Communication with QubeCL and QubeDL

ppqSense App.Note no. 1

ppqSense s.r.l

ppqSense Development Team

Revision 1.2 , 2025.06.06

ppqSense distributes its QubeCL and QubeDL laser drivers as an almost plug&play device. To make it possible to use each driver directly out of the box, ppqSense has developed a Control Software for QubeCL and QubeDL, distributed with each one of the drivers with a dedicated pendrive.
Nevertheless, it may be possible that more complex instrumental setups need to integrate the control of the Qube inside a bigger control software responsible of the management of the whole experiment.

In order to allow each user to fully integrate its Qube inside the experiment, this Application Note describes the Serial Connection of the drivers, along with some simple connection examples with different methods.

Contents

List of Tables

List of Figures

1 Introduction

In order to control, manage and monitor the operations of QubeCL and QubeDL laser drivers, ppqSense distributes a dedicated Control Software which provides an intuitive GUI and gives the user access to all the functionalities of the QubeCL/DL drivers, connecting to them by the mean of an USB serial port. Nevertheless, the Control Software is only meant to give the user direct control over the drivers and does not allow to be integrated into a third party control software.

Complex experimental setups may need to integrate the control of the Qube into their own control software, in order to perform automatized operations without the constant supervision of a user. In this Application Note, we provide the technical specification to control the QubeCL and QubeDL drivers with a third-party Control Software by the means of the same USB serial port used by ppqSense Qube Control Software, along with some specific examples. This will give the user the ability to fully integrate the Qube inside their experimental setup, controlling it with their own control software regardless of the programming method chosen (both LabVIEW and text-based programming languages such as C and Python can be used to achieve control of the Qubes over the Serial Port).

CAUTION

The Control Software provided by ppqSense includes some safety features that are useful to prevent any usage of the QubeCl and QubeDL drivers that may harm both the driver itself or the laser connected to it.
While using a third party control software, the user must be particularly cautious in order to avoid any possibly harmful condition for the driver or the laser.
Before using any third party control software, ppqSense strongly encourages the user to contact the ppqSense technical support to obaint counseling and advices from the techincal staff.

CAUTION

The QubeCL and QubeDL firmware does not prevent the user from sourcing current to the laser without previously activating the temperature controller, this safety measure is usually performed by the Qube Control Software provided by ppqSense.
While operating with a third party control software, care must be taken in order to never source current to the laser while the temperature controller module (if present) is not active.

2 Direct Communication

The Qubes operates on the basis of a series of commands received by a computer through the USB Serial Port. Each command is codified with a string, to be as friendly for a user as possible, being easily readable.

The commands share the same structure, each one being composed by a identifier and a value, separated by a colon character.

[identifier]:[value]

The *identifier* allows to differentiate all the available commands, each one acting on a specific functionality of the connected Qube. The *value* is used in association with the *identifier* to define the operation to be carried out by the Qube (for example *activate* or *deactivate* the modulations) or to pass numerical value to be used by the firmware of the device for its operations (for example laser temperature and bias current *setpoints*). The *value* parameter also differentiates the **write** commands and the **queries**.

- **Query:** It is possible to send queries to the Qube in order to receive information about its current status, read digital monitors and request informations about the driver. Not all the available **identifiers** have an associated query.
  In order to send queries to the Qube, the **value** of the command must be replaced by the **?** character, for example:

  iset:?

- **Write commands:** Allow to send the Qube numerical values or commands to enable/disable some of its functionalities, for example:

  iset:150

2.1 Serial Port Specifications

The ppqSense Control Software communicates with the Qubes by the mean of a Serial Connection. The same port can also be used by a third party software to interact with the driver. The baud rate of the Qubes Serial Port is 115200 bps, which has to be set on the third party software in order for it to work with the driver.
The connection of a third-party software is only possible if the Qube Control Software is not running, otherwise the Serial Port resources is kept busy and it is not accessible from the third party software.

