Quarch Technology Ltd

AN-015

Application Note

Automating with QPS

For use with:

**XLC Power Modules  
HD Power Modules**

**Quarch Power Studio (QPS)**



# Change History

|  |  |  |
| --- | --- | --- |
| -01 | June 2018 | Initial Release |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[Change History 2](#_Toc475542640)

[Introduction 4](#_Toc475542641)

[Modules Supported 6](#_Toc475542642)

[Installation and setup 7](#_Toc475542643)

[Linux setup 7](#_Toc475542644)

[Windows setup 10](#_Toc475542645)

[Provided Examples 12](#_Toc475542646)

[Locating devices 12](#_Toc475542647)

[Streaming capture example 13](#_Toc475542648)

[QIS built in terminal 14](#_Toc475542649)

[QIS remote terminal control 18](#_Toc475542650)

[Customer support from Quarch 20](#_Toc475542651)

[Access support from the Quarch website 20](#_Toc475542652)

[Find discussion topics, support information and testing ideas. 20](#_Toc475542653)

# Introduction

Quarch Power Studio (QPS) is a unique system for recording and analyzing power consumption of storage devices.

This application note demonstrates the ability to automate control of QPS, such that fully automated testing and recording is possible.

Now you can run complex overnight tests and have recorded, annotated data sets ready for analysis in the morning.

* Simple to install and run (Java JRE and Python required for this example)
* Virtually unlimited recording times
* Annotate recordings to assist with later analysis
* Add arbitrary data (drive temperature, IOPS and similar)

QPS runs on any Windows (And soon Linux) host PC. A USB cable or network connection to the Quarch power module allows QPS to control it.

The automated scripts that control QPS can now be run on the Test PC, or on any other PC on the network. While this application note demonstrates the use of Python, any language can be used to connect to QPS over TCP and issue commands.

QPS

Customer Drive

USB/LAN Connection

Power Module

Customer Test PC

# Modules Supported

XLC power modules

* QTL1824 (QTL1824-02A modules do NOT support streaming, but can be upgraded by Quarch)
* QTL1847

HD power modules

* QTL1995
* QTL1999

## Application Note Example Files

The **AN-015.zip** should be extracted to your preferred location.

**XXXXXX.py**[Explain purpose of file here]

# Installation and setup

## Python install

If you do not already have Python installed, download and install it from:

<https://www.python.org/downloads/>

Under Windows it is helpful to make sure the Python installation directory and PythonXX\Scripts are included in the PATH environment variable. See:

<https://docs.python.org/2/using/windows.html#excursus-setting-environment-variables>

## QuarchPy library install

The Quarch Python package can be installed from the Python web repository (assuming you have internet access) or via the download from our website.

### Web Install

From the command line:

**>pip install quarchpy**

If this fails, your path to “pip” may not be set, you can instead run:

**>python –m pip install quarchpy**

### Local Install

If you want to install from a downloaded folder, ensure the folder is unzipped to a local disk, navigate to the folder containing the setup.py file and run (noting the ‘.’ on the end):

**>pip install quarchpy .**

If this fails, your path to ‘pip’ may not be set, you can instead run:

**>python –m pip install quarchpy .**

### Upgrade

If you already have QuarchPy installed, you will get a failure message. If you want to upgrade to a new version, you need to add the ‘--upgrade’ command:

**>pip install --upgrade quarchpy**

The --upgrade command can similarly be used in any of the other examples, to load from a local install folder.

