APPLICATION NOTE

C181055 CPSC1 – PYTHON DEMO SCRIPTS

Driven by innovation

Important note: *any Graphical User Interface (GUI), script or programming code is provided 'As Is' without any express or implied warranty of any kind and is provided only to assist users getting acquainted with the CPSC1 controller, available commands and parameters and methods to communicate with the controller to drive positioners, actuators or stages.*

Please understand that all scripts and code are simply demo's and therefore are not thoroughly tested and may still contain bugs or have features missing.

JPE encourages users to develop their own higher-level control / user software.

1. INTRODUCTION

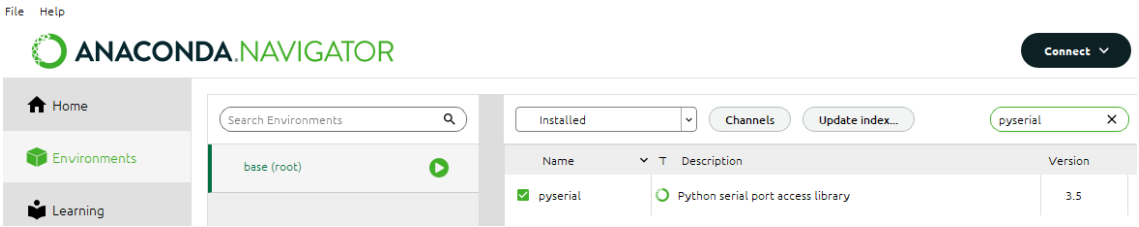
Demo scripts written in the Python programming language are available for download on the JPE website. These programs can be used for simple interaction with the CPSC1 controller and the supported modules installed (CADM RSM, OEM etc) to control positioners, actuators or stages.

2. REQUIREMENTS

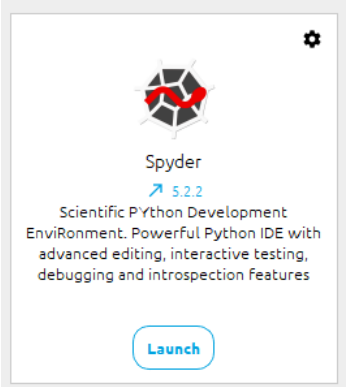
All code is written using Python version 3.9.

All demo scripts communicate with the CPSC1 (only) via the USB Virtual COM port interface. Therefor it is also required to connect the CPSC to the host device via USB and have the **pyserial** package installed. GUI's have been created using the **tkinter** package.

Tip: *install the Anaconda distribution. This Python development platform includes all required packages and modules. If necessary, install the* `pyserial` *package using the Anaconda Navigator:*



After installing Anaconda, start the ***Spyder*** *IDE and navigate to the folder containing the Python demo scripts. Use Spyder to run/start any of the scripts.*



Make sure to run Anaconda and Spyder using administrator rights/credentials.

