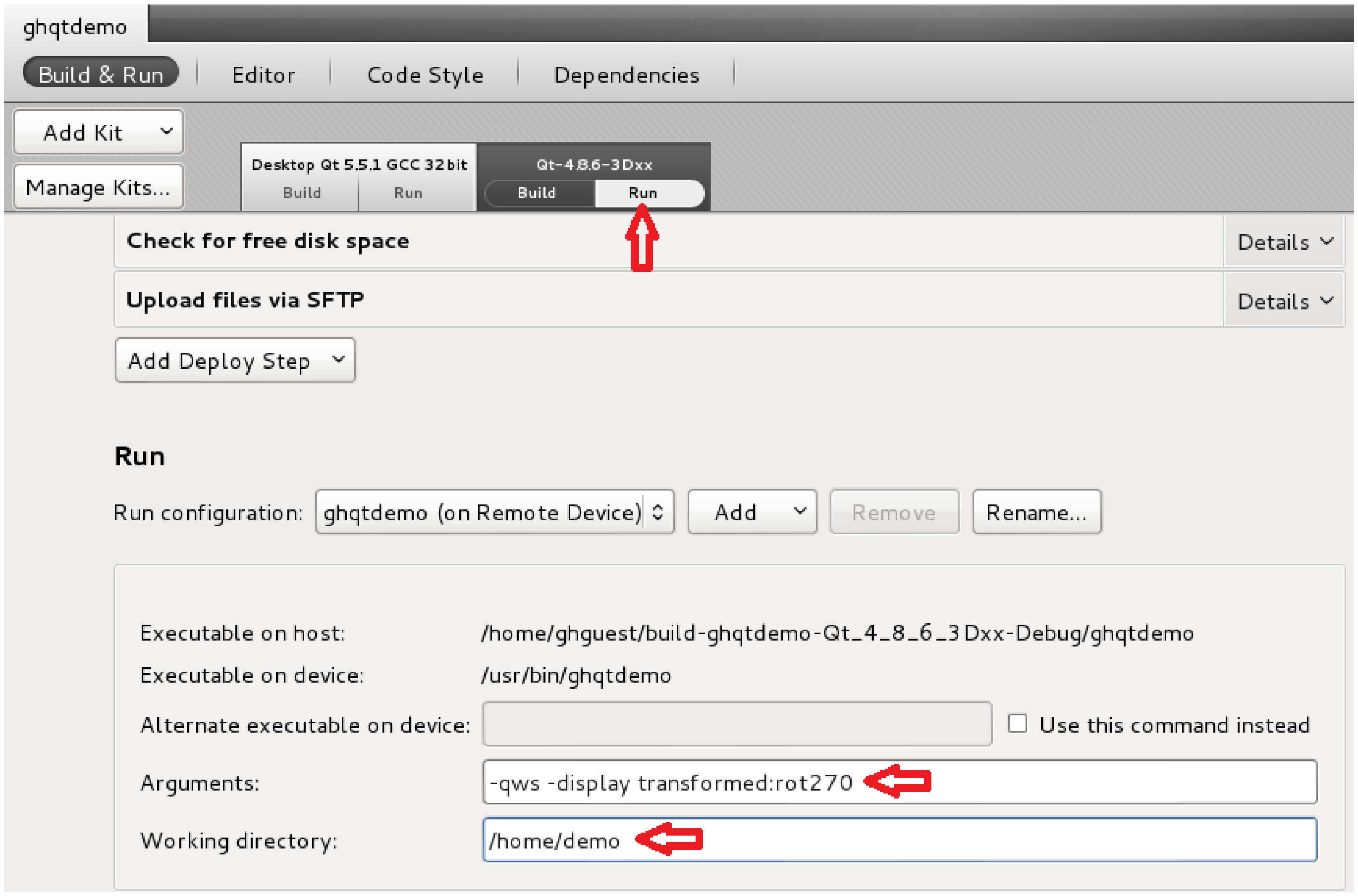
ghqtdemo	`-qws -display transformed:rot270`
gh7indemo	`-qws -display transformed:rot0 -display LinuxFb:/dev/fb1`

The items for the "Arguments:" box shown in the table above show two very important pieces of information about how to set up a Qt-4.8.6 program. (This information is not applicable to Qt-5.6.2).

First consider the “rot270” and “rot0” items. The “rot270” argument sets up the display for portrait mode and the “rot0” argument sets up the display for landscape mode. If the display output is to be flipped 180 degrees it is also possible to specify “rot90” or “rot180”. Next consider the “-display LinuxFb:/dev/fb1” argument. This argument tells Qt to send its graphics output to frame buffer 1 instead of the default frame buffer 0. This is required if trying to do graphic overlays on the camera displays. See the section on camera programming for more information.

- In the box labeled “Working directory:” type the appropriate entry for the chosen demo program:

ghqtdemo	`/home/demo`
gh7indemo	`/home/demo7in`



Run

Run configuration: ghqtdemo (on Remote Device)

Executable on host: /home/ghguest/build-ghqtdemo-Qt_4_8_6_3Dxx-Debug/ghqtdemo

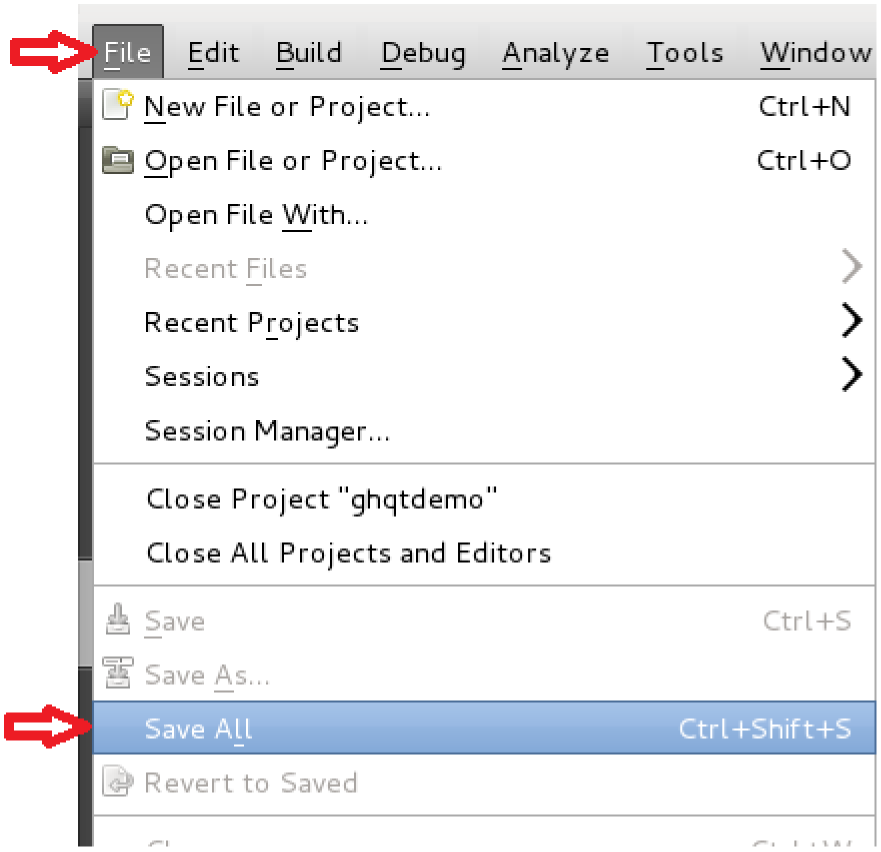
Executable on device: /usr/bin/ghqtdemo

Alternate executable on device:

Arguments: -qws -display transformed:rot270

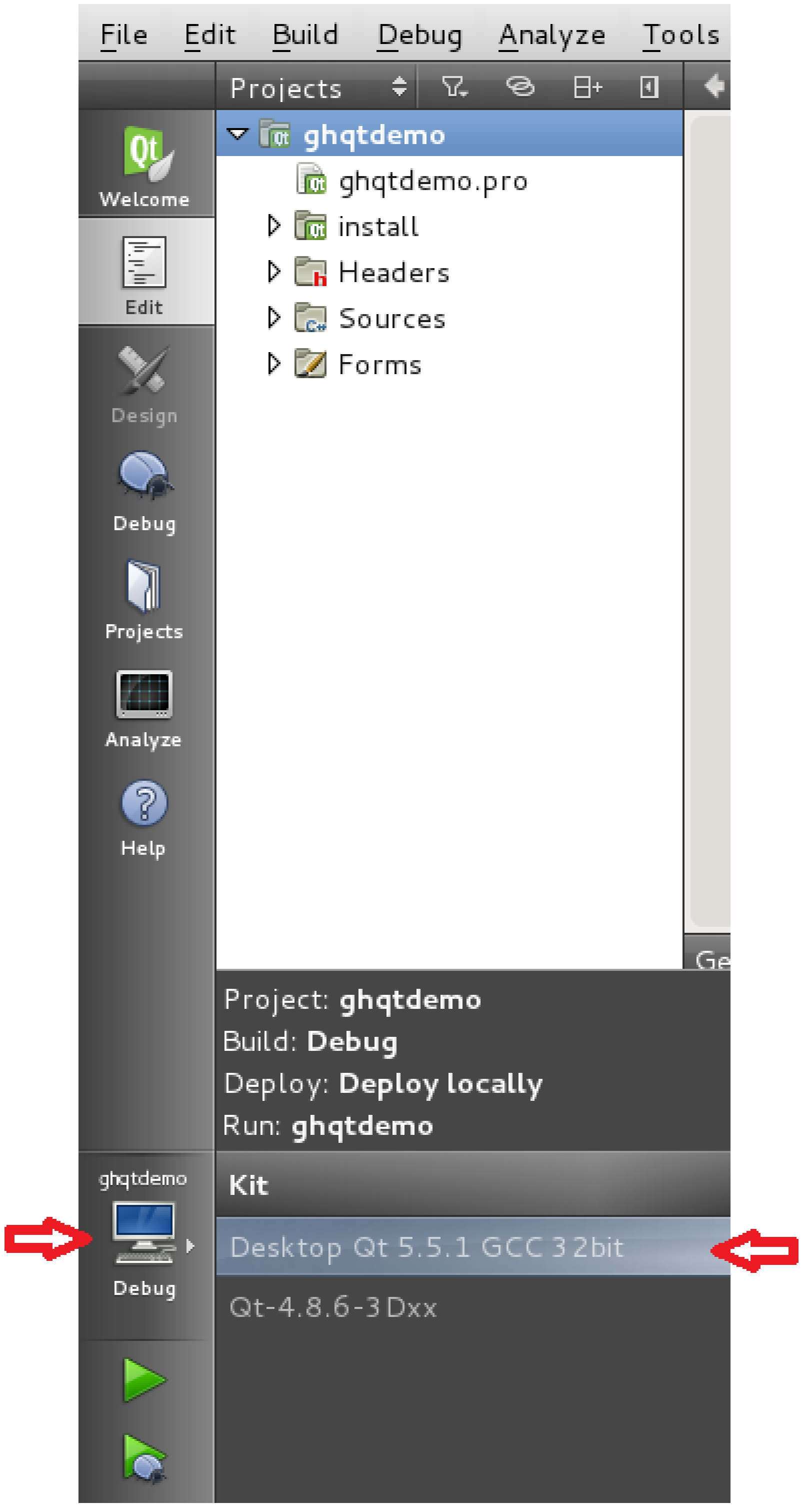
Working directory: /home/demo

- Next click on the “File” menu option in the upper left corner and click on “Save All” as shown here to save the project configurations:



