Quantis User's Guide

Version 2.4

Quantis User's Guide

Version 2.4

Revision History

Revision 2.4 16.09.2010

- Added instructions to install Quantis on Red Hat Enterprise Linux and CentOS.

Revision 2.3 25.06.2010

- Added scaling algorithms details.
- Improved EasyQuantis installation description on Linux.
- Added Troubleshooting appendix.

Revision 2.2 30.04.2010

- In Quantis PCI Linux driver installation section: fixed a wrong path and added two sub-sections.
- Updated EasyQuantis installation procedure under Linux.

Revision 2.1 26.04.2010

- Added EasyQuantis command line section.
- Added answers in the FAQ.

Revision 2.0 09.04.2010

- Initial version.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

Thank you for purchasing a Quantis Random Number Generator.

A random number generator is a device that produces sequences of numbers, whose outcome is unpredictable and which cannot subsequentially be reliably reproduced. There exist two main classes of random number generators: software and physical generators. From a general point of view, software generators produce so-called pseudo random numbers. Although they may be useful in some applications, they should not be used in most applications where randomness is required.

Quantis is a physical random number generator exploiting an elementary quantum optics process. Photons - light particles - are sent one by one onto a semi-transparent mirror and detected. The exclusive events (reflection - transmission) are associated to "0" - "1" bit values. The operation of Quantis is continuously monitored. If a failure is detected, the random bit stream is immediately disabled.

Quantum random number generators have the advantage over conventional randomness sources of being invulnerable to environmental perturbations and of allowing live status verification.

1.1. What You Need

To use your Quantis, you need:

- A PC with a supported operating system installed (see Table 1.1, “Supported operating systems.”) and one of the following slots/ports available:
  - A PCI 32-bit slot (for Quantis PCI).
  - A PCI Express x1 slot (for Quantis PCI Express).
  - An USB 2.0 port (for Quantis USB).
- An USB 2.0 port for the USB Flash drive.
- 50MB hard drive space.

Operating System	**Quantis PCI/PCIe**	**Quantis USB**
Microsoft Windows XP (32-bit)	✓	✓
Microsoft Windows XP (64-bit)	✗	✗
Microsoft Windows Server 2003	✓	✓
Microsoft Windows Vista (32-bit and 64-bit)	✓	✓
Microsoft Windows Server 2008 (32-bit and 64-bit)	✓	✓
Microsoft Windows 7 (32-bit and 64-bit)	✓	✓
Linux 2.6 (32-bit and 64-bit)	✓	✓

Table 1.1. Supported operating systems.

1.1.1. Additional Requirements

1.1.1.1. Linux

On Linux systems, you additionally need:

- Xorg 1.0 or higher (only required to use the EasyQuantis application).

Chapter 2. Hardware Installation

This chapter provides unpacking and installation information for the Quantis.

2.1. Quantis PCI and PCI Express Installation

Caution

Under ordinary circumstances, the Quantis PCI and Quantis PCI Express (PCIe) cards will not be affected by static charge as may be received through your body during handling of the unit. However, there are special circumstances where you may carry an extraordinarily high static charge and possibly damage the card and/or your computer. To avoid any damage from static electricity, you should follow some precautions whenever you work on your computer.

1. Turn off your computer and unplug power supply.
2. Use a grounded wrist strap before handling computer components. If you do not have one, touch both of your hands to a safely grounded object or to a metal object, such as the power supply case.
3. Place components on a grounded anti-static pad or on the bag that came with the components whenever the components are separated from the system.

The card contains sensitive electric components, which can be easily damaged by static electricity, so the card should be left in its original packing until it is installed.

Unpacking and installation should be done on a grounded anti-static mat. The operator should be wearing an anti-static wristband, grounded at the same point as the anti-static mat.

Inspect the card carton for obvious damage. Shipping and handling may cause damage to your card. Be sure there are no shipping and handling damages on the card before proceeding.

DO NOT APPLY POWER TO YOUR SYSTEM IF THE QUANTIS CARD IS DAMAGED.

2.1.1. Unpacking

Open the shipping carton and carefully remove all items, and ascertain that you have:

- Quantis PCI or Quantis PCIe card;
- Quick Install Guide;
- USB Flash Drive with Manual, Drivers and Samples.

If any item is found to be missing or damaged, please contact your local reseller for replacement.

2.1.2. Installing the Card

1. Shut down the computer, unplug its power cord and remove the chassis cover.
2. Locate the PCI 32-bits slot (for Quantis PCI) or the PCI Express x1 slot (for Quantis PCIe). If necessary, remove the metal cover from this slot, then align your Quantis card with the PCI or PCIe slot respectively and press it in firmly until the card is fully seated.

3. Install the bracket screw and secure the card to the computer chassis.
4. Cover the computer's chassis.
5. Switch the computer power on.
6. Install the driver (see next Chapter).

2.2. Quantis USB Installation

2.2.1. Unpacking

Open the shipping carton and carefully remove all items, and ascertain that you have:

- Quantis USB;
- USB cable;
- Quick Install Guide;
- USB Flash Drive with Manual, Drivers and Samples.

If any item is found to be missing or damaged, please contact your local reseller for replacement.

2.2.2. Installing the Device

1. Connect the Quantis device to a USB 2.0 port on your PC using the cable that came with the Quantis device.
2. Install the driver (see next Chapter).

Chapter 3. Driver Installation

To be able to access your Quantis device, you need to install a driver. This chapter contains instructions on how to install the driver on your operating system.

Important

Quantis PCI Express is software-compatible with Quantis PCI. This means that any software able to communicate with a Quantis PCI device (e.g driver) is also able to communicate with the Quantis PCI Express. More specifically:

- Quantis PCIe uses the Quantis PCI driver.
- Quantis PCIe is considered by the software (driver, application) as a Quantis PCI device.

3.1. Windows Operating Systems

This section contains instructions on how to install Quantis device on Windows Operating Systems.

Please insert the USB flash drive on an available USB port. This drive contains the Quantis drivers as well as software for your device.

Important

In this section, we assume that the letter of the USB flash drive provided by IDQ is Drive `D:`.

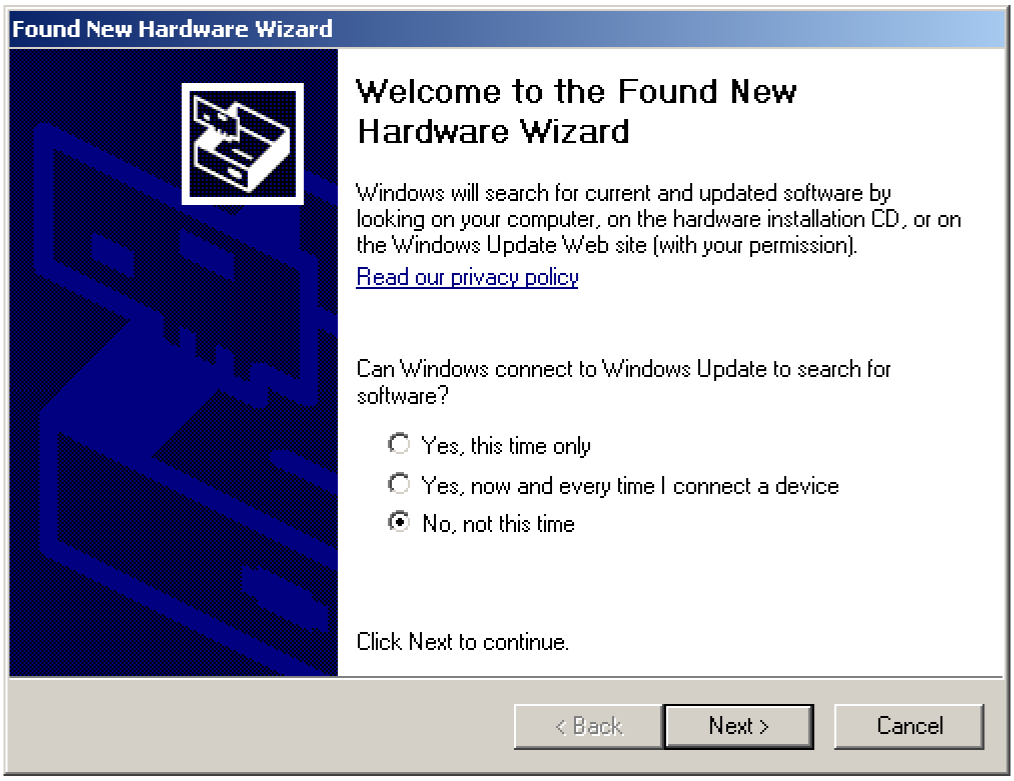
3.1.1. Windows XP

When a Quantis RNG is inserted into your computer for the first time, the operating system will detect the device automatically and display a *New Hardware Found* message. The following are step-by-step installation instructions.

3.1.1.1. Found New Hardware Wizard: Welcome

Windows will search for a driver on your computer, on removable media (e.g. CD-ROM) and on the Windows Update Web site.

The Quantis driver is not available on the Windows Update Web site. If asked, deny access to the Windows Update Web site and click the **Next** button.



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

Can Windows connect to Windows Update to search for software?

- ○ Yes, this time only
- ○ Yes, now and every time I connect a device
- [✓ No, not this time]

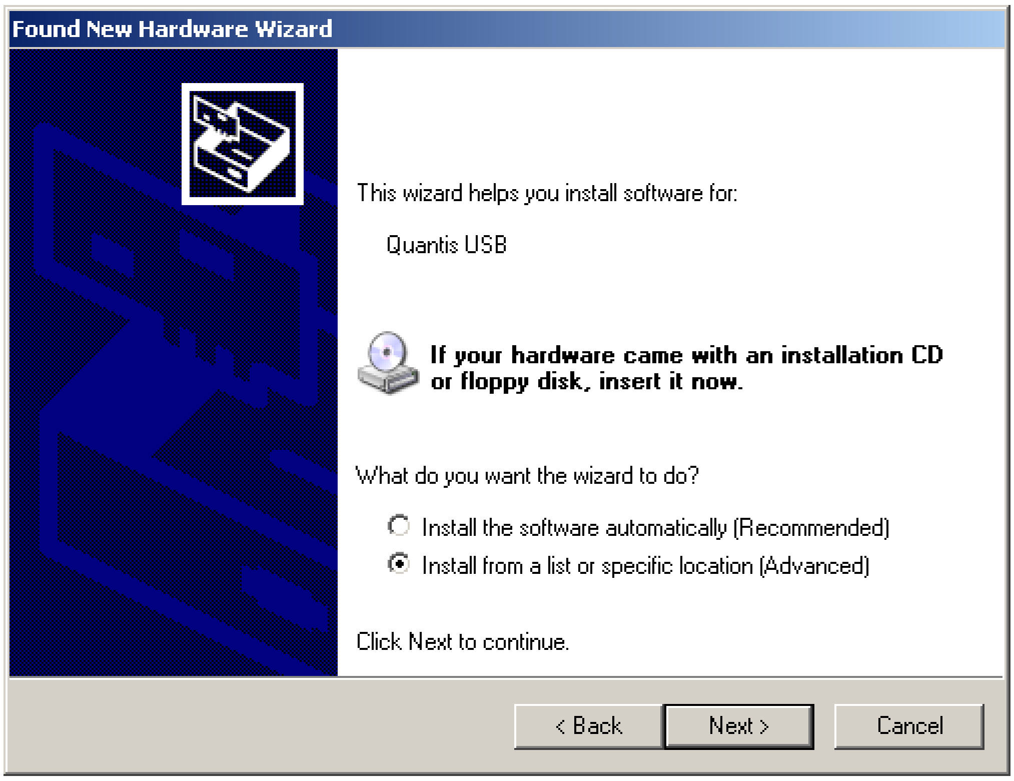
Click Next to continue.

Note

It is harmless to allow the wizard to connect to the Windows Update Web site. The installation process will only take a bit longer.

3.1.1.2. Found New Hardware Wizard: Quantis

When the wizard asks you what to do, select *Install from a list or specific location* and click the **Next** button.



This wizard helps you install software for:

Quantis USB

If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

- [ ] Install the software automatically (Recommended)
- [✓] Install from a list or specific location (Advanced)

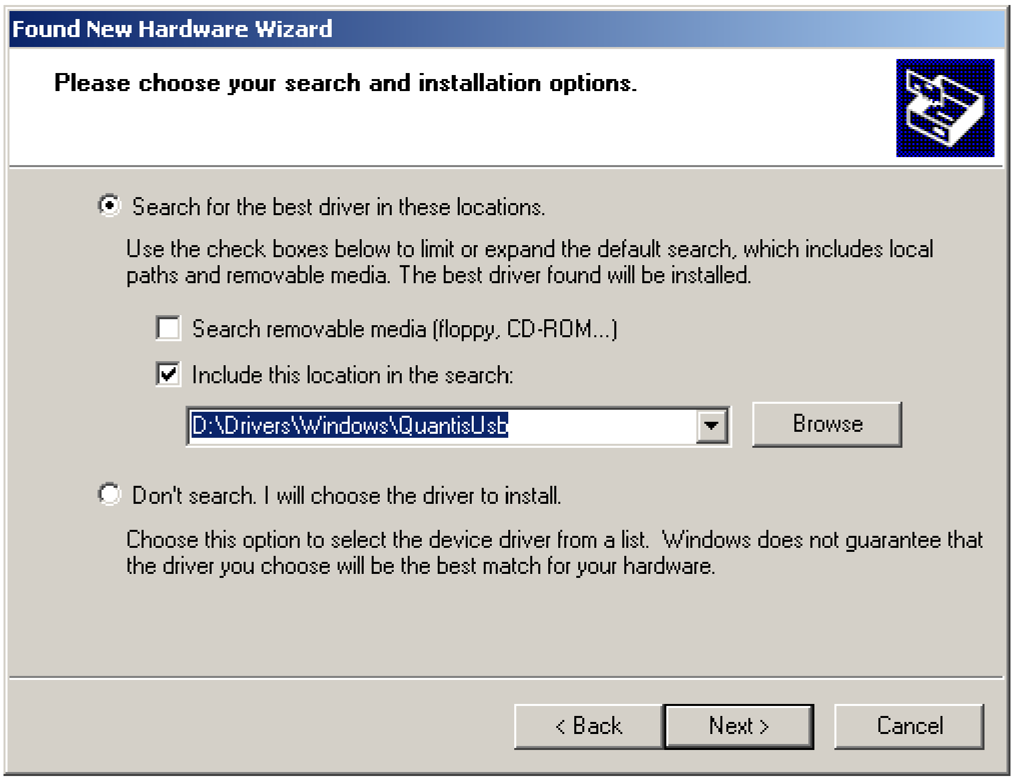
Click Next to continue.