## Java install

Check that the Java JRE is installed

You can find install instructions and files here:  
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

## QPS install

Current versions of QPS are provided as a portable (non-install) .jar file

The latest version can be downloaded from here:  
<https://quarch.com/products/quarch-power-studio>

## Linux USB Permissions

Linux systems require administrative rights to run python scripts for modules connected via USB. You can do that by running your script as root (sudo command) or changing the default USB permissions. This is done by creating a text file called **Quarch-permissions-usb.rules**

For ubuntu systems, you need to enter into that file:

SUBSYSTEM == “usb”, ATTRS{idVendor}==”16d0”, MODE=”0666”

SUBSYSTEM == “usb\_device”, ATTRS{idVendor}==”16d0”, MODE=”0666”

For Centos systems, you need:

SUBSYSTEM == “usb”, ATTRS{idVendor}==”16d0”, GROUP=”users”, MODE=”0666”

SUBSYSTEM == “usb\_device”, ATTRS{idVendor}==”16d0”, GROUP=”users”, MODE=”0666”

This file needs to be placed in /etc/udev/rules.d

Finally, the system either needs to be restarted or run the command:

**>sudo udevadm control -reload**

Then reconnect the USB device.

## Test the installation

Run the list devices script, to check which modules are available to you

>python QisListDevices.py

Below we can see that three devices were found, one local XLC module on USB and two remote devices on LAN.



If QIS did not run correctly, or is not accessible you instead will see:



# Provided Examples

## Locating devices

The file QisListDevices.py contains a simple example, querying QIS to get a list of available devices. This lets us see what is available, and conform that everything is working.

The name format will be used later for choosing the module to use. For example:

usb::qtl1847-01-016

This represents a USB connection to a QTL1847 (XLC Power Module), with a serial number of QTL1847-01-016

>python **QisListDevices**.py



## Streaming capture example

The file QisStreamExample.py connects to a selected module and runs a series of commands. These are split up into a few separate examples that are designed to do slightly different things.

The “powerMarginingExample” function demonstrates how the voltage being produced by the power module can be altered and measured without any streaming being involved. As no streaming is being performed, it requires much less CPU usage with the downside of simplistic power measurement.

The “simpleStream” function demonstrates how data can be streamed from the module. ‘Streaming’ involves the PC client software constantly puling data back from the Quarch module. If the PC client did not pull data from the module, its internal buffer would fill up and streaming would halt.

The module samples the voltage and current every 4µs and has the capability of averaging those samples before sending the data to the PC. An XLC, on a fast PC, can generally keep up with 16 samples being averaged in the device, whereas an HD can handle 4 samples. If any lower device averaging rates are selected, the device cannot transmit data fast enough and so its RAM will fill up. “simpleStream” uses the device default for averaging, which is no averaging or zero averaging. To change the device averaging rate, the command:

qis.sendCmd(module, "Record Averaging x")

Where x is any whole 2n from 0 to 32k.

“multiStreamExample” demonstrates a simple method for handling buffer overrun. This occurs when the device averaging bandwidth is greater than the rate data can be saved to file. When any overrun is detected, in either the module or QIS, the stream is stopped and restarted, saving the data into a new file.

|  |
| --- |
| **Note**: It can take time to empty either the QIS or device buffers so there will be sections of time where no data is present so if continuous data is required, use a high enough device sample rate such that overrun does not occur. |

The “averageStream” function demonstrates the script averaging feature built into QisInterface.py. This feature takes samples from the device and further averages them to create a user defined time base. In the example, data points will be saved to file that represent a 1s worth of data. All that needs to be set by the user is the final required averaging rate, and the averaging rate performed by the device.

|  |
| --- |
| **Note:** The implementation used has an absolute timing error for each data point equal to:  For example, having each data point represent 1s with a device averaging rate of 1k has a timing error of ±0.4096. This timing error does not translate to a cumulative timing error so 100 data points will represent 100s of time, in this case. |

Finally, the multiDeviceStreamExample function demonstrates how multiple modules can be controlled with a single script.

|  |
| --- |
| **Note:** As more devices stream to the same PC, the maximum achievable data rate is reduced. This means, buffer overrun will occur at higher device averaging rates. |

The example function demonstrates how all 6 ports of a QTL1999 6 port HD module can be communicated with and stream from each port simultaneously.