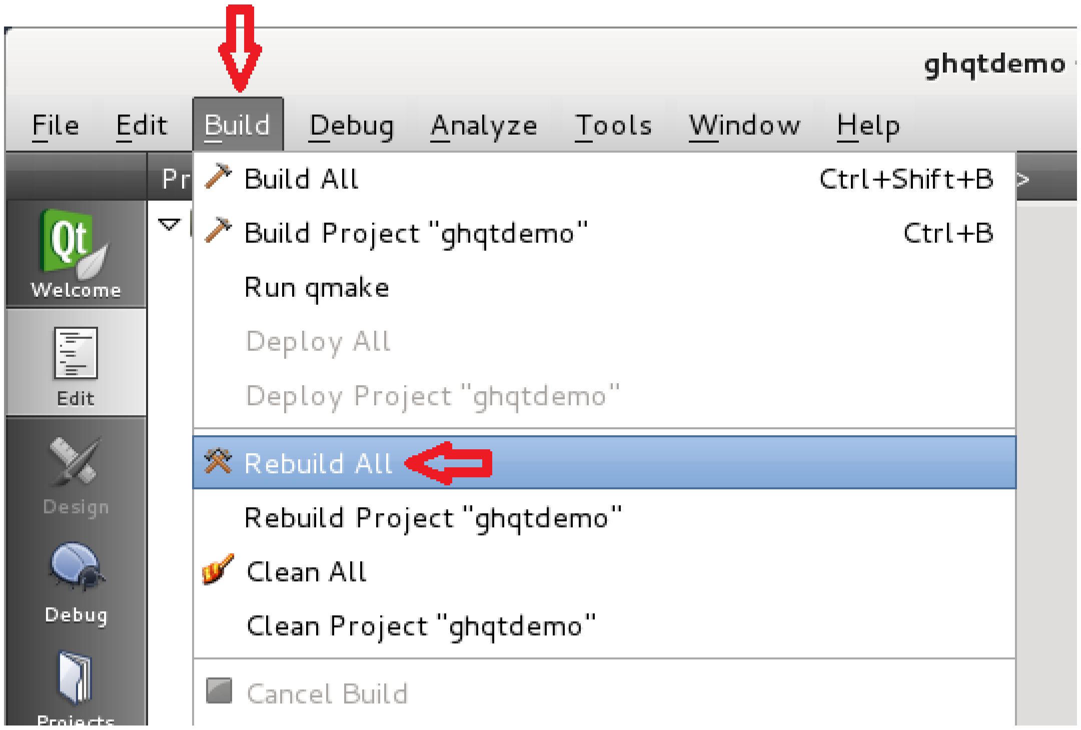
Build and Run 3Dxx Demo Project Desktop Version (Qt-5.5.1)

- The desktop version can be built by first selecting the “Desktop Qt 5.5.1 GCC 32-bit” item as shown below:

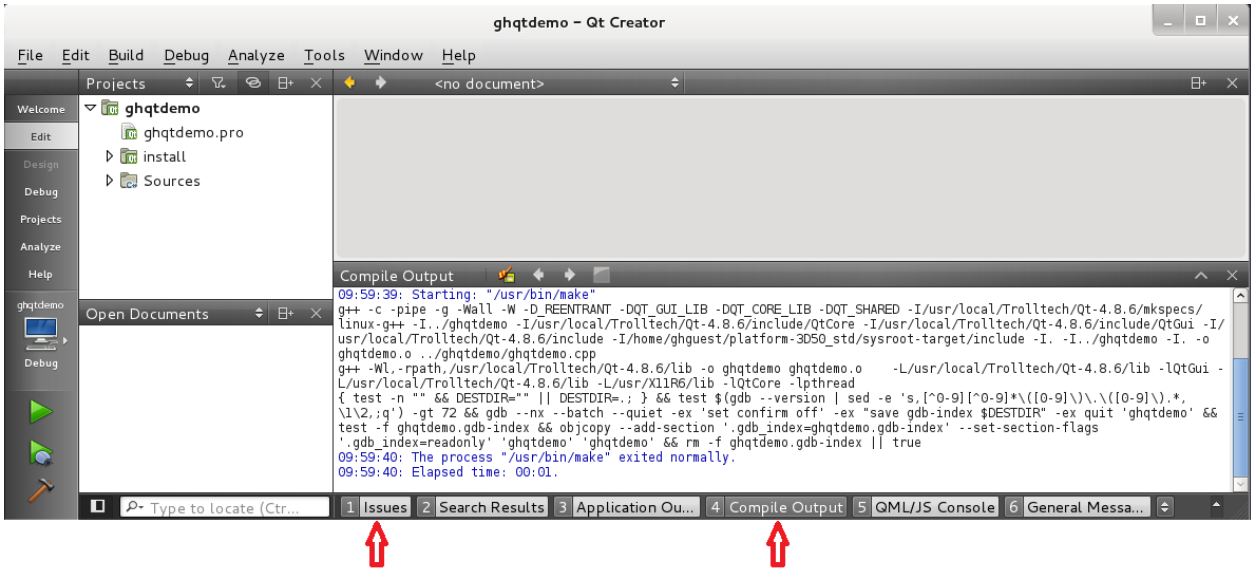


Project: **ghqtdemo**
Build: **Debug**
Deploy: **Deploy locally**
Run: **ghqtdemo**

- Next select the “Rebuild All” item under the “Build” menu item as shown here:

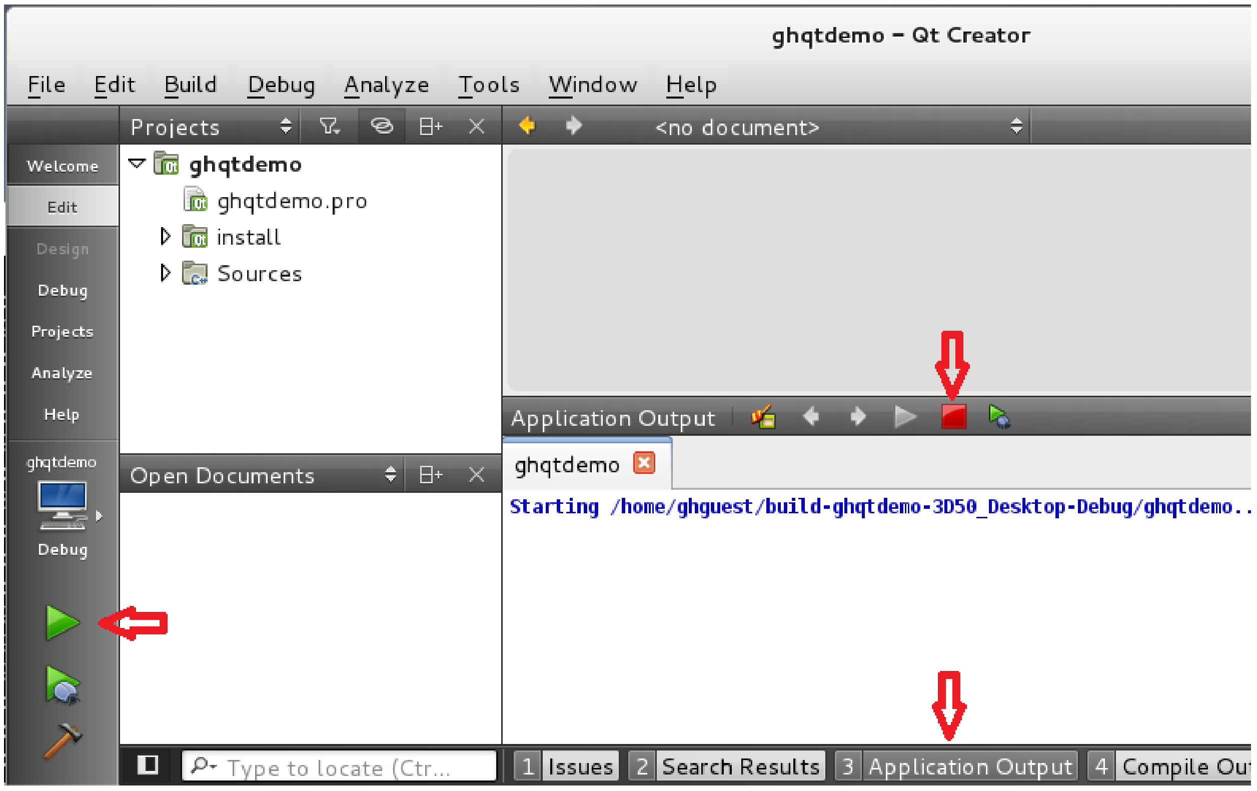


- Click on the “Compile Output” and “Issues” selectors on the bottom of the Qt Creator window to check for error messages and problems.



```
09:59:39: Starting: "/usr/bin/make"
g++ -c -pipe -g -Wall -W -D_REENTRANT -DQT_GUI_LIB -DQT_CORE_LIB -DQT_SHARED -I/usr/local/Trolltech/Qt-4.8.6/mkspecs/
linux-g++ -I../ghqtdemo -I/usr/local/Trolltech/Qt-4.8.6/include/QtCore -I/usr/local/Trolltech/Qt-4.8.6/include/QtGui -I/
usr/local/Trolltech/Qt-4.8.6/include -I/home/ghguest/platform-3D50_std/sysroot-target/include -I. -I../ghqtdemo -I. -o
ghqtdemo.o ../ghqtdemo/ghqtdemo.cpp
g++ -Wl,-rpath,/usr/local/Trolltech/Qt-4.8.6/lib -o ghqtdemo ghqtdemo.o    -L/usr/local/Trolltech/Qt-4.8.6/lib -lQtGui -
L/usr/local/Trolltech/Qt-4.8.6/lib -L/usr/X11R6/lib -lQtCore -lpthread
{ test -n "" && DESTDIR="" || DESTDIR=.; } && test $(gdb --version | sed -e 's,[^0-9][^0-9]*\([0-9]\)\.\([0-9]\).*,
\1\2,;q') -gt 72 && gdb --nx --batch --quiet -ex 'set confirm off' -ex "save gdb-index $DESTDIR" -ex quit 'ghqtdemo' &&
test -f ghqtdemo.gdb-index && objcopy --add-section '.gdb_index=ghqtdemo.gdb-index' --set-section-flags
'.gdb_index=readonly' 'ghqtdemo' 'ghqtdemo' && rm -f ghqtdemo.gdb-index || true
09:59:40: The process "/usr/bin/make" exited normally.
09:59:40: Elapsed time: 00:01.
```

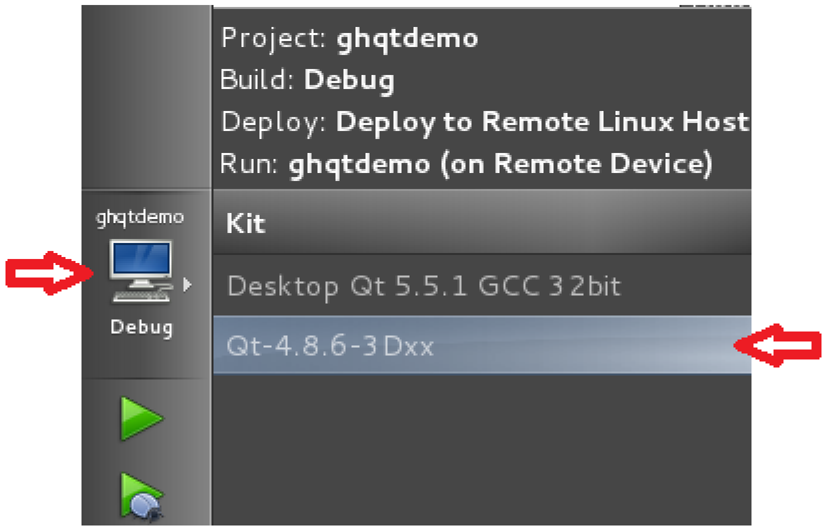
- The desktop version can now be run by clicking on the big green "Run" arrow on the lower left corner of the Qt Creator window.
- Click on the "Application Output" item on the bottom row to view application output.
- Click on red square on "Application Output" window to stop application.



```
Starting /home/ghguest/build-ghqtdemo-3D50_Desktop-Debug/ghqtdemo..
```

Build and Debug 3Dxx Demo Project Target Version (Qt-4.8.6)