3.1.1.3. Found New Hardware Wizard: Search Location

First select *Search for the best driver in these locations*. Then activate the option *Include this location in the search*. Click the Browse button and select the directory containing the right driver for your device:

- For the Quantis PCI and Quantis PCIe select `D:\Drivers\Windows\QuantisPci`.
- For the Quantis USB select `D:\Drivers\Windows\QuantisUsb`.

Click the **Next** button to validate.


Found New Hardware Wizard

Please choose your search and installation options.

(●) Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☐ Search removable media (floppy, CD-ROM...)

☑ Include this location in the search:

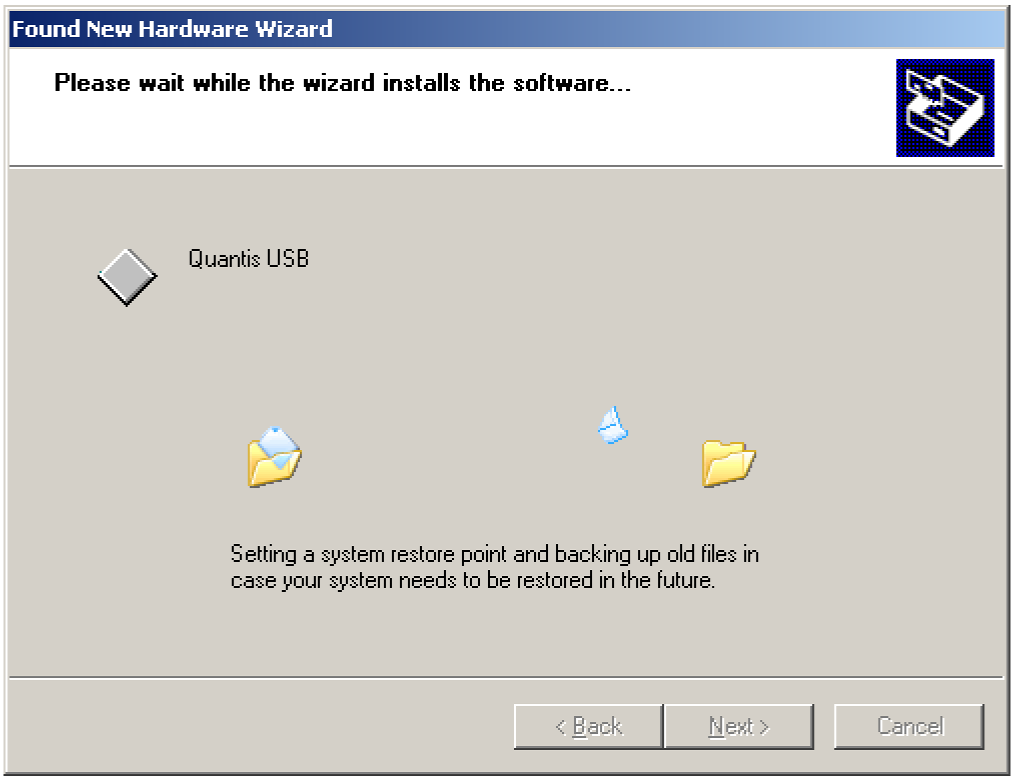
D:\Drivers\Windows\QuantisUsb

( ) Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

3.1.1.4. Found New Hardware Wizard: Installation

Wait while the wizard installs the Quantis driver.


Found New Hardware Wizard

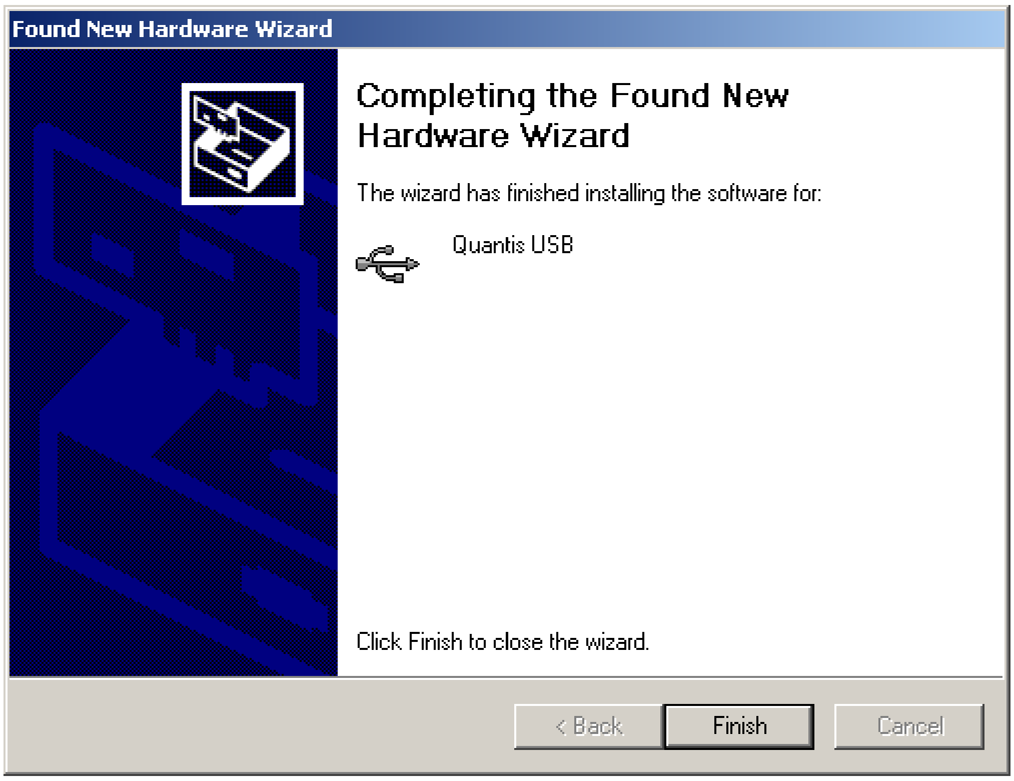
Please wait while the wizard installs the software...

Quantis USB

Setting a system restore point and backing up old files in case your system needs to be restored in the future.

3.1.1.5. Found New Hardware Wizard: Completed

When the wizard has finished installing the Quantis driver, click the **Finish** button to finish the installation. Reboot the computer if asked.


Found New Hardware Wizard

Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

Quantis USB

Click Finish to close the wizard.

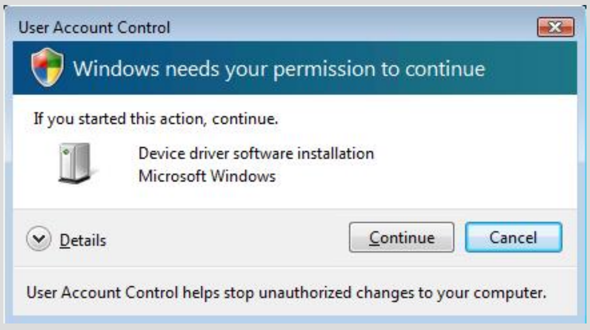
Your Quantis device is now installed. You can go to the next Chapter and install the software.

3.1.2. Windows Vista

When the Quantis RNG is inserted into your computer for the first time, the operating system will detect the device automatically and display a *New Hardware Found* message. The following are step-by-step installation instructions.

Note

One or more intermediate dialog boxes may appear during the process stating *Windows needs your permission to continue*. Click *Continue* to proceed.


User Account Control

Windows needs your permission to continue

If you started this action, continue.

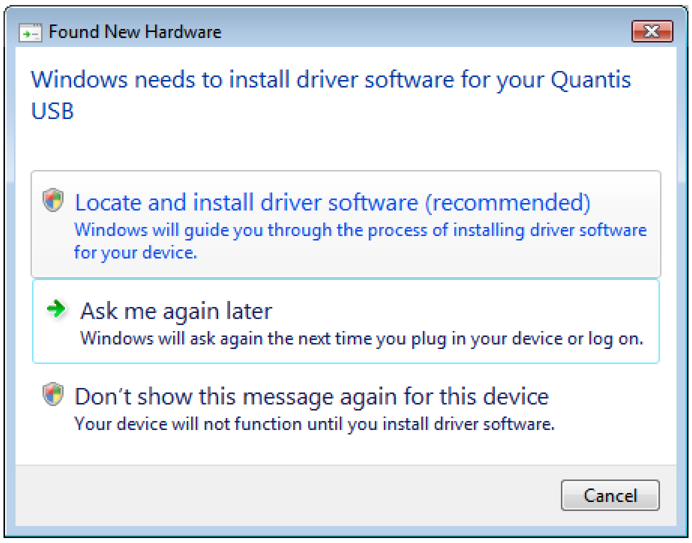
Device driver software installation
Microsoft Windows

User Account Control helps stop unauthorized changes to your computer.

3.1.2.1. Found New Hardware Wizard: Welcome

Windows will search for a driver on your computer, on removable media (e.g CD-ROM) and on the Windows Update Web site.

Let Windows try to locate the driver by clicking on *Locate and install driver software*.


Found New Hardware

Windows needs to install driver software for your Quantis USB

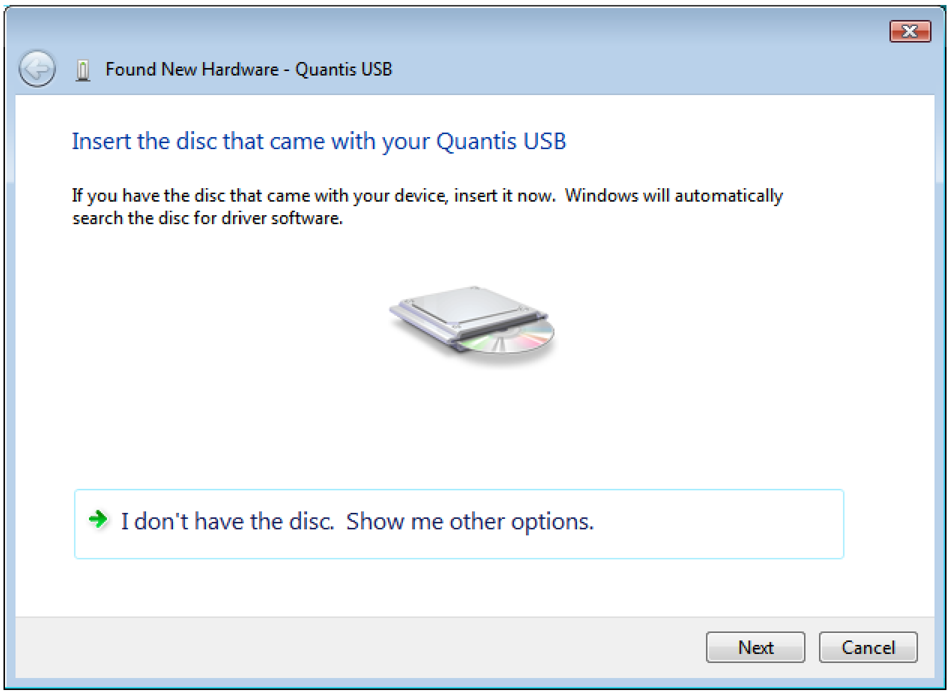
Locate and install driver software (recommended)
Windows will guide you through the process of installing driver software for your device.

Ask me again later
Windows will ask again the next time you plug in your device or log on.

Don't show this message again for this device
Your device will not function until you install driver software.

3.1.2.2. Found New Hardware Wizard: Insert Disc

When the wizard asks you to insert the disc that came with your Quantis USB, choose *I don't have the disc. Show me other options*. This allows you to specify the location of the driver available on the USB flash drive.


