## OEM instruments - Pressure Controller

# Communication Protocol

## Introduction

This document provides the information needed to communicate with the OEM Pressure Controller board through direct UART communication.

## Document status & Revision history

| Version | Author | Release date | Comments |
|---|---|---|---|
| V1.0.0 | Camille | 12/04/2024 | launch version |

# Serial connection settings

Baud rate: 230400

Data bits: 8

Stop bit: 1

Parity: none

Termination character: '\n'

# Syntax

## Command syntax

char 0: '<' to start the query

char 1 to 5: command name

char 6: '?' to read, '!' to write

then ':' to start a value. Can iterate over many arguments

## Error handling

In an answer, after the read/write character, '|xx|' with xx 2 hexadecimal numbers are also sent and indicating the error code associated with the request. '00' means non error. The following error codes are:

| Error code | Meaning |
|---|---|
| 00 | No error |
| C0 | Channel error: wrong channel requested |
| L0 | Locking error: you do not have writing access to this parameter |
| I0 | Impossible command: this query can not be processed |
| P0 | Pause error: this command can not be processed while pause is set to 1 |
| NS | No sensor connected to this channel |
| B0 | Argument value out of bound |

# Quickstart

A Pressure Controller module contains two main parts : a regulator unit capable of controlling an output pressure and a sensing communication head to plug sensors.

## Pressure control

The Pressure Controller internal regulator is a high-speed, high-quality pressure controller. It uses an input pressure to regulate an output pressure according to the user's request. The PRESS command is essential for controlling and measuring pressure, and it exclusively uses mbar values for writing and reading pressure values.

At startup, the pressure target is initially set to 0.

Calibration is typically unnecessary due to the high-quality calibration procedure conducted during device production. However, if you notice significant discrepancies between the pressure command and the physical output, it is crucial to carefully check the input pressure and inspect the tubing for any leaks. If a difference persists, you can adjust the command in the software using a linear calibration done with your own high precision pressure sensor, and contact in the meantime customer support for assistance.

## Sensing

A Pressure Controller sensing unit features a hybrid module that allows for direct analog reading from analog sensors, and includes an I2C head for communication with digital sensors.

## Sensor control & PI regulation

Elveflow digital sensors are automatically detected and read by the device. It’s possible to apply offset, slope and quadratic coefficients to the values measured by the sensor using SENCA command to set these coefficients.

Elveflow analog sensors can be utilized with the device as well. This is done by initially setting the correct sensor type using the SENSO command, in accordance with the sensor type correspondence table. This table matches analog sensors with their corresponding integer types for use in the SENSO command.

The module facilitates PI regulation of the pressure it generates.. This regulation can either be feedback looped to the module's own sensor or configured to 'listen' to a sensor from another module (such as Sensor Hub or another Pressure Controller). For instructions on setting up a remote feedback loop, refer to the motherboard documentation.

Example process to determine P and I values :
1. Manually find the max sensor value corresponding to max pressure value, if max pressure value leads to overflow of sensor or doesnt reach your target value for the PI control, adapt the microfluidic set up (increase of decrease set up's microfluidic resistance by adding or removing microfluidic resistance tubing).
2. Once you have a satisfying max pressure - max measurable flow rate correspondence, you can start by using I=0 and trying to find the best P possible (ie P value leading to best control)

3. Once you found a satisfying value, keep it and now work on the I value the same way
4. You can refine P with the fixed I value from previous step
5. Iterate between step 3 and 4 until you're satisfied

Once you find P and I parameters, they should be valid for every target you set within the range defined by step 1, as long as you keep the same microfluidic setup.