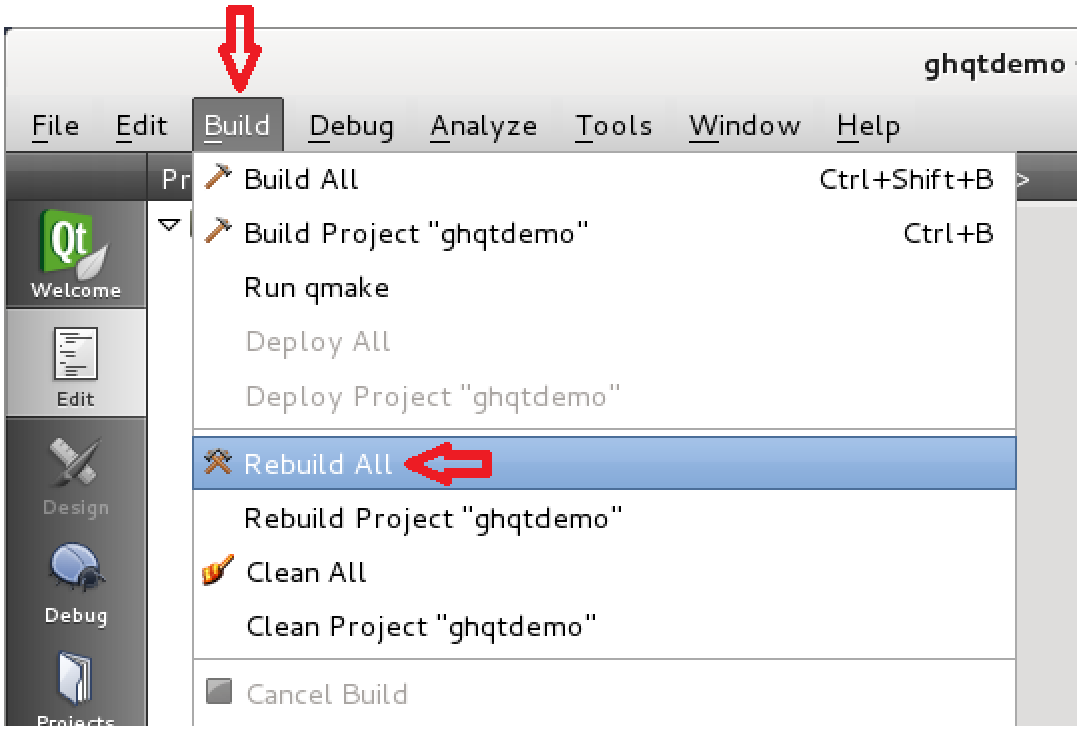
- The Qt-4.8.6 version of the 3Dxx demo project can be built by first selecting the "Qt-4.8.6-3Dxx" item as shown below:



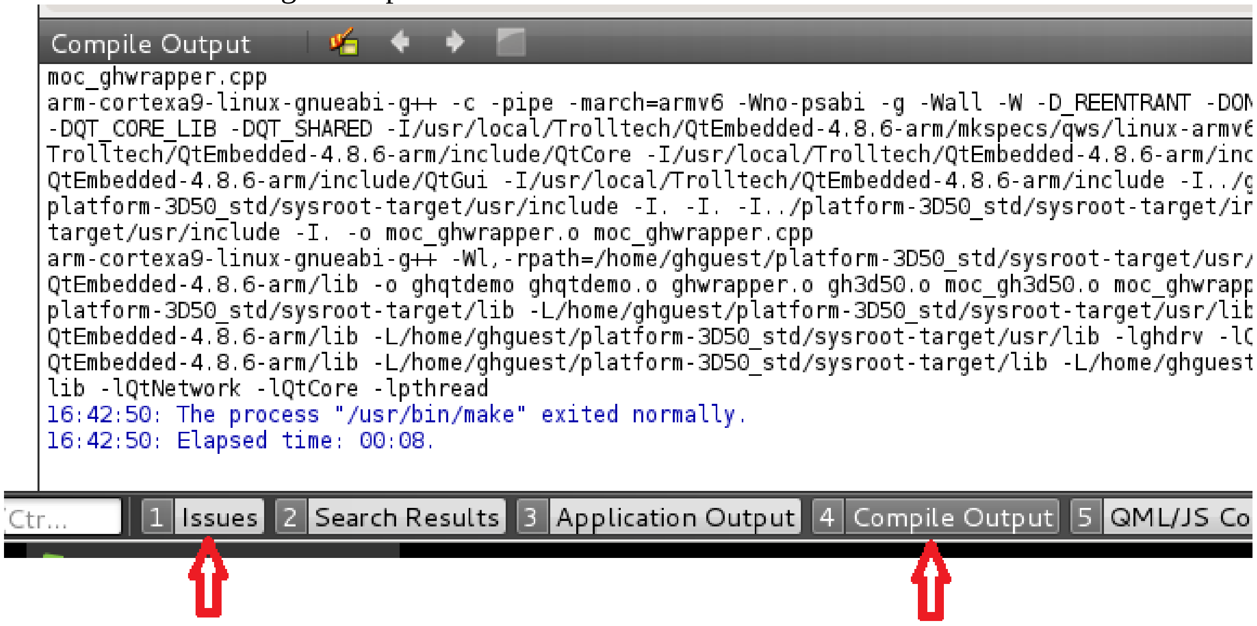
Project: **ghqtdemo**
Build: **Debug**
Deploy: **Deploy to Remote Linux Host**
Run: **ghqtdemo (on Remote Device)**

Kit
Desktop Qt 5.5.1 GCC 32bit
Qt-4.8.6-3Dxx

- Next select the "Rebuild All" item under the "Build" menu item as shown here:

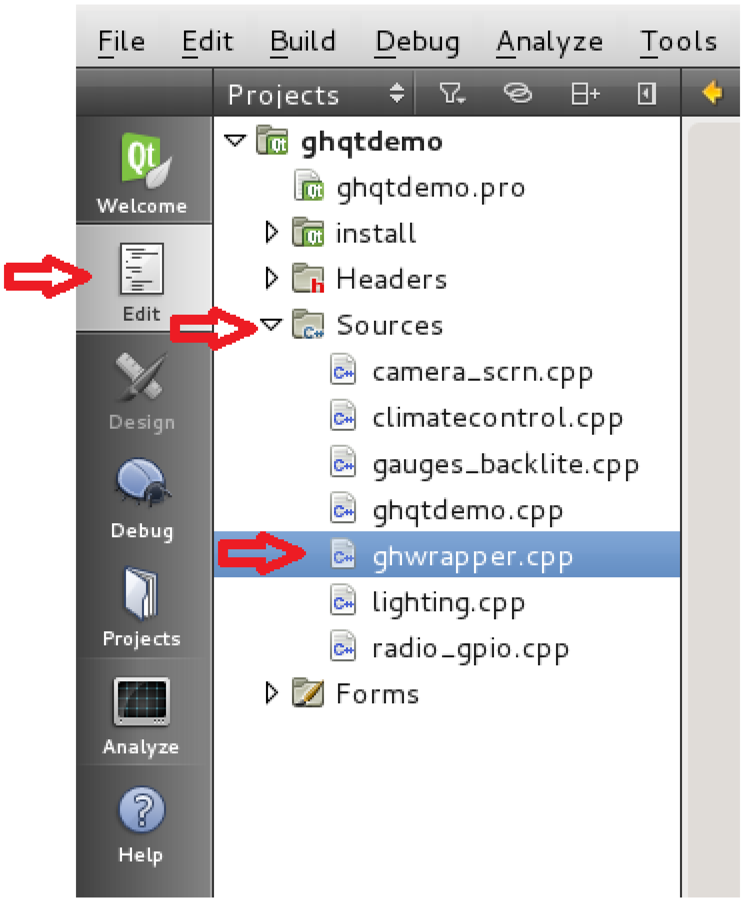


- Click on the “Compile Output” and “Issues” selectors on the bottom of the Qt Creator window to check for error messages and problems.

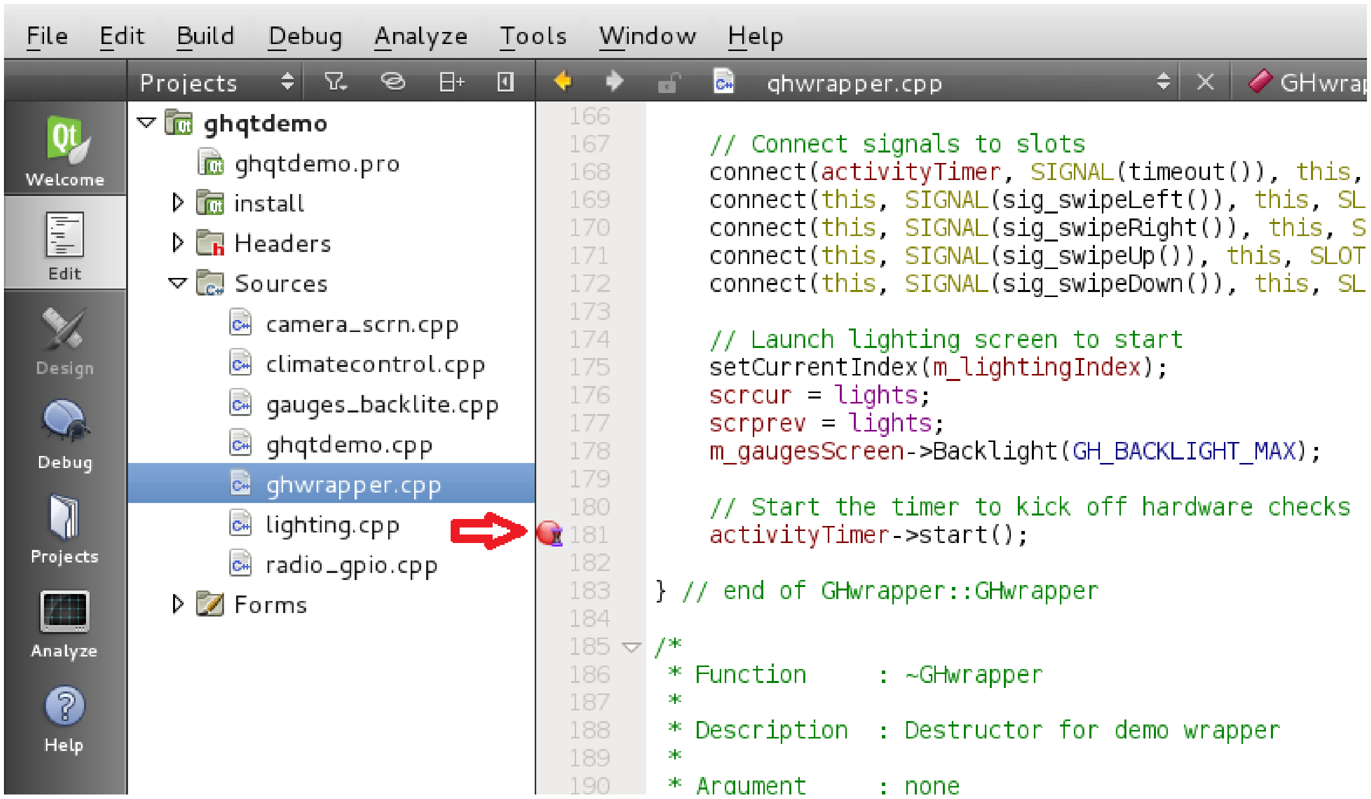


```
moc_ghwrapper.cpp
arm-cortexa9-linux-gnueabi-g++ -c -pipe -march=armv6 -Wno-psabi -g -Wall -W -D_REENTRANT -DON
-DQT_CORE_LIB -DQT_SHARED -I/usr/local/Trolltech/QtEmbedded-4.8.6-arm/mkspecs/qws/linux-armv6
Trolltech/QtEmbedded-4.8.6-arm/include/QtCore -I/usr/local/Trolltech/QtEmbedded-4.8.6-arm/inc
QtEmbedded-4.8.6-arm/include/QtGui -I/usr/local/Trolltech/QtEmbedded-4.8.6-arm/include -I../g
platform-3D50_std/sysroot-target/usr/include -I. -I. -I../platform-3D50_std/sysroot-target/ir
target/usr/include -I. -o moc_ghwrapper.o moc_ghwrapper.cpp
arm-cortexa9-linux-gnueabi-g++ -Wl,-rpath=/home/ghguest/platform-3D50_std/sysroot-target/usr/
QtEmbedded-4.8.6-arm/lib -o ghqtdemo ghqtdemo.o ghwrapper.o gh3d50.o moc_gh3d50.o moc_ghwrapp
platform-3D50_std/sysroot-target/lib -L/home/ghguest/platform-3D50_std/sysroot-target/usr/lib
QtEmbedded-4.8.6-arm/lib -L/home/ghguest/platform-3D50_std/sysroot-target/usr/lib -lghdrv -lQ
QtEmbedded-4.8.6-arm/lib -L/home/ghguest/platform-3D50_std/sysroot-target/lib -L/home/ghguest
lib -lQtNetwork -lQtCore -lpthread
16:42:50: The process "/usr/bin/make" exited normally.
16:42:50: Elapsed time: 00:08.
```

- The Qt-4.8.6-3Dxx Demo Project Target version is now ready to run.
- Set a breakpoint in the code by clicking on the “Edit” option on the left hand side of the Qt Creator window. Then click on the expansion arrow next to “Sources”.
- Double click on the “ghwrapper.cpp” file in order to open it in the editor screen to the right.