Found New Hardware - Quantis USB

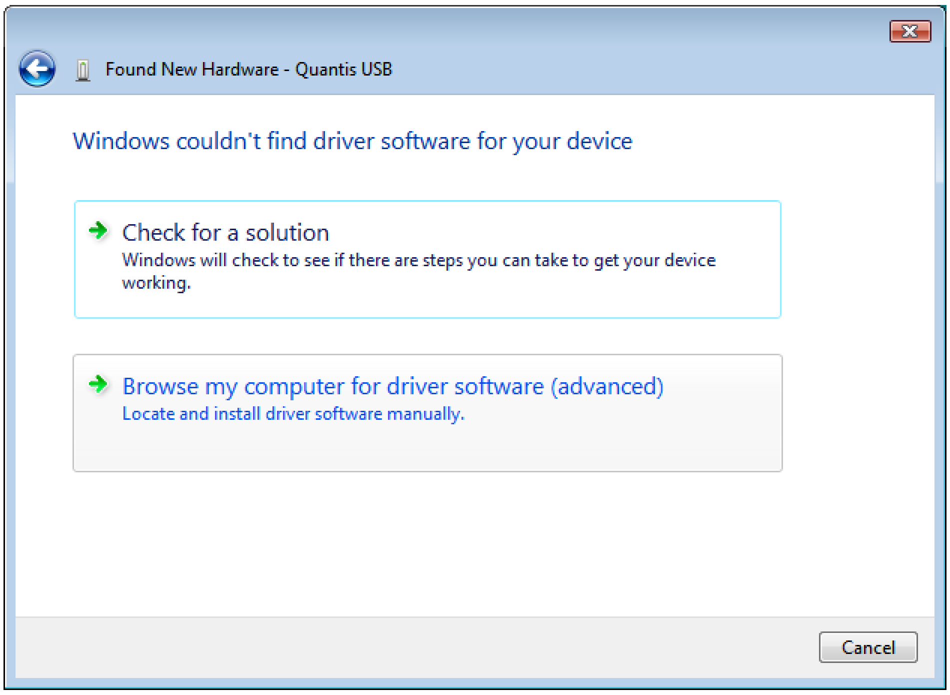
Insert the disc that came with your Quantis USB

If you have the disc that came with your device, insert it now. Windows will automatically search the disc for driver software.

I don't have the disc. Show me other options.

3.1.2.3. Found New Hardware Wizard: Search Location

Select *Browse my computer for driver software*.

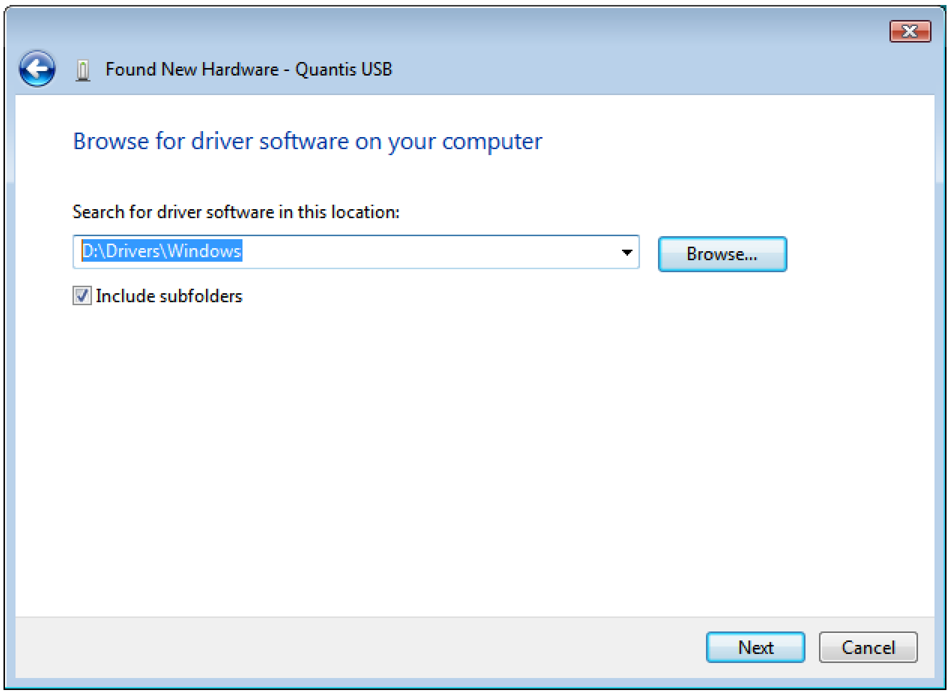

Found New Hardware - Quantis USB

Windows couldn't find driver software for your device

Check for a solution
Windows will check to see if there are steps you can take to get your device working.

Browse my computer for driver software (advanced)
Locate and install driver software manually.

On the next dialog, click the **Browse** button and select the directory `D:\Drivers\Windows`. This directory contains all drivers for Windows. Activate the option *Include subfolders* and validate your choices by clicking the **Next** button.


Found New Hardware - Quantis USB

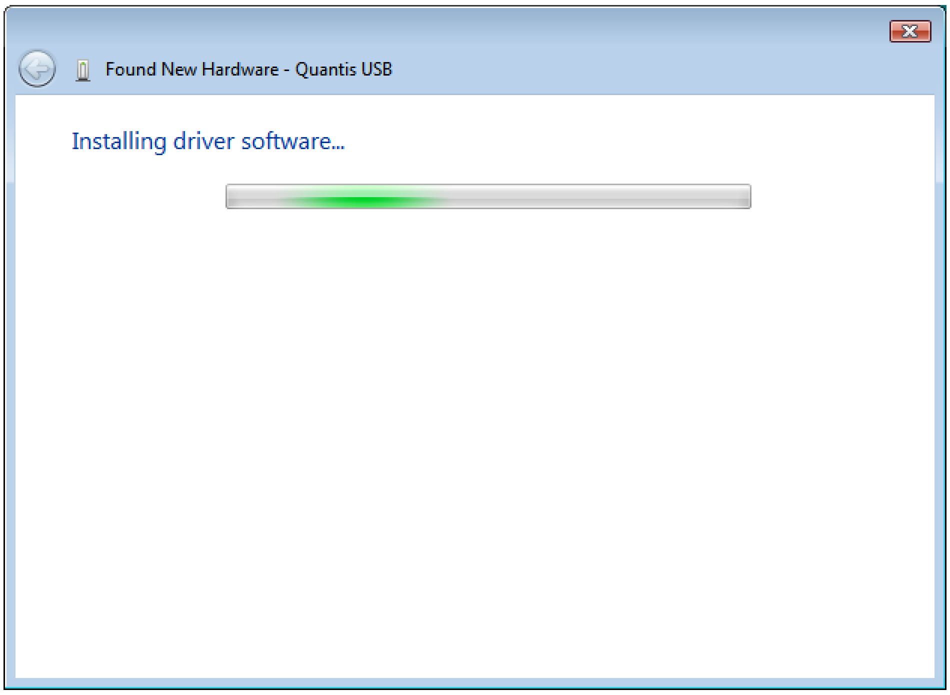
Browse for driver software on your computer

Search for driver software in this location:
D:\Drivers\Windows

☑ Include subfolders

3.1.2.4. Found New Hardware Wizard: Installation

Wait while the wizard installs the Quantis driver.



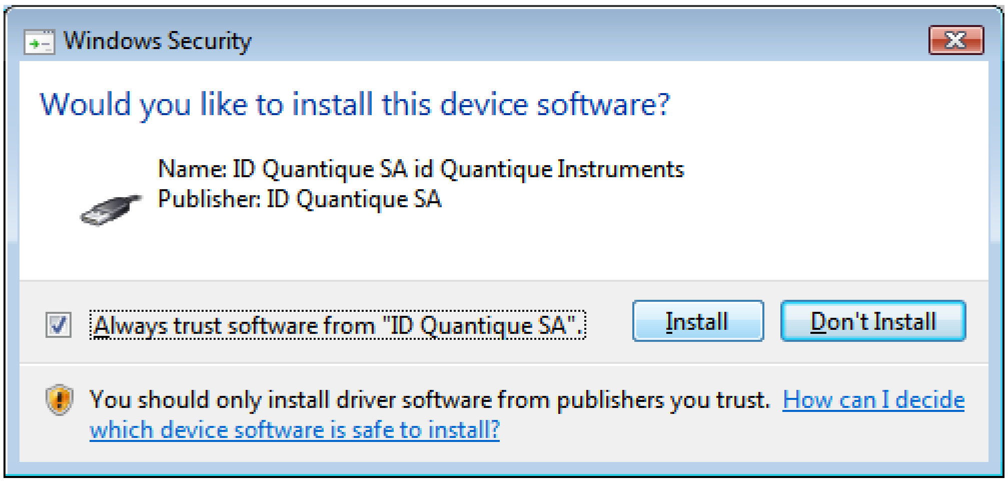
Found New Hardware - Quantis USB

Installing driver software...

3.1.2.5. Found New Hardware Wizard: Install

If asked, validate the installation by clicking the **Install** button.

You can select *Always trust software from "ID Quantique SA"*, to avoid this question in future. All software signed by ID Quantique will be automatically accepted as valid and will be installed without prompting.



Windows Security

Would you like to install this device software?

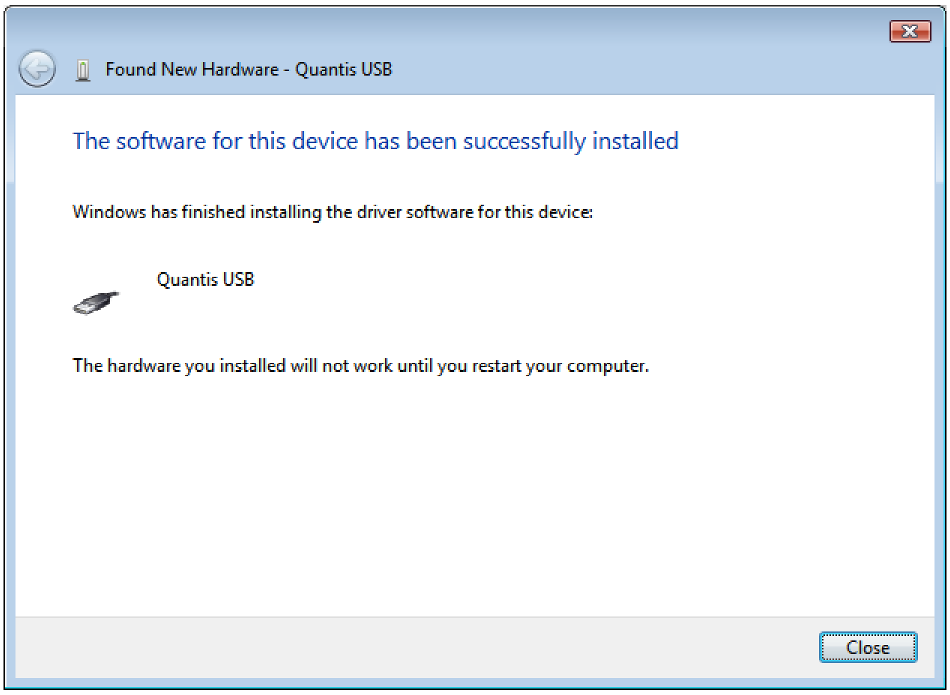
Name: ID Quantique SA id Quantique Instruments
Publisher: ID Quantique SA

☑ Always trust software from "ID Quantique SA".

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

3.1.2.6. Found New Hardware Wizard: Completed

When the wizard has finished installing the Quantis driver, click the **Close** button to finish the installation. Reboot the computer if necessary.


Found New Hardware - Quantis USB

The software for this device has been successfully installed

Windows has finished installing the driver software for this device:

Quantis USB

The hardware you installed will not work until you restart your computer.

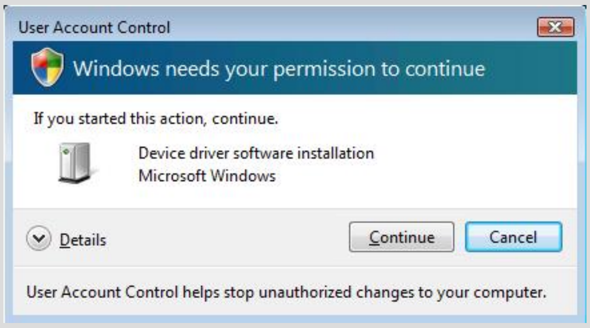
Your Quantis device is now installed. You can go to the next Chapter and install the software.

3.1.3. Windows 7

When the Quantis RNG is inserted into your computer for the first time, the operating system will detect the device automatically and search the Windows Update Web site for a driver.

Note

One or more intermediate dialog boxes may appear during the process stating *Windows needs your permission to continue*. Click *Continue* to proceed.


User Account Control

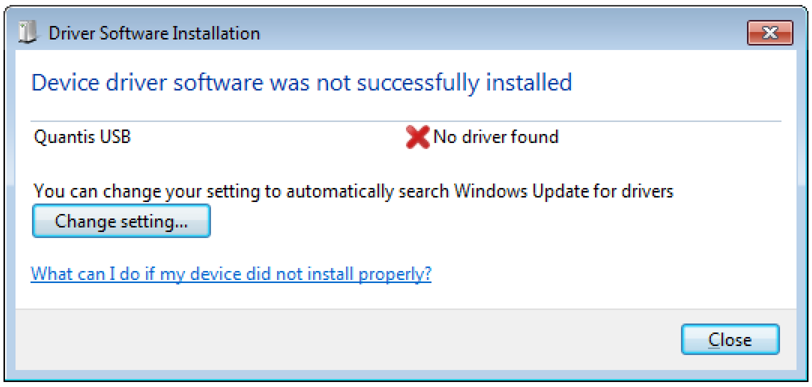
Windows needs your permission to continue

If you started this action, continue.