Here are behaviors to observe and use for the P and I refining :
- decreasing the P gain reduces the overshoot
- decreasing the P gain reduces oscillations
- decreasing the I gain reduces the overshoot
- decreasing the I gain reduces oscillations

As a general guide, here's also how to perform PI control, for instance, to achieve 500 uL/min :

1. Set pressure limits (applied when PI control is active) to prevent sensor saturation if desired, by using the UART command USRPL. It's important to note that the upper pressure limit should still permit the system to reach the sensor's maximum value.

| | Send | Receive |
|---|---|---|
| **Syntax** | <:USRPL!:0:750 | <USRPL!\|00\|00000.00:00750.00 |
| **Explanation** | 0 = pressure limit inf<br>750 = pressure limit sup | 00000.00 = pressure limit inf<br>00750.00 = pressure limit sup |

2. set the desired pressure/flow rate target.

| | Send | Receive |
|---|---|---|
| **Syntax** | <SENSC!:500 | >SENSC!\|00\|00:00500.00 |
| **Explanation** | Index 0 because there is only one channel on Pressure Controller (ie CF)<br><br>Arguments :<br>float: target value | |

3. set up the P and I values and launch PID control

| | Send | Receive |
|---|---|---|
| **Syntax** | <SETPI!:0**:0.15**:**0.23** | >SETPI!\|00\|00:**00000.15**:**00000.23** |
| **Explanation** | Index 0 because there is only one channel on Pressure Controller (ie CF)<br><br>Arguments :<br>**float: P value to set**<br>**float: I value to set** | |

4. launch PI control using PIRUN

# Custom waveforms

Four custom waveforms (indexed from 1 to 4) are available for pressure or sensor control (i.e., control based on internal pressure sensor values of the regulator or external sensor values).
These custom waveforms are expressed through 6000 value points separated in time by 1 ms. This implies that each custom waveform represents a 6-second period, which is set to repeat continuously.

These custom waveforms are saved in the device's hard memory (EEPROM) and loaded at startup. If you modify a waveform and wish to use it, the device must be restarted.

How to modify a custom waveform :

1) Use the WAVCI! command to set the value of the custom waveform value index you want to modify. Repeat this for as many value indices as necessary.
    - ex : <WAVCI!:1:149:20      sets the 149th value (out of 6000) of the 1st custom waveform to the value 20
2) Use the WAVCE! command to manually launch the saving of the custom waveform in the device's hard memory (EEPROM).
    - ex : <WAVCE!:1      This writes custom waveform 1 in the device's hard memory (EEPROM), ensuring the modification is saved.
3) Restart the device and verify that the correct value has been written by using the WAVCI? command.
    - Example: <WAVCI?:1:149> - This reads the 149th value (out of 6000) of the 1st custom waveform.

How to select a custom waveform to be used for control :

Use the WAVCT! command to set the custom waveform that you want to use for control.

- ex : <WAVCT!:1:150      custom waveform indexed 1 will be used for control with an offset of 150 points
- ex : <WAVCT!:0:0      sets control to static amplitude control (index 0 (first argument) corresponds to default amplitude control)

How to check what custom waveform is in use :

Use WAVCT? to get the custom waveform index currently used

- ex : <WAVCT?      answer : >WAVCT?|00|2:0000:0020.000      custom waveform used is index 2 with an offset of 0 and current pressure/sensor query is 20

# List of commands

<table>
<tr><th>Parameter</th><th>Mandatory arguments</th><th>Arguments</th><th>W</th><th>R</th><th>Number of characters returned</th><th>Example query</th><th>Typical answer</th><th>Note</th></tr>
<tr><td>PINGA</td><td></td><td>float: pressure value of regulator<br>float: sensor value<br>int: sensor type<br>bool : injecting</td><td></td><td>X</td><td>35</td><td><PINGA?</td><td>>PINGA?|00|00325.12:00124.13:04:00</td><td></td></tr>
<tr><td>_IDN_</td><td></td><td>str: device name</td><td></td><td>X</td><td>22</td><td><_IDN_?</td><td>>_IDN_?|00|OEMREGSEN</td><td></td></tr>
<tr><td>PRESS</td><td></td><td>float: pressure target in mBar</td><td>X</td><td>X</td><td>20</td><td><PRESS?<br><PRESS!:364</td><td>>PRESS?|00|00498.98<br>>PRESS!|00|00364.00</td><td></td></tr>
<tr><td>SENSC</td><td></td><td>float: sensor target value</td><td>X</td><td>X</td><td>20</td><td><SENSC?<br><SENSC!:500</td><td>>SENSC?|00|00500.00<br>>SENSC!|00|00500.</td><td>Set or ask sensor value target and start PID</td></tr>
</table>

