Quarch Technology Ltd
Quarch Instrument Server
Technical Manual

Change History

1.0		Initial Release
1.1		Switched QisInterface references to Quarchpy

Contents

Introduction

The Quarch Instrument Server, or QIS for short, is a Java based server app which allows for simple automation of our range of Power Modules.

QIS runs in the background on any PC and can connect to USB or remote LAN power modules. It handles module location and connection. No additional drivers are needed.

QIS exposes a TCP port, allowing local or remote scripts to access the full abilities of the Quarch power modules.

QIS allows you to send commands, get responses and to stream large blocks of power data from a module. Optimization and buffering are handled within QIS, allowing you to keep your automation scripts simple.

As QIS can run on a remote machine, you can offload the processing/storage requirements of power testing from your main test PC

Parts of QIS

1. QIS.jar *[And associated resources/libraries]*

   This is the core application and when run, activates the server process

2. Quarchpy

   This is our Python package, providing a simple API into all Quarch products, including those controlled via QIS. This is used by AN-012 (below)

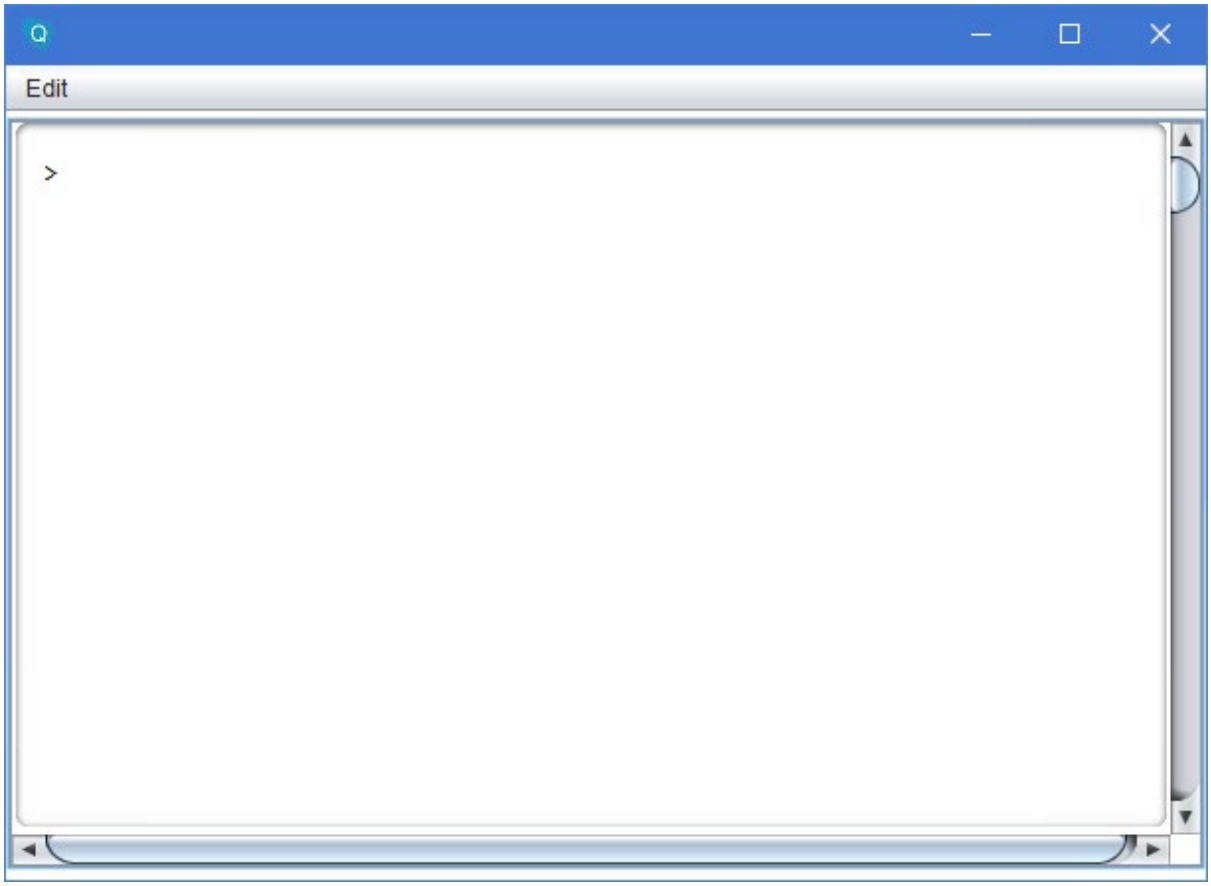
3. AN-012 [http://quarch.com/file/AN012-QIS-python-control-power-modules]

   This application note is not actually a part of QIS, but has detailed information on installation and common issues. It also has additional example scripts and is an excellent place to start

Basic setup - Terminal

QIS does not need installed, it can simple be copied to the host PC and then executed.