Device driver software installation
Microsoft Windows

User Account Control helps stop unauthorized changes to your computer.

Since the driver for your Quantis device is not available on this site, the installation will fail.


Driver Software Installation

Device driver software was not successfully installed

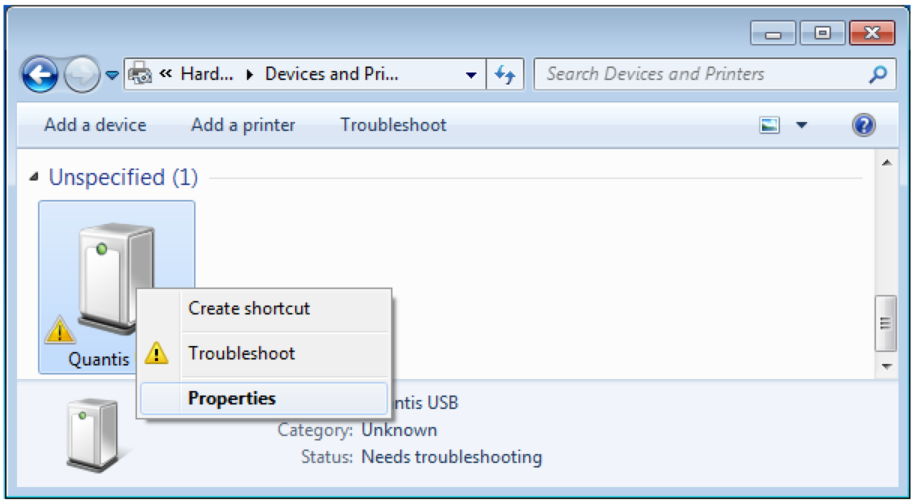
Quantis USB ❌ No driver found

You can change your setting to automatically search Windows Update for drivers

Close the dialog and read the following for the step-by-step installation instructions.

3.1.3.1. Devices and Printers

Open the *Start Menu* and select *Devices and Printers*. Scroll down until the Quantis device appears. Click on the Quantis device with the right button. Select *Properties* on the menu.



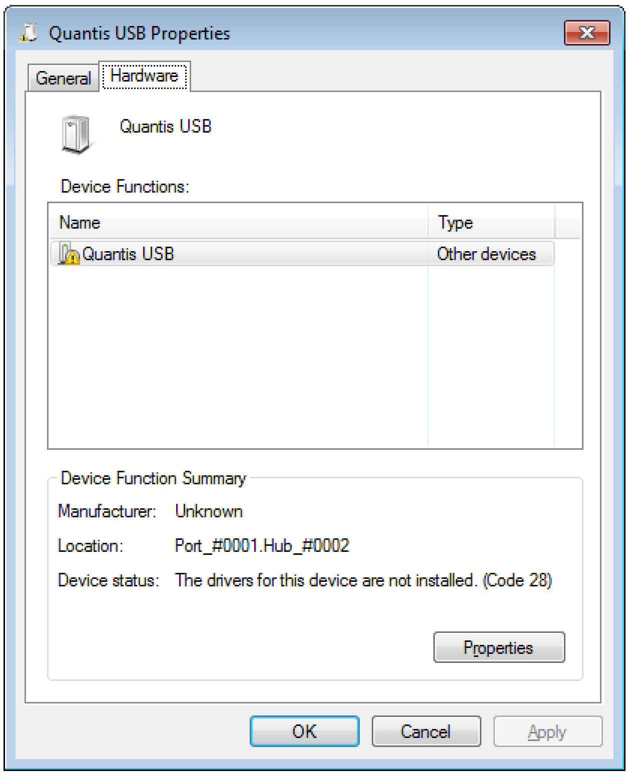
ntis USB

Category: Unknown

Status: Needs troubleshooting

3.1.3.2. Quantis Properties: Hardware

In the Quantis Properties' dialog, click on the *Hardware* tab and then on the *Properties* button.



Quantis USB

Device Functions:

Name	Type
Quantis USB	Other devices

Device Function Summary

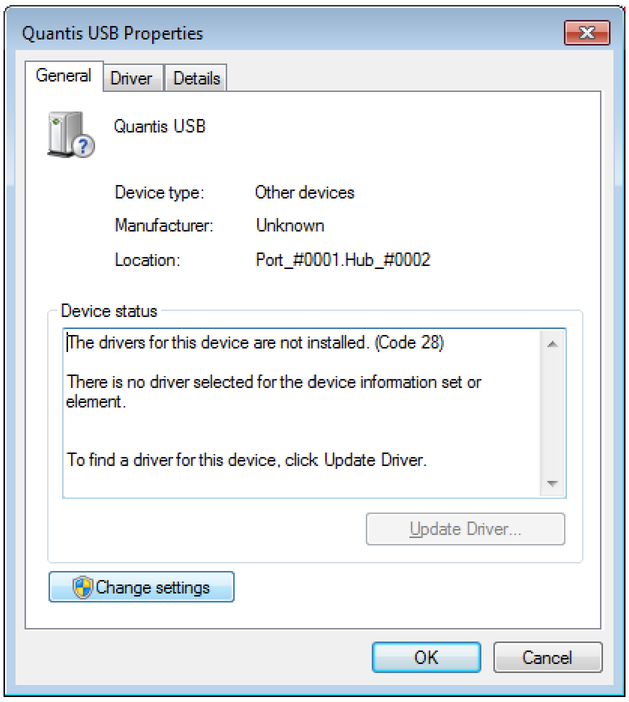
Manufacturer: Unknown

Location: Port_#0001.Hub_#0002

Device status: The drivers for this device are not installed. (Code 28)

3.1.3.3. Quantis Properties: Update Driver

First click on the button *Change settings*.



Quantis USB

Device type: Other devices

Manufacturer: Unknown

Location: Port_#0001.Hub_#0002

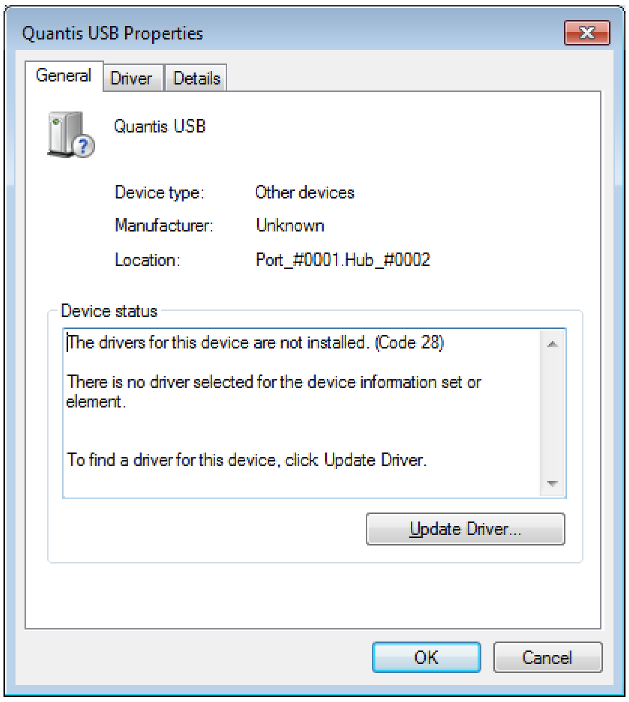
Device status

The drivers for this device are not installed. (Code 28)

There is no driver selected for the device information set or element.

To find a driver for this device, click Update Driver.

This will enable the *Update Driver* button. Click on it.


Quantis USB Properties

General | Driver | Details

Quantis USB

Device type: Other devices
Manufacturer: Unknown
Location: Port_#0001.Hub_#0002

Device status

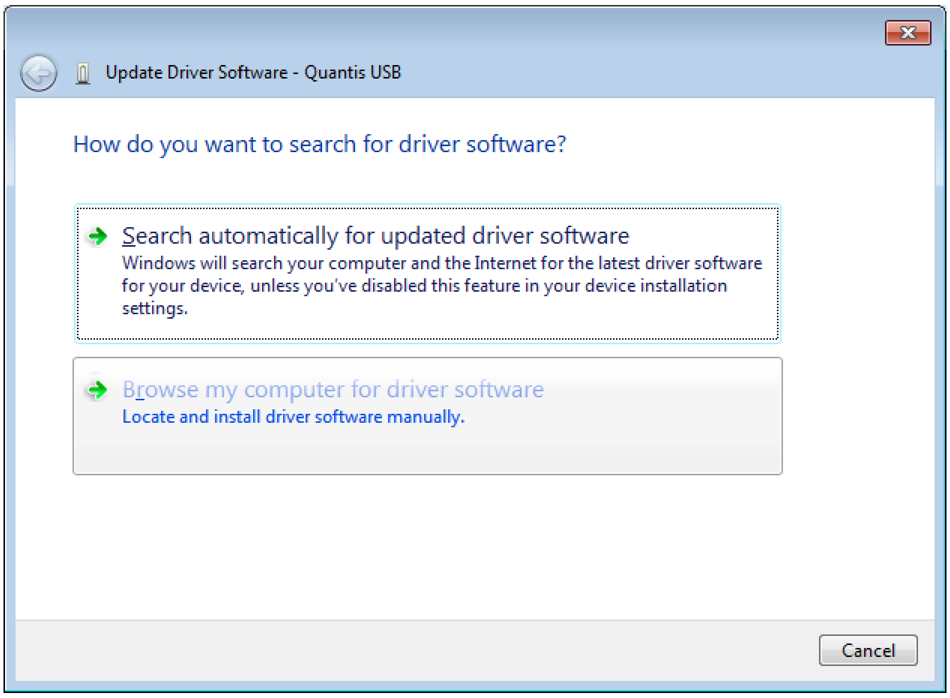
The drivers for this device are not installed. (Code 28)

There is no driver selected for the device information set or element.

To find a driver for this device, click Update Driver.

3.1.3.4. Update Driver Software: Search Driver

Driver is available on the USB flash drive provided. Select *Browse my computer for driver software*.


Update Driver Software - Quantis USB

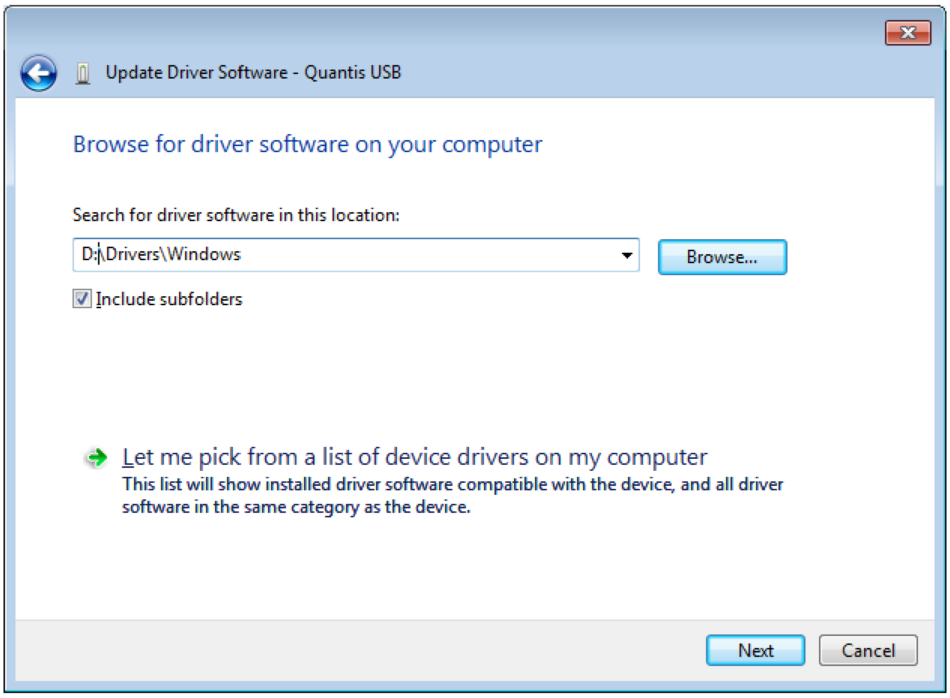
How do you want to search for driver software?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software
Locate and install driver software manually.

3.1.3.5. Update Driver Software: Search Location

Click the button **Browse** an select the directory `D:\Drivers\Windows`. This directory contains all drivers for Windows. Activate the option *Include subfolders* and validate your choices by clicking the **Next** button.