2.2 Serial Commands Specifications

User can find all the available commands listed in the subsequent tables. Those commands are the ones that are available for the final user to be integrated in a third party control software or to be used in the Qube Control Software to directly communicate with the driver via the dedicated form in the Advanced tab.
In order for the commands to work, they must be sent at the appropriate Baud Rate and each command must be terminated with the **n** character.
If a suitable *query* is issued, the Qube replies with a string containing the requested informations. All the strings sent as a reply to a *query* are terminated with the **r****n** characters.

2.3 List of available commands

Table 1: Available commands for Qube current generator.

Cmd	Value	R/W	Action	Output Format	Unit
Output Current and Modulations					
id:	?	R	Returns the identificative code of the instrument	Qube - ####	N.A.
ilas:	?	R	Returns the Output Current value readed from the Qube monitor.	####.##	mA
iset:	?	R	Returns the Current setpoint.	####.##	mA
	####.##	W	Sets the Current setpoint.	N.A.	mA
iout:	on	W	Switches the Current ON.	N.A.	N.A.
	off	W	Switches the Current OFF. If the modulations were active, they're also switched OFF.	N.A.	N.A.
imax:	?	R	Returns the Output Current Maximum Limit set by the user.	####.##	mA
	####.##	W	Sets the Output Current Maximum Limit.	####.##	mA
vlas:	?	R	Returns the measured Voltage Drop across the laser	####.##	V
mod:	on	W	Enables the overall Modulation Switch. This command becomes active 10 seconds after. the use of the **iout:on** command.	N.A.	N.A.
	off	W	Disables the overall Modulation Switch. If some of the modulations are active, they are switched off too.	N.A.	N.A.
mod1:	on	W	Enables the Modulation Channel 1.	N.A.	N.A.
	off	W	Disables the Modulation Channel 1.	N.A.	N.A.
mod2:	on	W	Enables the Modulation Channel 2.	N.A.	N.A.
	off	W	Disables the Modulation Channel 2.	N.A.	N.A.

Table 2: Available commands for Qube temperature controller.

Cmd	Value	R/W	Action	Output Format	Unit
Laser Temperature Controller					
tlas	?	R	Return the Temperature of the laser measured by the TC Module.	####.##	°C
tstab:	on	W	Switches the Temperature Stabilization on.	N.A.	N.A.
	off	W	Switches the Temperature Stabilization off.	N.A.	N.A.
tset:	?	R	Return the laser Temperature Setpoint	####.##	°C
	####.##	W	Sets the laser Temperature Setpoint	N.A.	°C
kp:	####.##	W	Sets the Proportional Gain Coefficient for the digital PID that manages the laser Temperature Stabilization.	N.A.	A/K
ki:	####.##	W	Sets the Integral Gain Coefficient for the digital PID that manages the laser Temperature Stabilization.	N.A.	A/Ks
kd:	####.##	W	Sets the Derivative Gain Coefficient for the digital PID that manages the laser Temperature Stabilization.	N.A.	As/K
pid:	?	R	Returns the Gain Values of the Digital PID that manages the laser Temperature Stabilization. kp ki kd	####.##: ####.##: ####.##	A/K A/Ks As/K
tecsign:	dir	W	Sets the Temperature Stabilization PID to work in direct mode.	N.A.	N.A.
	rev	W	Sets the Temperature Stabilization PID to work in reverse mode.	N.A.	N.A.
tlimax:	?	R	Returns the Maximum Temperature Setpoint for the Stabilization PID.	####.##	°C
	####.##	W	Sets the Maximum Temperature Setpoint for the Stabilization PID.	N.A.	°C
tlimin:	?	R	Returns the Minimum Temperature Setpoint for the Stabilization PID.	####.##	°C
	####.##	W	Sets the Minimum Temperature Setpoint for the Stabilization PID.	N.A.	°C

teclim:	?	R	Returns the Maximum Current Output for the TC Module.	####.##	A
	####.##	W	Sets the Maximum Current Output for the TC Module.	N.A.	A
teslim:	?	R	Returns the threshold for the Active Temperature Error Control.	####.##	°C*s
	####	W	Sets the threshold for the Active Temperature Error Control.	N.A.	°C*s

Table 3: Available commands for the DDS submodule.

