Digital Pump XP3000 User Interface

Nikolaos L. Koukis

June 22, 2014

Contents

List of Figures

1 Getting Started

1.1 About the software

For the development of the GUI, the PySide[1] framework was used. For the editing part Vim[2] was used. Please refer to section 5 for license information regarding the source code and the redistribution policy.

2 Hardware Setup

2.1 Equipment Used

For the hardware communication to be achieved the following equipment is suggested:

- USB->Serial (RS-232) adapter
- RS-232->RS-485 converter
- Power cables (Power + GND)
- Jumper wires (at least 2 Female to Male)
- Breadboard (optional)

2.2 Connecting the XP3000

The serial communication with the pump was implemented using the RS-485 serial protocol. Here is a sample configuration that was used:

[1] `http://qt-project.org/wiki/pyside`
[2] `http://www.vim.org`

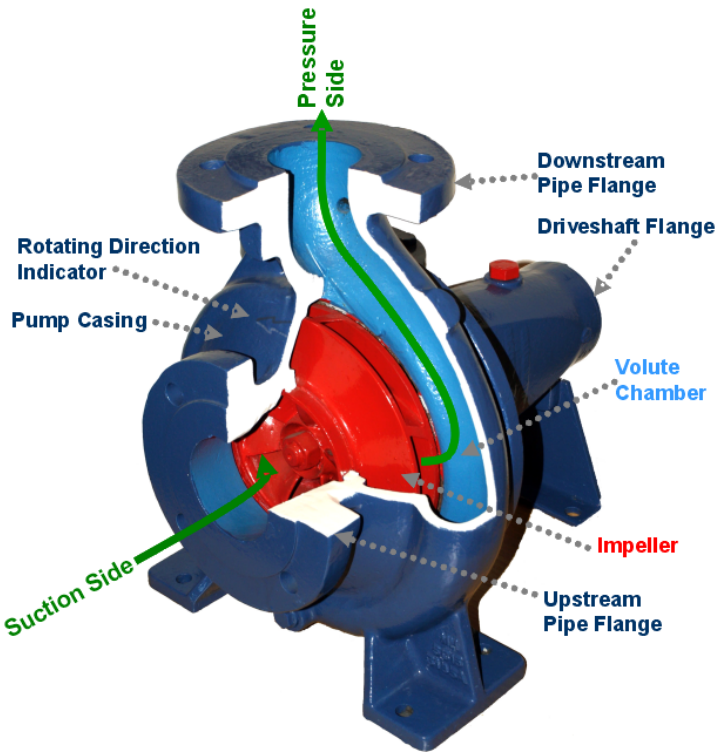

Figure 1: RS-485 - Typical configuration

The needed steps to achive a similar connection would be the following:

- First you have to supply the pump with the necessary power. As described in the manual the XP3000 requires a *24VDC power supply* with a current rating of *at least 1.5A*. As seen in figure 1, during the development part, the power was supplied indirectly to the pump first by running the supply cables into a breadboard and then using extra jumpers to connect to pump pins #1 (power) and #9(GND)
- For the serial communication, you need to connect the RS-485A, RS-485B signals from the pump PCB into the RS-485+, RS-485- of the data terminal (PC) used for the communication. Take exta notice when it comes to connecting the jumpers from the serial adapter to the pump PCB corresponding pins

To sum it up, here is the typical RS-485 pinout for the pump, as it is described above [3]:

RS-485 pin configuration		
Role	Pin No.	Signal to connect
Power	#1 [Power] #9 [GND]	24VDC Ground(of the power supply)
Protocol Signals	#11 [RS485B] #12 [RS-485B]	R-S485+ RS-485-

[3]for a detailed breakdown of the pin positions and roles, consult the pump manual

3 Software Setup

3.1 Getting the Software

There are two ways of acquiring and running the programme, You can either download directly the executable, extract and run the window_controller.exe , or you can download the source code and run it using any python distribution.

Running the .exe file
This option is pretty straightforward. A zipped version of the executable can be downloaded from: a.After the download completes, extract it with the tool of your choice and run it

Running from source
In order to directly run the python source you need to have the following modules in your *PYTHONPATH*:

- PySide (tested on PySide 1.2.2)
- serial
- signal
- Queue
- threading

For *nix users, downloading the needed modules is pretty straightfoward (either from source or using pip). For windows users, getting a full python distribution is recommended (i.e Enthought-Canopy, Anaconda) which are shipped with most of the above modules already installed and ready

3.2 User Interface

The first thing you have to do when you try to connect to the pump is to discover the computer port it is running on For *Windows users* this can be done in the following way:

1. Start Menu, right click on the 'My Computer' button and select 'Manage'
2. Click on the 'Device Manager' tab
3. Click on the 'Ports (COM & LPT)' tab and then select the port your adapter is on

