Quantis PCIe-40M and PCIe-240M User Manual

Version : 1.0
Date : 16.04.2020

ID Quantique SA
Ch. de la Marbrerie, 3bis
CH-1227 Carouge/Geneva
Switzerland

Tel: +41 (0)22 301 83 71
Fax: +41 (0)22 301 83 79
www.idquantique.com
info@idquantique.com

Please send any comment to
support@idquantique.com

Information in this document is subject to change without notice.

Disclaimer

Revision history

Version	Scope
Version 1.0	Initial version of the document

Contents

1. Introduction

Thank you for purchasing a Quantis Random Number Generator.

The goal of a random number generator device is to produce a sequence of numbers that is impossible to predict.

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

Quantis PCIe-40M and PCIe-240M are the latest versions of Quantis PCIe products.

While legacy PCIe products, PCIe-4M and PCIe-16M, embed legacy Quantis module, PCIe-40M and PCIe-240M embed IDQ20MC1 QRNG chips that rely on following elementary quantum optical process to produce random data. A multi-mode LED illuminates CMOS image sensor. The number of photo-electrons stored in each pixel follows poissonian distribution: that's where quantum randomness come from. After **digitization** by ADC, only the bits that cannot be influenced by non-quantum noise are kept. **Post-processing** can be done if required but Quantis entropy source are almost perfect and pass most of the statistical tests. An **auto-calibration** function controls LED optical power and CIS exposure time. This ensure random bits produced are unpredictable, independently of any environmental and operating conditions.

Note that Quantis lib and EasyQuantis application are able to handle a number of new and legacy PCIe cards at the same time.

1.1 Document scope

This document is the User Manual for QRNG PCI-Express card “Quantis PCIe-40M” and “Quantis PCIe-240M” that embed Quantis IDQ20MC1 QRNG chips.

1.2 What you need

To use your Quantis PCIe board, you need a PC with a supported operating system installed (see table below) and a PCI Express x1 slot available.

Operating System	**Quantis PCIe-40M**	**Quantis PCIe-240M**
Windows 10	supported	supported
Ubuntu 18.04	supported	supported
CentOS 7	supported	supported

1.3 Additional Requirements

1.3.1 Linux

On Linux systems, you additionally need:

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

2. Quantis PCIe-40M and PCIe-240M Hardware installation

This chapter provides unpacking and installation information for Quantis PCIe-40M and PCIe-240M.

CAUTION

Under ordinary circumstances, the 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 don't have one, touch with both of your hands a safely grounded object or 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 wrist band, 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 POWER YOUR SYSTEM IF THE QUANTIS PCIe IS DAMAGED.

2.1.Unpacking

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

- a Quantis PCIe card
- a USB Flash Drive with Manual, Drivers, Libraries, Applications and Samples

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

2.2.Installing the PCIe Card

1. Shut down the computer, unplug its power cord and remove the chassis cover
2. Locate the PCI Express x1 slot. If necessary, remove the metal blanking plate from this slot, then align your Quantis card with the PCIe slot and press it in firmly until the card is fully inserted.
3. Install the bracket screw and secure the card to the computer chassis
4. Close the computer's case chassis
5. Switch the computer power on
6. Install the driver (see next 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.

3.1. Windows Driver Installation

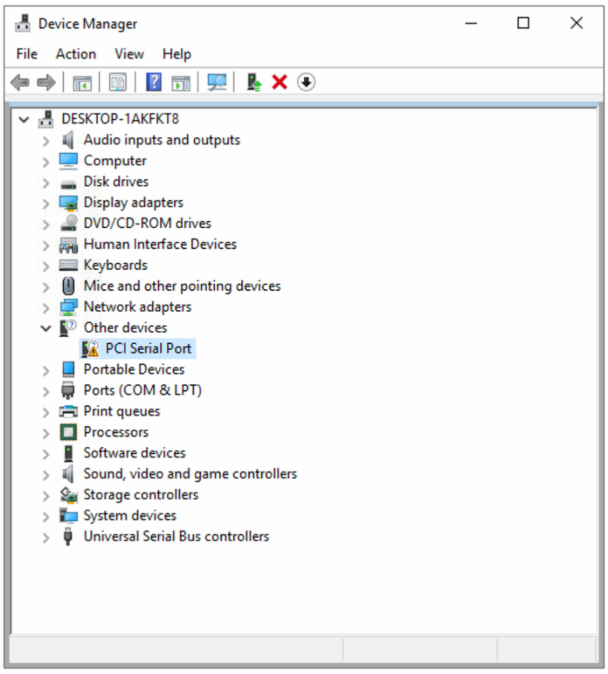
This section contains instructions on how to install a Quantis device on Windows Operating Systems. The software and drivers are supplied on a USB flash drive. Insert this USB flash drive into an available USB port.

Important

In this section, we assume that the letter of the USB flash drive provided by IDQ is drive D: . If this is different on your machine, substitute your corresponding drive name for D: in the appropriate places in this instruction.

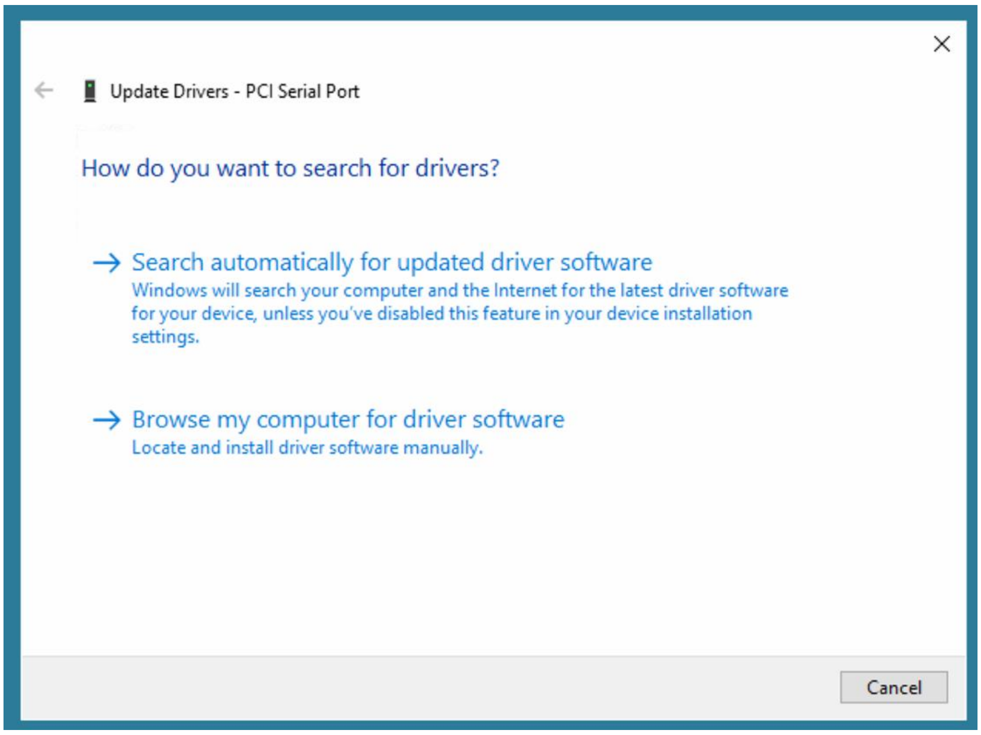
3.1.1. Windows 10 driver installation

The Quantis is not automatically detected by Windows 10. A manual installation is required. For this Run the *Device Manager* from the Control panel.



After connecting the Quantis-PCIe, the device is listed under other devices. Right click on the icon and select *Update Driver*

Then select *Browse my computer for driver software*.



Update Drivers - PCI Serial Port

How do you want to search for drivers?

→ 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.