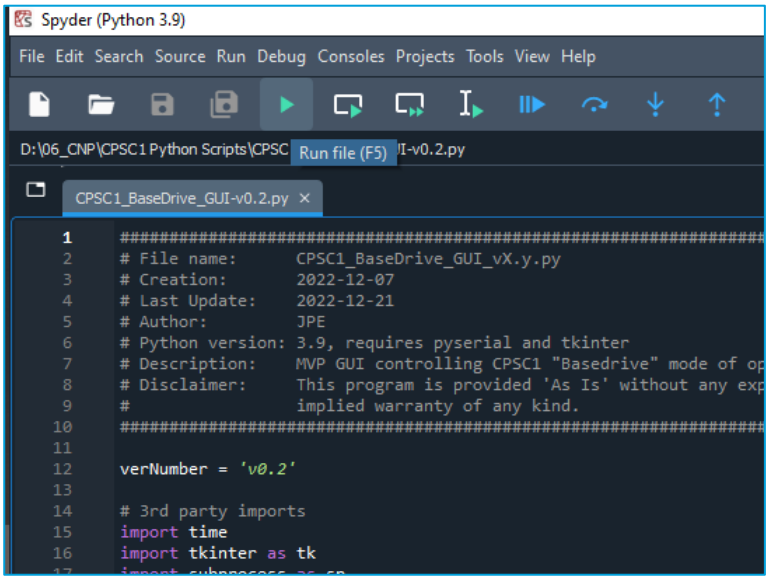
3. THE BASICS

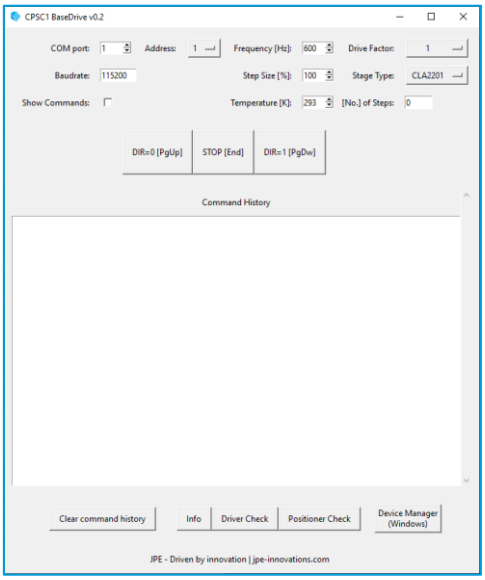
To get started simply download the zip file and unpack the contents to a folder of choice.

Important note: *inside this folder there's a (sub)folder called* `\CpscInterfaces\`*. Inside there is a Python file called* `CpscSerialInterface.py`*. Each demo script uses this file to communicate with the CPSC1, so do not change the name of this file or the name of the folder containing this file.*

3.1 Running a script

Using the Spyder IDE, double-click on a `.py` file and press the green 'Play' button to start the script. If the script contains a GUI, this should pop-up on the screen. Close a GUI using the "X"-button on the top-right of the GUI.





JPE
+31 (0) 43 3585 777
www.jpe.nl
Aziëlaan 12
6199 AG Maastricht-Airport
The Netherlands

BANK: ING Bank NV
IBAN: NL 93 INGB 0679351523
BIC: INGBNL2A
TRADE REGISTER: 14081561
VAT: NL 8134.92.841.B01

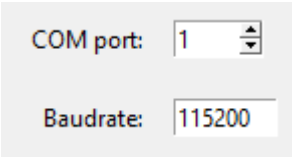
If a script does not contain a GUI, any feedback will be "printed" to the console (IPython console in the Spyder IDE). Also understand that if a script does not contain a GUI, parameters must be set inside the script code prior to running the script. This is typically done near the top of the script:

```
17  ##############################################
18  ### SET PARAMETERS BASED ON CONNECTED DEVICE ###
19  ##############################################
20
21  usbVcpSerialPort = 'COM18'
22  usbVcpBaudrate = 115200
23  duration = 10              # [Duration] of log/check movement in [sec] (default = 100)
24  oemAddr = '4'              # [Address] of OEM module, default set to 4 (=Slot4)
25  oemCh = '1'                # [Channel] of OEM module, default set to 1 (=Channel1)
26
27  ##########################################
28  ### DO NOT CHANGE SCRIPT BELOW THIS LINE ###
29  ##########################################
30
31  passedTime = 0
32  pollDelay = 0
33  startTime = time.time()
```

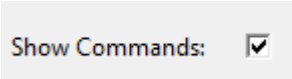
3.2 General tips and tricks

Most (GUI-)scripts should be straight forward to use, but this section contains a few tips and tricks to get up to speed more quickly.

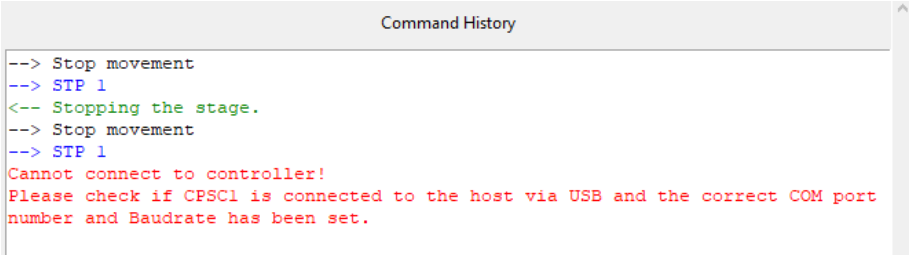
__Tip:__ each GUI contains a COM port selection box and a Baud Rate input field. Set accordingly.