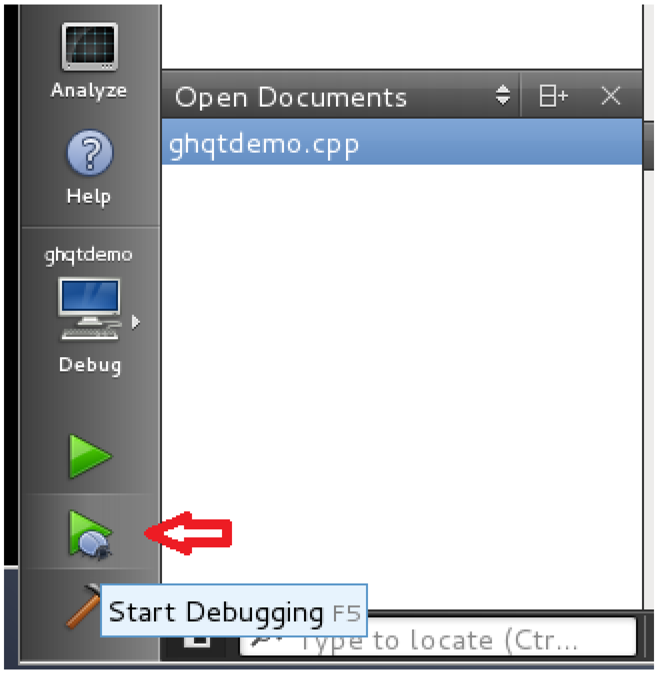


- Scroll down and select a line in the source code and then press the “F9” key to toggle on a breakpoint. An example is shown here:

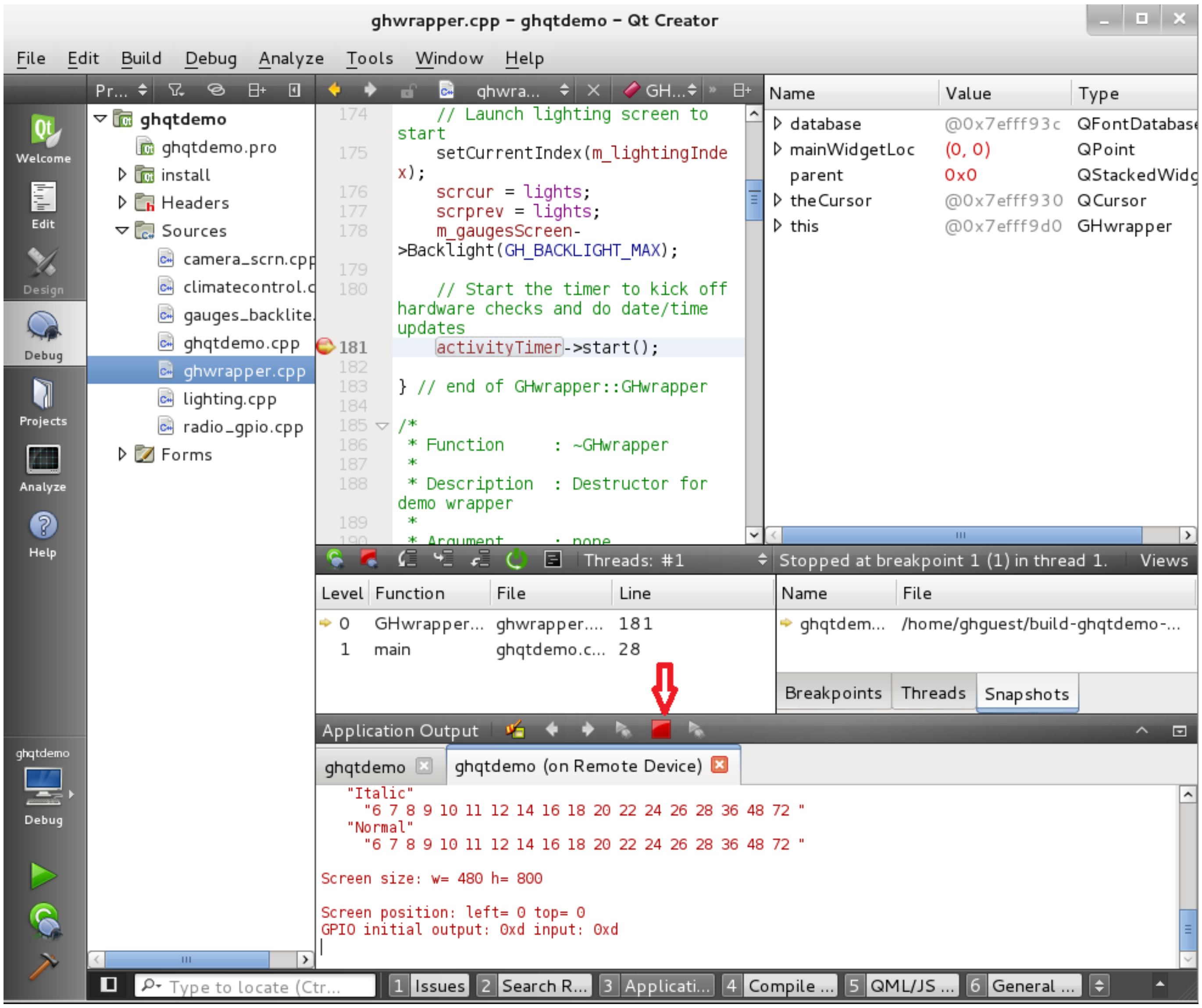


```
    // Connect signals to slots
    connect(activityTimer, SIGNAL(timeout()), this,
    connect(this, SIGNAL(sig_swipeLeft()), this, SL
    connect(this, SIGNAL(sig_swipeRight()), this, S
    connect(this, SIGNAL(sig_swipeUp()), this, SLOT
    connect(this, SIGNAL(sig_swipeDown()), this, SL

    // Launch lighting screen to start
    setCurrentIndex(m_lightingIndex);
    scrcur = lights;
    scrprev = lights;
    m_gaugesScreen->Backlight(GH_BACKLIGHT_MAX);

    // Start the timer to kick off hardware checks
    activityTimer->start();

} // end of GHwrapper::GHwrapper

/*
 * Function     : ~GHwrapper
 *
 * Description  : Destructor for demo wrapper
 *
 * Argument     : none
```

- Make sure that 3Dxx Display is powered up and that its Ethernet connection to the Qt host is working.
- Start debugging by clicking on the “Start Debugging” arrow as shown here or press the “F5” key:



- Qt will download the executable to the target along with any necessary support files and begin execution.
- Execution will stop when breakpoint is reached. At that time the Qt Creator window will look something like this:



```
    // Launch lighting screen to
start
    setCurrentIndex(m_lightingInde
x);
    scrcur = lights;
    scrprev = lights;
    m_gaugesScreen-
>Backlight(GH_BACKLIGHT_MAX);

    // Start the timer to kick off
hardware checks and do date/time
updates
    activityTimer->start();

} // end of GHwrapper::GHwrapper

/*
 * Function     : ~GHwrapper
 *
 * Description  : Destructor for
demo wrapper
 *
 * Argument     : none
```

Name	Value	Type
database	@0x7efff93c	QFontDatabase
mainWidgetLoc	(0, 0)	QPoint
parent	0x0	QStackedWidg
theCursor	@0x7efff930	QCursor
this	@0x7efff9d0	GHwrapper

Level	Function	File	Line
0	GHwrapper...	ghwrapper....	181
1	main	ghqtdemo.c...	28

Name	File
ghqtdem...	/home/ghguest/build-ghqtdemo-...

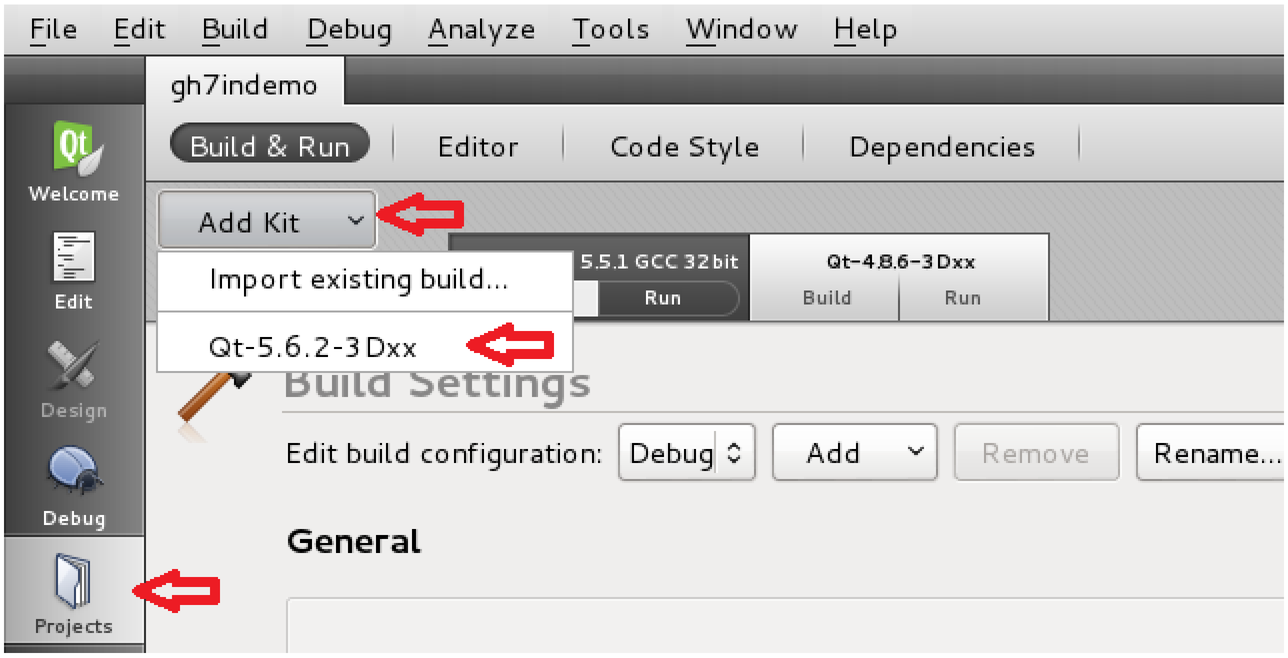
```
  "Italic"
     "6 7 8 9 10 11 12 14 16 18 20 22 24 26 28 36 48 72 "
  "Normal"
     "6 7 8 9 10 11 12 14 16 18 20 22 24 26 28 36 48 72 "

Screen size: w= 480 h= 800

Screen position: left= 0 top= 0
GPIO initial output: 0xd input: 0xd
```

- Execution can be continued by pressing the “F5” key.
- Execution can be terminated by clicking on the red stop square (highlighted by red arrow in above screen shot) on the “Application Output” window.