DDS Module					
Cmd	**Value**	**R/W**	**Action**	**Output Format**	**Unit**
dds1:	?	R	Returns the status of the DDS on modulation channel 1. **1**: DDS is active. **0**: DDS is inactive.	#	Bool.
	on	W	Activates the DDS on channel 1.	N.A.	N.A.
	off	W	Switches off the DDS on channel 1.	N.A.	N.A.
dds1w:	?	R	Returns the waveform set for DDS on channel 1. **1**: Sine wave. **2**: Triangular wave.	#	Int.
	#	W	Sets the waveform for the DDS on channel 1. Accepted values are: **1**: Sine wave. **2**: Triangular wave.	N.A.	Int.
dds1f:	?	R	Returns the frequency of the DDS on channel 1.	####.##	Hz
	####.##	W	Sets the frequency for the DDS on channel 1.	N.A.	Hz
dds1a:	?	R	Returns the amplitude of the DDS on channel 1.	####.##	mA
	####.##	W	Sets the amplitude for the DDS on channel 1.	N.A.	mA
dds1p:	?	R	Returns the phase of the DDS on channel 1.	####.##	Deg.
	####.##	W	Sets the phase for the DDS on channel 1.	N.A.	Deg.
dds2:	?	R	Returns the status of the DDS on modulation channel 2. **1**: DDS is active. **0**: DDS is inactive.	#	Bool.
	on	W	Activates the DDS on channel 2.	N.A.	N.A.
	off	W	Switches off the DDS on channel 2.	N.A.	N.A.
dds2w:	?	R	Returns the waveform set for DDS on channel 2. **1**: Sine wave. **2**: Triangular wave.	#	Int.
	#	W	Sets the waveform for the DDS on channel 2. Accepted values are: **1**: Sine wave. **2**: Triangular wave.	N.A.	Int.

dds2f:	?	R	Returns the frequency of the DDS on channel 2.	####.##	Hz
	####.##	W	Sets the frequency for the DDS on channel 2.	N.A.	Hz
dds2a:	?	R	Returns the amplitude of the DDS on channel 2.	####.##	mA
	####.##	W	Sets the amplitude for the DDS on channel 2.	N.A.	mA
dds2p:	?	R	Returns the phase of the DDS on channel 2.	####.##	Deg.
	####.##	W	Sets the phase for the DDS on channel 2.	N.A.	Deg.
syncf:	?	R	Returns the frequency of the DDS channel to which the Sync. Out signal is synchronized.	####.##	Hz
	ch1	W	Synchronizes the Sync. Out signal to the DDS1	N.A.	N.A.
	ch2	W	Synchronizes the Sync. Out signal to the DDS2	N.A.	N.A.

Table 4: Available commands for the PLL locking module.

PLL Locking Module					
Cmd	**Value**	**R/W**	**Action**	**Output Format**	**Unit**
mux:	#	W	Allow the user to select which signal is present on the DIVIDER MON output. Allowed values are: **0** DIVIDER MON switched off. **2** DIVIDER MON set on RF. **4** DIVIDER MON set on LO.	N.A.	Int.
sig:	#	W	Set the sign of the PLL correction loop. Accepted values are: **0**: negative correction loop. **1**: positive correction loop.	N.A.	Int.
cp:	?	R	Returns the value of cp, which is the gain of the PLL. Must be switched ON.	##	HEX.
	on	W	Switches ON the proportional gain of the PLL chip.	N.A.	N.A.
	off	W	Switches OFF the proportional gain of the PLL chip.	N.A.	N.A.
	#	W	Set the proportional gain. The value can range from 1 to 8.	N.A.	Int.
ndiv:	####	W	Sets the divisor coefficient for the RF frequency input.	N.A.	Int.
rdiv:	####	W	Sets the divisor coefficient for the LO frequency input.	N.A.	Int.
pby:	off	W	Sets the PLL correction loop in proportional only mode.	N.A.	N.A.
	on	W	Sets the PLL correction loop in proportional-integrative mode.	N.A.	N.A.
tp:	#	W	Sets the Integrative time constant of the PLL correction loop. Accepted values range from 0 to 3.	N.A.	Int.
tz:	#	W	Sets the Proportional gain of the PLL correction loop. Accepted values range from 0 to 3.	N.A.	Int.
hg:	#	W	Sets the proportional gain of the PLL chip. Accepted values ranges from 0 to 3.	N.A.	Int.