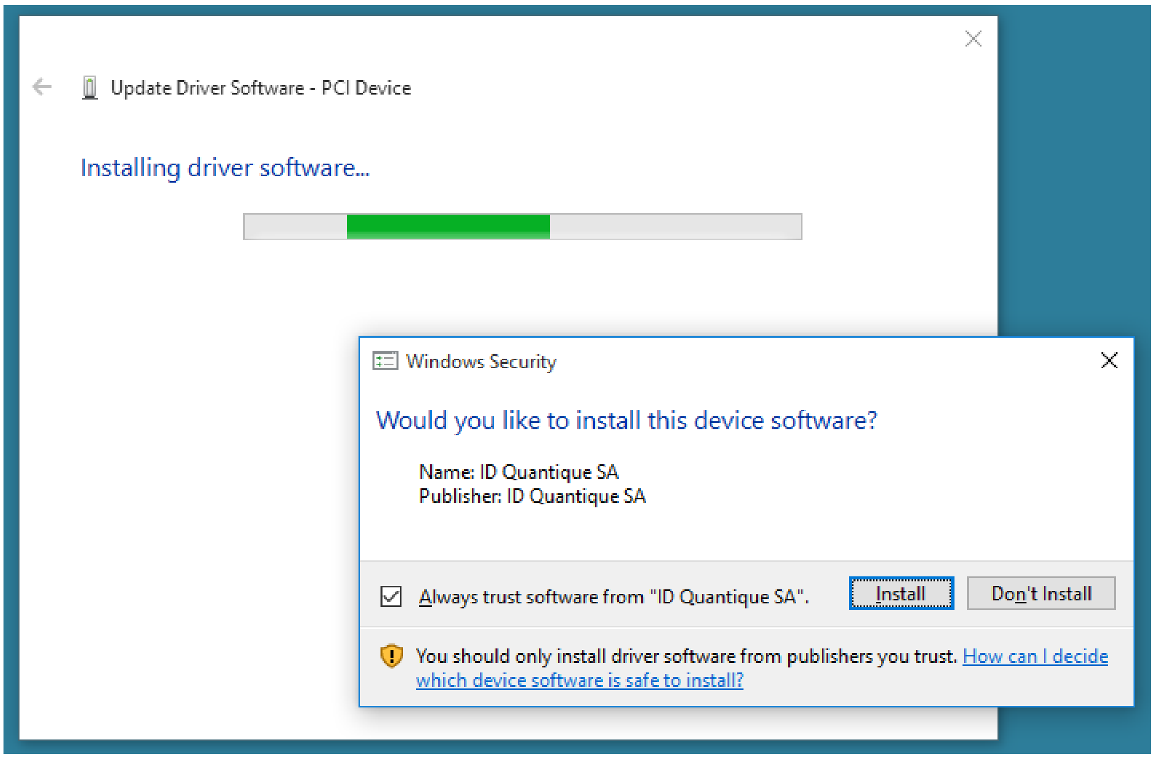
→ Browse my computer for driver software
Locate and install driver software manually.

Then click the button **Browse** and select the directory `D:\pcie-chip-x.x.x\Drivers\windows`. This directory contains all drivers for Windows. Activate the option

Include subfolders and validate your choices by clicking the **Next**.

Wait while Windows installs the driver.

Windows ask you to confirm the installation. Click the **install** button.


Update Driver Software - PCI Device
Installing driver software...
Windows Security
Install
Don't Install

Would you like to install this device software?

Name: ID Quantique SA
Publisher: ID Quantique SA

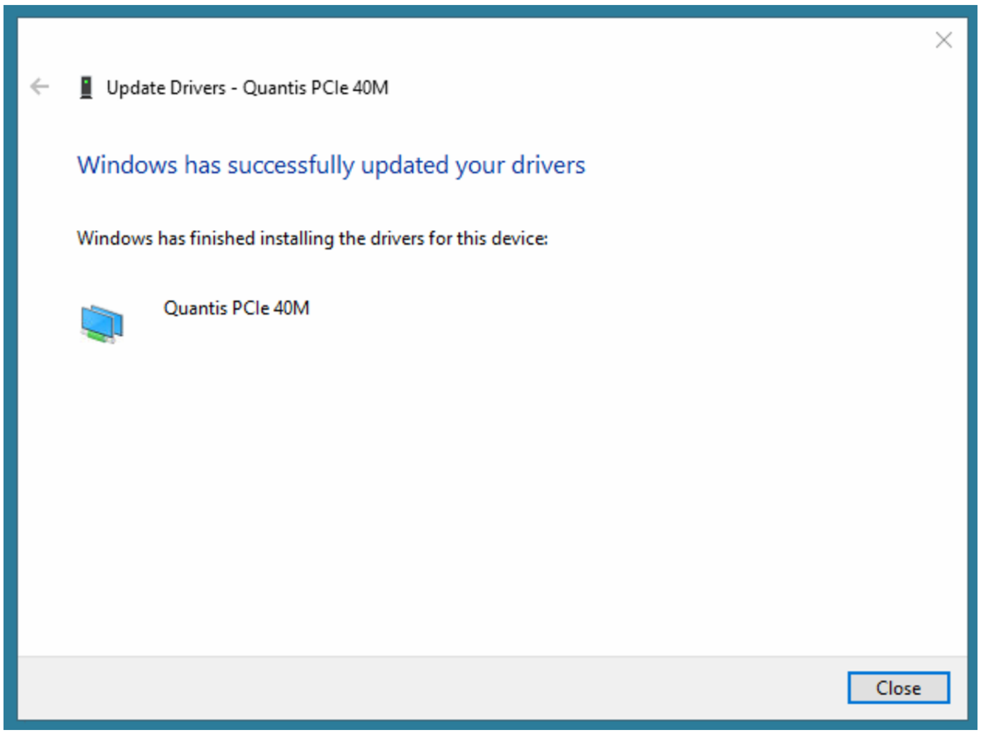
☑ Always trust software from "ID Quantique SA".

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

Update Driver Software: Completed

When the wizard has finished installing the Quantis driver, click the **Close** button to exit the installation.

Reboot the computer if asked.

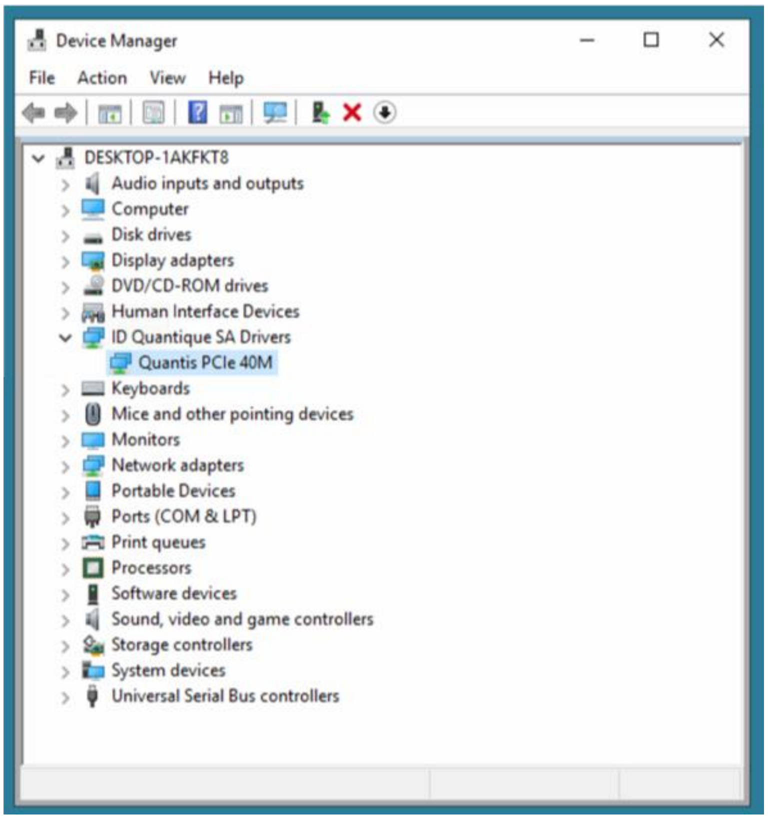

Update Drivers - Quantis PCIe 40M

Windows has successfully updated your drivers

Windows has finished installing the drivers for this device:

Quantis PCIe 40M

Your Quantis device is now installed. The Quantis PCIe is now listed under IDQ devices.



DESKTOP-1AKFKT8
- Audio inputs and outputs
- Computer
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- ID Quantique SA Drivers
  - Quantis PCIe 40M
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Portable Devices
- Ports (COM & LPT)
- Print queues
- Processors
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers

You can go to the next chapter and install the application software.