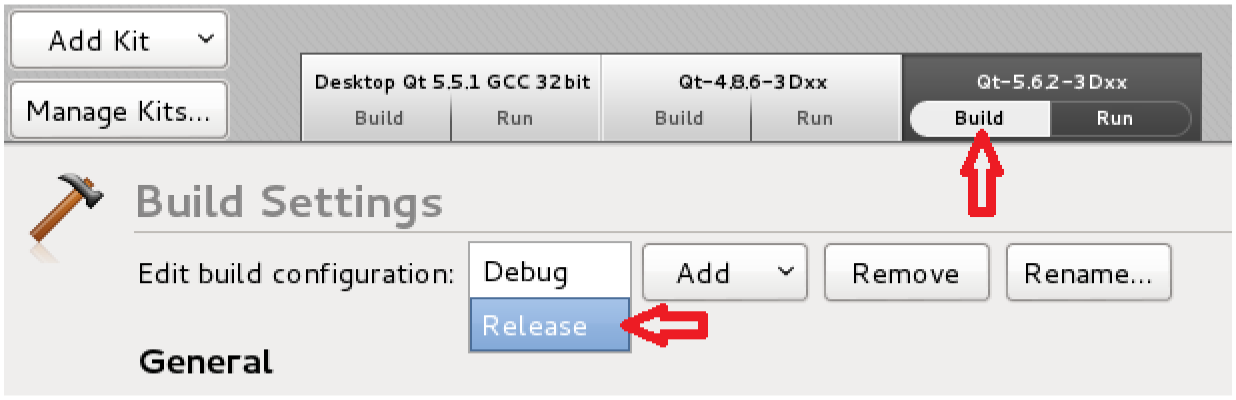
Configure a 3Dxx Demo Project to Build and Run (Qt-5.6.2)

These instructions explain how to configure the “gh7indemo” program to run on a 3Dxx Display using Qt-5.6.2. The instructions are also applicable to demonstration program “gh10indemo”, with noted differences. The example and screen shots shown below are for the “gh7indemo” demonstration program. This configuration procedure is very similar to one described in section **Configure a 3Dxx Demo Project to Build and Run (Qt-4.8.6)** and assumes that those instructions have already been completed.

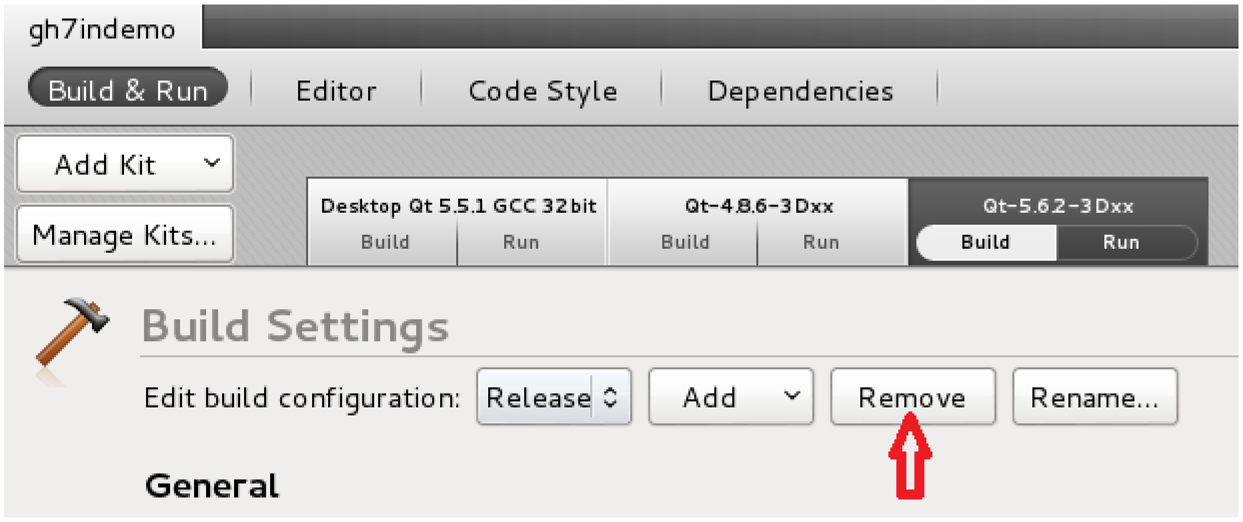
- Open the “gh7indemo” project in “Qt Creator”.
- On the main “Qt Creator” window click on the “Projects” icon on the left side of the window.
- On the “Projects” screen click on the “Add Kit” option and select the “Qt-5.6.2-3Dxx” kit.



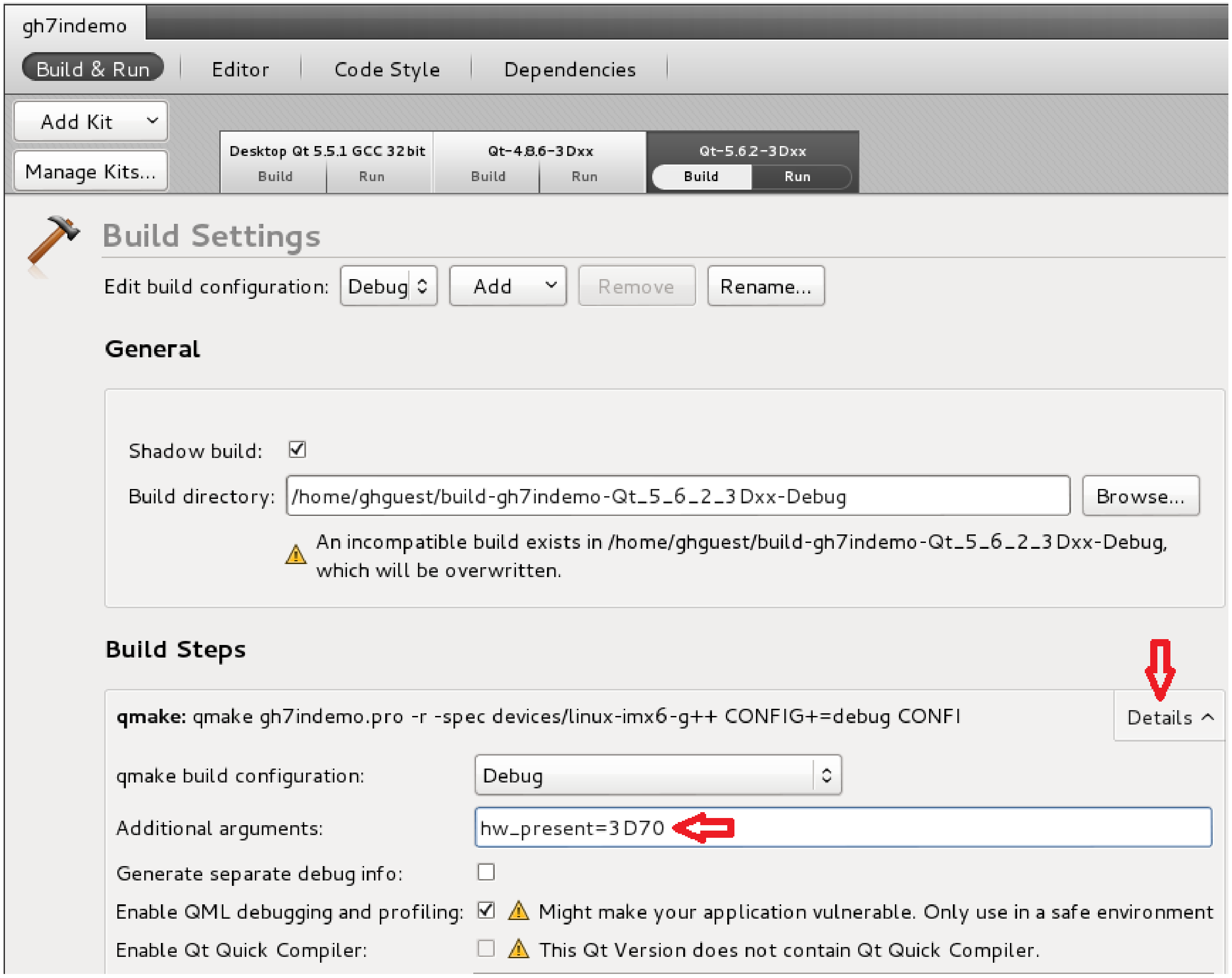
- Click on the “Build” button under the “Qt-5.6.2-3Dxx” kit and then select the “Release” configuration as shown here:



- Then click on the “Remove” button to remove the “Release” configuration..



- Click on the "Details" button to the right of the "qmake:" box under the "Build Steps" group. In the "Additional arguments:" box type "hw_present=3D70" as shown below.
- NOTE: for the gh10indemo program, use hw_present=3D2104 for the "Additional arguments".



Shadow build: ✓

Build directory: /home/ghguest/build-gh7indemo-Qt_5_6_2_3Dxx-Debug

An incompatible build exists in /home/ghguest/build-gh7indemo-Qt_5_6_2_3Dxx-Debug, which will be overwritten.

qmake: qmake gh7indemo.pro -r -spec devices/linux-imx6-g++ CONFIG+=debug CONFI

qmake build configuration: Debug

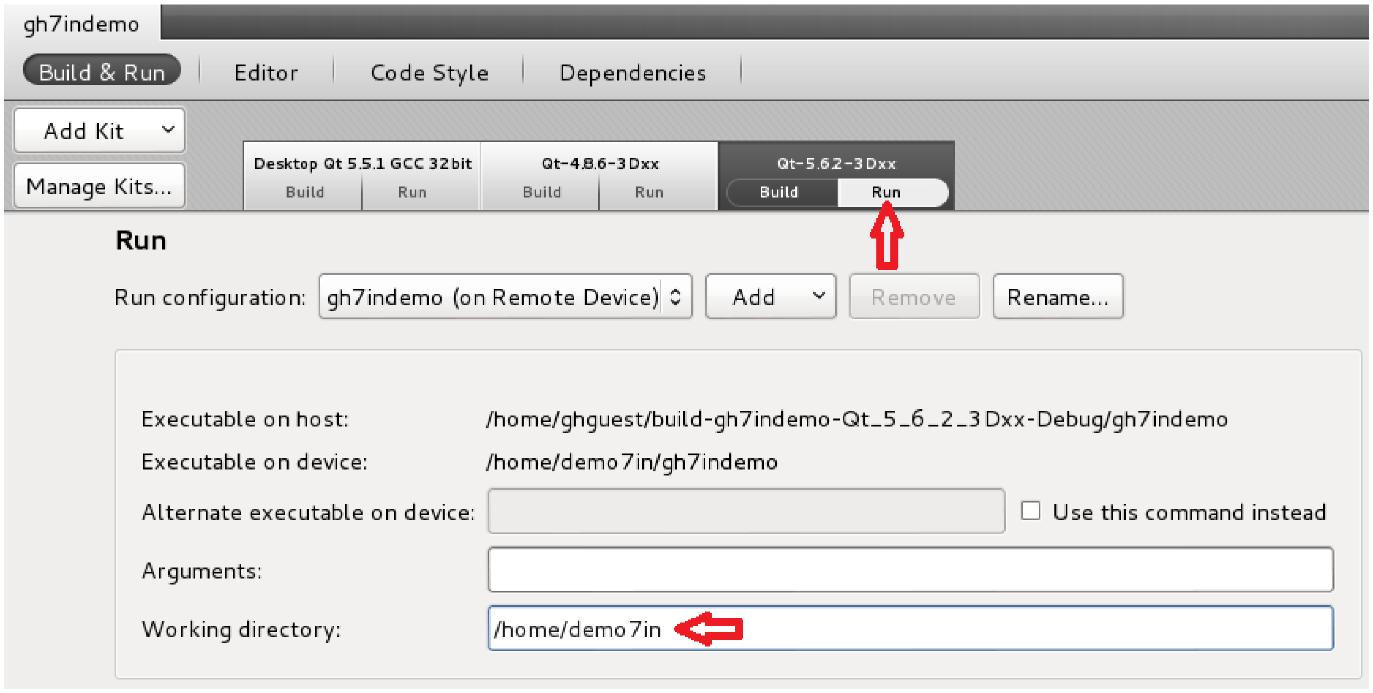
Additional arguments: hw_present=3D70

Generate separate debug info: ☐

Enable QML debugging and profiling: ✓ Might make your application vulnerable. Only use in a safe environment

Enable Qt Quick Compiler: ☐ This Qt Version does not contain Qt Quick Compiler.