Update Driver Software - Quantis USB

Browse for driver software on your computer

Search for driver software in this location:

D:\Drivers\Windows

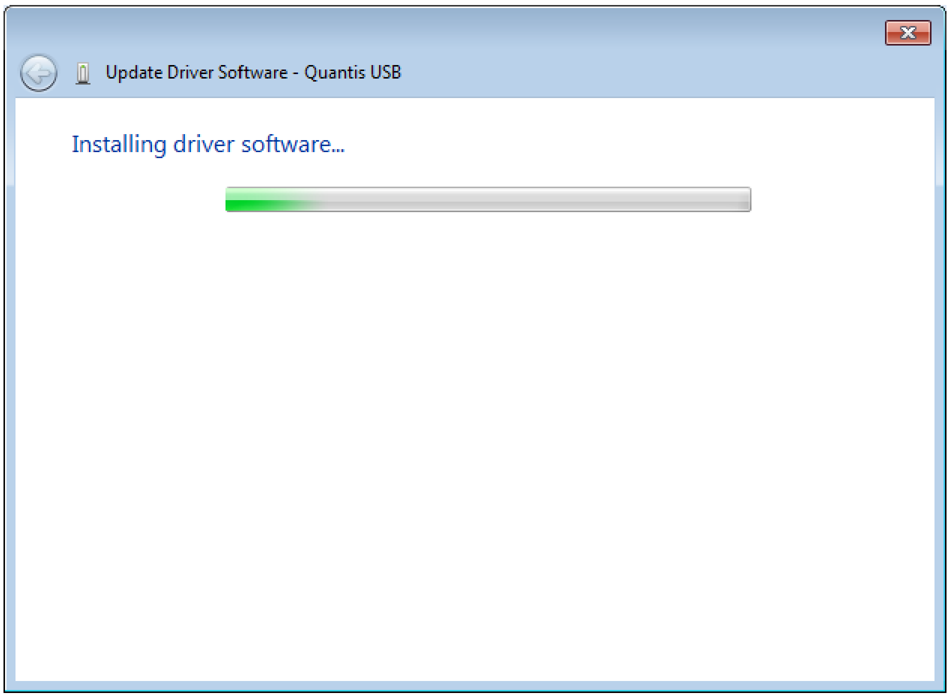
☑ Include subfolders

Let me pick from a list of device drivers on my computer

This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

3.1.3.6. Update Driver Software: Installation

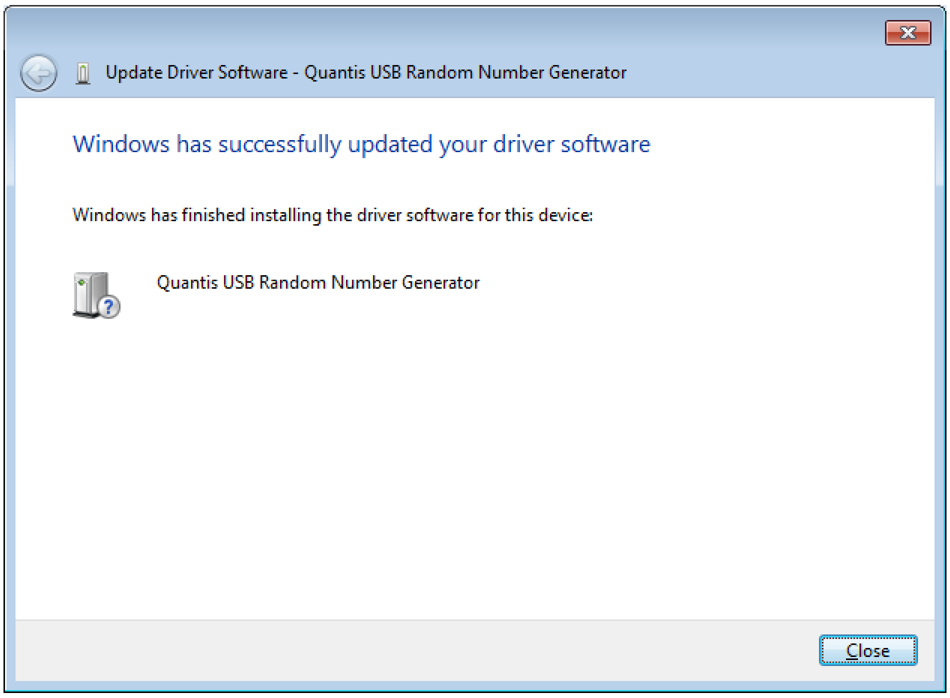
Wait while the Windows installs the driver.


Update Driver Software - Quantis USB

Installing driver software...

3.1.3.7. Update Driver Software: Completed

When the wizard has finished installing the Quantis driver, click the **Close** button to finish the installation. Reboot the computer if necessary.


Update Driver Software - Quantis USB Random Number Generator

Windows has successfully updated your driver software

Windows has finished installing the driver software for this device:

Quantis USB Random Number Generator

Your Quantis device is now installed. You can go to the next Chapter and install the software.

3.2. Linux Operating System

This section contains instructions on how to install Quantis devices on Linux Operating Systems.

Note

In this section, we assume that the USB flash drive with the software is mounted on `/media/USB_FLASH`.

Important note for Ubuntu users

Ubuntu does not include the `root` user. Instead, administrative access is given to individual users, who may use the **sudo** application to perform administrative tasks. To use **sudo** on the command line, preface the command with **sudo**:

```
$ sudo my_command_requiring_administrative_access
```

In this document, when a command must be executed as `root`, preface the command with **sudo**.

Please refer to the Ubuntu guide for more details about the command **sudo**.

3.2.1. Quantis PCI and Quantis PCI Express

The Quantis PCI and Quantis PCIe cards require a kernel module to be compiled and installed to work correctly. The following are step-by-step installation instructions.

3.2.1.1. Install Pre-Requirements

Before being able to compile a Quantis PCI kernel module, you must install a compiler and the Linux kernel sources.

Note

Generally, you do not need the full source tree in order to build a module against the running kernel. Most of the time you just need the kernel headers.

3.2.1.1.1. Debian-based Distributions

Debian-based distributions have an extremely powerful tool for building kernel modules: **module-assistant**. Module assistant aims to facilitate the process of building kernel modules from source. Type following command as `root` to install module assistant:

```
# apt-get install module-assistant
```

To download the headers corresponding to the current kernel and other mandatory tools, simply run (as `root`):

```
# m-a prepare
```

This command determines the name of the required kernel-headers package, installs it if needed and creates the `/usr/src/linux` symlink if needed. Also installs the **build-essential** package to ensure that the same compiler environment is established.

All required software has been installed. You can skip to Section 3.2.1.2, “Compile and Install Driver”.

3.2.1.1.2. Red Hat Enterprise Linux and CentOS Distributions

To build kernel modules on Red Hat Enterprise Linux and CentOS distributions it is not necessary to download the entire kernel. To build a module for the currently running kernel, only the matching **kernel-devel** package is required. Run the following command to install the **kernel-devel** package using **yum**:

```
# yum install kernel-devel
```

Important

The previous command installs the kernel headers for the latest kernel available in the repository. If your system is not up-to-date you need first to update the kernel and then boot the new kernel before installing the **kernel-devel** package:

```
# yum update kernel*
# reboot
# yum install kernel-devel
```

To compile the kernel driver you also need to install the developer tools such as GNU GCC C/C++ compilers, make and others. You can install them with the following command (as `root`):

```
# yum groupinstall "Development Tools"
```

All required software has been installed. You can skip to Section 3.2.1.2, “Compile and Install Driver”.

3.2.1.1.3. Other Distributions

Install the GNU GCC compiler and the header corresponding to the current kernel (or the whole source kernel). Please refer to the guide of your distribution for help on installing packages.

3.2.1.2. Compile and Install Driver

Now that all pre-requirements have been installed you can compile and install the driver.

First copy the driver's source code to `/tmp`:

```
$ cp -R /media/USB_FLASH/Drivers/Unix /tmp/
```

Change to the driver's directory and compile the driver:

```
$ cd /tmp/Unix/QuantisPci/
$ make
```

When compilation finish, install and load the driver with following commands (as `root`):

```
# make install
# modprobe quantis_pci
```

You can verify that the driver has been successfully loaded and all your Quantis PCI and PCIe cards have been detected with the command **dmesg**:

```
$ dmesg | grep quantis_pci
quantis_pci: Initializing Quantis PCI RNG driver version 2.0
quantis_pci:   driver build Feb 12 2010 14:26:14
quantis_pci:   support enabled up to 10 PCI card(s)
quantis_pci: Found card #0
quantis_pci:    core version 0x040a1201
quantis_pci:    device registered at /dev/qrandom0
quantis_pci: Driver loaded. Found 1 card(s)
```

Important

If you update your kernel, you must recompile and reinstall the driver!

3.2.1.3. Autoload the Driver on Boot

Instead of using the **modprobe** command each time you want to load the driver, you can let the system to automatically load the driver on boot.

Note

Some distributions already load the driver on boot for each detected device (if available).

Before continue it is thus suggested to reboot your computer and run the command **dmesg** as explained in previous section. If the driver has been loaded and all Quantis devices have been detected, you can skip this section.

3.2.1.3.1. Debian-based Distributions

To automatically load the driver on boot, simply add the driver's name at the end of `/etc/modules`. You can type following command (as `root`) to add the entry:

```
# echo "quantis_pci" >> /etc/modules
```

3.2.1.3.2. Red Hat Enterprise Linux and CentOS Distributions

Red Hat Enterprise Linux checks for the existence of the `/etc/rc.modules` file at boot time, which contains various commands to load modules. The following commands configure loading of the `quantis_pci` module at boot time (as `root`):

```
# echo modprobe quantis_pci >> /etc/rc.modules
# chmod +x /etc/rc.modules
```

3.2.1.3.3. Other Distribution

Please consult your distribution's guide to know how to autoload a driver on boot.

3.2.1.4. Modify the Device's Permissions

Depending on the distribution, the Quantis PCI device might be accessible only to user `root`. UDEV (the device manager for the Linux 2.6 kernel series) must be instructed to allow other users to access the Quantis.

3.2.1.4.1. The `plugdev` group

IDQ provides a rule for UDEV that allows all users in group `plugdev` to access the Quantis device. The group `plugdev` is generally created on all modern distributions.

First check if your system already has the group `plugdev`:

```
$ grep plugdev /etc/group
```

If previous command display a line beginning as:

```
plugdev:x:
```

then your system has the group `plugdev`. When the **grep** command do not display any message, then the `plugdev` group do not exists on your system. Type following command (as `root`) to create the `plugdev` group:

```
# groupadd --gid 46 plugdev
```

3.2.1.4.2. Adding users to the `plugdev` group

Every user who is a member of the `plugdev` group can access hotpluggable devices (digital cameras, USB drives etc.).

You can use the command **groups** to display the groups your user is in:

```
$ groups
users adm dialout cdrom plugdev lpadmin admin sambashare
```

If your user is not in the group `plugdev`, use the **usermod** command (as `root`) to add the user `LOGIN` to the group `plugdev`:

```
# usermod -G plugdev -a LOGIN
```

3.2.1.4.3. UDEV rules

In the directory `Drivers/Unix/` on the USB flash there are two files with UDEV rules:

- `idq-quantis-rhel.rules` for Red Hat Enterprise Linux and CentOS distributions.
- `idq-quantis.rules` for all other distributions.

Copy (as `root`) the right file into `/etc/udev/rules.d/` directory:

- On Red Hat Enterprise Linux and CentOS distributions:

  ```
  # cp /media/USB_FLASH/Drivers/Unix/idq-quantis-rhel.rules
  ```

```
  /etc/udev/rules.d/
```

- On all other distributions:

```
# cp /media/USB_FLASH/Drivers/Unix/idq-quantis.rules
  /etc/udev/rules.d/
```

Note

The files `idq-quantis.rules` and `idq-quantis.rules` contain UDEV rules for both Quantis PCI and Quantis USB devices.

The udev daemon must now reload the rules. Type following command (as `root`) to reload the rules:

```
# udevadm control --reload-rules
```

Note

On Red Hat Enterprise Linux and CentOS distributions the udevadm command do not exists. User following command instead to reaload the rules:

```
# udevcontrol reload_rules
```

Note

Udev daemon only apply rules when creating the device's node (when the drivers loads). If the Quantis PCI driver is already loaded you need thus to unload and reload it to have the right permissions on the device:

```
# rmmod quantis_pci
# modprobe quantis_pci
```

3.2.1.5. Check Your Device

The driver has been installed and the system configured. You can now check if your device works correctly by reading some random bytes from the device. Following command reads 100 bytes from the first Quantis PCI device

```
$ head -c 100 /dev/qrandom0
```

Important

It is important not to run the **head** command as root but with the standard user who will use the Quantis PCI device. This is the only way to verify that permission has been granted to access the device.

Above command will display some random characters on the console.

Important

If you get one or more *Operation not permitted* messages, the device doesn't have the right permissions to access the Quantis device. In such a case:

- Verify that `/etc/udev/rules.d/idq-quantis.rules` exists and is the same as the one provided on the USB flash drive.
- Verify that your user is in `plugdev` group.
- Reboot the system to ensure that the new rules are loaded by the udev daemon.

You Quantis device is now installed. You can go to the next Chapter to install the software.

3.2.2. Quantis USB

Quantis USB only requires USB support enabled in the kernel[1]. The Quantis USB device is accessed through the open source library libusb-1.0 .

The following are step-by-step installation instructions.

3.2.2.1. libusb-1.0 Installation

Quantis USB device is accessed through the library libusb-1.0[2]. This library is available on all recent distributions and can be installed using the package manager of the distribution.

Warning

Do not confuse libusb-0.1 with libusb-1.0! libusb-0.1 is the legacy release and is not developed any more. As of December 2008, libusb-1.0 is the current stable branch. This new branch, used to access the Quantis USB, adds features missing from the first release.

3.2.2.1.1. Debian-based Distributions

Note for Debian users

libusb-1.0 is only available on Debian Squeeze and newer releases. It is also available on Debian Lenny backport. Please refer to the Debian help on how to enable backport packages. On all other Debian releases, you need to manually install libusb-1.0. Please refer to Section 3.2.2.1.4, “Manually Compile libusb-1.0”.

Note for Ubuntu users

libusb-1.0 is only available on Ubuntu Jauntry (9.04) and newer releases. On previous Ubuntu releases you need to manually install libusb-1.0. Please refer to Section 3.2.2.1.4, “Manually Compile libusb-1.0”.