3.2.Linux Driver Installation

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

Note: In this section, we assume that the USB flash drive with the software is mounted on `/media/USB_FLASH`. If this is different on your machine, substitute your corresponding drive name in the appropriate places in this instruction.

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

3.2.1. 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. Debian-based Distributions

Debian-based distributions have a 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 Compile and Install Driver "Compile and Install Driver".

3.2.1.2. 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 their 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 Compile and Install Driver "Compile and Install Driver".

3.2.2. Compile and Install Driver

IMPORTANT
A kernel update can break the driver compilation. If it is the case and you have a supported OS, please contact our support at support@idquantique.com.

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

First copy the source code of the driver to `/tmp`:

```
$ cp -R /media/USB_FLASH/pcie-chip-x.x.x/Drivers/linux /tmp/
```

Change to the directory which contains the driver and compile the driver:

```
$ cd /tmp/linux/driver/
$ make
```

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

```
# make install
# modprobe quantis_chip_pcie
```

The **modprobe** command will load the driver. By default it is set to RNG mode that generates post-processed data (see part 3.2.3).

You can verify that the driver has been successfully loaded and all your Quantis PCIe cards

have been detected with the command **dmesg**:

```
$ dmesg | grep quantis
…
[    1.783471] quantis_chip_pcie vx.x.x
[    1.899741] quantis_chip_pcie: PCB revision:  x
[    1.899743] quantis_chip_pcie: Firmware version: x.x.x
[    1.899745] quantis_chip_pcie: Firmware build date: x
[    1.899752] quantis_chip_pcie: Serial number:xxxxxxxxxxx
[    1.899752] quantis_chip_pcie: RNG MODE (qrng_mode:0)
```

IMPORTANT

1. If you update your kernel, you must recompile and reinstall the driver!
2. Please wait at least 1 second between the **`modprobe`** command and any data acquisition

3.2.3. Entropy and RNG modes

By default, the driver will configure the board in RNG mode (see table below). For Quantum entropy mode use the following command:

```
# modprobe quantis_chip_pcie default_qrng_mode=1
```

"default_qrng_mode" argument:

Argument value	Mode name	Acquisition speed PCIe-40M	Acquisition speed PCIe-240M	Mode description
1	SAMPLE	38.3 Mbits/s	232 Mbits/s	Entropy data (also called "Quantum entropy")
0 (default)	RNG	9.6 Mbits/s	58 Mbits/s	Cryptographic secure random data. Quantum entropy data has been post-processed using NIST 00-90 A/B/C compliant DRBG

See the IDQ20MC1 datasheet and documentation for more details on entropy data and RNG data.

3.2.4. Loading legacy and PCIe-40M/240M in the same system

If you want to have legacy devices PCIe-4M/16M and new generation devices PCIe-40M/240M in the same computer, you should use the parameter **`device_file_first_index`** while loading the driver. Example:

```
# modprobe quantis_chip_pcie ... device_file_first_index=X
```

The integer **X** is the first index for PCIe-40M/240M devices in **/dev/qrandomX** which is 0 by default.

In the case where legacy devices are also installed, **X** should be bigger than the number of legacy devices.

3.2.5. Auto-load the Driver on Boot-up

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

3.2.5.1. 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 the following command (as root) to add the entry:

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

3.2.5.2. 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 the loading of the quantis_pci module at boot time (as root):

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

3.2.6. Modify the Device's Permissions

Depending on the distribution, the Quantis PCIe device might be accessible only to user root. UDEV (the device manager) must be instructed to allow other users to access the Quantis device.

3.2.6.1. 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 the previous command displays a line beginning with:

```
plugdev:x:
```

then your system has the group plugdev.

If the grep command does not display any message, then the plugdev group doesn't exists on your system. Type following command (as root) to create the plugdev group:

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

3.2.6.2. Adding users to the plugdev group

Every user who is a member of the plugdev group can access hot-pluggable 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` (substitute your own login name for `LOGIN`):

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

3.2.6.3. UDEV rules

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

```
# cp /media/USB_FLASH/pcie-chip-x.x.x/Drivers/linux/udev/idq-quantis.rules
/etc/udev/rules.d/
```

Note: The file `idq-quantis.rules` contain UDEV rules for all Quantis devices (legacy included).

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

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

Note: The 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_chip_pcie
# modprobe quantis_chip_pcie
```

3.2.7. 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. The following command reads 100 bytes from the first Quantis PCI device found on the system (not as root):