Ik:	off	W	Switches OFF the correction current injected by the PLL module into the Bias Current.	N.A.	N.A.
	on	W	Activates the correction current injected by the PLL module into the Bias Current. Overall modulation must be enabled for it to work.	N.A.	N.A.
Im:	?	R	Returns the value of the PLL correction loop output voltage.	####.##	mV

Table 5: Available commands for the PDH locking module.

PDH Locking Module					
Cmd	**Value**	**R/W**	**Action**	**Output Format**	**Unit**
pdhint:	off	W	Sets the PDH correction loop in proportional only mode.	N.A.	N.A.
	on	W	Sets the PDH correction loop in proportional-integrative mode	N.A.	N.A.
pdhhold:	off	W	Deactivates the **HOLD** input of the PDH module.	N.A.	N.A.
	on	W	Activates the **HOLD** inptut of the PDH module.	N.A.	N.A.
pdhlock:	off	W	Deactivates the correction current injected into the Bias current by the PDH module.	N.A.	N.A.
	on	W	Activates the correction current injected into the Bias current by the PDH module. Overall modulation enable must be active for it to work.	N.A.	N.A.
pdhrint:	#	W	Sets the first proportional gain of the PDH correction loop. Accepted values range from 0 to 3.	N.A.	Int.
pdhtz:	#	W	Sets the second proportional gain of the PDH correction loop. Also affect the Integrative time constant. Accepted values range from 0 to 3.	N.A.	Int.
pdhtp:	#	W	Sets the time constant of the PDH correction loop. Accepted values range from 0 to 3.	N.A.	Int.
pdhmon:	#	W	Sets which signal is present on the Monitor Output of the PDH module. Accepted values are: **0**: Error Signal. **1**: Correction signal.	N.A.	Int.
pdhmonint:	?	R	Returns the values of the Error and Correction signals converted by the on board ADCs.	####.##: ####.##	mV
pdhvoff:	?	R	Returns the input offset of the PDH correction loop.	####.##	mV
	####.##	W	Sets the input offset of the PDH correction loop. The value can range from 0 to 5000. Default is 2500.	N.A.	mV

pdhdp:	?	R	Returns the PDH correction loop pre-amplifier gain.	####.##	mV
	####.##	W	Sets the PDH coorection loop pre-amplifier gain. Accepted values range from 0 to 63.	N.A.	mV

Table 6: Available commands for the LIA locking module.

LIA Module					
Cmd	**Value**	**R/W**	**Action**	**Output Format**	**Unit**
lkpi:	?	R	Returns the mode of the correction loop: **0** stands for **P** mode. **1** stands for **PI** mode.	#	Bool.
	0	W	Sets the correction loop in P mode.	N.A.	Int.
	1	W	Sets the correction loop in PI mode.	N.A.	Int.
lkflt:	?	R	Returns the status of the Low Pass Filter of the correction loop: **0** stands for inactive. **1** stands for active.	#	Bool.
	en	W	Activates Low Pass Filter	N.A.	N.A.
	dis	W	Deactivates Low Pass Filter	N.A.	N.A.
lkgain:	?	R	Returns the VGA Gain.	####.##	dB
	####.##	W	Sets the VGA Gain	N.A.	dB
lktp:	?	R	Returns the pole of the correction loop.	#	Int.
	#	W	Sets the pole of the correction loop. Accepted value ranges from 0 to 3.	N.A.	Int.
lktz:	?	R	Returns the proportional gain of the correction loop.	#	Int.
	#	W	Sets the proportional gain of the correction loop. Accepted values ranges from 0 to 3.	N.A.	Hz
lktpb:	?	R	Returns the frequency setting for the Low Pass Filter of the correction loop.	#	Int.
	#	W	Sets the frequency setting for the Low Pass Filter of the correction loop. Accepted value ranges from 0 to 3.	N.A.	mA
lklock:	on	W	Activates the LIA correction loop.	N.A.	N.A.
	off	W	Deactivates the LIA correction loop.	N.A.	N.A.
lkdemod:	?	R	Returns the demodulation mode of the LIA. **0** stands for **f** mod. **1** stands for **2f** mode. **2** stands for no deomdulation acrive.	#	Int.
	f	W	Sets the **f** demodulation mode	N.A.	N.A.
	2f	W	Sets the **2f** demodulation mode	N.A.	N.A.
	free	W	Deactivates the demodulation	N.A.	N.A.