For *nix users (Linux, MacOs etc.) this can be done in the following way:

1. Open the terminal application
2. Issue the following commands:

```
cd /dev
ls -lart |grep tty
```

This will give you a list of the available ports. Run the second command after ejecting and inserting again the serial adapter so that you can discover the needed port

3. Click on the 'Ports (COM & LPT)' tab and then select the port your adapter is on

Make sure that *you have selected the correct port*, otherwise the pump will not respond and will not raise any error Message

After running the software you will be prompted with the *New Device Dialog*. Select the port on which the pump is connected. You are now in the *Main Window Screen*. From here you can command a volume delivery, change the speed of the plunger, or issue a quick command to the pump *push al the fluid, full the syringe etc.* You can also select the editor's tab in order to issue a series of commands to the pump and either run it or save it for later use. A typical view of the main window is the following, running on a MacOs 10.9 environment: From the main window screen you can navigate to a series of other dialog windows

4 Contact Information

For information, bug reports, or just to drop a comment about the software, contact using one of the methods below:

by email: nickkouk@gmail.com
by phone: +30 210 7721516

Mailing Address:
Department of Mechanical Engineering, Ktirio M
National Technical University of Athens
Heroon Polytechniou 9
15780 Zografou, GREECE

5 Appendix A: Software license

The Software is licensed under the LGPL v2.1[4]. The source code as well as the documentation code – written in Latex – is freely available for download and modification from the following github page: `https://github.com/bergercookie`. Everyone interested, is encouraged to contribute in the developement of the program either by suggesting changes to the UI or by working on the open issues[5]of the software

6 Appendix B: Pump Commands

The commands below consist the main way for communicating with the pump. They can be given directly by the user from the Editor Tab. The user can also issue common python commands according to his own needs.

6.1 pump.send_Command(self, command, bits_on_return = 0)

Arguements

[4] `http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html`
[5] This should be the link to that!

- command -> python string
- bits_on_return -> integer [default value = 0]

Description
Sending Commands to the pump. Consult the manual for a detailed list of the pump commands for the RS-485 Protocol. Using this method you must have already defined the pump address to send to (Via the main window or the Editor tab) and you should issue neither the Run character [R] nor the terminating character [\r]
Examples
pump.send_Command('A3000', 10) -> move the plunger to abs position 3000, read back 10 bytes
pump.send_Command('T') -> Terminate Plunger move in progress

6.2 pump.connect_new(self, port_name)

Description
Connect to the given serial port address. Make sure that the given port address isn't already in use
Examples
pump.connect_new('/dev/tty.-SerialPort1-1') -> Configuring connection on Mac

6.3 pump.initialize_pump(self, output = 'right')

Description
Initialize the default output valve. The default output value, unless indicated differently is the right valve **Examples**
pump.initialize_pump() -> Initializing the pump with default output valve (right)
pump.initialize_pump(output = 'left') -> Initializing the pump with output valve set to left

6.4 pump.valve_command(self, position)

Arguements

- Input_Valve: 'I'
- Output_Valve: 'O'
- Bypass_Valve: 'B'

Description
Set the valve to certain position
Examples
pump.valve_command('B') -> Set the valve to Bypass position

6.5 pump.property_set(self, a_property, value)

Arguements

Property_str,	[Command,[Setting_Limits]]
'speed':	['S', [1, 40]]
'backlash':	['K', [0, 31]]
'slope':	['L', [1, 20]]
'start_velocity':	['v', [50, 1000]]
'top_velocity':	['V', [5, 5800]]
'cutoff_velocity':	['c', [50, 2700]]
'cutoff_velocity_steps':	['C', [0, 25]]

Description
The user can specifically set a property of the pump to a specific value
Examples
pump.property_set('speed', '10') -> Set the plunger speed to 10
pump.property_set('backlash', '15') -> set the backlash steps to 15

6.6 pump.volume_command(self, direction = 'P', vol = '0' [in microlt],special=None)

Arguements

- special = 'push_all'
- special = 'pull_all'

Description
Volume Pushing / Drawing mechanism. direction -> python_string
vol -> python_string [!]
The user can issue a special action as well.
Examples
pump.volume_command(direction = 'D', vol = '5') -> Dispense 5 microlitres
pump.volume_command(special = 'push_all') -> Dispense fluid volume

6.7 pump.ser.write(string_to_pump)

Description
This command should be used when a direct serial command has to be sent to the pump. User must issue the pump to which he is addressing to as well as the terminating character
Syntax
pump.ser.write(address + command + character_to_send + termination_char)
Examples
pump.ser.write('/2ZR$\overset{\circ}{}$') -> Initialize the pump with the address *1*