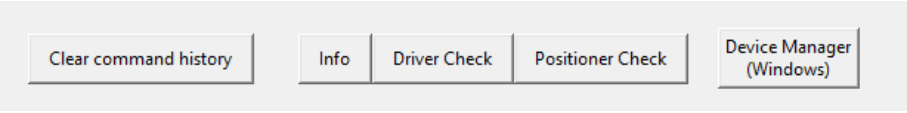
__Tip:__ check the Send Commands checkbox to display the actual commands send to the controller when pushing a button. Together with the Software User Manual this is a quick way to get an understanding on how to use the available software commands.

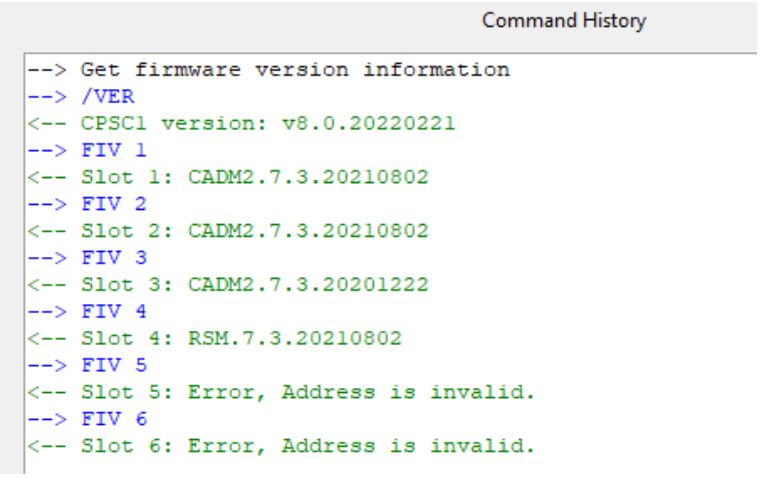


__Tip:__ the Command History window shows commands to the controller (if the Show Commands checkbox is set) and responses from the controller. Internal script messages are displayed in black text, commands to the controller in blue and responses in green. Any errors will be displayed in red:

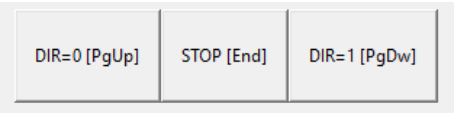


__Tip:__ some GUI scripts have additional buttons to display controller (firmware) Information (Info button) or to perform some checks (Driver Check, Positioner Check).





__Tip:__ some buttons have a keyboard shortcut. This way you can quickly start or stop a movement using a keystroke. The corresponding keyboard shortcut is displayed between []-brackets.



4. SERIAL COMMUNICATION DETAILS

Most demo scripts contain a GUI created with the `tkinter` package and most code in such a script purely serves to create a GUI, setup the buttons and user inputs etc. Basically, only the following is required to send commands to and receive messages from the controller:

Import CpscSerialInterface.py:

```
from CpscInterfaces import CpscSerialInterface as CpscSerial
```

And execute the function:

```
with CpscSerial.CpscSerialInterface([COMx], [BaudRate]) as usbVcp:
    response = usbVcp.WriteRead([command], [termination])
```

The WriteRead function sends a `[command]` (formatted as string) via the USB Virtual COM port to the controller. Any return message from the controller will be stored in `response`. If `[termination]` is set to `1`, the function will add a "`\r\n`" after each `[command]` and will strip "`\r\n`" from any received message. `[termination]` set to 0 (zero) won't do this and it'll be up to the user to add/remove these string termination characters. By default `[termination]` is set to 1 in each demo script. `[COMx]` and `[BaudRate]` should be formatted as strings, for example: `COM1` and `115200`.

Good practice is to use a `try - except` when sending a command, so that the script won't crash immediately if the Virtual COM port cannot be opened/used. For example:

```
try:
    with CpscSerial.CpscSerialInterface('COM1', '115200') as usbVcp:
        response = usbVcp.WriteRead(cmdStp, 1)
except IOError:
    print('Communication error')
```