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | 00 | |
| WAVET | | **int:** waveform type<br>**float:** max value<br>**float:** min value<br>**float:** period in s<br>**int:** phase in degree | X | X | 50 | <WAVET?<br><WAVET!:500:200:100:0 | >WAVET?\|00\|01:00500.00:00200.00:00100.00:00000.00<br>>WAVET!\|00\|01:00500.00:00200.00:00100.00:00000.00 | **Set the specified classic waveform to be used for PID control**<br>**Waveform types:**<br>**00:** basic amplitude control<br>**01:** sine<br>**02:** square<br>**03:** triangle<br>**04:** linear |
| PIRUN | | **bool :** pid runs (switch between pressure control and sensor control)<br>**bool :** pid pause | X | X | 17 | <PIRUN?<br><PIRUN!:1:0 | >PIRUN?\|00\|00:00<br>>PIRUN!\|00\|01:00 | **Get or set PI control (0 for pressure (regulator) or 1 for sensor (external)**<br>**and pause status (pressure output will freeze to last set pressure value when paused, control will resume where it was left of when play)**<br>**/!\ changing PI control automatically reset PI parameters (target & accumulation error)** |
| DEVSN | | **str:** SN | | X | 27 | <DEVSN? | >DEVSN?\|00\|48V111 | |
| FIRMV | | **str:** firmware version | | X | 21 | <FIRMV? | >FIRMV?\|00\|v01.03.01 | |
| RESET | | | | | | <RESET | | **reset firmware** |
| REGTY | | **int:** regulator type | | X | 14 | <REGTY?<br><REGTY!:2 | >REGTY?\|00\|02<br>>REGTY!\|00\|02 | |
| SENSO | **int** : channel | **int:** sensor type | X | X | 14 | <SENSO?:1 | >SENSO?\|00\|01:04 | **Sensor types:** |

<table>
<tr><td></td><td>(only 1)</td><td></td><td></td><td></td><td></td><td><SENSO!:0:21</td><td>>SENSO!|00|01:21</td><td>00: no sensor<br>01: MFSD1<br>02: MFSD2<br>03: MFSD3<br>04: MFSD4<br>05: MFSD5<br>10: Universal<br>21 to 50: analog sensors<br>(W only compatible with analog sensors)</td></tr>
<tr><td>SENRE</td><td>int : channel<br>(only 1)</td><td>int: sensor resolution</td><td>X</td><td>X</td><td>17</td><td><SENRE?:1<br><SENRE!:1:8</td><td>>SENRE?|00|01:04<br>>SENRE!|00|01:8</td><td>Sensor resolution corresponds to the number of bits used to code the sensor value.<br>(see Acquisition time table for sensor resolution for details)<br>compatible only with digital sensors type 1 to 5</td></tr>
<tr><td>REGSN</td><td></td><td>str: regulator SN</td><td></td><td>X</td><td>22</td><td><REGSN?</td><td>>REGSN?|00|XXXXXXXX</td><td></td></tr>
<tr><td>SETPI</td><td></td><td>float: p<br>float: i</td><td>X</td><td>X</td><td>29</td><td><SETPI?<br><SETPI!:11:2.2</td><td>>SETPI?|00|00010.00:00003.00<br>>SETPI!|00|00011.00:00002.20</td><td></td></tr>
<tr><td>LISTN</td><td></td><td>int: channel listened</td><td>X</td><td>X</td><td>26</td><td><LISTN?<br><LISTN!:2</td><td>>LISTN?|00|02:01:00234.01<br>>LISTN!|00|02:01:00234.01</td><td>Listen to another module's sensor</td></tr>
<tr><td>ERLOG</td><td></td><td>float : PI error</td><td>X</td><td>X</td><td>27</td><td><ERLOG?</td><td>>ERLOG?|00|0000</td><td>Get pid error or set pid error to 0</td></tr>
</table>