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



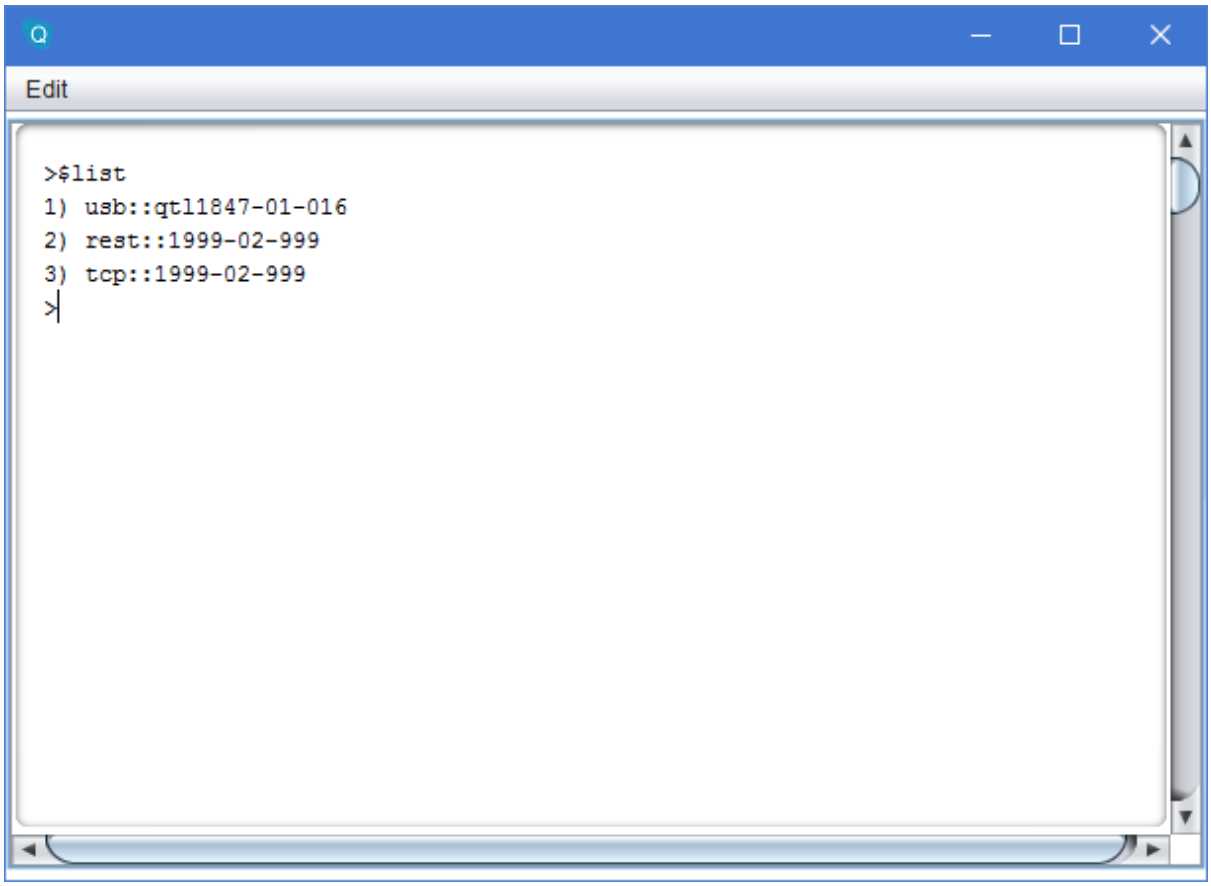
Enter the following command to scan for all local and remote devices:

```
>$scan
```

Now request a list of those devices:

```
>$list
```

If everything is working, you should see a list of the modules available.