```
$ head -c 128 /dev/qrandom0 | hexdump -C
```

IMPORTANT
It is important not to run the head command as root but as the standard user who will use the Quantis PCI device. Otherwise you won't be verifying that permission has been granted to you to access the device.

The above command will display 128 bytes of random data on the console.

IMPORTANT
If you get one or more *Operation not permitted* messages, you don'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 has the same content as the one provided on the USB flash drive.
- Verify that your user is in the `plugdev` group.
- Reboot the system to ensure that the new rules are loaded by the udev daemon.

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

4. EasyQuantis Application

Quantis is delivered with the EasyQuantis application that allows you to quickly and easily generate random data.

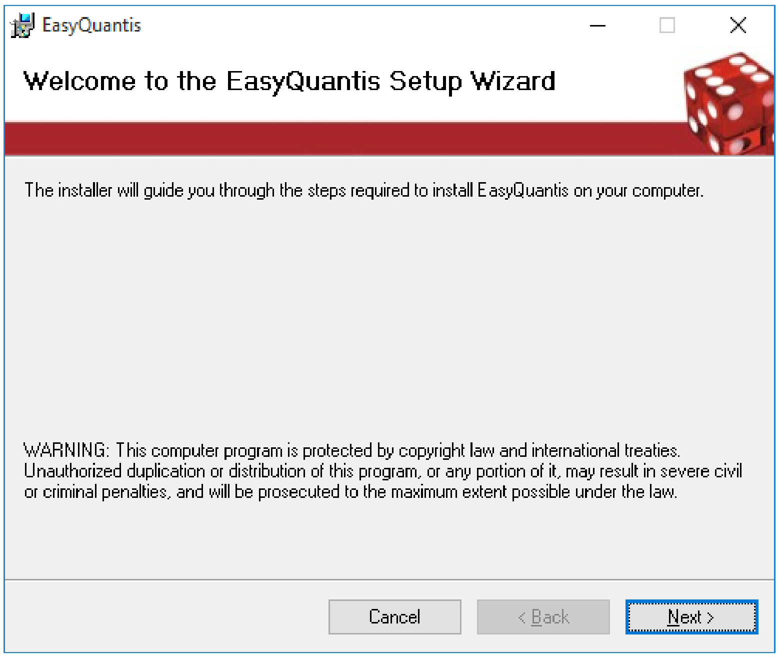
IMPORTANT
Easy Quantis can handle several legacy and new generation Quantis PCIe cards if needed. Note that the new Quantis PCIe-40M and PCIe-240M can be identified by their serial number wheras legacy PCI devices don't have serial numbers.

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

4.1. EasyQuantis Application Installation

4.1.1. Windows Operating Systems

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



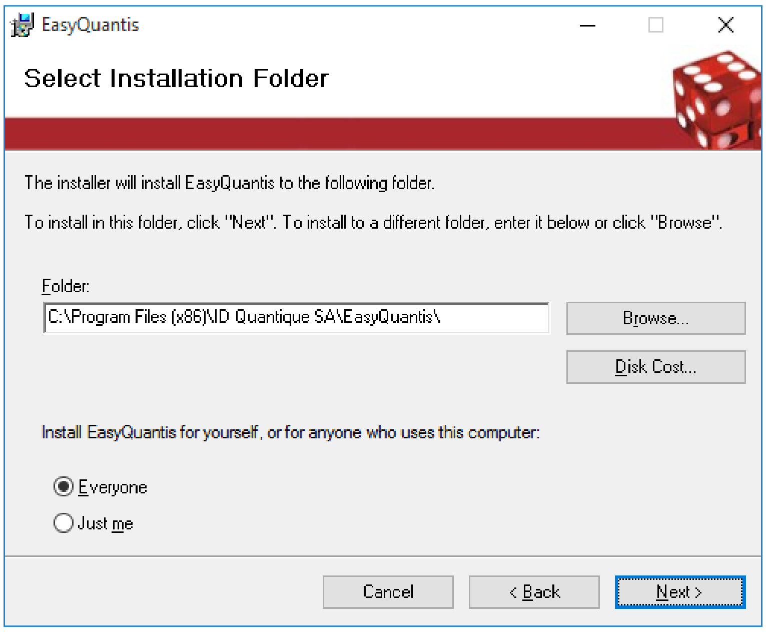
EasyQuantis

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.

Then you may select a directory to install EasyQuantis.



EasyQuantis

Select Installation Folder

The installer will install EasyQuantis to the following folder.

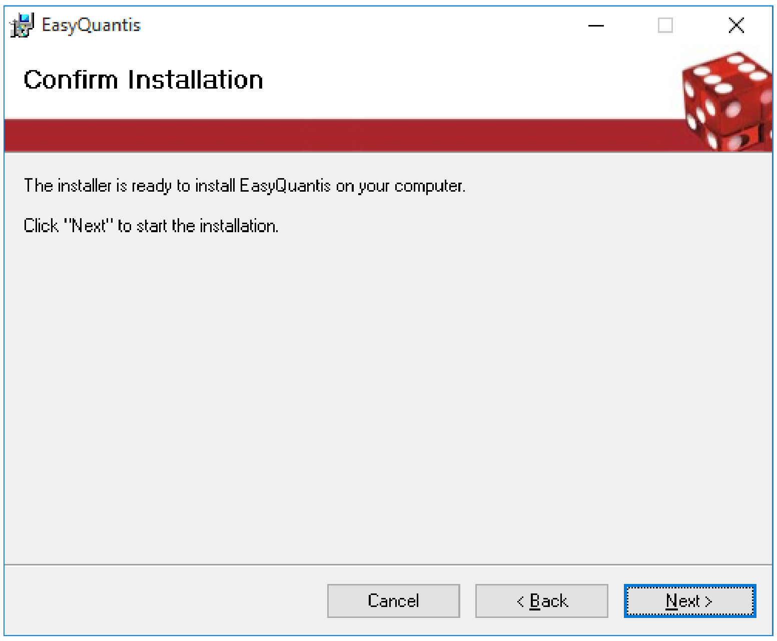
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
C:\Program Files (x86)\ID Quantique SA\EasyQuantis\

Install EasyQuantis for yourself, or for anyone who uses this computer:

[✓ Everyone]
Just me

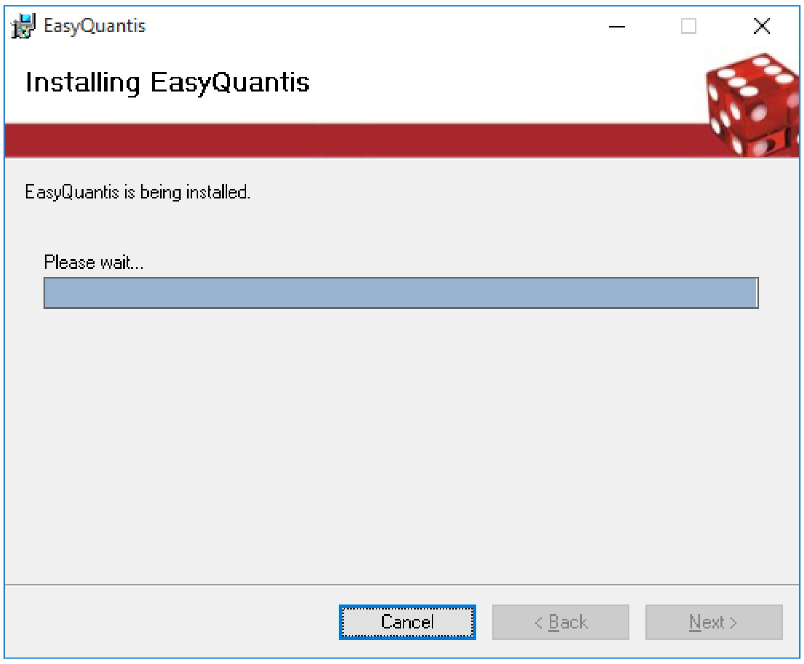
Then confirm you are ready to install:



The installer is ready to install EasyQuantis on your computer.

Click "Next" to start the installation.

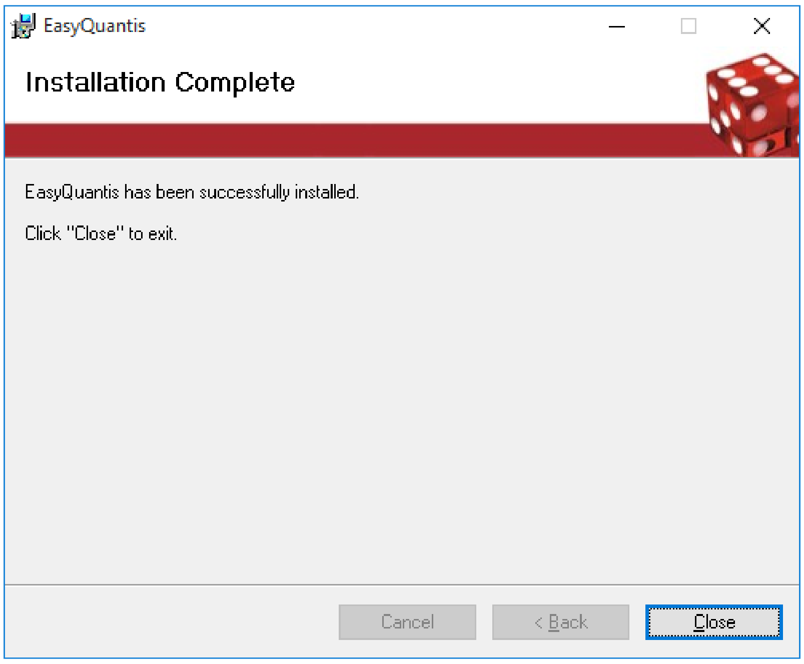
The installation starts. For security reasons you will be asked to allow the installation. Click yes.



EasyQuantis is being installed.

Please wait...

After copying all required files, the final screen shows the result. Click Close to quit.



EasyQuantis has been successfully installed.

Click "Close" to exit.

To launch the application, click on the EasyQuantis icon in the start screen.

4.1.2. Linux Operating Systems

4.1.2.1. Install Requirements

EasyQuantis requires the `libusb-1` and Qt4 libraries (`qt4-core` and `qt4-gui`) to work.

If you have a Quantis PCIe card, please follow instructions in Section 3.2.2.1, “libusb-1.0 Installation”.
If you have 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-dev using the following command (as root):

```
# apt-get install libqt4-dev
```

4.1.2.1.2. Other Distributions

Install `libqt4-dev` (or `libqt4-devel`) using the package manager of your distribution.

4.1.2.2. Install the Application

The EasyQuantis and Quantis libraries are provided in a bz2 archive.

On 32-bit systems, you can install the application using the following commands as `root` (replace x.y with your version number):

```
# 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 systems, use the 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/
```

On some distributions you need to copy the 64 bits libraries to /lib64:

```
# mv lib64/libQuantis* /lib64/
```

On other distributions you need to copy to /lib/x86_64-linux-gnu

```
# mv lib64/libQuantis* /lib/x86_64-linux-gnu
```

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 as follows (as `root`):

For 32 bits system:

```
# rm -Rf /bin/EasyQuantis
# rm -Rf /lib/libQuantis*
```

For 64 bits system:

```
# rm -Rf /bin/EasyQuantis
# rm -Rf /lib64/libQuantis*
```

Or

```
# rm -Rf /bin/EasyQuantis
# rm -Rf /lib/x86_64-linux-gnu/libQuantis*
```

4.2. EasyQuantis functions

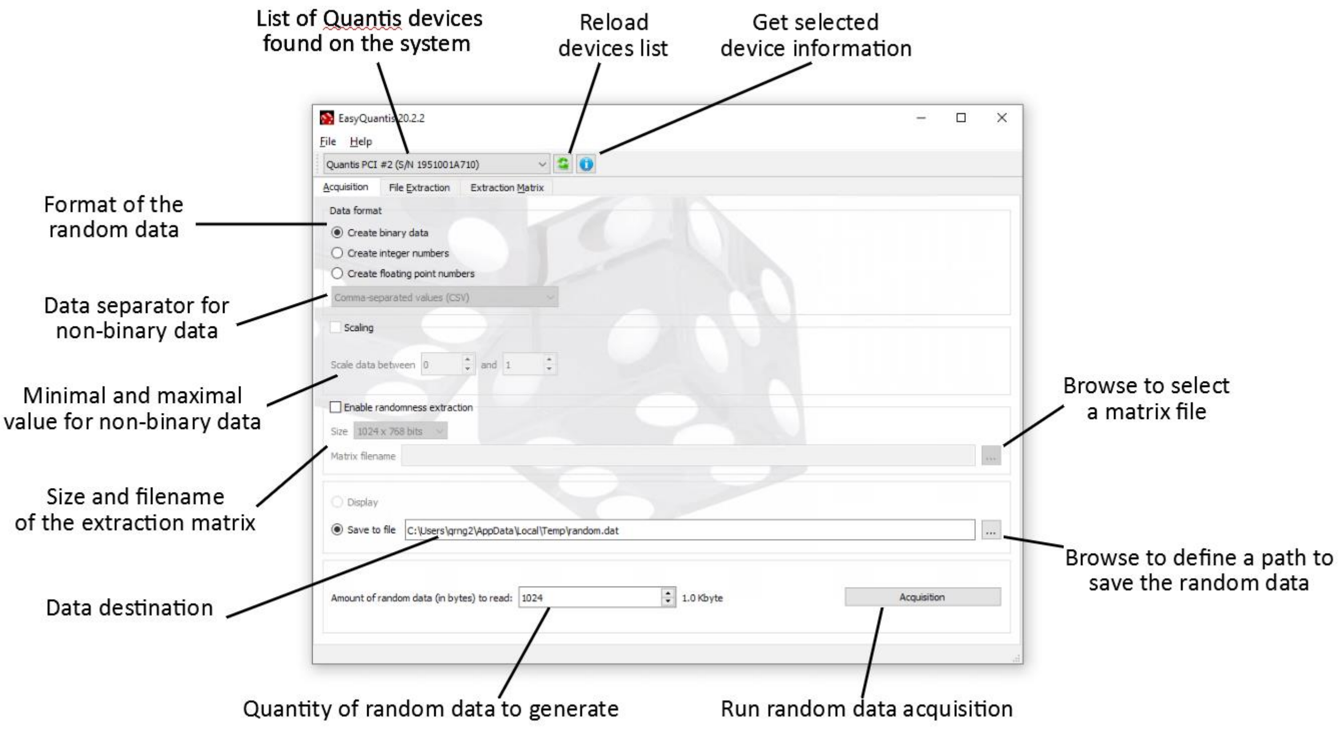
EasyQuantis is made of 3 tabs:

- **Acquisition**: Make acquisition of random data from a Quantis device.
- **File extraction**: Process randomness extraction form a file.
- **Extraction matrix**: Create your own matrix file.

Note: Extraction is out of the scope of this document because the new Quantis devices PCIe-40M and PCIe-240M have integrated hardware extraction.

4.2.1. Acquisition

The Acquisition tab allows to generate random numbers from a Quantis device.


List of Quantis devices found on the system
Reload devices list
Get selected device information
Format of the random data
Data separator for non-binary data
Minimal and maximal value for non-binary data
Size and filename of the extraction matrix
Data destination
Browse to select a matrix file
Browse to define a path to save the random data
Quantity of random data to generate
Run random data acquisition
EasyQuantis 20.2.2
File
Help
Acquisition
File Extraction
Extraction Matrix
Acquisition

Quantis PCI #2 (S/N 1951001A710)

Data format
- [✓ Create binary data]
- Create integer numbers
- Create floating point numbers

Comma-separated values (CSV)

Scaling

Scale data between 0 and 1

Enable randomness extraction

Size 1024 x 768 bits

Matrix filename

Display

[✓ Save to file] C:\Users\qrng2\AppData\Local\Temp\random.dat

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

To generate random data from a Quantis device using EasyQuantis:

1. Select a Quantis device from the list.
2. Select a data format:
   - **Binary data**. Data is read from the Quantis and returned as bytes.

- **Integer numbers**. Generates 32-bit random numbers ranging between -2'147'483'648 and 2'147'483'647 (inclusive).
- **Floating point numbers**. Generates numbers 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 subsequent 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 value is on a separate line:

  Value1

  Value2

  ...

  ValueN

Note: When generating binary data, you cannot select a data separator.

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

Note: For more details about the scaling algorithms, please refer to section Integral Values: The Scaling Algorithm ”Integral Values: The Scaling Algorithm”.

5. Optionally you can enable the randomness extraction post processing:

- **Size**. Select the size of the matrix
- **Matrix filename**. Select a matrix filename. The size of the file must be greater or equal to the matrix size. IDQuantique provide a default matrix located here: `Libs-Apps/QuantisExtensions/default_idq_matrix.dat`

6. 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.

7. **Save to file**. The data is written to a file. Use this option to generate large amounts of random data or to generate data for later use.

8. Select the amount of data to generate. File size is limited to 2 GBytes (2'147'483'647 bytes).

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

4.2.2. The EasyQuantis Command Line

EasyQuantis includes a command line parser, allowing you to use the application from the console in a script.

4.2.2.1. Generic Options

`-h [ --help ]`	Display a summary of available options.
`-v [ --version ]`	Return the version of EasyQuantis (available since EasyQuantis 2.1).

4.2.2.2. Quantis Options

`-l [ --list ]`	List all devices available on the system.
`-p [ --pci ] arg`	Select the given Quantis PCIe device as input device. arg is the number of the Quantis PCIe 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.

4.2.2.3. 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`	Sets 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. If specified, requires `--max` to be specified as well.
`--max arg`	Specify the maximal value for integers and floats. If specified, requires–`min` to be specified as well.

4.2.3. Usage Examples

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

4.2.3.1. List connected Quantis devices

```
$ EasyQuantis -l
```

List all USB and PCIe devices found on the system. Using this command allows to get the device number required for acquisition.

4.2.3.2. Generate Binary Data

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

Generates a file named random.dat containing 1GByte of binary random numbers using the Quantis PCIe device number 0.

4.2.3.3. Generate Numbers

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

Generates a file named `integers.dat` with 1000 integer numbers using Quantis-USB device number 0.

4.2.3.4. Generate Scaled Numbers

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

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

5. The Quantis Library

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

Note: We use the name "modules" for the QRNG chip IDQ20MC1.

5.1. Library location

You can find the library files (Quantis.so, Quantis.dll, Quantis.lib, ...) in the following paths.

For Windows users, or for Linux users after decompressing:

```
<path-to-quantis>\Packages\Windows\lib\<your system arch>\
```

under Linux:

```
<path-to-quantis>/Packages/QuantisRNG-<version>-Linux-<your system
arch>.tar.gz
```

If you recompile your libraries yourself as described in later chapters, you will find the Windows libraries under

```
<path-to-filename>\Libs-Apps\Quantis\<your system arch>\
```

and the Linux libraries under

```
<path-to-quantis>/Libs-Apps/build/Quantis
```

5.2.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 Express or PCI (legacy and new generation)
- QUANTIS_DEVICE_USB to specify a Quantis USB.

5.3.Basic Functions

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

5.3.1. QuantisCount

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

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

Parameters:

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

5.3.2. QuantisGetDriverVersion

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

Returns the version of the driver as a number composed of a major and a minor version number. The value before the point represents the major version number, while the value after the point represents the minor version number.
Returns a `QUANTIS_ERROR` code (cast to float) on failure.

Parameters:

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

5.3.3. QuantisGetLibVersion