These examples all use a number of functions built into this script or QisInterface.py. These functions are:

debugPrint(*string, setting*)

Prints string to console or file dependant on debugPrintSetup. When *setting* = 1 the string can be saved to file but will not print to console.

debugPrintSetup(*setting, filename)*

Setting can be either “Command Line” or “File”. Filename is required when saving to file.

qis.sendCmd(*module, command*)

Sends commands to the module via QIS. Module is the net bios or IP address of the module being communicated with. Command is the string being sent to the module. It returns text from the device dependent on the command.

qis.startStream(*module, filename, fileMaxMB, streamName, streamAverage*)

Starts device streaming. *module* is the module being commanded. *fileMaxMB* is the max file size for streamed data files. *streamName* is saved into the header for the stream data file. *streamAverage* is the script average performed in seconds. Can be set to None if not required.

qis.streamRunningStatus(*module*)

Returns the running status of the stream from the module

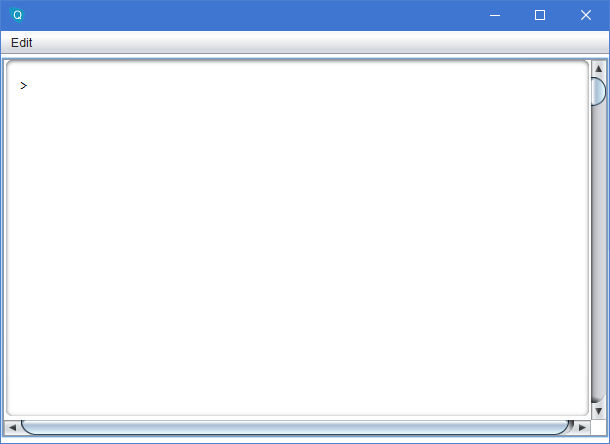
# Debugging

There are a number of built in options for debugging QIS and Python scripts, allowing you to track down problems

* The simple script to ‘locate devices’ (described above) checks that QIS is running and accessible, without needing a device attached
* QIS has a built in terminal which can be used to check access to devices, without needing to run a script
* A remote terminal (putty for example) can be used to check access to a QIS server, allowing you to check it is accessible needing access to Python

## QIS built in terminal

QIS has a built in Terminal. When running, there will be an icon in the task bar. Right click this and select ‘Show Terminal’. If the icon does not work on your linux distro, run QIS with the –terminal option

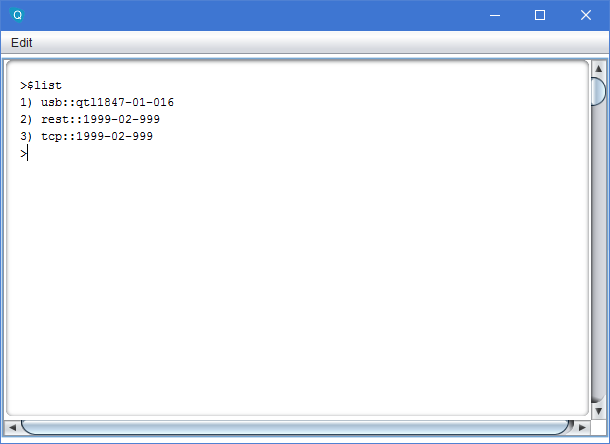


To get a list of commands enter:

>$help

To get a list of available modules enter:

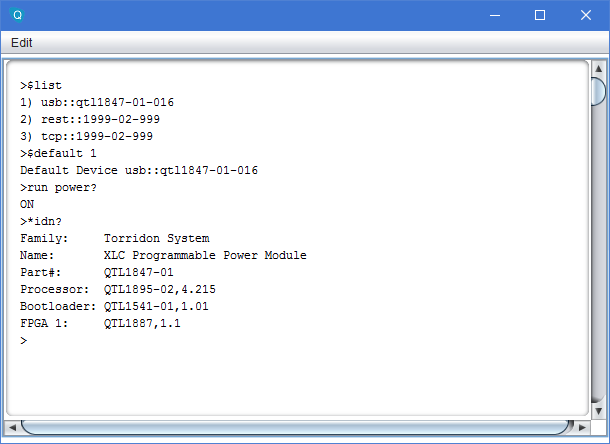
>$list



To select a device to control (We want the USB device in this case), use the command:

>$default 1

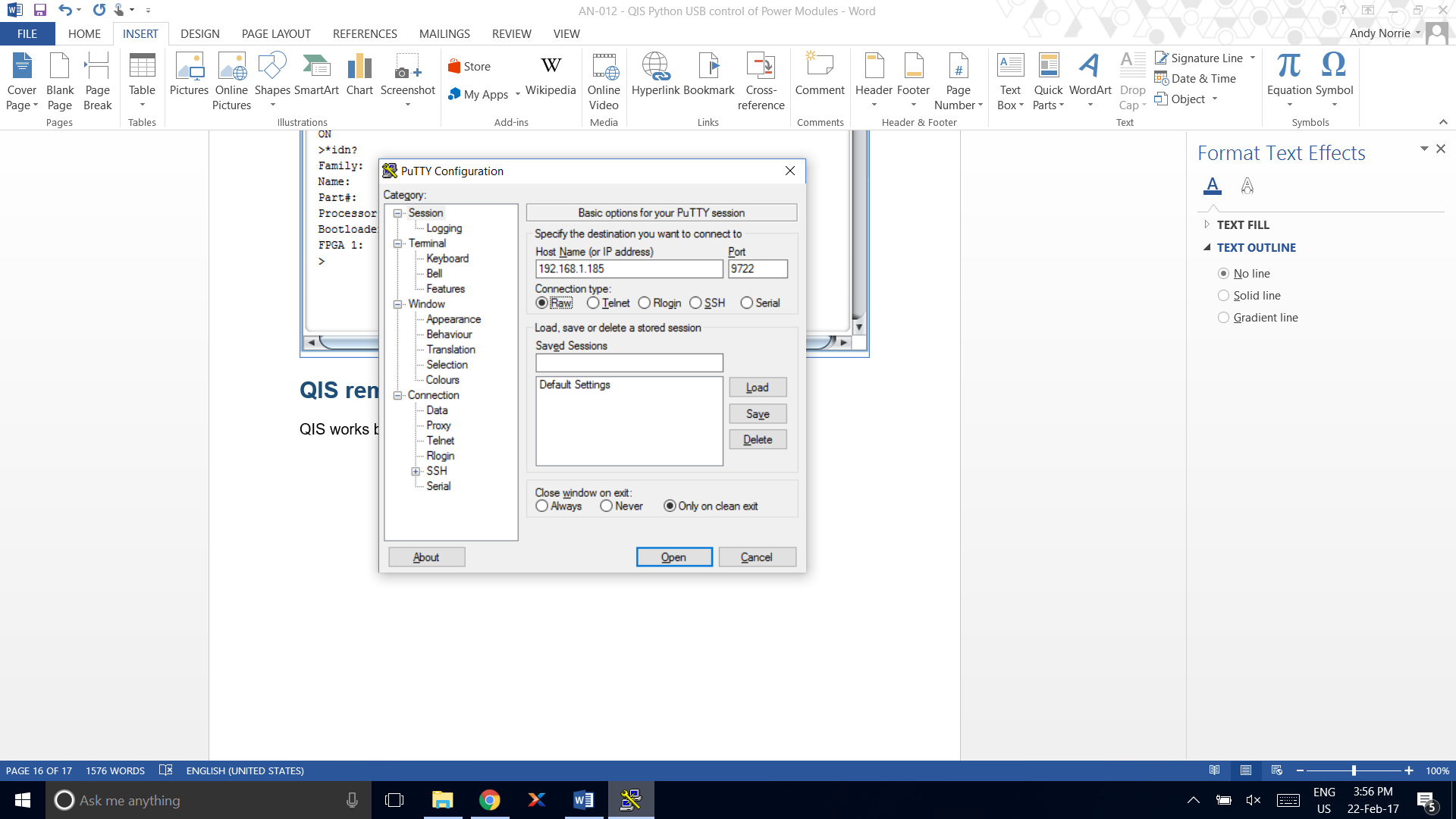
You can now enter any standard command for the chosen device (see the device technical manual for command lists).

