Quarch Technology Ltd
QuarchPy Python Package
User Guide

Friday, 19 October 2018

Change History

1.0		Beta Release
1.2		Changed object model, first full release
1.3.1		Added QIS functions
1.3.2		Fixed bug when locally running QIS
1.3.3		Added additional QIS useful functions
1.3.4		Improved QIS launch mechanism Implemented QIS resampling
1.5.5		Added FIO functions and various bug fixes

Contents

Introduction

Quarch modules can be easily controlled using Python through multiple interface options:

- USB
- Serial
- REST
- Telnet

QuarchPy is a python package designed to provide an easy to use API which will work seamlessly over any of the communication options.

The package contains all prerequisites needed as a single PyPI project which can be installed from the python online repository or a downloaded local copy.

This makes it easy to install and keep up to date, while also providing you with full access to the source code if you want to make changes or additions.

Modules Supported

- All breaker/hot-swap modules
- All Power modules
- All Physical layer switch modules

Requirements

- Windows or Linux PC
- Python 2.x or Python 3.x
- Quarch USB driver (Windows only)

Installation

Python

If Python is not installed on your system, go to https://www.python.org/downloads/ and choose the relevant version for your system.

Under Windows it is sensible to ensure the Python installation directory and PythonXX\Scripts are included in the PATH environment variable. See HERE for instructions on how to do this.

QuarchPy Package

The Quarch Python package can be installed from the Python web repository (assuming you have internet access) or via the download from our site which contained this document.

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 .whl file, download the file from https://pypi.org/manage/project/quarchpy/releases/, navigate to the folder containing the quarchpy-x.y.z-py2.py3-none-any.whl file and run:

```
>pip install quarchpy-x.y.z-py2.py3-none-any.whl
```

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

```
>python –m pip install quarchpy-x.y.z-py2.py3-none-any.whl
```

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.

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.

Basic Setup

We strongly recommend you download a relevant application note to demonstrate the use of the Python API, only the basics are shown here.

Below is a simple example to import and use a Quarch module.

```
# Imports QuarchPy package
from quarchpy import *

# Open a connection to a module on USB
myDevice = quarchDevice("usb:QTL1999-01-002")
```

For USB connections, you specify a full, or partial serial number, as found on the product label.

For LAN connections (ReST and Telnet) you can specify either an IP address, or NetBIOS name (if supported by your network). The default NetBIOS name for most modules is the serial number:

```
myDevice = quarchDevice ("telnet:192.168.1.55")
```

Or

```
myDevice = quarchDevice ("telnet:QTL1079-03-137")
```

For Serial connections, specify the COM port on windows

```
myDevice = quarchDevice ("SERIAL:COM4")
```

Or the linux serial port ID

```
myDevice = quarchDevice ("SERIAL:ttyS0") or you may need
myDevice = quarchDevice ("SERIAL:/dev/ttyS0")
```

You can now send commands to the module. The full command list for each module can be found in its Technical Manual, which can be downloaded from www.quarch.com.

To send the simple 'hello?' command, use:

```
myDevice.sendCommand ("hello?")
```

This function will return the command response, so you can easily see the result with:

```
print (myDevice.SendCommand("hello?"))
```

Finally you must close the connection with

```
myDevice.closeConnection()
```

QuarchPy API Listings

Class descriptions

quarchDevice

Starts a connection and implements basic control over any Quarch device.

quarchDevice (ConString, [ConType = "PY"], [timeout = "5"])

argument string ConString
Specifies the device to connect to by COM port, IP address or USB ID

USB:QTL1177 or USB:QTL1177-02 or USB:QTL1177-02-012
Will connect to the first device found matching the string.

SERIAL:COM4 or SERIAL:ttys0
Linux only: ls /dev/serial/ lists serial connections in your system.

TELNET:192.168.1.102 or TELNET:QTL1079-02-004
You can use NetBIOS name if your networks supports it.

REST:192.168.1.102 or REST:QTL1079-02-004
You can use NetBIOS name if your networks supports it.

argument string **ConType**
Allows additional information on connection type for QIS and similar

QIS
Local instance of QIS

QIS:192.168.100.102:9722
QIS:ip:port

QPS
Local instance of QPS

PY
Python connection

argument string **timeout**
Specifies a timeout when searching for device

return object quarchDevice

sendCommand(command)
Sends a command to the device.

argument string **command**
Torridon command to send

return string
Answer from module.

openConnection()
Opens a connection with device.

closeConnection()
Closes the connection with device and any open communication ports.

qisInterface
Implements direct communication with QIS.

qisInterface([host=127.0.0.1], [port=9722])

argument string **host**
Specify the remote host ip.

argument integer **port**
Specify the remote host port.

return object qisInterface

getDeviceList()
Returns a list of devices connected to the QIS interface object.

return list
Returns a list of devices connected to the QIS interface object.

quarchPPM

Extends quarchDevice to provide additional functions for power modules.

quarchPPM(quarchDevice)

argument quarchDevice object **quarchDevice**
Basic quarchDevice object to use as a power module.

startStream([fileName='streamData.txt'], [fileMaxMB=2000], [streamName ='Stream With No Name'], [streamAverage = None], [releaseOnData = False)]

argument string **fileName**
Base file name for the output files.

argument int **fileMaxMB**
Maximun size of a file before creating a new one (default = 2000).