```
float QuantisGetLibVersion();
```

Returns the version of the library as a number composed of a major and a minor version number.
The value before the point represents the major version number, while the value after the point represents the minor version number.

5.3.4. QuantisGetManufacturer

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

Returns a pointer to the manufacturer name string of the Quantis device. Currently only the USB version supports manufacturer name retrieval. On PCI Express, the string "Not available" is returned. The returned pointer points to a statically allocated string, which may not be modified by the enclosing application.
The string "*Not available*" is returned on failure as well.

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.

5.3.5. QuantisGetSerialNumber

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

Returns a pointer to the serial number string of the Quantis device. Currently only the USB version supports serial number retrieval. On PCI and PCI Express, the string "*S/N not available*" is returned. The returned pointer points to a statically allocated string, which may not be modified by the enclosing application.
The string "*S/N not available*" is returned on failure as well.

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.

5.3.6. QuantisRead

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

Reads random data from the Quantis device. This function performs an open, a read of the requested data and close the device.
Returns the number of bytes read from the device on success or `QUANTIS_ERROR` code on failure.

Notes: Use QuantisReadHandled() for best performance and / or thread safe operations.

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.
`buffer`	a pointer to the 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 data, the library will deliver unexpected results and may even cause a crash of the enclosing application.

5.3.6.1. Reading Large Amounts of Data

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

```
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;
}
```

5.3.6.2. Reading Basic Data Types

The function `QuantisRead` 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 designed for this purpose.

5.3.6.2.1. 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_SUCCESS` on success or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.
`value`	a pointer to the destination value.
`min`	the minimal value that the returned numbers can take
`max`	the maximal value that the returned numbers can take.

5.3.6.2.2. 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 Quantis.

To perform the scaling, the largest permitted multiple of the output range is selected. Random values equal or higher this limit is discarded. Below you will find a simplified version of the C code implementing `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 = 232;
/* 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 wastes data when Quantis generates random values equalling or exceeding 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 their 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.

5.3.6.2.3. 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). The scaled versions read a random floating point value between min (inclusive) and max (exclusive). Returns `QUANTIS_SUCCESS` on success or a `QUANTIS_ERROR` code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.
`value`	a pointer to the destination value.
`min`	the minimal value that the returned numbers can take
`max`	the maximal value that the returned numbers 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}and2^{64}respectively).

WARNING
In certain conditions, the distribution of numbers produced by the floating point scaling algorithm has a small bias that favours numbers at the lower end of the output range. If you need unbiased random numbers, please consider using `QuantisRead-ScaledShort` 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;
```

5.3.7. QuantisOpen

```
int QuantisOpen(QuantisDeviceType deviceType,
unsigned int deviceNumber,
QuantisDeviceHandle** deviceHandle);
```

Open the Quantis device for handled reads, e.g. QuantisReadHandled() function. Opened handles should be closed with a QuantisClose() call.

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

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.
`deviceHandle`	a pointer to a pointer to a handle the device.

5.3.8. QuantisClose

```
void QuantisClose(QuantisDeviceHandle* deviceHandle);
```

Close the Quantis device that was opened with a QuantisOpen() call.

Parameters:

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

5.3.9. QuantisReadHandled

```
int QuantisReadHandled(QuantisDeviceHandle* deviceHandle,
void* buffer,
size_t size);
```

Read data from Quantis. This function expects a Quantis device handle from the QuantisOpen() function, which must be called before.

Returns the number of bytes read on success or a QUANTIS_ERROR code on failure.

Parameters:

`deviceHandle`	a pointer to a handle the device.
`buffer`	a pointer to the 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`).

Note:

Increasing the request size minimizes system calls and therefore improve the performance in terms of acquisition speed. To reach the maximum speed 16MiB should be large enough, but it's highly dependent of the OS and CPU.

5.3.10.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 may not be modified by the enclosing application. Further calls to this function will

overwrite its content.

Parameters:

`errorNumber`	the number assigned to a particular type of error.

5.3.11.Advanced Functions

5.3.11.1. 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 the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.

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

5.3.11.2. QuantisGetBoardVersion

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

Returns the internal version of the board or QUANTIS_ERROR code on failure.

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.

5.3.11.3. QuantisGetModulesCount

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

Returns the number of modules/QRNG chips that have been detected on a Quantis device or a `QUANTIS_ERROR` code on failure.
For PCIe-40M, 2 chips are normally operational. For PCIe-240M, there are 12 chips.

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.

5.3.11.4. QuantisGetModulesMask

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

Returns a bitmask of the modules/QRNG chips that have been detected on a Quantis device or a `QUANTIS_ERROR` code on failure.
It is providing the position on the PCB of the chips installed.

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.

5.3.11.5. QuantisGetModulesStatus

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

Returns the status of the modules (QRNG chips) on the given device as a bitmask or a `QUANTIS_ERROR` code on failure.

Bit `n` is set in the bitmask if module/chip `n` is enabled and functional. For example, 13 (1101 in binary) is returned when modules/QRNG chips 0, 2 and 3 are functional.
A module (chip) is functional if it is operational and it has a positive health check, including LED, sensor, dark sum check, RNG proportion and repetition errors etc.. For more information on the health check, please consult the IDQ20MC1 chip documentation. If the health check of a chip on the PCIe card is not positive, its output is not considered and the throughput of the PCIe card is reduced.

Parameters:

`deviceType`	the type (PCI or USB) of the Quantis device.
`deviceNumber`	the number of the Quantis device. Note that device numbering starts at 0.

5.1.How to use the Quantis Library

5.1.1. Quantis Library Windows

Follow the following steps to use the Quantis library in Visual Studio 2019:

1. Open the solution in `<path-to-Quantis>\Libs-Apps\Quantis_Libs-Apps.sln`
2. Create a new project according to the type of application you want
3. Under your project structure right click on "References" and click on "Add Reference…"
4. Check "Quantis" and press OK
5. Include the Quantis library header files in your application. Example :
   ```
   #include "../Quantis/Quantis.h"
   #include "../Quantis/Quantis.hpp"
   ```
6. Right click on your project and set the following output directory:
   ```
   $(SolutionDir)$(Platform)\$(Configuration)\
   ```
7. Build the Quantis project
8. Build your project.

It will generate the application under: `<path-to-Quantis>\Libs-Apps\Win32`

5.1.2. Quantis Library Linux

Inside `<path-to-Quantis>\Sample\` you will find trivial makefiles to use the Quantis library.

5.2.Recompiling the Quantis Library

5.2.1. Windows compilation

Before compiling the Quantis libraries, you have to prepare your machine as described below. If you already have some of these components, you may of course skip the relevant section.

5.2.1.1. Visual Studio 2019

You can download and install Visual Studio 2019 via https://visualstudio.microsoft.com/vs/.

5.2.1.2. Installing the Boost C++ libraries

You need first to download and compile boost.

Download Boost from https://www.boost.org/users/history/version_1_72_0.html.

1. Extract the Zip in `C:\`
2. Open as Admin cmd.exe in `C:\`
3. Build Boost library for x86 with VS2019 (msvc-14.2) with the following script:

-

```
@echo off
rem Directory to boost root
set boost_dir=boost_1_72_0

rem Number of cores to use when building boost
set cores=%NUMBER_OF_PROCESSORS%

rem What toolset to use when building boost.

rem Visual Studio 2012 -> set msvcver=msvc-11.0
rem Visual Studio 2013 -> set msvcver=msvc-12.0
rem Visual Studio 2015 -> set msvcver=msvc-14.0
rem Visual Studio 2017 -> set msvcver=msvc-14.1
rem Visual Studio 2019 -> set msvcver=msvc-14.2

set msvcver=msvc-14.2

rem Start building boost
echo Building %boost_dir% with %cores% cores using toolset %msvcver%.

cd %boost_dir%
call bootstrap.bat

rem Static libraries
rem b2 -j%cores% toolset=%msvcver% address-model=64 architecture=x86 link=static
threading=multi runtime-link=shared --build-type=minimal stage --
stagedir=stage/x64
b2 -j%cores% toolset=%msvcver% address-model=32 architecture=x86 link=static
threading=multi runtime-link=shared --build-type=minimal stage --
stagedir=stage/win32
pause
```

In Visual Studio, EasyQuantis project settings, add the following paths:

- "C/C++" -> "Additional Include Directories": ensure the root boost path is correct (example "C:\boost_1_72_0")
- "Linker" -> "Additional Library Directories": ensure the lib boost path is correct (example "C:\boost_1_72_0\stage\x64\lib")

5.2.1.3. Installing Java

Download Java SDK from https://adoptopenjdk.net/.