Type the following command (as `root`) to install libusb-1.0 and the development package (needed if you want to write your own application):

```
# apt-get install libusb-1.0-0 libusb-1.0-0-dev
```

3.2.2.1.2. Red Hat Enterprise Linux and CentOS Distributions

libusb-1.0 is currently not available on Red Hat Enterprise Linux nor CentOS distributions. You need to manually install libusb-1.0. Please refer to Section 3.2.2.1.4, “Manually Compile libusb-1.0”.

[1]USB support in the kernel is generally enable on all modern Linux distributions.
[2]`http://libusb.org/wiki/Libusb1.0`

3.2.2.1.3. Other Distributions

Use the package manager of your distribution to install the library libusb-1.0. If the package is not available, please refer to Section 3.2.2.1.4, “Manually Compile libusb-1.0”.

3.2.2.1.4. Manually Compile libusb-1.0

If library libusb-1.0 can not is not available on the list of packages in the package manager of your distribution, you can easily compile it by hand.

First you need to download the library's sources. Go to the address `http://sourceforge.net/projects/libusb/files/libusb-1.0/` and download latest version.

Open the **Terminal** application and change the working directory to the one which contains the downloaded libusb-1.0 archive. Unpack the archive and compile the library (replace *x* with your version):

```
$ tar xvjf libusb-1.0.x.tar.bz2
$ cd libusb-1.0.x/
$ ./configure --prefix=/usr
$ make
```

When the library has been compiled, install it with following command (as `root`):

```
# make install
```

3.2.2.2. Modify the Device's Permissions

By default, the Quantis USB device is accessible only to user `root`. UDEV (the device manager for the Linux 2.6 kernel series) must be instructed to allow other users to access the Quantis. Please follow instructions on Section 3.2.1.4, “Modify the Device's Permissions”.

Important

If the Quantis USB device was already plugged in before reloading the udev rules, please unplug and replug the Quantis device, otherwise the device will have the wrong permissions.

3.2.2.3. Check Your Device

All requirement have been installed. You can now plug your Quantis USB device into your computer.

You can now check if your device works correctly with the **lsusb** command as following:

```
$ lsusb -d 0aba:0102 -v
```

Important

It is important not to run the **lsusb** command as root but with the standard user who will use the Quantis USB device. This is the only way to verify that permission has been granted to access the device.

Note

If above command returns the message:

```
lsusb: command not found
```

then the command **lsusb** is not installed. Install the **usbutils** package to fix the problem.

The output of above command should be similar to the following:

```
Bus 002 Device 035: ID 0aba:0102 Ellisys
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass          255 Vendor Specific Class
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x0aba Ellisys
  idProduct          0x0102
  bcdDevice            2.00
  iManufacturer           1 id Quantique
  iProduct                2 Quantis USB
  iSerial                 3 070001A410
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                 9
    bDescriptorType         2
    wTotalLength           25
    bNumInterfaces          1
    bConfigurationValue     1
    iConfiguration          0
    bmAttributes         0x80
      (Bus Powered)
    MaxPower              300mA
    Interface Descriptor:
      bLength                 9
      bDescriptorType         4
      bInterfaceNumber        0
      bAlternateSetting       0
      bNumEndpoints           1
      bInterfaceClass       255 Vendor Specific Class
      bInterfaceSubClass      0
      bInterfaceProtocol      0
      iInterface              0
      Endpoint Descriptor:
        bLength                 7
        bDescriptorType         5
        bEndpointAddress     0x86  EP 6 IN
        bmAttributes            2
          Transfer Type            Bulk
          Synch Type               None
          Usage Type               Data
        wMaxPacketSize     0x0200  1x 512 bytes
        bInterval               0
Device Qualifier (for other device speed):
  bLength                10
  bDescriptorType         6
  bcdUSB               2.00
  bDeviceClass          255 Vendor Specific Class
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  bNumConfigurations      1
Device Status:     0x0000
  (Bus Powered)
```

Important

Please verify that you have the *Device Qualifier* and *Device Status* information sections and not messages such as:

```
can't get device qualifier: Operation not permitted
can't get debug descriptor: Operation not permitted
cannot read device status, Operation not permitted
```

If you get one or more *Operation not permitted* messages, the device doesn't have the right permissions to access the Quantis device. In such a case:

- Verify that `/etc/udev/rules.d/idq-quantis.rules` or `/etc/udev/rules.d/idq-quantis-rhel.rules` exist and is the same as the one provided on the USB flash drive.
- Verify that your user is in `plugdev` group.
- Reboot the system to ensure that the new rules are loaded by the udev daemon.

You Quantis device is now installed. You can go to the next Chapter to install the software.

Chapter 4. The EasyQuantis application

The Quantis is delivered come with the EasyQuantis application. This application allows you to quickly generate random data.

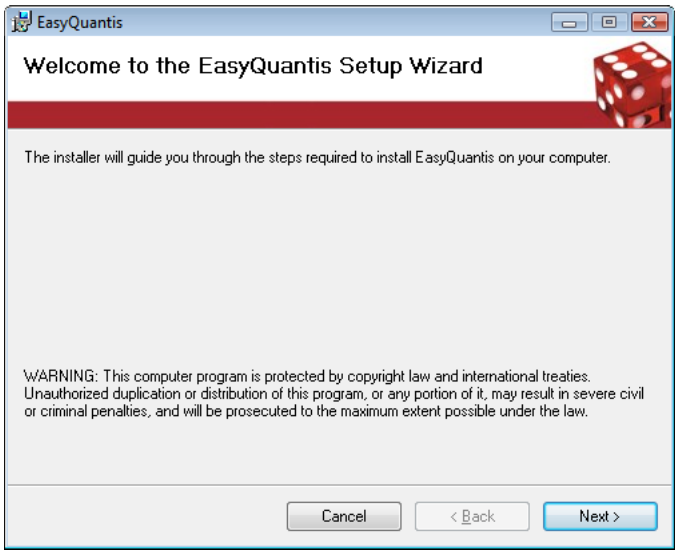
Important

Quantis PCI Express is software-compatible with Quantis PCI. EasyQuantis considers Quantis PCIe devices as Quantis PCI devices.

4.1. Installation

4.1.1. Windows Operating Systems

EasyQuantis is provided as Micrsoft Installer (MSI) package for easy installation. Just double-click on `EasyQuantis.msi` file and follow on-screen instructions.



Welcome to the EasyQuantis Setup Wizard

The installer will guide you through the steps required to install EasyQuantis on your computer.

WARNING: This computer program is protected by copyright law and international treaties.
Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

Figure 4.1. EasyQuantis Setup Wizard welcome

To launch the application click on the `EasyQuantis` icon in `Start -> Program`.

4.1.2. Linux Operating Systems

4.1.2.1. Install Requirements

EasyQuantis requires `libusb-1` and Qt 4 libraries (`qt4-core` and `qt4-gui`) to work.

If you installed a Quantis PCI card, plase follow instructions in Section 3.2.2.1, "libusb-1.0 Installation". If you installed a Quantis USB device, `libusb-1` has already been installed on your system.

Qt libraries are available on all major Linux distributions and they can be installed using the package manager of the distribution.

4.1.2.1.1. Debian-based Distributions

Open a terminal and install `libqt4-core` and `libqt4-gui` with following command (as `root`):

```
# apt-get install libqt4-core libqt4-gui
```

4.1.2.1.2. Other Distributions

Install `libqt4-core` and `libqt4-gui` with the package manager of your distribution.

4.1.2.2. Install the Application

EasyQuantis as well as Quantis libraries are provided in a bz2 archive.

On 32-bit system, you can install the application with following commands as `root` (replace *x.y* with current version):

```
# cd /mnt/USB_FLASH/
# tar xvjf QuantisRNG-2.x.y-Linux-i386.tar.bz2 -C /tmp/
# cd /tmp/QuantisRNG-2.x.y-Linux-i386/
# mv bin/EasyQuantis /bin/
# mv lib/libQuantis* /lib/
```

On 64-bit system, use following commands as `root` instead:

```
# cd /mnt/USB_FLASH/
# tar xvjf QuantisRNG-2.x.y-Linux-amd64.tar.bz2 -C /tmp/
# cd /tmp/QuantisRNG-2.x.y-Linux-amd64/
# mv bin/EasyQuantis /bin/
# mv lib64/libQuantis* /lib64/
```

You can run the EasyQuantis application by typing **EasyQuantis** on a terminal:

```
$ EasyQuantis
```

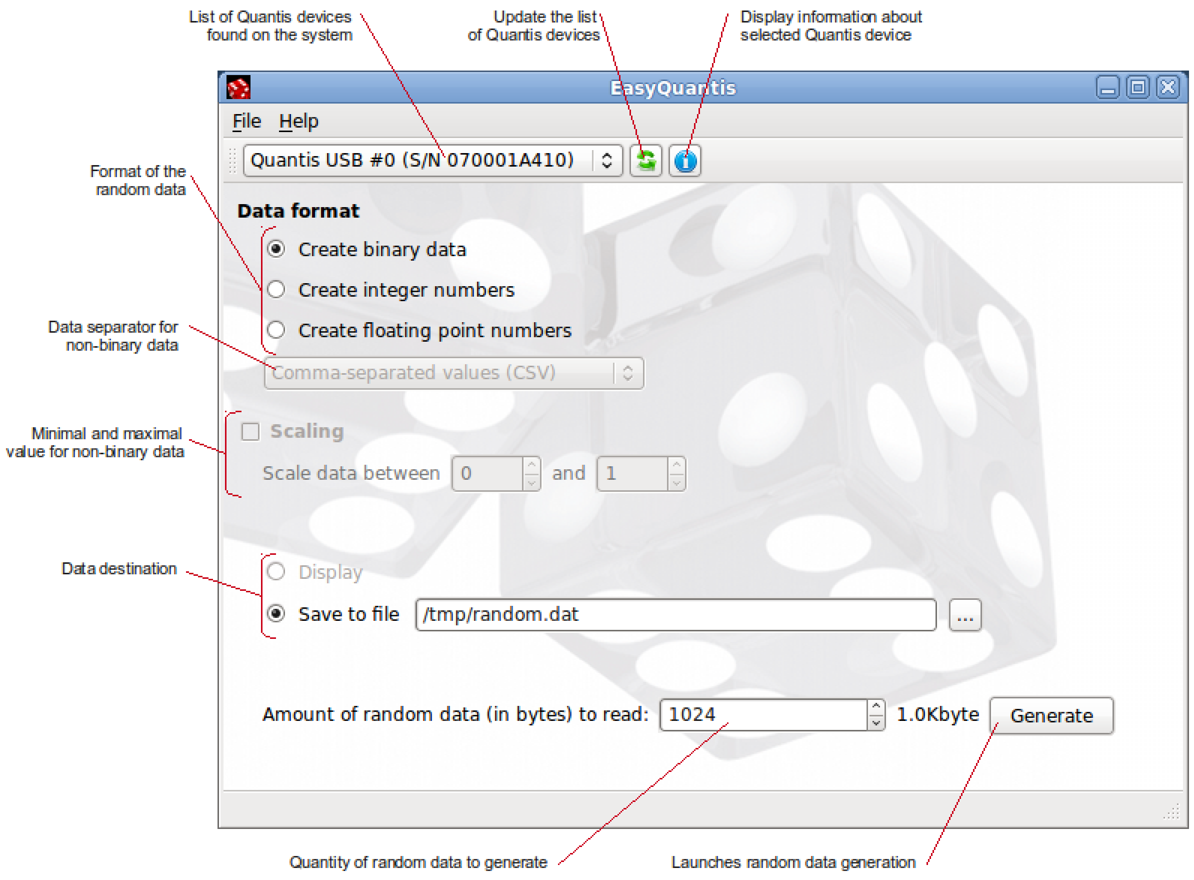
4.1.2.3. Uninstall the Application

To uninstall, manually remove installed files with following commands (as `root`):

```
# rm -Rf /bin/EasyQuantis
# rm -Rf /lib/libQuantis* # on 32-bit systems
# rm -Rf /lib64/libQuantis* # on 64-bit systems
```

4.2. Using EasyQuantis

Figure 4.2, “EasyQuantis main window” shows the main window of the **EasyQuantis** application.


List of Quantis devices found on the system
Update the list of Quantis devices
Display information about selected Quantis device
EasyQuantis
File
Help
Format of the random data
Data separator for non-binary data
Minimal and maximal value for non-binary data
Data destination
Quantity of random data to generate
Launches random data generation
Generate

Quantis USB #0 (S/N 070001A410)

Data format

[✓ Create binary data]
Create integer numbers
Create floating point numbers
Comma-separated values (CSV)

Scaling
Scale data between 0 and 1

Display
[✓ Save to file] /tmp/random.dat

Amount of random data (in bytes) to read: 1024 1.0Kbyte

Figure 4.2. EasyQuantis main window

To generate random data using EasyQuantis:

1. Select a Quantis device from the list.

2. Select a data format:

   - **Binary data.** Data is read from the Quantis and used AS IS
   - **Integer numbers.** Generates 32-bit random numbers ranging between -2'147'483'648 and 2'147'483'647 (inclusive).
   - **Floating point numbers.** Generates a number between 0.0 (inclusive) and 1.0 (exclusive).