- Next click on the "Run" option under the "Qt-5.6.2-3Dxx" kit and scroll down to the "Run" area.
- In the box labeled "Working directory:" type "/home/demo7in".
  (for the gh10indemo , type "/home/demo10in")



Run

Run configuration: gh7indemo (on Remote Device)

Executable on host: /home/ghguest/build-gh7indemo-Qt_5_6_2_3Dxx-Debug/gh7indemo

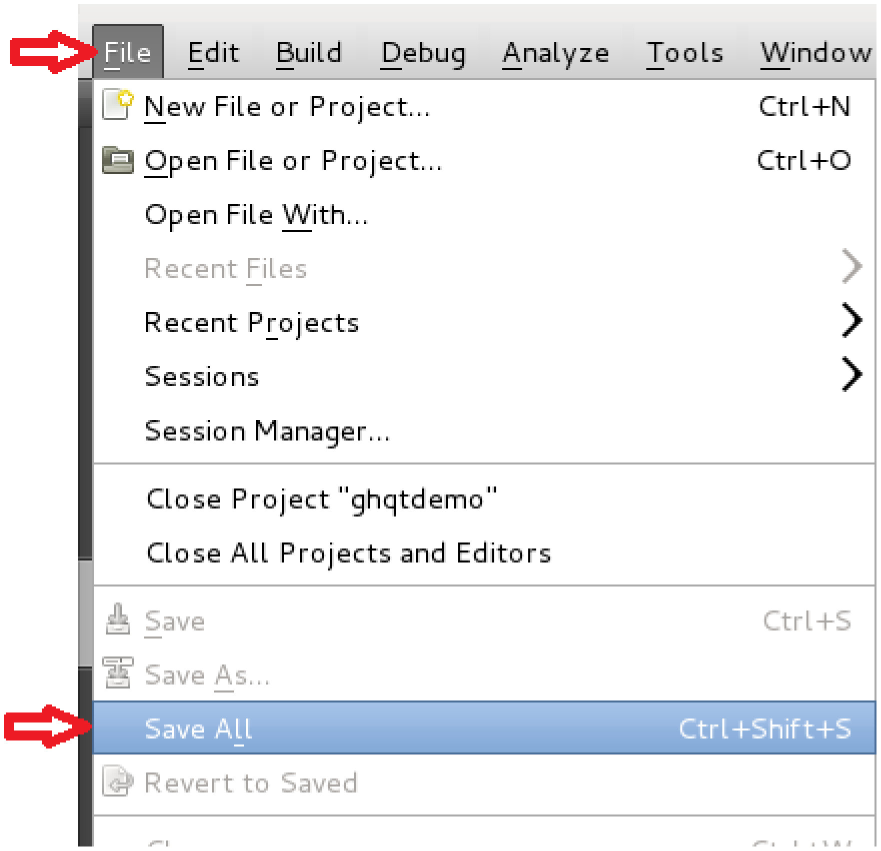
Executable on device: /home/demo7in/gh7indemo

Alternate executable on device: ☐ Use this command instead

Arguments:

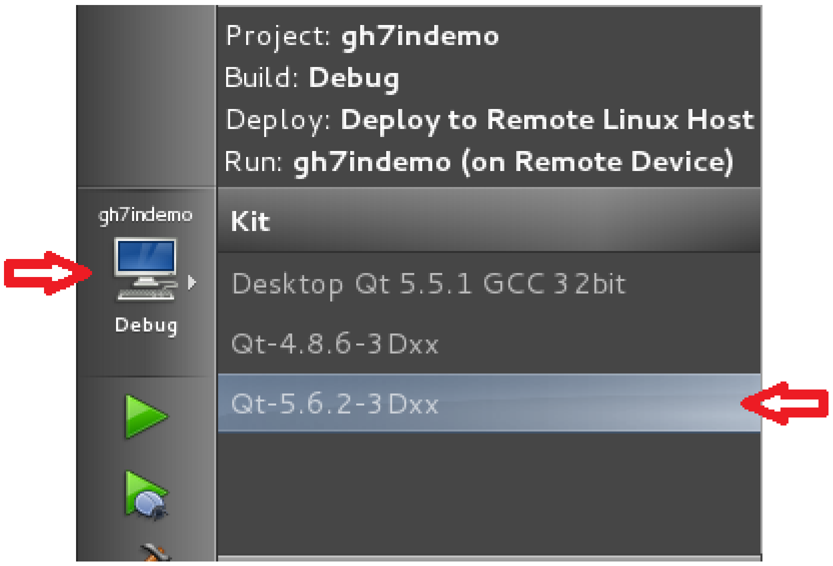
Working directory: /home/demo7in

- Next click on the “File” menu option in the upper left corner and click on “Save All” as shown here to save the project configurations:



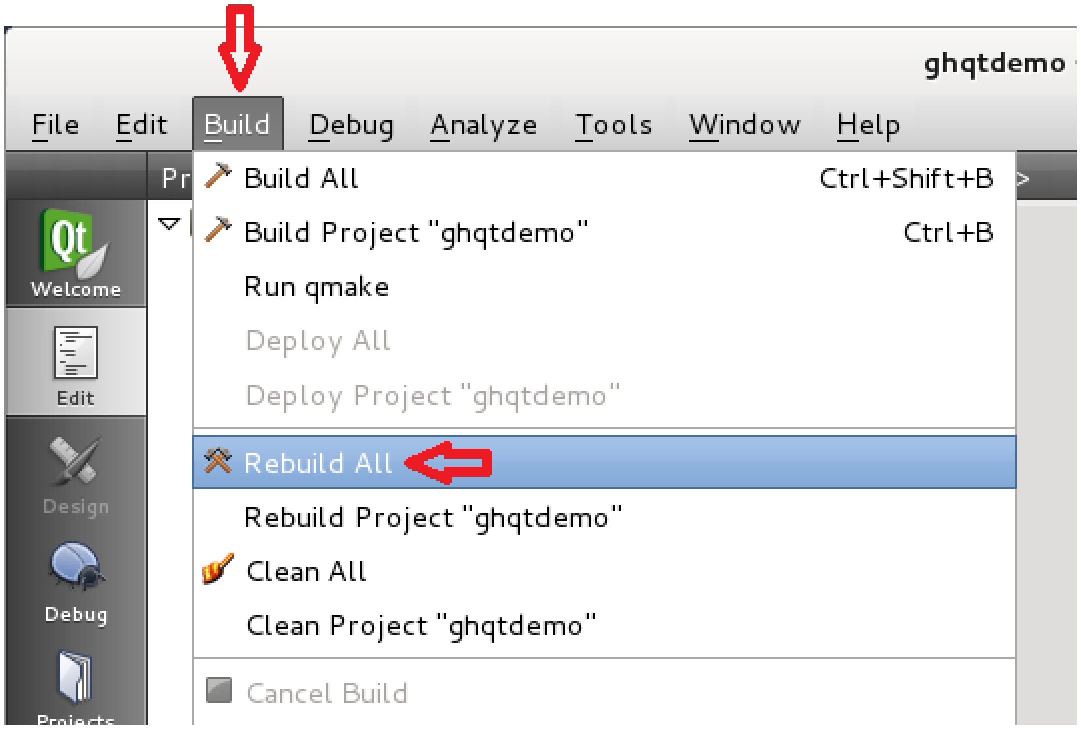
Build and Run 3Dxx Demo Project Target Version (Qt-5.6.2)

- The Qt-5.6.2 version of the 3Dxx demo project can be built by first selecting the “Qt-5.6.2-3Dxx” item as shown below:

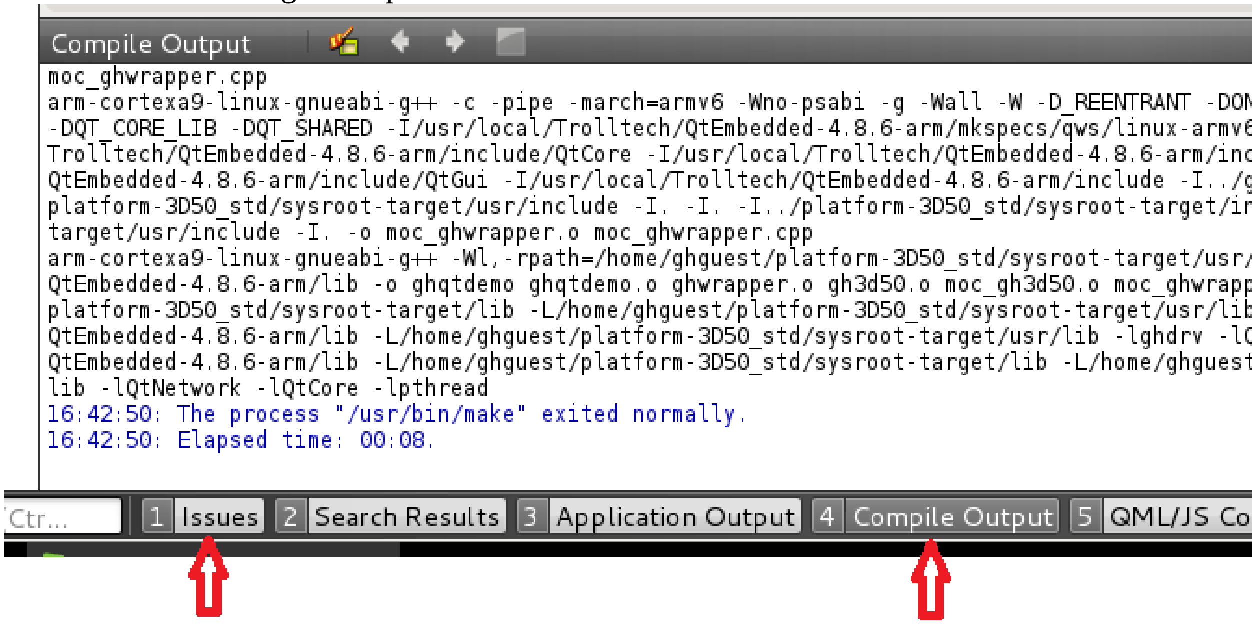


Project: gh7indemo
Build: Debug
Deploy: Deploy to Remote Linux Host
Run: gh7indemo (on Remote Device)

- Next select the “Rebuild All” item under the “Build” menu item as shown here:

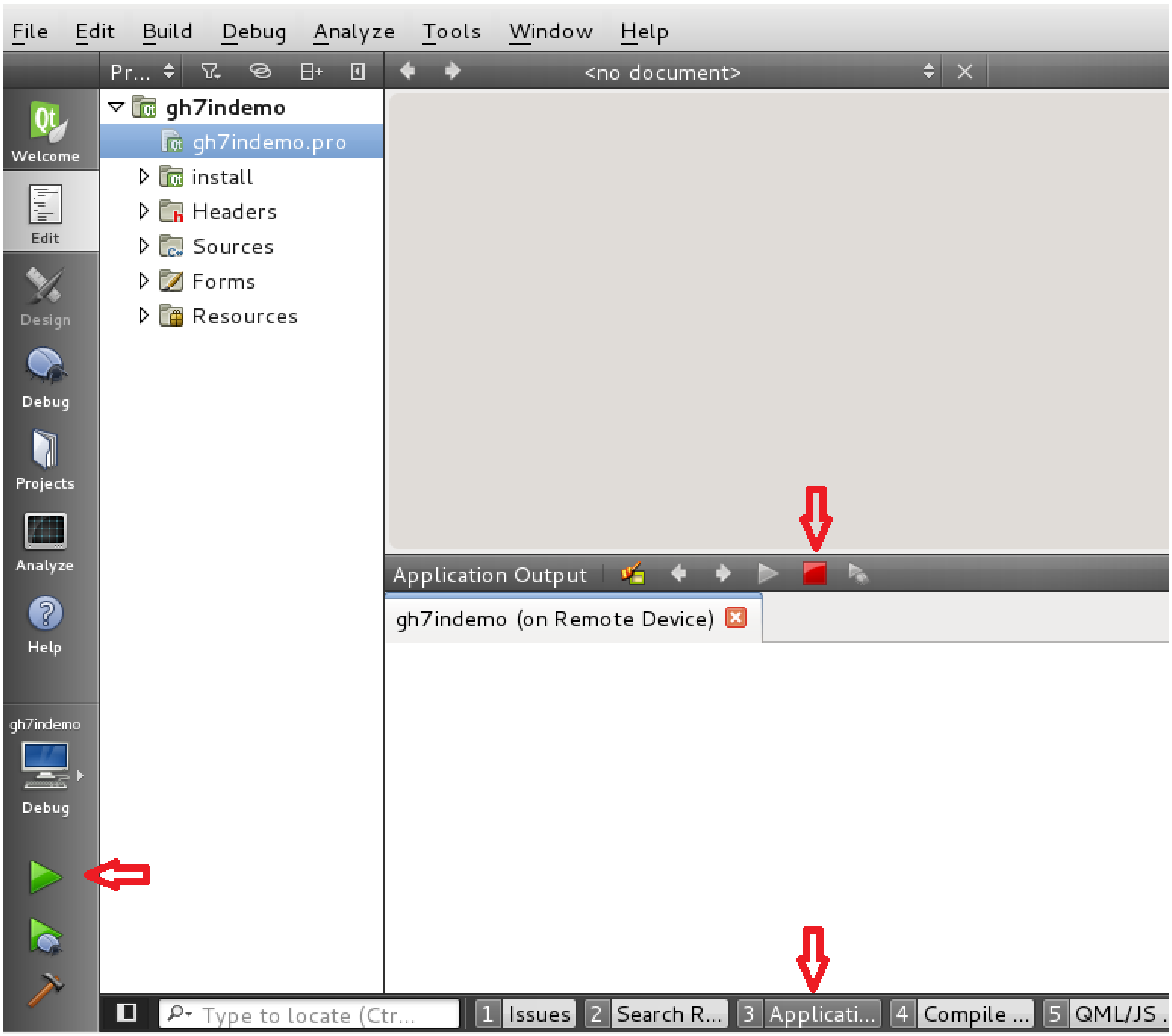


- Click on the “Compile Output” and “Issues” selectors on the bottom of the Qt Creator window to check for error messages and problems.



```
moc_ghwrapper.cpp
arm-cortexa9-linux-gnueabi-g++ -c -pipe -march=armv6 -Wno-psabi -g -Wall -W -D_REENTRANT -DON
-DQT_CORE_LIB -DQT_SHARED -I/usr/local/Trolltech/QtEmbedded-4.8.6-arm/mkspecs/qws/linux-armv6
Trolltech/QtEmbedded-4.8.6-arm/include/QtCore -I/usr/local/Trolltech/QtEmbedded-4.8.6-arm/inc
QtEmbedded-4.8.6-arm/include/QtGui -I/usr/local/Trolltech/QtEmbedded-4.8.6-arm/include -I../g
platform-3D50_std/sysroot-target/usr/include -I. -I. -I../platform-3D50_std/sysroot-target/ir
target/usr/include -I. -o moc_ghwrapper.o moc_ghwrapper.cpp
arm-cortexa9-linux-gnueabi-g++ -Wl,-rpath=/home/ghguest/platform-3D50_std/sysroot-target/usr/
QtEmbedded-4.8.6-arm/lib -o ghqtdemo ghqtdemo.o ghwrapper.o gh3d50.o moc_gh3d50.o moc_ghwrapp
platform-3D50_std/sysroot-target/lib -L/home/ghguest/platform-3D50_std/sysroot-target/usr/lib
QtEmbedded-4.8.6-arm/lib -L/home/ghguest/platform-3D50_std/sysroot-target/usr/lib -lghdrv -lQ
QtEmbedded-4.8.6-arm/lib -L/home/ghguest/platform-3D50_std/sysroot-target/lib -L/home/ghguest
lib -lQtNetwork -lQtCore -lpthread
16:42:50: The process "/usr/bin/make" exited normally.
16:42:50: Elapsed time: 00:08.
```

- The Qt-5.6.2-3Dxx Demo Project Target version is now ready to run.
- The Qt-5.6.2-3Dxx Demo Project Target version can now be run by clicking on the big green "Run" arrow on the lower left corner of the Qt Creator window.
- Click on the "Application Output" item on the bottom row to view application output.
- Click on red square on "Application Output" window to stop application.



- The "gh7indemo" and "gh10indemo" demonstration programs include a "QML Demo" button on the "Settings" screen; however, this button will just exit the Qt program when run on the 3Dxx target. In order to make this button work, the QML demonstration program must be loaded separately on the 3Dxx target and the demonstration program must be launched using a script as shown below.

For gh7indemo:

```
# Loop to run QML demo from Qt demo
while true; do
   /home/demo7in/gh7indemo &> /run/qt.log
   /home/samegame/samegame &> /run/qml.log
done
```

For gh10indemo:

```
# Loop to run QML demo from Qt demo
while true; do
   /home/demo10in/gh10indemo &> /run/qt.log
   /home/samegame/samegame &> /run/qml.log
done
```

Build and Run QML Demonstration Program (Qt-5.6.2)

This describes how to build and run the QML demonstration program “Samegame”.

- From Qt Creator open the “~/Samegame/samegame.pro” project.
- Select the desired kit, either “Qt-5.6.2-3Dxx” or “Desktop Qt 5.5.1 GCC 32-bit”.
- Click on “Build->Rebuild All” to build program.
- Click on green arrow “Run” button to run program.

Setting up 3Dxx Qt Program to Run at Boot Up

Tis describes how to configure one of the demo programs to start automatically when the 3Dxx Display is powered up and not connected to the Qt development system. Using this example any Qt program can be configured to run automatically at boot up.

- There is a launch script provided for each demo program. Using the information below, select the correct "scp" copy command to copy the desired script to the target 3Dxx Display. The "scp" command is entered from a terminal window in the Linux development environment and assumes that the Ethernet link to the target 3Dxx Display is working and configured to use the host name "gmd".
    - Demo Program: ghqtdemo
        - Orientation: Portrait
        - Sends graphics to fb0; no graphics overlay for camera
        - "scp" copy command:

```
scp /media/sf_vmshare/3Dxx_launchqtdemo root@gmd:/etc/init.d/launchqtdemo
```

    - Demo Program: gh7indemo
        - Orientation: Landscape
        - Sends graphics to fb1; can do graphics overlay for camera
        - "scp" copy command:

```
scp /media/sf_vmshare/3Dxx_launchqt7indemo root@gmd:/etc/init.d/launchqt7indemo
```

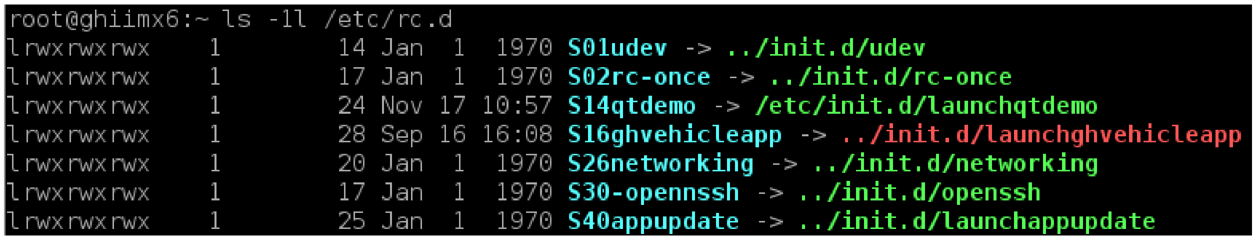
    - Demo Program: gh10indemo
        - Orientation: Landscape
        - Sends graphics to fb1; can do graphics overlay for camera
        - "scp" copy command:

```
scp /media/sf_vmshare/3Dxx_launchqt10indemo
root@gmd:/etc/init.d/launchqt10indemo
```

- Next a link to the launch script must be placed in the directory "/etc/rc.d" on the target 3Dxx Display. This is done by entering the correct command on the 3Dxx Linux console using the "Minicom" application. (The section **Configuring 3Dxx Display** describes how to launch "Minicom".) Note that the order in which the links appear in the directory "/etc/rc.d" controls the order in which the various applications are started at boot up. The example shows the recommended ordering for the fastest possible boot up of the Qt application.
    - Demo Program: ghqtdemo
        - Link command entered on "Minicom" console:

```
ln -s /etc/init.d/launchqtdemo /etc/rc.d/S14qtdemo
```

        - `Result (get listing with command: "ls -ll /etc/rc.d"):`

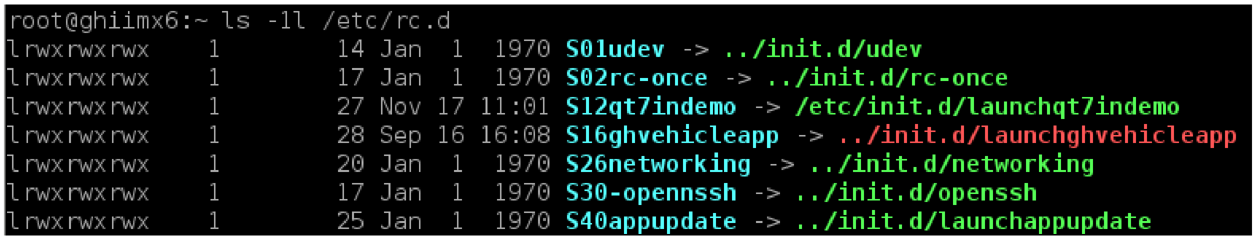


```
root@ghiimx6:~ ls -1l /etc/rc.d
lrwxrwxrwx    1        14 Jan  1  1970 S01udev -> ../init.d/udev
lrwxrwxrwx    1        17 Jan  1  1970 S02rc-once -> ../init.d/rc-once
lrwxrwxrwx    1        24 Nov 17 10:57 S14qtdemo -> /etc/init.d/launchqtdemo
lrwxrwxrwx    1        28 Sep 16 16:08 S16ghvehicleapp -> ../init.d/launchghvehicleapp
lrwxrwxrwx    1        20 Jan  1  1970 S26networking -> ../init.d/networking
lrwxrwxrwx    1        17 Jan  1  1970 S30-opennssh -> ../init.d/openssh
lrwxrwxrwx    1        25 Jan  1  1970 S40appupdate -> ../init.d/launchappupdate
```

- Demo Program: gh7indemo
  - Link command entered on “Minicom” console:

```
ln -s /etc/init.d/launchqt7indemo /etc/rc.d/S12qt7indemo
```

  - Result (get listing with command: “ls -1l /etc/rc.d”):

