QP-CryoSwitch Controller installation guide

Packaged goods

- 1x CryoSwitch Controller

Preparation

GIT repository

For user convenience, a small repository was created containing all the files necessary to operate the QP-CryoSwitch Controller from a PC.

The GIT repository can be found here.

Inside the repository, you will find a copy of the installation guide (this file), the controller datasheet, the Labjack installation file, and finally a library containing python code to interface with the CryoSwitch Controller.

Python setup

The controller's library doesn't need any specific python distribution, but some extra python library must be installed.

The easiest way to do so is by using 'pip'. From your base or virtual environment use the command line to run:

```
> pip install -r requirements.txt
```

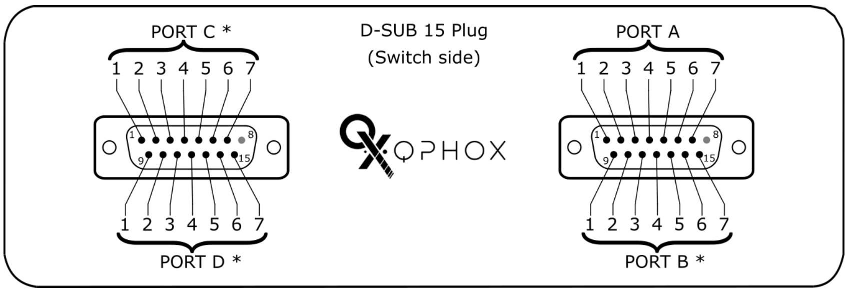
Preparation Summary

- Clone the GIT repository.
- Install the necessary python dependencies.

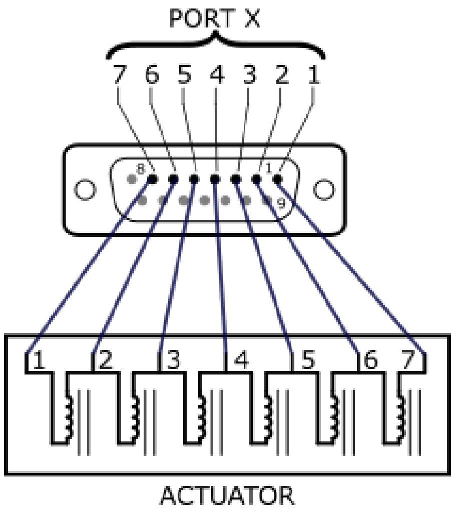
Connections and ports information

The QP-CryoSwitch Controller is specifically designed for driving inductive loads at cryogenic temperatures.

The controller supports a maximum of 4 ports (version dependent) and 7 contacts or channels per port. The following figure shows the back plate and pinout of the controller.



For example, if a SP6T relay/actuator is used the connections would be as follows:



Python Library

The library contains the ‘Cryoswitch’ class which is an easy way of interfacing between the QP-CryoSwitch Controller and your computer.

A basic implementation of the CryoSwitchController class can be done with the following functions:

- start()

- Input: None
- Default: None
- Enables the voltage rails, voltage converter and output channels

- set_output_voltage(Vout)

- Input: Desired output voltage (Vout)
- Default: 5V
- Sets the converter voltage to Vout. The output stage later utilizes the converter voltage to generate the positive/negative pulses.

- set_pulse_duration_ms(ms_duration)

- Input: Pulse width duration in milliseconds (ms_duration).
- Default: 10ms.
- Sets the output pulse (positive/negative) duration in milliseconds.

- connect(port, contact)

- Input: Corresponding port and contact to be connected. Port={A, B, C, D}, contact={1,...,6}
- Default: None.
- Connects the specified contact of the specified port (switch).

- disconnect(port, contact)

- Input: Corresponding port and contact to be disconnected. Port={A, B, C, D}, contact={1,...,6}
- Default: None.
- Disconnects the specified contact of the specified port (switch).

Advanced functions

- enable_OCP()

- Input: None
- Default: None.
- Enables the overcurrent protection.

- set_OCP_mA(OCP_value)

- Input: Overcurrent protection trigger value (OCP_value).
- Default: 100mA.
- Sets the overcurrent protection to the specified value.

- enable_chopping()

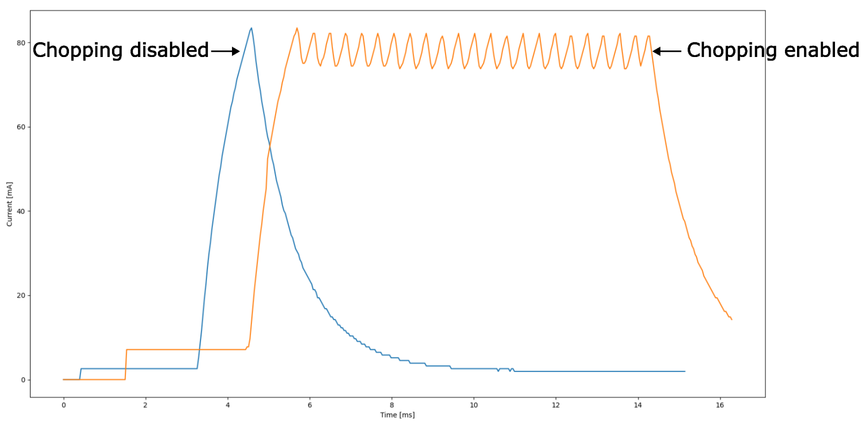
- Input: None.
- Default: None.
- Enables the chopping function. When an overcurrent condition occurs, the controller will 'chop' the excess current instead of disabling the output. Please refer to the installation guide for further information.

- disable_chopping()

- Input: None.
- Default: None.
- Disables the chopping function. When an overcurrent condition occurs, the controller will disable the output voltage. Please refer to the installation guide for further information.

The following image shows the current waveforms when the chopping function is enabled and disabled. The Blue trace shows the case when the chopping feature is disabled, as soon as the current exceeds the preset threshold the output is disabled.

On the other hand, the orange trace shows the case when the chopping feature is enabled. As soon as the current exceeds the preset threshold, the controller reduces the voltage in order to reduce the current. After the current drops, the controller will increase the voltage in order to increase the current. This cycle continues until the end of the pulse.


Chopping disabled
Chopping enabled
Current [mA]
Time [ms]