QP-CryoSwitch Controller installation guide

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.

Labjack setup

The current version of the QP-CryoSwitch Controller is designed to be interfaced with the Labjack T4 DAQ device.

In order to communicate with the Labjack, the appropriate drivers must be installed first. This can be done by running the LabJack.exe file included in the GIT repository or by downloading it from the official Labjack [webpage](#).

Python setup

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

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

```
> pip install labjack-ljm
```

Preparation Summary

- Clone the GIT [repository](#).
- Install the Labjack drivers.
- Install the necessary python dependencies (`pip install labjack-ljm`).

Python Library

The library contains the 'radiall_switch_controller' class which is an easy way of interfacing between the QP-CryoSwitch Controller and the Labjack T4.

NOTE$_1$: Since the Radiall switch R583423141 has a latching actuator, it's important to start the program with the switches in a known state. By default, the program assumes all actuators are in an open state.

The class contains 3 useful functions, these are:

- *radiall_switch_controller.connect_switch_port(port)*

  Used for connecting the actuator RF input number "port" to the common RF output.

- *radiall_switch_controller.disconnect_switch_port(port)*

  Used for disconnecting the actuator RF input number "port" from the common RF output.

- *radiall_switch_controller.read_switch_state()*

  Returns the current switch state, please remember **NOTE$_1$**.

For more information regarding the library please refer to the README.md file inside the repository.
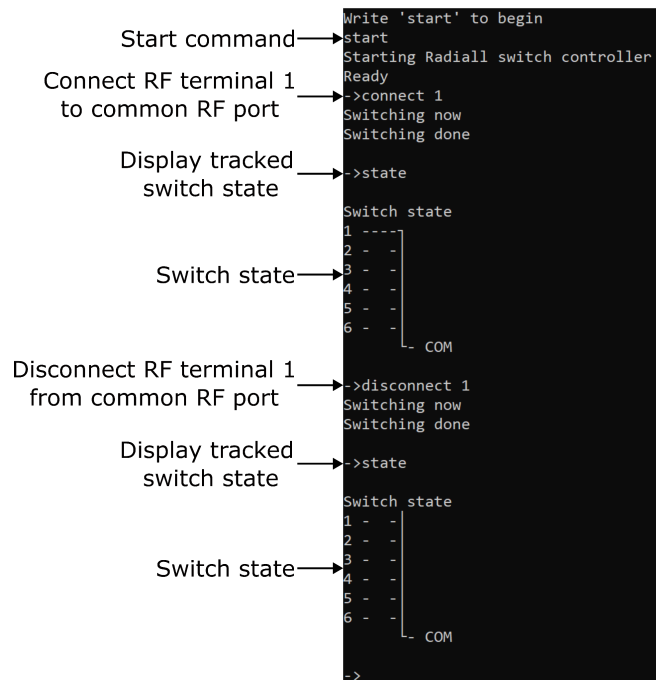
Implementation: cmd_radiall_switch_controller.py

The *cmd_radiall_switch_controller.py* is a simple implementation of the library described above.

The script can be executed either from the command line or from your favorite IDE. It contains 4 commands:

- *start*: starts the communication with the Labjack, it's the first command that needs to be executed after running the script.
- *connect x* : connects port **x** to the common terminal. It is equivalent to the radiall_switch_controller.connect_switch_port(port) function implemented in the library shown in the previous section.
- *disconnect x* : disconnects port **x** from the common terminal. It is equivalent to the *radiall_switch_controller.disconnect_switch_port(port)* function implemented in the library shown in the previous section.
- *state* : displays the current state of the switch.
- *stop* : stops the program.

An example of how the interface with the program would look is displayed below:

Physical connections

The following image shows the physical connections needed to operate the CryoSwitch Controller.
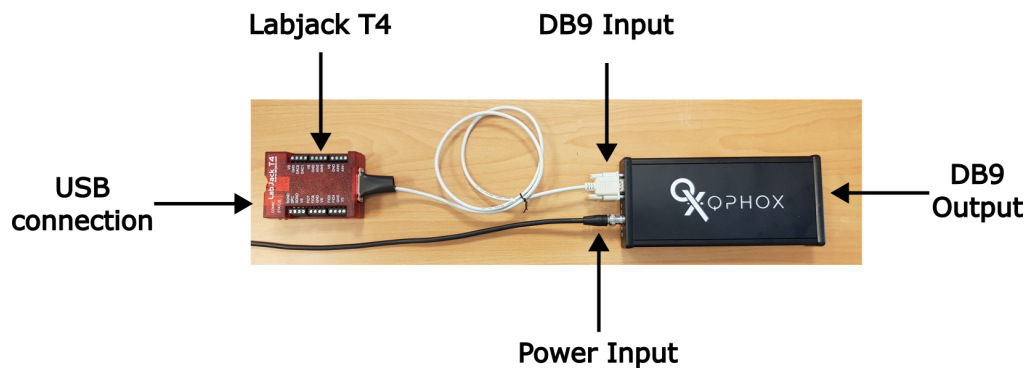
Necessary connections:

From	Cable	To
Computer	User preferred to USB-B	Labjack
Labjack	DB15 to DB9	IN (Controller)
Power Supply	Banana plug to BNC Female	POWER(Controller)
OUT (Controller)	DB9 to user preferred	Radiall Actuator

The output pinout of the controller can be seen in the following image:

D-SUB 9 Plug
(Controller)
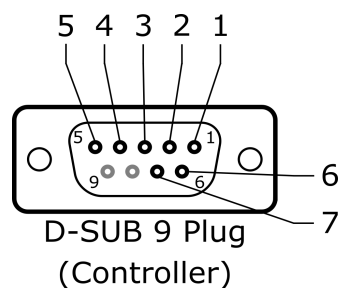
Please refer to the controller's datasheet for further information regarding the electrical characteristics.