3. Select a data separator:

   - **Comma-separated values (CSV).** CSV is a type of delimited text data, which uses a comma to separate values. The benefit of CSV is that they allow for the transfer of data across different applications.

     The following is an example of CSV:

     ```
     Value1,Value2,Value3,...,ValueN
     ```

   - **One entry per line.** Each values is on a separate line, as following:

     ```
     Value1
     Value2
     Value3
     ```

```
...
ValueN
```

Note

When generating binary data you cannot select data separator.

4. If needed, you can scale the random values to be within a smaller range.

Note

For more details about the scaling algorithms, please refear to Section 5.2.6.2.1.1, “Integral Values: The Scaling Algorithm”.

5. Select the data destination:

   - **Display.** Data is displayed on screen. You can copy-paste the data to your application.

     Use this option for *small* amounts of random data that you want to use it right away.

     Note

     This option is not available for binary data.

   - **Save to file.** Data is written on a file. Use this option to generate large amount of random data or to generate data for further use.

6. Select the amount of data to generate.

7. Click the Generate button and wait while the application generates the random data.

4.3. The EasyQuantis Command Line

EasyQuantis v1.1 and newer includes a command line parser, allowing you to use the application from the console or in a script.

4.3.1. Options

4.3.1.1. Generic Options

`-h [ --help ]`	Display a summary of available options.

4.3.1.2. Quantis Options

`-l [ --list ]`	List all devices available on the system.
`-p [ --pci ] arg`	Select the given Quantis PCI device as input device. `arg` is the number of the Quantis PCI device to use.
`-u [ --usb ] arg`	Select the given Quantis USB device as input device. `arg` is the number of the Quantis USB device to use.

4.3.1.3. Acquisition Options

`-n [ --size ] arg`	Set the number of bytes or values that should be read. If nothing is specified 1024 is used.

`-b [ --binary ] arg`	Generates a binary file. `arg` designates the filename.
`-i [ --integers ] arg`	Generates a file of integers numbers. `arg` designates the filename.
`-f [ --floats ] arg`	Generates a file of floating point numbers. `arg` designates the filename.
`-s [ --separator ] arg`	Set the separator string for non-binary files. The default format is one entry per line.
`--min arg`	Specify the minimal value for integers and floats numbers. If specified, requires `--max` to be specified too.
`--max arg`	Specify the maximal value for integers and floats numbers. If specified, requires `--min` to be specified too.

4.3.2. Usage Examples

In this section you will find some examples of usage of the EasyQuantis command line.

4.3.2.1. Generate Binary Data

```
EasyQuantis -p 0 -b random.dat -n 1073741824
```

Generates a binary file named `random.dat` of 1Gbyte using the Quantis PCI device number 0.

4.3.2.2. Generate Numbers

```
EasyQuantis -u 0 -i integers.dat -n 1000
```

Generates file named `integers.dat` with 1000 integer numbers.

4.3.2.3. Generate Scaled Numbers

```
EasyQuantis -u 0 -i integers.dat -n 1000 --min 1 --max 6
```

Generates file named `integers.dat` with 1000 integer numbers whose values are between 1 and 6.

Chapter 5. The Quantis Library

To easily access the Quantis device from your application, IDQ provides an abstraction library on all supported operating systems. The library allows you to easily write your (multi-platform) application without knowing how the Quantis devices internally works.

Important

API changed with Quantis library version 2.0. If your application uses a prior Quantis library version, pleas read to the Appendix C, *Migrating to the New API*.

Note for C++ users

Each time you request an operation, the Quantis library:

1. Opens the Quantis device;
2. Performs the requested operation;
3. Closes the Quantis device.

The C++ library has been optimized to open the Quantis device in the class constructor and close it in the class deconstructor. If your application is written in C++, it is suggested to use the C++ wrapper. Please read Chapter 6, *Quantis Library Wrappers* for further information.

5.1. Device Type

Almost all Quantis library functions require the device type to be specified. Currently there are two types:

- `QUANTIS_DEVICE_PCI` to specify a Quantis PCI or a Quantis PCI Express.
- `QUANTIS_DEVICE_USB` to specify a Quantis USB.

Important

Quantis PCI Express is software-compatible with Quantis PCI. There is no distinction between Quantis PCI and Quantis PCIe devices within the library. They are both considered as PCI devices.

5.2. Basic Functions

This section introduces minimal functions you need to use to read random data within your application.

5.2.1. QuantisCount

```
int QuantisCount(QuantisDeviceType deviceType);
```

Returns the number of devices that have been detected. It returns 0 when no card is installed or on error.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.

5.2.2. QuantisGetDriverVersion

```
float QuantisGetDriverVersion(QuantisDeviceType deviceType);
```

Returns the version of the driver as a number composed of the major and minor number. Integral value represent major number. Decimal places after the decimal point represent the minor number.

Returns a `QUANTIS_ERROR` code (casted to float) on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.

5.2.3. QuantisGetLibVersion

```
float QuantisGetLibVersion();
```

Returns the version of the library as a number composed of the major and minor number. Integral value represent major number. Decimal places after the decimal point represent the minor number.

5.2.4. QuantisGetModulesDataRate

```
int QuantisGetModulesDataRate(QuantisDeviceType deviceType,
                              unsigned int deviceNumber);
```

Returns the data rate (in bytes per second) provided by the Quantis device or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.

5.2.5. QuantisGetSerialNumber

```
char* QuantisGetSerialNumber(QuantisDeviceType deviceType,
                             unsigned int deviceNumber);
```

Get a pointer to the serial number string of the Quantis device. Currently only USB support serial number retrieval. On PCI and PCI Express device, the string "*S/N not available*" is returned.

On failure the string "*S/N not available*" is also returned.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.

5.2.6. QuantisRead

```
int QuantisRead(QuantisDeviceType deviceType,
                unsigned int deviceNumber,
                void* buffer,
                size_t size);
```

Reads random data from the Quantis device.

Returns `QUANTIS_SUCCES` on success or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.
`buffer`	a pointer to a destination buffer. The buffer **MUST** already be allocated. Its size must be **at least** `size` bytes.
`size`	the number of bytes to read (cannot be larger than `QUANTIS_MAX_READ_SIZE`).

Warning

If `buffer` is not allocated or the allocated size of memory is insufficient to store the whole data, the library will have unexpected results and may even lead to the crash of the application!

5.2.6.1. Reading Large Amount of Data

QuantisRead is not meant to read large amount of data. The maximal size of a request is defined by value `QUANTIS_MAX_READ_SIZE`. If you try reading a larger value `QuantisRead` will return an error. To read large amount of data you have to use a loop as in following example:

```
/* Chunk size. Recommended values are 2048 or 4096 */
chunkSize = CHUNK_SIZE;

remaining = SIZE;

while(remaining > 0u)
{
  /* Chunk size */
  if (remaining < chunkSize)
  {
    chunkSize = remaining;
  }

  /* Read data */
  result = QuantisRead(deviceType, 0, buffer, NUM_BYTES);

  /*
   * TODO:
   *  1. Check result (see example at the end of the chapter)
   *  2. Handle buffer (e.g. store data in a file)
   */

  /* Update info */
  remaining -= chunkSize;
}
```

5.2.6.2. Reading Basic Data Types

The function `QuantiRead` is useful to read a high quantity of raw random data. Depending on the application, it can be useful to be able to directly read basic data types. This section introduces functions for this purpose.

5.2.6.2.1. Integral Values

```
int QuantisReadShort(QuantisDeviceType deviceType,
                     unsigned int deviceNumber,
                     short* value);

int QuantisReadScaledShort(QuantisDeviceType deviceType,
                           unsigned int deviceNumber,
                           short* value,
                           short min,
                           short max);

int QuantisReadInt(QuantisDeviceType deviceType,
                   unsigned int deviceNumber,
                   int* value);

int QuantisReadScaledInt(QuantisDeviceType deviceType,
                         unsigned int deviceNumber,
                         int* value,
                         int min,
                         int max);
```

Reads a random 16-bit or 32-bit integral value. Returns `QUANTIS_SUCCES` on success or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.
`value`	a pointer to the destination value.
`min`	the minimal value the random number can take.
`max`	the maximal value the random number can take.

5.2.6.2.1.1. Integral Values: The Scaling Algorithm

Random numbers required by an application are very often in a range (much) smaller than the (fixed) range of the random number produced by the Quantis.

To perform the scaling, the highest number that is the largest multiple of the output range is selected. Random values equal or higher this limit are discarded. Following is the (simplified) C code of `QuantisReadScaledInt` which produces an *unbiased* number between `minValue` and `maxValue` (inclusive):

```
int rnd;

/* Output range */
unsigned long long range = maxValue - minValue + 1;

/* Range of the rnd value */
unsigned long long maxRange = 2^32;

/* Largest multiple of the output range */
unsigned long long limit = maxRange - (maxRange % range);

/* Read raw random number until it is lower the limit */
do
{
  QuantisReadInt(deviceType, deviceNumber, &rnd);
} while (rnd >= limit);
```

```
/* Scale value */
value = (rnd % range) + minValue;
```

Note

This scaling algorithm waste data when the Quantis generates random values equals or higher the limit. In the worst case (when `range = maxRange / 2 + 1`), the probability to drop a generated value is roughly 50%!

Warning

Raw random values are often scaled using the modulus operator, using something like:

```
minValue + (rawRndValue % (maxValue - minValue + 1))
```

where `%` represents the modulus operator. This formula produces a number between `minValue` and `maxValue` (inclusive), but in certain conditions (when range is not a multiple of the output range) the distribution of these numbers has a small bias that favours numbers at the lower end of the output range.

5.2.6.2.2. Floating Point Values

```
int QuantisReadFloat_01(QuantisDeviceType deviceType,
                        unsigned int deviceNumber,
                        float* value);

int QuantisReadScaledFloat(QuantisDeviceType deviceType,
                           unsigned int deviceNumber,
                           float* value,
                           float min,
                           float max);

int QuantisReadDouble_01(QuantisDeviceType deviceType,
                         unsigned int deviceNumber,
                         double* value);

int QuantisReadScaledDouble(QuantisDeviceType deviceType,
                            unsigned int deviceNumber,
                            double* value,
                            double min,
                            double max);
```

Reads a random floating point value between 0.0 (inclusive) and 1.0 (exclusive). Scaled versions reads a random floating point value between `min` (inclusive) and `max` (exclusive). Returns `QUANTIS_SUCCES` on success or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.
`value`	a pointer to the destination value.
`min`	the minimal value the random number can take.
`max`	the maximal value the random number can take.

Note

Floating point values are computed by dividing a random integral value (32-bit for `floats` and 64-bit for `double`) by the integral's value range (2^{32} and 2^{64} respectively).

Warning

Floating point scaling algorithm in certain conditions the distribution of these numbers has a small bias that favours numbers at the lower end of the output range. If you need unbiased random numbers, please consider to use `QuantisReadScaledShort` or `QuantisReadInt` instead:

```
/* Example: how to generate a random number between
 * 1.001 and 75.5 (inclusive)
 */

float min = 1.001;
float max = 75.5;
float multiplier = 1000.0;

int rndInt;
if (QuantisReadScaledInt(deviceType,
                         deviceNumber,
                         &rndInt,
                         (int)(min * multiplier),
                         (int)(max * multiplier)) < 0)
{
  /* Handle error */
}

float randomValue = (float)rndInt / multiplier;
```

5.2.7. QuantisStrError

```
char* QuantisStrError(QuantisError errorNumber);
```

Get a pointer to the error message string. This function interprets the value of `errorNumber` and generates a string describing the error. The returned pointer points to a statically allocated string, which shall not be modified by the program. Further calls to this function will overwrite its content.

Parameters:

`errorNumber`	the error number.

5.3. Advanced Functions

This section introduces advanced functions that allow a high control of the Quantis devices. Most users don't need to use these functions.

5.3.1. QuantisBoardReset

```
int QuantisBoardReset(QuantisDeviceType deviceType,
                      unsigned int deviceNumber);
```

Resets the Quantis board. Returns `QUANTIS_SUCCESS` on success or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.

Note

You generally don't need to reset the Quantis device: the Quantis library already resets the device when needed.

5.3.2. QuantisGetBoardVersion

```
int QuantisGetBoardVersion(QuantisDeviceType deviceType,
                           unsigned int deviceNumber);
```

Get the internal version of the board.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.

5.3.3. QuantisGetModulesCount

```
int QuantisGetModulesCount(QuantisDeviceType deviceType,
                           unsigned int deviceNumber);
```

Returns the number of modules that have been detected on a Quantis device or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.

5.3.4. QuantisGetModulesMask

```
int QuantisGetModulesMask(QuantisDeviceType deviceType,
                          unsigned int deviceNumber);
```

Returns a bitmask of the modules that have been detected on a Quantis device or a `QUANTIS_ERROR` code on failure.

Bit `n` is set in the bitmask if module `n` is present. For instance 5 (1101 in binary) is returned when modules 0, 2 and 3 have been detected.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.

5.3.5. QuantisGetModulesPower

```
int QuantisGetModulesPower(QuantisDeviceType deviceType,
                                                    unsigned int deviceNumber);
```

Get the power status of the modules. Returns 1 if the modules are powered, 0 if the modules are not powered and a `QUANTIS_ERROR` code on failure.

Note

This function is useful only for Quantis USB devices. Modules of Quantis PCI devices are always powered, thus the function always returns 1 on such devices.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.