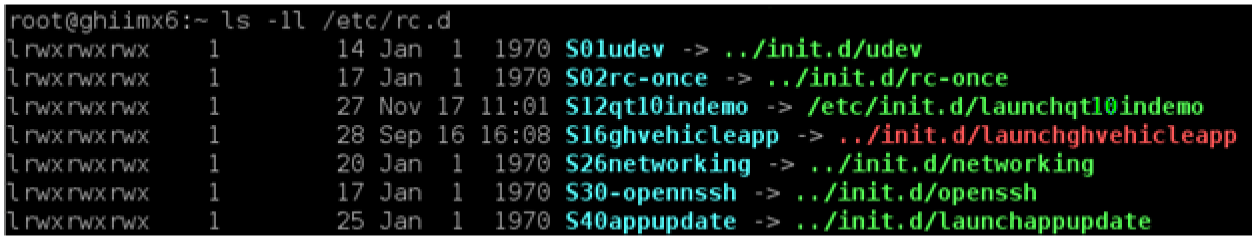


```
root@ghiimx6:~ ls -1l /etc/rc.d
lrwxrwxrwx    1        14 Jan  1  1970 S01udev -> ../init.d/udev
lrwxrwxrwx    1        17 Jan  1  1970 S02rc-once -> ../init.d/rc-once
lrwxrwxrwx    1        27 Nov 17 11:01 S12qt7indemo -> /etc/init.d/launchqt7indemo
lrwxrwxrwx    1        28 Sep 16 16:08 S16ghvehicleapp -> ../init.d/launchghvehicleapp
lrwxrwxrwx    1        20 Jan  1  1970 S26networking -> ../init.d/networking
lrwxrwxrwx    1        17 Jan  1  1970 S30-opennssh -> ../init.d/openssh
lrwxrwxrwx    1        25 Jan  1  1970 S40appupdate -> ../init.d/launchappupdate
```

- Demo Program: gh10indemo
  - Link command entered on “Minicom” console:

```
ln -s /etc/init.d/launchqt10indemo /etc/rc.d/S12qt10indemo
```

  - Result (get listing with command: “ls -1l /etc/rc.d”):



```
root@ghiimx6:~ ls -1l /etc/rc.d
lrwxrwxrwx    1        14 Jan  1  1970 S01udev -> ../init.d/udev
lrwxrwxrwx    1        17 Jan  1  1970 S02rc-once -> ../init.d/rc-once
lrwxrwxrwx    1        27 Nov 17 11:01 S12qt10indemo -> /etc/init.d/launchqt10indemo
lrwxrwxrwx    1        28 Sep 16 16:08 S16ghvehicleapp -> ../init.d/launchghvehicleapp
lrwxrwxrwx    1        20 Jan  1  1970 S26networking -> ../init.d/networking
lrwxrwxrwx    1        17 Jan  1  1970 S30-opennssh -> ../init.d/openssh
lrwxrwxrwx    1        25 Jan  1  1970 S40appupdate -> ../init.d/launchappupdate
```

In the above directory listings observe that the link for the “S16ghvehicleapp” item is shown in red. This is because the target for this link was renamed to “launchghvehicleappXX” and this “broke” the link. Since this link is “broken”, this item (which is the VUI Builder© application) will not be run at boot up. This is a convenient way to switch what application is to be started at boot up. Just rename the launch script located in the directory “/etc/init.d”.

CAUTION!
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 will conflict with one another.

NOTE
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..

Interfacing 3Dxx Hardware from QT Software

The 3Dxx Display contains the following custom component interfaces:

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

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.

LCD

The Grayhill 3Dxx Series Display uses a 16 bit per pixel LCD screen. The pixel dimensions of various 3Dxx Display products are should in the section **Supported Hardware Products**. The default orientation of the frame buffer is landscape mode (wider pixel dimension is in horizontal direction).

The following only applies to using Qt-4.8.6:

The frame buffer can be rotated to portrait orientation and can even be flipped upside down by using the "-display transformed:rot" parameter when launching the Qt application. Setting of this parameter when configuring a Qt project for development and debug is shown in section **Configure a 3Dxx Demo Project to Build and Run (Qt-4.8.6).** In order to set this parameter when launching a Qt application at boot up, examine the launch scripts discussed in the section **Build and Run QML Demonstration Program** (Qt-5.6.2)

This describes how to build and run the QML demonstration program "Samegame".

- From Qt Creator open the "~/Samegame/samegame.pro" project.

Select the desired kit, either "Qt-5.6.2-3Dxx" or "Desktop Qt 5.5.1 GCC 32-bit".

- Click on "Build->Rebuild All" to build program.
- Click on green arrow "Run" button to run program.

Setting up 3Dxx Qt Program to Run at Boot Up**.**

Images can be displayed on frame buffer 0 or 1 using standard Qt API. Which frame buffer a Qt application uses is set by the presence or absence of the parameter "-display LinuxFb:/dev/fb1" when launching the Qt application. The Qt application may not switch frame buffers once it starts. Setting of this parameter when configuring a Qt project for development and debug is shown in section **Configure a 3Dxx Demo Project to Build and Run (Qt-4.8.6).** In order to set this parameter when launching a Qt application at boot up, examine the launch scripts discussed in the section **Build and Run QML Demonstration Program** (Qt-5.6.2)

This describes how to build and run the QML demonstration program "Samegame".

- From Qt Creator open the "~/Samegame/samegame.pro" project.

Select the desired kit, either "Qt-5.6.2-3Dxx" or "Desktop Qt 5.5.1 GCC 32-bit".

- Click on "Build->Rebuild All" to build program.
- Click on green arrow "Run" button to run program.

Setting up 3Dxx Qt Program to Run at Boot Up**.**

LCD Backlight

The LCD Backlight setting is a value between 0 and 100 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();
}
```

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 displayed at 30fps.
Note: Only one camera input can be active at a time.

Interface:

The Qt application can interface with the Camera driver using the Camera class.

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)

// 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;
```

Function Prototypes:

Camera::Camera

Camera class constructor

Syntax

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

Parameters

int camnum
[in]
Camera Number. Valid range 1-2 for Model 3D50, 1-3 for Model 3D70, 1-4 for Model 3D2104

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

int fbdev
[in]
The "fbdev" value must indicate 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 GRPAHICS 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

Return Value

none

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);
```

Parameters

PDISPLAYPARAMS p

[in]
refer to DISPLAYPARAMS structure

Return Value

int
0 indicates success, -1 indicates failure

Camera::setcolorparams

Sets the following camera color parameters

- Brightness
- Saturation
- Contrast
- Hue

Syntax

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

Parameters

PCOLORPARAMS	p [in] refer to COLORPARAMS structure

Return Value

int
0 indicates success, -1 indicates failure

Camera::setsensorparams

Sets the camera sensor parameters.

Syntax

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

Parameters

PSENSORPARAMS	psensor [in] refer to SENSORPARAMS structure

Return Value

int
always returns 0

Camera::show

Enables or disables the camera

Syntax

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

Parameters

int	enable [in] 1 = enable, 0 = disable

Return Value

int
0 indicates success, -1 indicates failure

Required Files:

Header File: camera.h
Link Library : libghdrv.so

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 l+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.

Interface:

The qt demo application can interface with the CAN bus driver using the CAN class.

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;
```

Function Prototypes:

CAN::CAN
CAN class constructor

Syntax

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

Parameters

int num
[in]
CAN Port Number. Valid range 1-2 for Models 3D50, 3D70; 1-3 for Model 3D2104

Return Value

none

CAN::OpenPort
Opens the CAN socket

Syntax

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

Parameters
none

Return Value
int
non-zero value indicates success, -1 indicates failure

CAN::WritePort
Writes a single CAN frame to the CAN port.

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

Parameters
PCANMSG	TxMsg [in] Contains the CAN frame to be written

Return Value
int
0 indicates success, -1 indicates 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);
```

Parameters
PCANMSG	RxMsg [out] Contains the CAN frame received

Return Value
int
contains the number of bytes read, -1 indicates failure

CAN::ClosePort
Closes the CAN socket

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

Parameters
none

Return Value

none

Required Files:

Header File: can.h
Link Library : libghdrv.so

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)	

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);`

Parameters

int ch
[in]
Input pin to configure (`GHIOLIB_CH1, GHIOLIB_CH2, GHIOLIB_CH3, or GHIOLIB_CH4`)
uint8_t config
[in]
`GHIOLIB_DIG_IN_FLOAT, GHIOLIB_DIG_IN_PULL_DN, or GHIOLIB_DIG_IN_PULL_UP`

Return Value

int
`GHIOLIB_RET_OK, GHIOLIB_RET_ERROR, or GHIOLIB_RET_NOTSUPPORTED`

ghiolib_getDigIn

This function reads the state of an input pin.

Syntax

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

Parameters

int ch
[in]
Input pin to read (`GHIOLIB_CH1, GHIOLIB_CH2, GHIOLIB_CH3, or GHIOLIB_CH4`)
uint8_t *value
[out]
Returns 0 if input is low, else returns 1

Return Value

int
`GHIOLIB_RET_OK, GHIOLIB_RET_ERROR, or GHIOLIB_RET_NOTSUPPORTED`

ghiolib_getDigOut

Reads the current state of an output pin.

Syntax

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

Parameters

int ch
[in]
Output pin to read (`GHIOLIB_CH1, GHIOLIB_CH2, GHIOLIB_CH3, or GHIOLIB_CH4`)
uint8_t *value
[out]

Returns 0 if output is set low, else returns 1

Return Value

int
`GHIOLIB_RET_OK, GHIOLIB_RET_ERROR, or GHIOLIB_RET_NOTSUPPORTED`

ghiolib_setDigOut

This function sets the current state of an output pin.

Syntax

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

Parameters

int ch
[in]
Output pin to set (`GHIOLIB_CH1, GHIOLIB_CH2, GHIOLIB_CH3, or GHIOLIB_CH4`)
uint8_t value
[in]
If 0 sets output pin low, else sets output pin high (Vbatt)

Return Value

int
`GHIOLIB_RET_OK, GHIOLIB_RET_ERROR, or GHIOLIB_RET_NOTSUPPORTED`

Required Files:

Header File:	ghiolib.h
Link Library:	libghiodrv.so

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);
```

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).

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 (Model 3D70 only)
This function configures an analog input for one of five different reading modes.

Syntax

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

Parameters

int ch
[in]
Input to configure (`GHIOLIB_CH1 or GHIOLIB_CH2`)

uint8_t config
[in]
`GHIOLIB_ANALOG_5V, GHIOLIB_ANALOG_10V, GHIOLIB_ANALOG_1500OHM, GHIOLIB_ANALOG_5000OHM, or GHIOLIB_ANALOG_20MA`

Return Value

int
`GHIOLIB_RET_OK, GHIOLIB_RET_ERROR, or GHIOLIB_RET_NOTSUPPORTED`

ghiolib_getADCIn (Model 3D70 only)
This function gets a reading from an analog input pin.

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

Parameters

int ch
[in]
Input to read (`GHIOLIB_CH1 or GHIOLIB_CH2`)

PADCVALUES p
[out]
Reading is returned in member "adcch" of this structure. Other items in this structure can be ignored.

Return Value
int
`GHIOLIB_RET_OK, GHIOLIB_RET_ERROR, or GHIOLIB_RET_NOTSUPPORTED`

Required Files:

Header File:	ghiolib.h
Link Library:	libghiodrv.so

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);
```

Buzzer (Models 3D70, 3D2104)

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

Interface:

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

Required Files:

Header File: none
Link Library: none

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;
}
```

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).

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.

Required Files:

Header File:	none
Link Library:	none
Executable:	mpg123 (normally installed on Model 3D70 Display)

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`");
```

Appendix A: VirtualBox Linux Passwords

- Password for user ghguest on Linux system: !admin!
- Password for user root on Linux system: !rty32999!

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

- To immediately set the 3Dxx Display file system to read-write mode enter this console command:

```
mount -o remount,rw /
```

- The above command only remains in effect until the next reboot and is usually stored in a script file here: /home/writeablefs.

- To have the 3Dxx Display file system set 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 remout,rw /`" line near the end of the file (or comment it out by putting a "#" in column one of that line).

CAUTION!
It is not recommended to leave the 3Dxx Display flash file system in read-write mode on fielded units because this may result in file system corruption and slower boot-up times.

- Another way to have the 3Dxx Display file system set to read-write mode on boot-up, is to add a link to the "writeablefs" script in the home directory like this:

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