lkmon:	?	R	Returns the currently set for the output monitor of the LIA module: **0** stands for correction monitor. **1** stands for ERROR monitor.	####.##	Deg.
	err	W	Sets the output monitor to show the ERROR signal.	N.A.	N.A.
	lock	W	Sets the output monitor to show the correction signal.	N.A.	N.A.
lkIIR:	?	R	Returns the digital filter currently in use: **1** stands for **BP0**. **2** stands for **BP1**. **3** stands for **BP2**. **4** stands for **LP1**. **5** stands for **LP2**. **6** stands for **Notch**. **7** stands for **All Pass**.	#	Int.
	BP0	W	Sets the BP0 filter	N.A.	N.A.
	BP1	W	Sets the BP1 filter	N.A.	N.A.
	BP2	W	Sets the BP2 filter	N.A.	N.A.
	LP0	W	Sets the LP0LP0 filter	N.A.	N.A.
	LP1	W	Sets the LP1 filter	N.A.	N.A.
	NOTCH	W	Sets the NOTCH filter	N.A.	N.A.
	ALLPASS	W	Sets the ALLPASS filter	N.A.	N.A.

Table 7: Available commands to use the Slow Loop with locking modules.

Qube Slow Loop					
pllock:	?	R	Returns the current status of the Slow Loop: **0** stands for inactive. **1** stands for active.	#	Bool.
	on	W	Activates the Slow Loop for the mounted locking module, if any.	N.A.	N.A.
	off	W	Deactivates the Slow Loop.	N.A.	N.A.
pllocka:	?	R	Returns which actuator is currently set to be used by the Slow Loop: **0** Slow Loop uses laser current. **1** Slow Loop uses laser temperature.	#	Bool.
	temp	W	Sets the Slow Loop to use the laser temperature as actuator.	N.A.	N.A.
	curr	W	Sets the Slow Loop to use the laser current as actuator.	N.A.	N.A.

pllocks:	?	R	Returns the sign of the Slow Loop: **0** Slow Loop acting in reverse mode. **1** Slow Loop acting in direct mode.	#	Bool.
	dir	W	Sets the Slow Loop to work in direct mode.	N.A.	N.A.
	rev	W	Sets the Slow Loop to work in reverse mode.	N.A.	N.A.
pllockt:	?	R	Returns the time interval at which the Slow Loop reads the locking module output in order to operate its correction. The smaller the reading interval, the faster the Slow Loop reacts.	####	ms
	####	W	Sets the time Interval at which the Slow Loop checks the Locking Module output amplitude. Acceptable values range from 1 to 2000.	N.A.	ms
pllocki:	?	R	Returns the maximum laser current change that the Slow Loop can produce.	####	mA
	####	W	Sets the maximum change the Slow Loop can produce to the current output to the laser. This parameter must be set for safety reason in order not to damage the laser with excessive bias current.	N.A.	mA

Table 8: Available commands to check the instrument status.

Qube Instrument Status					
vcc:	?	R	Returns the Supply Voltage for the CM Module measured by the Qube .	####.##	V
tsense:	?	R	Returns the internal Temperature of the Qube instrument.	####.##	°C

3 Examples

The commands listed in the previous pages of this document are simple strings that can be sent to the Qube over its Serial Port, hence it is possible to establish the connection, send and receive commands, queries and replies with a variety of different programming languages and software.
Here we provide a couple simple example to show how to establish the communication with a QubeCL or QubeDL using different methods. In both cases, some of the previously described commands are used.