5.3.6. QuantisGetModulesStatus

```
int QuantisGetModulesStatus(QuantisDeviceType deviceType,
                            unsigned int deviceNumber);
```

Returns the status of the modules on the device as a bitmask or a `QUANTIS_ERROR` code on failure.

Bit `n` is set in the bitmask if module `n` is enabled and functional. For instance 5 (1101 in binary) is returned when modules 0, 2 and 3 are enabled and functionals.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.

5.3.7. QuantisModulesDisable

```
int QuantisModulesDisable(QuantisDeviceType deviceType,
                          unsigned int deviceNumber,
                          int modulesMask);
```

Disable one or more modules. Returns `QUANTIS_SUCCES` on success or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.
`modulesMask`	a bitmask of the modules (as specified in `QuantisGetModulesMask` function) that must be disabled.

5.3.8. QuantisModulesEnable

```
int QuantisModulesEnable(QuantisDeviceType deviceType,
                         unsigned int deviceNumber,
                         int modulesMask);
```

Enable one or more modules. Returns `QUANTIS_SUCCES` on success or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.
`modulesMask`	a bitmask of the modules (as specified in `QuantisGetModulesMask` function) that must be enabled.

5.3.9. QuantisModulesReset

```
int QuantisModulesReset(QuantisDeviceType deviceType,
                        unsigned int deviceNumber,
                        int modulesMask);
```

Reset one or more modules. Returns `QUANTIS_SUCCES` on success or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of Quantis device.
`deviceNumber`	the number of the Quantis device. Note that first device is 0.
`modulesMask`	a bitmask of the modules (as specified in `QuantisGetModulesMask` function) that must be reset.

Note

This function executes sequentially `QuantisModuleDisable` and `QuantisModuleEnabled` with the given parameters.

5.4. Example

The following is a simple highly commented example for the Quantis library:

```
/* Global includes */
#include <stdio.h>
#include <stdlib.h>

/* Includes Quantis library's header */
#include "Quantis.h"

/* Define the number of bytes that should be read */
#define NUM_BYTES 100

int main()
{
  QuantisDeviceType deviceType;
  unsigned char* buffer;
  int result;
  int i;

  /* Select device type */
  if (QuantisCount(QUANTIS_DEVICE_PCI) > 0)
  {
    /* There is one ore more Quantis PCI device... */
    deviceType = QUANTIS_DEVICE_PCI;
  }
  else if (QuantisCount(QUANTIS_DEVICE_USB) > 0)
  {
    /* There is one ore more Quantis USB device... */
    deviceType = QUANTIS_DEVICE_USB;
  }
  else
  {
    /* No Quantis device has been found on the system */
    printf("No Quantis device found\n");
    return -1;
  }

  /* Allocate buffer's memory */
  buffer = (unsigned char*)malloc(NUM_BYTES);
  if (!buffer)
  {
    fprintf(stderr, "Unable to allocate memory\n");
    return -1;
  }

  /* Read random data from the Quantis*/
  result = QuantisRead(deviceType, 0, buffer, NUM_BYTES);
  /* Check if there are some errors */
  if (result < 0)
  {
    /* An error occured. Print the error message */
    fprintf(stderr,
            "An error occured when reading random bytes: %s\n",
            QuantisStrError(result));
    goto cleanup;
  }
  else if (result != NUM_BYTES)
  {
    /* Quantis did not return the number of bytes asked */
    fprintf(stderr,
            "Asked to read %d byts but received %d bytes\n",
            NUM_BYTES,
            result);
    goto cleanup;
  }

  /* Display buffer in HEX format */
  printf("Displaying %d random bytes in HEX format:\n",
         NUM_BYTES);
  for(i = 0; i < NUM_BYTES; i++)
  {
    printf("%02x ", buffer[i]);
  }
  printf("\n");

  /* Cleanup */
cleanup:

  if(buffer)
  {
    free(buffer);
  }

  return 0;
```

```
}
```

A more detailed example is available on the USB flash drive in the `Samples` directory.

Chapter 6. Quantis Library Wrappers

IDQ provides several wrappers to allow you to use the Quantis device with your preferred programming language.

Currently wrappers for following languages are available:

- C++
- C#
- Java
- VB.NET

Wrappers are for Object-oriented programming languages and they all have the same structure:

- The class is named `Quantis`.
- On class instantiation, you must provide the `deviceType` and the `deviceNumber`.
- Names of public functions are the same as the name of functions of Quantis library without the prefix *Quantis*, for instance `QuantisCount` is named `Count` in the wrapper. Functions definitions do not require `deviceType` and `deviceNumber` variables since defined globally within the class. Only exception are static function which have the same definition.

Please refer to the sample available with each wrapper for further details.

Note for wrapper for C++

Since the Quantis device is kept open until the Quantis class is not destroyed, it is highly recommended to reduce the scope of the Quantis variable as much as possible. n particular it is discouraged to define the Quantis variable global.

6.1. Example

Here is the example presented in previous chapter modified to use C++ wrapper.

```
/* Global includes */
#include <iostream>
#include <cstdlib>
#include <string>

/* Includes Quantis library's header */
/* Note the hpp extension! */
#include "Quantis.hpp"

/* Define the number of bytes that should be read */
#define NUM_BYTES 100

using namespace std;

int main()
{
  QuantisDeviceType deviceType;
  int result;
  /* Select device type */
  if (Quantis::Count(QUANTIS_DEVICE_PCI) > 0)
  {
    /* There is one ore more Quantis PCI device... */
    deviceType = QUANTIS_DEVICE_PCI;
  }
  else if (Quantis::Count(QUANTIS_DEVICE_USB) > 0)
  {
    /* There is one ore more Quantis USB device... */
    deviceType = QUANTIS_DEVICE_USB;
  }
  else
  {
    /* No Quantis device has been found on the system */
    cout << "No Quantis device found" << endl;
    return -1;
  }

  try
  {

    /* Creates a quantis object */
    Quantis quantis(deviceType, 0);

    /* Read random data from the Quantis*/
    string buffer = quantis.Read(NUM_BYTES);
    if (buffer.length() != NUM_BYTES)
    {
      /* Quantis did not return the number of bytes asked */
      cerr << "Asked to read " << NUM_BYTES
           << " byts but received " << buffer.length()
           << " bytes" << endl;
      return -1;
    }

    // Display buffer in HEX format
    cout << "Displaying " << NUM_BYTES
         << " random bytes in HEX format:" << endl;
    string::iterator it = buffer.begin();

    while (it != buffer.end())
    {
      cout << setw(2) << setfill('0') << hex
           << static_cast<int>(static_cast<unsigned char>(*it++))
           << " ";
    }
    cout << endl;

  catch (runtime_error &ex)
  {
    cerr << "Error while accessing Quantis device: "
         << ex.what() << endl;
    return -1;
  }

  return 0;
}
```

Appendix A. Troubleshooting

A.1. EasyQuantis

A.1.1. EasyQuantis crash on Linux, with one of the following errors:

- *Segmentation fault*.
- *symbol lookup error: EasyQuantis: undefined symbol: _ZN14QPlainTextEditC1EP7QWidget*.

These errors are generally caused by a binary incompatible Qt library. The binary of EasyQuantis provided by ID Quantique has been linked against Qt version 4.3.4. Or Qt libraries are only backward compatible, which means that you need to installe QT4 version 4.3.4 or newer.

To solve this issue please upgrade your Qt libraries to version 4.3.4 or newer.

If Qt version 4.3.4 (or newer) is not available on your system, you can still use EasyQuantis in command line, which is not affected by this issue. Please refear to Section 4.3, “The EasyQuantis Command Line” for more help.

This issue can also be solved recompiling the Quantis library and EasyQuantis on your system.

Appendix B. Frequently Asked Questions (FAQ)

B.1. Quantis Library

B.1.1. Can I use the 32-bit Quantis library on a 64-bit system?

Yes, you can use the 32-bit Quantis library with a 32-bit application on 64-bit systems. Not however that you can not use the 32-bit Quantis library with a 64-bit application nor the 64-bit Quantis library with a 32-bit application.

B.1.2. On Microsoft Windows I must copy the `Quantis.dll` library on system directory (`C:\Windows\System32`)?

No,it is not mandatory. IDQ recommends to install the `Quantis.dll` library in your application's directory.

B.1.3. On Microsoft Windows, when I use `Quantis.dll` within my application I have the error "*The application has failed to start because WINUSB.DLL was not found. Re-installing the application may fix this problem*". What should I do?

This problem occurs with `Quantis.dll` v2.1 (and older) when the Quantis USB driver is not installed. This issue has been fixed in `Quantis.dll` v2.2. Please update your `Quantis.dll` to the latest available version.

B.2. EasyQuantis

B.2.1. On Microsoft Windows, when I launch EasyQuantis I have the error *"The application has failed to start because WINUSB.DLL was not found. Re-installing the application may fix this problem*". What should I do?

This problem occurs with EasyQuantis 1.0 when the Quantis USB driver is not installed. This issue has been fixed in EasyQuantis 1.1. Please update EasyQuantis to the latest available version.

B.2.2. When I launch EasyQuantis on Microsoft Windows a console appears for a few seconds. What is wrong?

EasyQuantis integrates a command line interface and a graphical interface. Or on Microsoft Windows it is not possible to build an hybrid Windows/Console application. EasyQuantis has been built as a Console application. When launched, the system automatically creates a console windows and if no argument has been provided to the application, the console window is hidden and the graphical interface is displayed. There is unfortunately no easy way to avoid this issue.

Appendix C. Migrating to the New API

Quantis library version 2.0 slightly changed API. This is mainly due to the merge of the old Quantis library (used to access Quantis PCI devices) and Quantis-USB library (used to access Quantis USB devices) into a unique library.

Main difference between 1.x and 2.0 versions is the addition of the parameter `deviceType`, which allows you to specify the type of device to use (PCI/PCIe or USB). Additionally functions names have been modified when ambiguous. See Table C.1, “API 1.x and 2.0 functions equivalences.” for equivalences between API 1.x and 2.0.

API 1.x functions	**API 2.0 functions**
`int quantisBoardReset( int cardNumber);`	`int QuantisBoardReset( QuantisDeviceType deviceType, unsigned int deviceNumber);`
`int quantisBoardVersion( int cardNumber);`	`int QuantisGetBoardVersion( QuantisDeviceType deviceType, unsigned int deviceNumber);`
`int quantisCount();`	`int QuantisCount( QuantisDeviceType deviceType);`
`int quantisDriverVersion();`	`float QuantisGetDriverVersion( QuantisDeviceType deviceType);`
`int quantisGetModules( int cardNumber);`	`int QuantisGetModulesMask( QuantisDeviceType deviceType, unsigned int deviceNumber);`
`char* quantisGetSerialNumber( int cardNumber);`	`char* QuantisGetSerialNumber( QuantisDeviceType deviceType, unsigned int deviceNumber);`
`int quantisLibVersion();`	`float QuantisGetLibVersion();`
`int quantisModuleDataRate( int cardNumber);`	`int QuantisGetModulesDataRate( QuantisDeviceType deviceType, unsigned int deviceNumber);`
`int quantisModulesDisable( int cardNumber, int moduleMask);`	`int QuantisModulesDisable( QuantisDeviceType deviceType, unsigned int deviceNumber, int modulesMask);`
`int quantisModulesEnable( int cardNumber, int moduleMask);`	`int QuantisModulesEnable( QuantisDeviceType deviceType, unsigned int deviceNumber, int modulesMask);`
`int quantisModulesPower( int cardNumber);`	`int QuantisGetModulesPower( QuantisDeviceType deviceType, unsigned int deviceNumber);`
`int quantisModulesReset( int cardNumber, int moduleMask);`	`int QuantisModulesReset( QuantisDeviceType deviceType, unsigned int deviceNumber, int modulesMask);`

API 1.x functions	API 2.0 functions
`int quantisModulesStatus( int cardNumber);`	`int QuantisGetModulesStatus( QuantisDeviceType deviceType, unsigned int deviceNumber);`
`int quantisRead( int cardNumber, void* buffer, unsigned int size);`	`int QuantisRead( QuantisDeviceType deviceType, unsigned int deviceNumber, void* buffer, size_t size);`

Table C.1. API 1.x and 2.0 functions equivalences.

C.1. Compatibility Wrapper

IDQ provides a compatibility wrapper that allows you to use old API with the new library. This is meant to facilitate the migration of your application to the new API.

Important

It is highly suggested to update your application to the new API as soon as possible.

To use the compatibility wrapper, define `QUANTIS_DEVICE_TYPE` and then include `Quantis-Compat.h` instead of `quantis.h` and recompile your application:

```
/*
 * Define Quantis type:
 *   - set QUANTIS_DEVICE_TYPE to 1 for Quantis PCI/PCIe
 *   - set QUANTIS_DEVICE_TYPE to 2 for Quantis USB
 */
#define QUANTIS_DEVICE_TYPE 1

/* Includes compatibility wrapper */
#include "Quantis-Compat.h"
```

Note

On Microsoft Windows systems, you can try to rename `QuantisPci-Compat.dll` to `Quantis.dll` or `QuantisUsb-Compat.dll` to `Quantis-Usb.dll` and replace your old library. This way you normally do not need to recompile your application.

Appendix D. Notes

D.1. Images

Some images used on this manual and on Quantis software are from VistaICO.com.

Bibliography

Websites

[USB] *Official USB website*. http://www.usb.org/ .