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

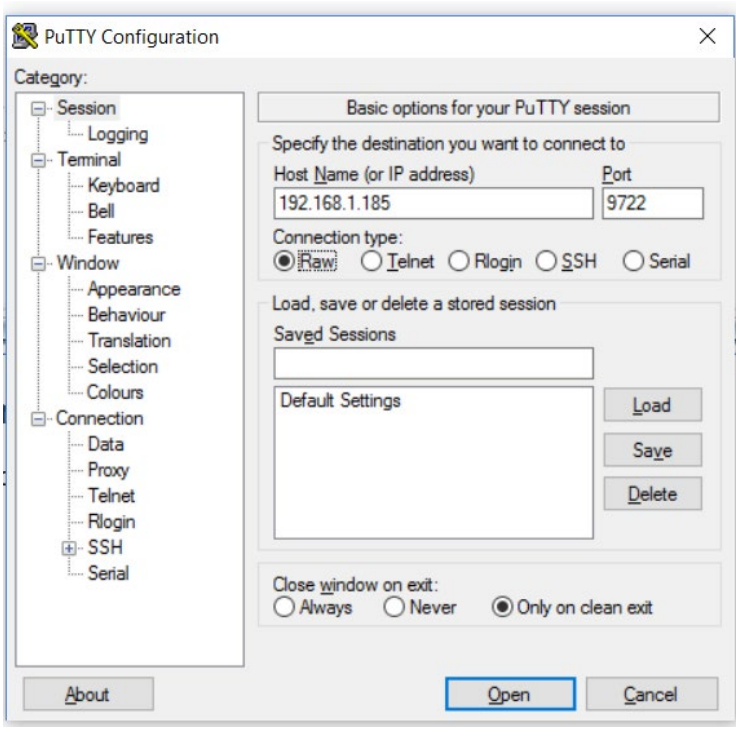
```
>$default 1
```

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

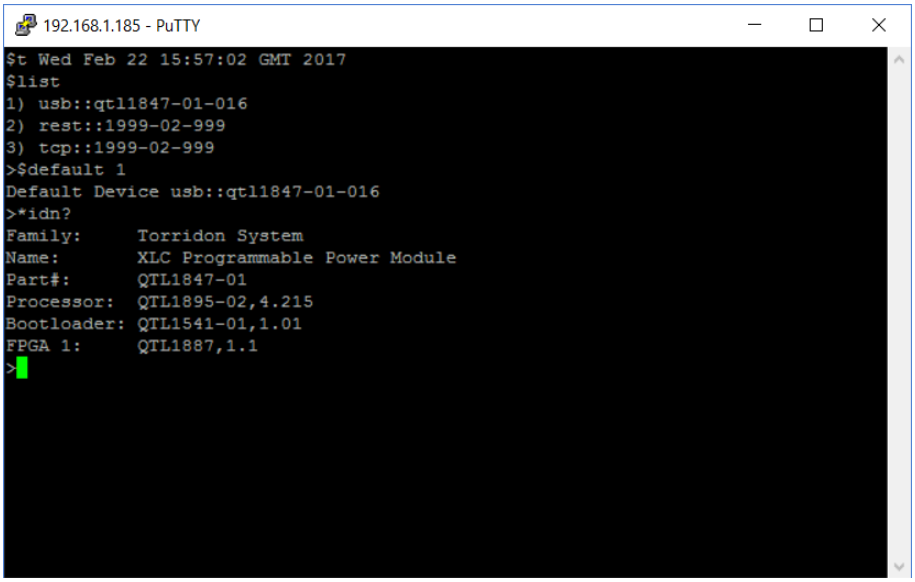
Basic Setup – Remote terminal

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.



Command Listing

Commands line parameters

`-terminal`

Displays the terminal window on application start

`-Djava.awt.headless=true`

This is a Java environment setting where Java is configured to run without GUI components and QIS will run in headless mode, no system tray icon or terminal window so interaction is only possible via a separate REST or TCP application.

QIS Setup commands

These commands are interpreted and responded to by QIS and perform basic setup

$help

Displays a basic help screen

$scan

Begins a scan of devices, both local USB modules and those visible over the network will be located. QIS uses a broadcast UDP packet to locate LAN devices.

$list

Lists all local and LAN devices. $scan command must have been run previously.

$list details

As $list but with additional device information displayed

$debug [on|off]
$debug?

Sets/returns the debug mode. This generated additional debug information on the Java debugging terminal while running

$default [DeviceConnection]

Sets the connection to send all following commands to the given default device:

$default tcp::qtl1944-01-020

$default [DeviceNumber]

Sets the connection to send all following commands to the given default device. Uses the list number from the $list command

$default 2

$sysinfo

Displays memory usage and stream task information

$shutdown

Closes QIS

QIS Data commands

Although these are executed internally by QIS, they differ from the $ commands in that they require a module identifier in the command, or a currently selected default module (via the $default command).