3.1 LabVIEW

To control a Qube by the mean of a LabVIEW based control software, it is sufficient to properly configure a Serial COM port using the basic tools already provided by National Instrument.
ppqSense made available a simple VI (Qube_Serial_Interface.vi) that can be integrated into a LabVIEW program in order to send and receive textual strings to a QubeCL with the proper format which can be downloaded from the website of requested to customer assistance (qube.support@ppqsense.odoo.com).

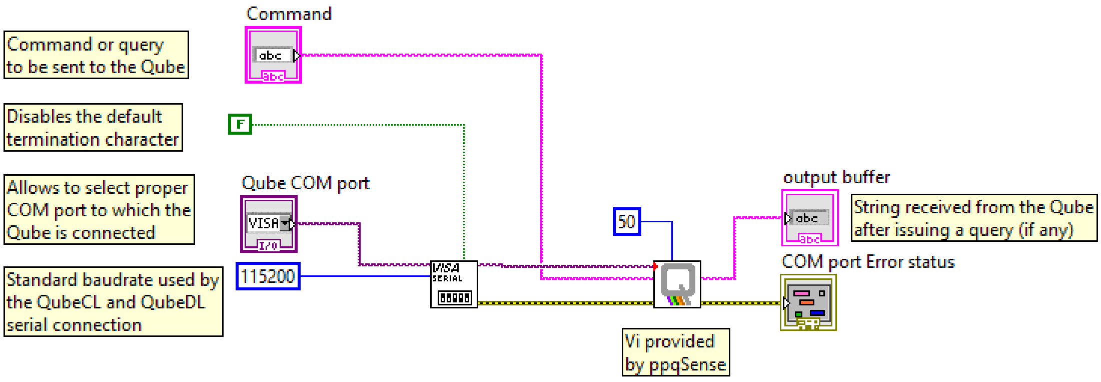

Command
Command or query to be sent to the Qube
abc
Disables the default termination character
F
Allows to select proper COM port to which the Qube is connected
Qube COM port
VISA
Standard baudrate used by the QubeCL and QubeDL serial connection
115200
VISA SERIAL
50
output buffer
abc
String received from the Qube after issuing a query (if any)
COM port Error status
Vi provided by ppqSense

Figure 1: A simple LabVIEW program using the Qube_Serial_Interface.vi to communicate with a Qube

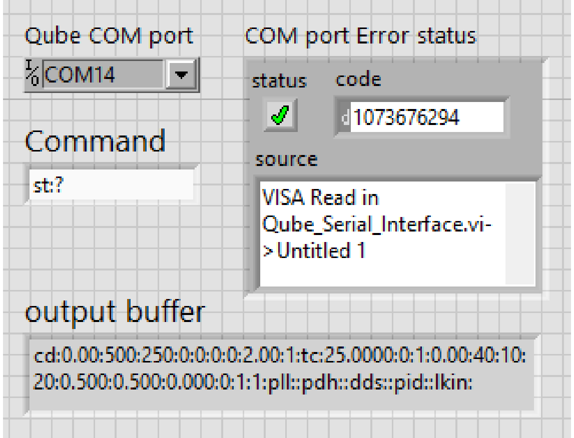
The Qube_Serial_Interface.vi needs a few inputs:

- **Serial Port:** The serial port to which the Qube is connected, properly set up to work at 115200 baud without any termination character.
- **VISA Error:** The error for the VISA COM port made available by LabVIEW
- **Command:** A string containing the command with the **[identifier]:[value]** format to be sent to the Qube

- **Milliseconds to wait:** A constant defining how much time (in milliseconds) it is necessary to wait before accessing the Serial Buffer to read the reply received from the Qube (if any).