argument string **streamName**
Name for stream file header (default = "Stream With no Name").

argument string **streamAverage**
Sets the averaging for the stream.

argument boolean releaseOnData
If True, only returns from startStream after th first row of data is returned form the module.

streamResampleMode (resampleRate)

Sets the resampling mode.

argument resampleRate
off, [x]ms or [x]us

stopStream()

Stops stream.

quarchQPS

Implements the interface with QPS.

quarchQPS (quarchDevice)

argument quarchDevice object **quarchDevice**
quarchDevice Object.

startStream (directory)

argument path **directory**
Declares where the data from the stram will be located.

return object **quarchStream**

quarchStream

Implements an interface with QPS stream.

quarchStream (quarchQPS)

argument quarchQPS object **quarchQPS**
quarchQPS object connected to QPS.

openConnection()

Re-open the connection if it has been closed.

closeConnection()

Closes the device connection and any open communication ports.

createChannel (channelName, channelGroup, baseUnits, usePrefix)

Creates a new channel in QPS.

argument string **channelName**
Name of channel that will be add to QPS.

argument string **channelGroup**
Name of channel group that the channel will be add to.

argument string **baseUnits**
Unit for the new channel.

argument boolean **usePrefix**
Wether the new channel will automatically use metric prefixes to represent numbers.

hideChannel (channelSpecifier)

Hides a channel within QPS.

argument string **channelSpecifier**
The channel name the user wants to hide.

hideAllDefaultChannels

Hides as default channels. Only user specified channels will be visible.

showChannel (channelSpecifier)

Show a specific channel in QPS.

argument string **channelSpecifier**
The channel name the user wants to show.

addAnnotation (annotationString, [annotationTime=0])

Adds an annotation at a specified point in graph.

argument string **annotationString**
Annotation string that will appear in QPS.

"text to show"
Will show this whole string on graph.

"<<text> text to show</text> <extraText> text to hide</extraText>>"
Add the option to hide part of the string. This can be accessed righ clicking in the annotation.

argument string **annotationTime**
Either UNIX time or as elapsed time since the start of the dataset.

1528370532877
Unix time.

e12D13:14:15.167
Specifies 12 days, 13 hours, 14 minutes, 15.167 seconds

addComment (commentString, [commentTime=0])

Adds an annotation at a specified point in graph.

argument string **commentString**
Comment string that will appear in QPS.

"text to show"
Will show this whole string on graph.

"<<text> text to show</text> <extraText> text to hide</extraText>>"
Add the option to hide part of the string. This can be accessed righ clicking in the comment

argument string **commentTime**
Either UNIX time or as elapsed time since the start of the dataset.

1528370532877
Unix time.

e12D13:14:15.167
Specifies 12 days, 13 hours, 14 minutes, 15.167 seconds

addDataPoint (channelName, groupName, dataValue, [dataPointTime])

Adds a new data point to the specified channel for representation on graph.

argument string **channelName**
Name of channel that will be add to QPS.

argument string **groupName**
Name of channel group that the channel will be add to.

argument int **driveTemp**
Value of the data point.

argument string **dataPointTime**

1528370532877
Unix time.

e12D13:14:15.167
Specifies 12 days, 13 hours, 14 minutes, 15.167 seconds

stopStream()

Stops the current stream.

myChannels()

Returns the active channel.

return list
List of channels currently active in QPS.

Helper functions

QIS

isQisRunning ([host="localhost"], [port"9722"])

Returns wheter QIS is running or not.

argument string **host**
Specify the remote host ip.

argument integer **port**
Specify the remote host port.

return boolean
True if QIS is running or False if not.

startLocalQis ([QisPath=/path/inside/QuarchPy])

Start a copy of QIS.

argument path **QisPath**
Path to a custom QIS folder.

closeQis (host, port)

Closes QIS.

argument string **host**
Specify the remote host ip.

argument integer **port**
Specify the remote host port.

QPS

isQpsRunning ([host="localhost"], [port"9722"])

Returns wheter QPS is running or not.

argument string **host**
Specify the remote host ip.

argument integer **port**
Specify the remote host port.

return boolean
True if QPS is running or False if not.

startLocalQps ([QisPath=/path/inside/QuarchPy], [keepQisRunning=False])

Starts a new instance of QPS.

argument path **QisPath**
Path to a custom QPS folder.

argument boolean **keepQisRunning**
Will keep QIS running after the execution of the script.

closeQps (host, port)

argument string **host**
Specify the remote host ip.

argument integer **port**
Specify the remote host port.

FIO specific functions

FIO

runFIO (myStream, mode, fioCallbacks, user_data, [arguments], [file_name])

Runs FIO.

argument qpsStream object **myStream**
Your QPS stream where you want to send the data points to.

argument string **mode**
Selects the execution mode.

file
QuarchPy with parse the paraemters from the variable "fioFile".

arg
QuarchPy will parse the parameters from the variable "arguments".

cli
FIO will run the same way it would in a shell.

argument dictionary **fioCallbacks**
A dictionary containing all the callbacks for FIO.

argument list **user_data**
List of arguments to return from FIO.

argument dictionary **arguments**
List of arguments to send to FIO.

argument string **file_name**
File with FIO configuration.

GetQpsModuleSelection (QpsConnection)

Prompts the user to choose a module.

argument qpsStream object **QpsConnection**
Your QPS connection.

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 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.