****

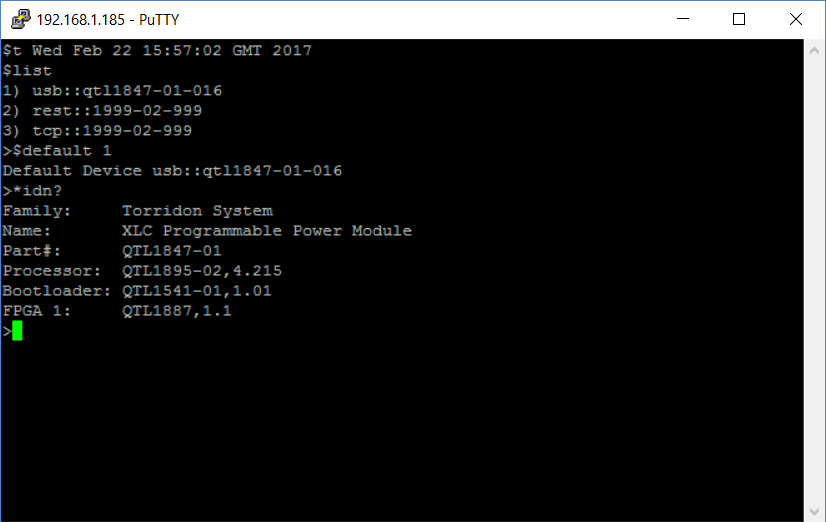
## QIS remote terminal control

QIS works by exposing a TCP port on the PC it runs on. You can connect to this via a standard terminal program such as Putty. Below you can see the setting used.

The IP address is the IP of the PC running QIS  
The port is 9722  
The connection type must be ‘Raw’



When connected, you can use the same commands as were described in the ‘QIS built in terminal’ section.



## Know Issues

* CentOS 7 with Gnome:

Running QIS will show an icon for the server application, but it cannot be right-clicked to get to the menu.

To use the terminal, it must be invoked with:   
java –jar qis.jar –terminal

When the terminal is running, the icon on the task bar is over sized, but is functional. If the terminal is closed, you will not be able to open it again.

As the taskbar icon does not work, you may have to terminate the QIS process manually.

# Customer support from Quarch

There are multiple ways to access the support you need. You can contact us directly or access an extensive range of valuable support materials from <http://quarch.com/support>.

* Contact us direct
* Get going quickly and easily, with help direct from the engineers:
* Call +44 1343 508 140 or email [support@quarch.com](mailto:support@quarch.com) during UK office hours.
* Our international partners are well trained in the use of our products and can deal with many basic technical queries from within your time zone, if you prefer. Check <http://quarch.com/resellers> for the contact details of your regional supplier.

## Access support from the Quarch website

You can download up-to-date software and drivers, technical manuals, datasheets and more from our website. To help you get started quickly we provide additional documents, such as examples in Perl, Python and C# and Telnet and Serial instructions.

* Key **places to visit on** the Quarch website
* Register your Quarch product to confirm your international warranty: <http://quarch.com/product-registration>
* Download a wide range of documentation, free applications and drivers to help you make the best possible use of your Quarch tools: <http://quarch.com/content/downloads>
* Access the Quarch support forum (<http://quarch.com/forum>):

## Find discussion topics, support information and testing ideas.

Browse existing topics or login to your user account to ask for information and advice.

Sign up for Quarch Technical Updates to get the most out of your Quarch products. Updates are published approximately once a quarter and include news about the latest features, tools, application notes and software updates. See <http://quarch.com/content/sign-quarch-technical-updates>.