> For example, the following command will execute on the default module:
> **`stream text header`**
>
> If a default is not set, you must specify the DeviceConnection string before the command, for example:
> **`usb::qtl1824-03-161 stream text header`**

`stream text header`

Returns the stream header information from the most recent stream to be run on the device. There are several versions of the header available which have been extended over time (set with **`stream mode header`** command)

V1 *// Initial 'basic' header*

```
Version: 5      // Underlying stream version, safe to ignore
Format: 15      // Binary coding for the 4 channel enable states
Average: 14     // 2^n value for averaging (32k in this case)
```

V2 *// Extended information header*

```
Version: 5      // V1 header information
Format: 15
Average: 10
V2              // Start of v2 header
@Channels       // Start of channel data
Status status NA // Status channel (triggering)
5V voltage mV   // [Channel] [Measurement] [Unit]
5V current uA
12V voltage mV
12V current uA
5V power uW
12V power uW
@Channels_End   // End of channel data
```

V3 *// XML based information header*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<header>
        <version>V3</version>
        <mainPeriod>4096uS</mainPeriod>
        <legacyVersion>5</legacyVersion>
        <legacyFormat>15</legacyFormat>
        <legacyAverage>10</legacyAverage>
        <channels>
                <channel>
                        <name>Status</name>
                        <group>status</group>
                        <units>NA</units>
                        <dataPosition>0</dataPosition>
                </channel>
                <channel>
                        <name>5V</name>
                        <group>voltage</group>
                        <units>mV</units>
                        <dataPosition>1</dataPosition>
                </channel>
                <channel>
                        <name>5V</name>
                        <group>current</group>
                        <units>uA</units>
                        <dataPosition>2</dataPosition>
                </channel>
                <channel>
                        … Additional channels…
                </channel>
        </channels>
</header>
```

`stream mode header [v1|v2|v3]`

Sets the header mode to use. This must be set **before** the start of a stream

`stream mode power [enable|disable]`

Enable/disable the internal power calculation. Must be set before the start of the stream (defaults to disabled). When enabled AND both current and voltage is enabled for a channel, the power values will be calculated and streamed back along with current/voltage.

`stream?`

Returns running status and buffer information for the current stream operation, examples:

Running *// Stream is running*
Stripes Buffered: 1167 of 8388608 *// 1167 stripes in buffer*

Stopped: User *// Stream stopped at user command*
Stripes Buffered: 94 of 8388608 *// 94 stripes still to read in buffer*

stream text [all|number of stripes]

The option ‘all’ is subject to a limit of 4096 stripes. To ease timing considerations, QIS implements an internal buffer of approximately 8 million stripes.

Returns N stripes in formatted ASCII text from the stream buffer. A ‘stripe’ is defined as all available measurements for a single point in time.

The number of data points will depend on the current record settings of the module you are connected to

[RecordNumber] [StatusFlags] [n data points, as stream header]

```
stream text 3

0 0 7 2 8 49

1 0 7 2 8 50

2 0 7 2 8 50
```

In this case we see 3 stripes. These are numbered 1, 2, 3 since the start of the stream. From this number and the stripe averaging mode we can know the exact time of each stripe.

[StatusFlags] value of 0 = No Trigger, NonZero = Trigger Present

4 data points are present (power is disabled). Default channel ordering is:

5V Voltage, 5V Current, 12V Voltage, 12V Current, 5V Power, 12V Power

When there is no data to return, a line containing ‘eof’ is returned

stream bin [all|number of stripes]

The option ‘all’ is subject to a limit of 4096 stripes. To ease timing considerations, QIS implements an internal buffer of approximately 8 million stripes.

Returns raw format data from the instrument stream, with each sample encoded as a Java native 32 bit integer.

This is more complex than processing the text data, but the processing overhead is potentially lower if you want to save data directly without converting it to strings.

Module Commands

Any commands that are not listed above are assumed to be aimed at a module.

If a default module is selected, the command format it:

```
>hello?
HD Programmable Power Module +Triggering
```

If no default is selected, the DeviceConnection string must be added to each command:

```
>usb::qtl1824-03-161 hello?
XLC Programmable Power Module +Triggering
```

This combination allows easy control of the module, and QIS features at the same time

Example Usage

Simple setup and streaming

Below is a simple example, setting up the system and recording the power up of a device.

```
# Refresh the scan table
$scan
# Connect to the specified module
$default tcp::QTL1999-02-010
# Set the record averaging to 1024 = 4.096mS/sample
record:average 1024
# Set the voltages levels for the output
signal:12v:voltage 12000
signal:5v:voltage 5000
# Enable power calculation
stream mode power enable


# Begin streaming, QIS will now start buffering stripes
record stream
# Enable the outputs to power up the device
run:power up


# Sleep here, until the device has fully powered up…


# Stop streaming
record stop


# Start downloading the buffered data in 500 stripe sections (this
can also be done while still streaming)

Loop Until response contains ‘eof’
    stream text 500
```