In Visual Studio, for the project EasyQuantis and QuantisExtensions, ensure that you have these tow paths in "C/C++" -> "Additional Include Directories":

- Include path: for example, in "C:\Program Files\AdoptOpenJDK\jdk-8.0.242.08-hotspot\include"
- Include win32 path: for example, in "C:\Program Files\AdoptOpenJDK\jdk-8.0.242.08-hotspot\include\win32"

5.2.1.4. Installing QT 5.14

Qt can be installed from http://download.qt.io/official_releases/qt/5.14/5.14.1/qt-opensource-windows-x86-5.14.1.exe.

In installation, you need to check the following packages: msvc2017 and msvc2017_64.

Quantis has been tested with QT 5.14.1, ideally you should install this version. If your version is too new or too old, and you find that you have problems with compiling or executing Quantis samples or applications, you might have to update it or to roll back as appropriate.

In VisualStudio 2019's menu click on: "Extensions" → "Manage Extensions" → Search for "Qt Visual Studio Tools", install Qt Visual Studio Tools.

Click on “Extensions” → “Qt VS Tools” -> “Qt Options” and ensure the path to the Qt installation is correct (example "C:\Qt\Qt5.14.1\5.14.1\msvc2017").

5.2.1.5. Installing OpenSSL

Download OpenSSL from https://slproweb.com/products/Win32OpenSSL.html.

In Visual Studio, for the project EasyQuantis, ensure the path is correct (example "C:\Program Files (x86)\OpenSSL-Win32\include") in "C/C++" -> "Additional Include Directories" and also ensure the path is correct (example "C:\Program Files (x86)\OpenSSL-Win32\lib\VC\static;") in "Linker" -> "Additional Library Directories".

5.2.1.6. Compilation

Start Visual Studio and open the Quantis library solution file located in `<path-to-Quantis>\Libs-Apps\Quantis-Libs-Apps.sln`

Select “Release” and build EasyQuantis project (it will build all dependencies).

5.2.1.7. Change acquisition mode in Windows

In Windows the acquisition mode is chosen by a define in the library code.

For Linux see Linux driver installation section.

By default, the RNG mode is set. To change it to SAMPLE mode, uncomment the line `//#define SAMPLE_MODE` in file `<path-to-Quantis>\Libs-Apps\Quantis\QuantisPcie_Windows.c`.

Mode name	Acquisition speed (PCIe-40M)	Acquisition speed (PCIe-240M)	Mode description
SAMPLE	38.3 Mbits/s	232 Mbits/s	Entropy data (also called “Quantum entropy”)
RNG	9.6 Mbits/s	58 Mbits/s	Cryptographic secure random data. Quantum entropy data has been post-processed using NIST 00-90 A/B/C compliant DRBG.

See the IDQ20MC1 datasheet for more details.

5.2.2. Linux

Modern versions of Linux are not able to build the Quantis Library due to old dependencies. You need to use an Ubuntu 14.04 Docker container as build environment. Generated binaries will be compatible with modern Linux systems.

5.2.2.1. Install Docker and build the image

First on update the package manager information:

On Ubuntu run: `sudo apt-get update`

On Cent OS run: `sudo yum check-update`

Run the following commands to install and start Docker:

```
curl -fsSL https://get.docker.com/ | sh
sudo systemctl start docker
sudo usermod -aG docker $(whoami)
```

Create a file named `Dockerfile` with the following contents:

```
FROM ubuntu:trusty-20190515
RUN apt-get update \
    && apt-get install -y \
    gcc \
    git \
    g++ \
    linux-headers-generic \
    make \
    cmake \
    gcc-4.7 \
    g++-4.7 \
    default-jdk \
    libusb-1.0-0-dev \
    libssl-dev \
    libboost-all-dev \
    libqt4-dev \
    linux-headers-generic \
    make \
    pkg-config \
    && apt-get clean autoclean \
    && apt-get autoremove --yes \
    && rm -rf /var/lib/{apt,dpkg,cache,log}/
RUN  \
    mv /usr/bin/g++ /usr/bin/g++_bkp && \
    mv /usr/bin/gcc /usr/bin/gcc_bkp && \
    ln -s /usr/bin/g++-4.7 /usr/bin/g++ && \
    ln -s /usr/bin/gcc-4.7 /usr/bin/gcc
```

Then open a terminal in the same folder where you created the Dokerfile and build the container with:

```
docker build -t quantis-lib-compiler .
```

5.2.2.2. Compilation

Run the Docker container:

```
docker run -v <absolute-path-to-Quantis>:/root/quantis -it quantis-lib-compiler
```

Run the following commands to build the lib and EasyQuantis:

```
mkdir /root/quantis/Libs-Apps/build
cd /root/quantis/Libs-Apps/build
```

```
cmake ..
make
```

You can exit the docker container with `exit`.

Back to the host computer, the following binaries were generated:

- Quantis library:
- `<absolute-path-to-Quantis>/Libs-Apps/build/Quantis/libQuantis.so*`
-
- EasyQuantis application:
- `<absolute-path-to-Quantis>/Libs-Apps/build/ EasyQuantis/EasyQuantis`

You can delete the Docker image if you don't need it anymore:

```
docker rmi -f quantis-lib-compiler
```

6. Quantis Library Wrappers

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

Currently wrappers for the following languages are available:

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

The 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`.
- The names of public functions in the Wrappers are the same as those in the Quantis C library but without the prefix Quantis, for instance QuantisCount is named Count in the wrapper. Functions other than the constructors do not require providing values for deviceType and deviceNumber since these are defined globally within the class. The only exception to this rule is static functions which have the same definition in both the C- and the Wrapper code.

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

Furthermore, Quantis can be accessed via the C++11 "random_device" interface which standardises true random number generator access. Usage of this interface is described below.

Note: for the C++ Wrapper Since the Quantis device is kept open until the Quantis class is destroyed, it is highly recommended to reduce the scope of the Quantis variable as much as possible. In particular it is discouraged to make the Quantis variable global.

6.1.The C++11 random_device interface

6.1.1. About the interface and our implementation

The standard C++11 "random_device" interface allows to access true random number generation in a standardised manner. It is derived from the boost "random_device" class, so if your application uses either the C++11 or the boost version of the random_device class, you can switch to Quantis very easily by including the Quantis implementation - i.e. the file "Quantis_random_device.h" – in your code and commenting your previous include, and making a very small number of changes described in the following.

At the time of releasing this interface, C++11 is a very new standard. Many compilers have already implemented parts of it, but which parts are supported varies widely. In order to avoid compile time issues, the C++11-specific keywords have been commented, and you should uncomment those that your compiler supports in order to be as compatible as possible.

Note that our implementation is a little different from the standard one in that the standard interface accesses a device mounted on the file system, while we must create a Quantis C++ object and access it to get to Quantis. For this reason, we have included a destructor in our interface that you should call when you no longer need Quantis.
A sample executable in your code shows how to use this interface.

6.1.2. Library Compilation for C++11

It is important to not to compile EasyQuantis when compiling with C++11. If you want to compile both do two separate compilations.

If you want to test the interface without C++11 support, i.e. with the keywords suppressed, simply compile it as usual, with no options given to cmake. The sample will work either way.

IMPORTANT

For this feature to work, your compiler must support the C++11standard. Many compilers support parts of it but not all keywords needed in the "random_device" interface. It may thus happen that even though your compiler has a C++0x/C++11 option, the full feature version will still not be run. In this case, you can still use the "random_device" interface, but the C++11-specific keywords are automatically suppressed to pass compilation in C++ 98. In most applications using "random_device", it should still be easy to replace your old implementation with Quantis even if you don't have C++11 supported.

The full version of this feature is only supported on Linux and MacOSX systems. On the other supported OSs, the interface of the class will be available along with the Sample code for C++11, but the C++11 specific keywords will be automatically suppressed. Again, this means that you can use the class, but it is not compatible with the C++11 standard in the strict sense. In most applications using "random_device", it should still be easy to replace your old implementation with Quantis even if you don't have C++11 supported.

If you use cmake and the gnu C/C++ compiler version 4.6 or newer, it is possible that the features will work for you even on other systems than Linux and MacOSx.