<table>
<tr><td></td><td></td><td>bool : physical error marker</td><td></td><td></td><td></td><td>————————<br><ERLOG!</td><td>02345.32:00<br>————————<br>>ERLOG!|00|000000000.00:00</td><td>Physical error means the PI control accumulated error is drifting (if 1, automatically pauses PI control and deco/reco sensor)</td></tr>
<tr><td>USRPL</td><td></td><td>float : pressure min allowed during PI sensor control<br>float : pressure max allowed during PI sensor control</td><td>x</td><td>x</td><td>29</td><td><USRPL?<br>————————<br><USRPL!:500:1200</td><td>>USRPL?|00|00500.00:01200.00<br>————————<br>>USRPL!|00|00500.00:01200.00</td><td></td></tr>
<tr><td>WAVCI</td><td>int: waveform index (1 to 4)<br>int: waveform value index (0 to 5999)</td><td>float: waveform value</td><td>X</td><td>X</td><td>28</td><td><WAVCI?:1:149<br>————————<br><WAVCI!:1:149:20</td><td>>WAVCI?|00|01:0149:0020.000<br>————————<br>>WAVCI!|0001:|0149:0020.000</td><td>get or set the value of a specified custom waveform (identified by its index) given index (from 0 to 5999)</td></tr>
<tr><td>WAVCE</td><td>int: waveform index (1 to 4)</td><td></td><td>X</td><td>X</td><td>14</td><td><WAVCE?:1<br>————————<br><WAVCE!:1</td><td>>WAVCE?|00|01<br>————————<br>>WAVCE!|00|01</td><td>read or write from/to the device's EEPROM the specified custom waveform (identified by its index)</td></tr>
<tr><td>WAVCZ</td><td>int: waveform index (1 to 4)</td><td></td><td>X</td><td></td><td>14</td><td><WAVCZ!:1</td><td>>WAVCZ!|00|01</td><td>reset specified custom waveform (identified by its index) values to 0</td></tr>
<tr><td>WAVCT</td><td></td><td>int: custom waveform type<br>int: offset point to start custom waveform</td><td>X</td><td>X</td><td>19</td><td><WAVCT?<br>——————————<br><WAVT!:01:100</td><td>>WAVCT?|00|01:0100<br>————————<br>>WAVCT!|00|01:0100</td><td>Set the specified custom waveform to be used for PID control and immediately starts running the custom waveform</td></tr>
</table>

# Correspondence table for sensor types

| Type id | unit in output | Sensor type |
|---|---|---|
| 0 | | No sensor connected |
| 1 | uL/min | MFS1 digital |
| 2 | uL/min | MFS2 digital |
| 3 | uL/min | MFS3 digital |
| 4 | uL/min | MFS4 digital |
| 5 | uL/min | MFS5 digital |
| 6-20 | | reserved |
| 21 | uL/min | MFS1 analog |
| 22 | uL/min | MFS2 analog |
| 23 | uL/min | MFS analog |
| 24 | uL/min | MFS3 analog |
| 25 | uL/min | MFS4 analog |
| 26 | uL/min | MFS5 analog |
| 27-29 | | reserved |

| | | |
|---|---|---|
| 30 | mbar | MPS0 analog |
| 31 | mbar | MPS1 analog |
| 32 | mbar | MPS2 analog |
| 33 | mbar | MPS3 analog |
| 34 | mbar | MPS4 analog |
| 35 | mbar | MFP |
| 36-39 | | reserved |
| 40 | mV | bubble detector |
| 44 | mV | Custom |

## Acquisition time table for sensor resolution

| Resolution mode | Resolution (bit) | Processing Time (ms) Min. | Typ. (ms) | Max. (ms) |
|---|---|---|---|---|
| 1 | **9** | 0.5 | 0.8 | 0.9 |
| 2 | **10** | 1.0 | 1.3 | 1.5 |
| 3 | **11** | 2.0 | 2.4 | 2.6 |
| 4 | **12** | 4.1 | 4.6 | 4.9 |

| | | | | |
|---|---|---|---|---|
| 5 | **13** | **8.2** | **8.9** | **9.4** |
| 6 | **14** | **16.4** | **17.5** | **18.5** |
| 7 | **15** | **32.8** | **34.8** | **36.7** |
| 8 | **16** | **65.5** | **69.3** | **73.2** |