The Qube_Serial_Interface.vi add the necessary terminating character in order for the command to be correctly executed by the Qube and gives back two outputs that can be used inside the program:

- **Output buffer:** A string containing the reply from the Qube. It is empty for write commands or if the query gave no reply.
- **COM port Error status:** A VISA COM port error variable useful to check if anything went wrong during the communication with the Qube.



Qube COM port: COM14

Command: st:?

COM port Error status

status: ✓

code: 1073676294

source: VISA Read in Qube_Serial_Interface.vi->Untitled 1

output buffer:

```
cd:0.00:500:250:0:0:0:0:2.00:1:tc:25.0000:0:1:0.00:40:10:
20:0.500:0.500:0.000:0:1:1:pll::pdh::dds::pid::lkin:
```

Figure 2: The panel screen of the same LabVIEW program used to send an **st:?** query to a Qube, showing the reply it sent back

3.2 Python

Python gives all the necessary tools to seamlessly connect to a Qube, sending commands and receiving replies to issued queries. As for the previous example, after a proper set up of the Serial connection, the communications is easy to implement.
The example below shows a very simple function (*write_read_from_Qube*); it takes the command, with the **[identifier]:[value]** format, adds the necessary terminating character to it and send it to the Qube over a predefined Serial Port.
The example also contains a simple script that uses the *write_read_from_Qube* function to perform a couple operations on the Qube.

```
import serial
import time

Qube_COM_num = "COM18" #replace with the actual COM on your PC
```

```
Qube_standard_Baudrate = 115200   #do not change!

# initiate the serial connection
Qube_Serial = serial.Serial(port = Qube_COM_num, baudrate =
    Qube_standard_Baudrate, timeout=0.6)



# Basic fucntion to send commands to the Qube and receive reply to
    queries, if any
def write_read_from_Qube(command):
    Qube_Serial.write(bytes(command + '\n', 'utf-8')) # adds the proper
        terminating character
    time.sleep(0.05)      # wait for 50 ms
    reply = Qube_Serial.readline().decode('utf8')
    return reply

########################################################################
# Here a small demonstrative script to assess if a Qube is connected and
    working

print("Testing the Serial Connection to the Qube")
print("The Qube is connected to the Serial Port " + Qube_COM_num)
print("The connection Baudrate is " + str(Qube_standard_Baudrate) + "\n")

print("Issuing the command id:?")
Serial_number = write_read_from_Qube("id:?")
print("The Serial Number of the Qube is: " + Serial_number)

print("Checking the current status of the Qube by issuing the command
    st:?\n")
Qube_status = write_read_from_Qube("st:?")
print("The Status of the Qube is: " + Qube_status)

print("Setting the output current at 157 mA\n")
write_read_from_Qube("iset:157")


print("Checking if the setpoint for the output current has been received
    by issuing the command st:?\n")
Qube_status = write_read_from_Qube("st:?")
print("The Status of the Qube is: " + Qube_status)
```

Once the proper COM port to use has been set, in this case COM18, it is possible to run the script and obtain a few information from the Qube, as shown below:

```
Testing the Serial Connection to the Qube
The Qube is connected to the Serial Port COM18
The connection Baudrate is 115200

Issuing the command  id:?
The Serial Number of the Qube is: QubeCL-185


Checking the current status of the Qube by issuing the command st:?

The Status of the Qube is:
    cd:810.03:900:2000:0:0:0:0:2.00:1:tc:5.0000:0:1:3.00:100:25:-10:
       0.500:0.221:0.000:1:1:1:pll::pdh::dds::pid::lkin:


Setting the output current at 157 mA

Checking if the setpoint for the output current has been received by
    issuing the command st:?

The Status of the Qube is:
    cd:157.00:900:2000:0:0:0:0:2.00:1:tc:5.0000:0:1:3.00:100:25:-10:0.500:
       0.221:0.000:1:1:1:pll::pdh::dds::pid::lkin:
```

Revision History

Revision	Date	Author(s)	Description
1.0	2025.04.11	LM	First redaction.
1.1	2025.05.29	LM	Available command tables updated.
1.2	2025.06.06	LM	Available command tables updated. Corrected typos in the Python example code.