6.1.2.1. Windows

The Windows Visual Studio 2008/2010 compilers do not support all the necessary C++11 features, so if you want to use C++11 under Windows you must revert to a compilation as under Linux using CMake and GCC 4.6 (see below).

6.1.2.2. Linux

The distribution available by default uses C++98, so you need to recompile the library to use C++11. You will need to have installed CMake and GCC 4.6 or higher. Proceed as follows:

```
cd <your-path-to-Quantis>/Libs-Apps/
mkdir build
cd build
cmake .. -DUSE_CXX11=1 -DDISABLE_EASYQUANTIS=1
make
```

6.1.3. C++11 Sample compilation

For how to compile the C++11 Sample, see the relevant section in the chapter "Sample Code" below.

7. Sample Code

You will find sample code on how to use each library wrapper in the subdirectory `<path-to-Quantis>/ Samples/`.

7.1.Java Sample

Install Java and Apache Ant - All OS

To use the Java wrapper, install the Java Standard Edition (JSE) JDK matching your OS. For instance, the Java SE 64 bits JDK should be installed with a Windows 64 bits. You will find the JSE JDK in all available versions and a detailed installation instruction on the website of OpenJDK or on the webpage of Oracle (at the time of writing this guide,
http://openjdk.java.net/ and http://www.oracle.com/technetwork/java/javase/overview/index.html respectively). In addition to Java, Apache Ant should be installed. The Apache Ant website contains a user manual that explains how to install it on your system (at the time of writing, http://ant.apache. org).
From here, go to the section describing your OS.

Linux

In order to be able to work with Java and Ant, you need to add their binaries to the PATH system variable, if they aren't there yet. Additionally, Java needs you to set the JAVA_HOME variable. You can set these variables as follows:

Open the file ~/.bashrc (on some systems this may be called ~/.bash_profile). This is a script that will run automatically and set the variables in it for you, so you won't need to set them by hand every time you open a new terminal.

In the file, add the following entry on a separate line:

```
export PATH=$PATH:<path-to-java-bin-directory>:<path-to-ant-bin-directory>
```

To find out where your Java bin directory is, type in your terminal

```
which java
```

which will result in something like `/usr/bin/java`. Your <path-to-java-bin-directory> is therefore `/usr/bin`. The same procedure applies to finding your ant directory. To apply the changes, execute the command

```
source ~/.bashrc
```

To compile the Java samples, the easiest solution is to

- Change to the directory where the Java samples are located:

```
cd <path-to-quantis>/Samples/Java
```

- In order for Java to be able to use the Quantis library, execute the following command which will add the Quantis library to the Java library path:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<directory-containing-libQuantis.so>
```

The location of the Quantis library is explained in section "Library location" in the chapter "The Quantis Library".

You may also add this line to the ~/.bashrc file as you did before when adding Java and Ant to the PATH variable, if you want it to be loaded automatically when you open a new shell.

- Type the command `ant`. The Java archive created is `dist/Quantis.jar`.
- To execute the program, pass the command `java -jar dist/Quantis.jar`. Alternatively, type `ant run`.

Windows

In order to be able to work with Java and Ant, you need to add their binaries to the `PATH` system variable, if they aren't there yet. Additionally, you need to add the directory containing the Quantis libraries to `PATH` as well.

You can set it in Control Panel -> System Properties -> Tab 'Advanced' -> Environment Variables, note that usually you have to be Administrator user to do this. Choose or create the variable PATH (upper case important) under 'System Variables', and add `<path-to-ant>\bin;<pathto-java>\bin;<path-to-Quantis-libs`, making sure that all entries are separated by a semicolon. To find your Quantis libraries, refer to chapter "The Quantis Library", section "Library Location". E.g., if you have decompressed the Ant folder in the location `C:\Program Files\Ant\apacheant-x.y.z`, your Java JDK is under `C:\Program Files\Java\jdk-xxx` and your Quantis libraries are under `<path-to-Quantis\ Packages\Windows\lib\Win32`, you should add the following line:

```
C:\Program Files\Ant\apache-ant-x.y.z;C:\Program Files\Java\jdkxxx;
<path-to-Quantis\Packages\Windows\lib\Win32
```

IMPORTANT

On some 64b Windows systems, you may have to recompile the Quantis libraries before executing the Java sample, in order to avoid JNI related errors. For how to do this, refer to chapter "TheQuantisLibrary", section "RecompilingtheQuantisLibrary". When you have done that, you will need to update the entry in the PATH variable pointing to the Quantislibraries. The section "LibraryLocation" tells you where you can find your recompiled Quantis libraries.

Now you can execute the Java Sample:

```
cd <path-to-Quantis>\Samples\Java
ant
ant run
```

Instead of **ant run** you can also execute **java -jar dist\Quantis.jar**.

8. Appendix: Troubleshooting

8.1. EasyQuantis

***Q:* EasyQuantis crashes on Linux, with one of the following errors:**

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

A: Such errors are generally caused by an incompatible Qt library binary. The binary of EasyQuantis provided by ID Quantique has been linked against Qt version 4.3.4. To solve this issue, you need to install Qt4 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 mode, which is not affected by this issue. Please refer to Section 4.5, “The EasyQuantis Command Line”.
This issue can also be solved by recompiling the Quantis library and EasyQuantis on your system.

8.2. Quantis Samples

***Q1:* When I try to run the C++ sample code after a successful build, it crashes and gives me an Access Violation error. What should I do?**

A1: If the C++ sample crashes upon build and you get an Access Violation error, restart VS2010 in administrator mode since there is some memory that cannot be accessed by a normal user. To do that, right-click on the VS2010 icon and choose "Run in administrator mode". Then reopen your solution/project and proceed as normal.

***Q2:* I am on a Windows 64 bits machine. When I execute the Java sample, I get JNI-related errors. What should I do?**

A2: This error occurs on some Windows 64 bits systems, and can be solved by recompiling the Quantis libraries on your own machine. Instructions for recompiling the library can be found in chapter "The Quantis Library" in section "Recompiling the Quantis Library".

9. Appendix: FAQ

9.1. Quantis Library

Q1: Can I use the 32-bit Quantis library on a 64-bit system?

A1: Yes, you can use the 32-bit Quantis library within a 32-bit application on 64-bit systems. Note however, that you can neither use the 32-bit Quantis library within a 64-bit application nor the 64- bit Quantis library within a 32-bit application.

Q2: On Microsoft Windows, is it necessary to copy the Quantis.dll library to the system directory (C:\Windows\System32)?

A2: No, this is not mandatory. IDQ recommends installing the Quantis.dll library in the directory in which your application resides.

Q3: On Microsoft Windows, when I use Quantis.dll within my application I get 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?

A3: 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.

Q4: I have changed the name of the Java package Java for Quantis from com.idquantique.quantis to something else/I have moved the Java code to another directory. Now I can load the dynamic library Quantis.dll but I get the error message
Exception in thread "main" java.lang.UnsatisfiedLinkError: random.utils.Quantis.QuantisCount(I)I.

A4: The Java classes use mainly native functions (by using the JNI interface). A native function must obey to strict rules, and if you don't abide by them your application may not execute or compile properly.
For instance, the name of the package your code resides in is used to define the name of functions in that package. So, if you change the package name, say to *random.quantis*, the function names will change in the Java code, but not match the native code function names anymore and thus produce errors. The best solution is to keep the original name *com.idquantique.quantis* for the package.
If you absolutely need the Java code in another package, change the two files `Libs-Apps/Quantis/Quantis_Java.h` and `Libs-Apps/Quantis/Quantis_Java.cpp` to reflect the new package name in function names. After the modification, the dynamic Quantis library must be recompiled and reinstalled.

9.2. EasyQuantis

Q1: ***When I launch EasyQuantis on Microsoft Windows, a console appears for a few seconds and disappears when the GUI window comes up. Why does this happen?***

A1: EasyQuantis integrates a command line interface and a graphical interface. However, on Microsoft Windows it is not possible to build a hybrid Windows/Console application. EasyQuantis has been built as a Console application. When launched, the system automatically creates a console window. If no argument has been provided to the application (giving arguments would invoke the console version), the console window is hidden, and the graphical interface is displayed. Avoiding this issue is very difficult.