



# Mitos Fluika Pressure and Vacuum Pumps

## SOFTWARE USER INSTRUCTIONS



# Contents

<b>1.</b>	<b>Introduction</b>	<b>3</b>
<b>2.</b>	<b>Product description</b>	<b>3</b>
<b>3.</b>	<b>Communicating</b>	<b>4</b>
<b>4.</b>	<b>Control commands</b>	<b>4</b>
<b>4.</b>	<b>Using the .NET library</b>	<b>8</b>

## 1. Introduction

This software user guide lists all of the control commands for the Mitos Fluika Low Pressure Pump (Part no. 3200418) and Mitos Fluika Low Vacuum Generator Pump (Part no. 3200419), followed by detailed descriptions, syntax and usage examples of each command.

It also includes detailed information of how to use the .NET library to control the Mitos Fluika Low Pressure Pump and Mitos Fluika Low Vacuum Generator Pumps.

## 2. Product description

These compact pressure and vacuum generators are designed to provide pressurized air or vacuum with fast and accurate pressure control. They are easy to set up and use, requiring no gas supplies or external vacuum pumps. This makes them perfect for OEM integration. Moreover, power and control is supplied through a USB port only.

There are 2 versions:

- Mitos Fluika Low Pressure Pump – for pressures above atmospheric pressure
- Mitos Fluika Low Vacuum Generator Pump – for pressures below atmospheric pressure

Both generators have a single pressure channel/outlet, which is accessed via a 4mm tube push-fit pneumatic connector on the front panel of the device.

The pumps can be used with the **Mitos Fluika Control Valve** for multiplexing four output channels between two input channels, making them ideal for applications that require fast switching between two pressure levels.

**Warning! If you are using the Vacuum Pump, make sure that the substrate reservoir is sufficiently large enough to contain all the liquid for your experiment with a space for the remaining vacuum. It is very important to protect the device from any liquid ingress.**

### 3. Communicating with the Dolomite Fluika Pumps

The USB connection is a virtual COM port so Dolomite Fluika Devices appear as COM ports. In addition hardware parameters such as baud rate, handshaking, etc. are irrelevant in case of the Dolomite Fluika Device.

The fastest initial testing can be done using a terminal program, such as Putty (Open Source program available [www.putty.org](http://www.putty.org)) or HyperTerminal on Windows XP.

In order to open Fluika Device in Putty, choose **Connection Type: Serial**. Type COM port number into serial line (eg. COM10). Click **Open**. In terminal you can use Dolomite Fluika command line syntax. (eg. type "ls", in order to get listing of available commands) see below for a list of control commands.

Note :- USB device parameters are: vendor ID: 04D8 and product ID: F882

### 4. Control Commands for Mitos Fluika Pressure and Vacuum Pumps

All control commands for the pressure and vacuum pumps are listed below followed by detailed descriptions, syntax and usage examples of each command. Each command line is finished by entering (**'r'** or **ASCII: 13**), thereafter the command is executed and depending on the command, it may return a value. Returned values are on one line, separated by '\r' + '\n' symbols. Key parts of returned values are in parenthesis '(...)' and contain two parts: name of return value and then the value itself. These are separated by a colon ':'. The part outside of parenthesis is for verbal explanation.

#### List of commands:

- ls
- sn
- dev
- setp
- getp
- geta
- smrt
- gmax
- gmin
- stp

### Detailed description:

Command	Description	Syntax	Example
<b>ls</b>	Commands list. ( <i>parameters: no</i> ) Displays all possible commands in this device	<b>ls</b>	>ls conf dev
<b>sn</b>	Serial number. ( <i>parameters: no</i> ) Displays 6-digit serial number of the device. Name of return value is 'SN'	<b>sn</b>	>sn (SN:010002) >
<b>dev</b>	Device name. ( <i>parameters: no</i> ) Displays name of the device. Name of return value is 'DEV'	<b>dev</b>	>dev (DEV:PG500) >
<b>setp</b>	Set pressure. ( <i>parameters: value</i> ) Pressure value ( <i>vvvvv</i> ) is positive integer (up to 5 digits). Unit of the pressure is an internal digital unit, linearly related to actual pressure. After pressure has been reached the $t_{reach}$ time will be returned in ms. Name of the return value is 'T'.	<b>setp vvvvv</b>	>setp 3400 SETP > ... (T:500) <i>Set pressure to 3400 [internal unit]. Return value after that it took 500ms.</i>
<b>getp</b>	Get pressure. Returns actually measured pressure value Unit of the pressure is an internal digital unit, linearly related to actual pressure. Name of the return value is 'P'	<b>getp</b>	>getp (P:2530) > <i>Pressure was 2530 [internal unit]</i>
<b>geta</b>	Get average pressure. Similar to getp, but pressure value has been averaged over 64 readings. Name of return value is 'A'	<b>geta</b>	>getp (A:12000) >
<b>smrt</b>	Set maximum response time ( <i>parameter: time in seconds, up to 100</i> ). The maximum response time is the maximum allowed continuous pumping time. If this time has been	<b>smrt ddd</b>	>smrt 10 >

	exceeded, pumping stops and error 3 (ERR:003) will be returned. This is to protect the pump against leak and continuous run, which can eventually shorten its lifetime. Default value: 10		
<b>gmax</b>	<p>Get maximum pressure (<i>parameter: unit</i>) of the device,</p> <p>Unit 'u' can carry two values:</p> <ul style="list-style-type: none"> <li>'p' pressure in mbars <math>p_{max}[mbar]</math></li> <li>'d' pressure in internal digital units <math>p_{max}[d]</math></li> </ul> <p>Name of return value is 'MAX'</p>	<b>gmax u</b>	<p>&gt;gmax p (MAX:500p)</p> <p>&gt;</p> <p>Maximum pressure of the device is 500 [mbar]</p>
<b>gmin</b>	<p>Get minimum pressure (<i>parameter: unit</i>) of the device,</p> <p>Unit 'u' can carry two values:</p> <ul style="list-style-type: none"> <li>'p' pressure in mbars <math>p_{min}[mbar]</math></li> <li>'d' pressure in internal digital units <math>p_{min}[d]</math></li> </ul> <p>Name of return value is 'MAX'</p>	<b>gmin u</b>	<p>&gt;gmin d (MIN:2300d)</p> <p>&gt;</p> <p>Minimum pressure of the device is 2300 [internal unit]</p>
<b>stp</b>	<p>Stop (<i>parameters: stop_level</i>)</p> <p>Parameter can have two values:</p> <ul style="list-style-type: none"> <li>'a' stop all and release pressure</li> </ul> <p>'p' stop pumping, but do not release pressure</p>	<b>stp p</b>	<p>&gt;stp p</p> <p>&gt;</p>

If a mistake has been made and the command has not been recognized, an error message will show: "**Unknown command**".

#### **Device errors:**

- (ERR:001) – reserved
- (ERR:002) – reserved
- (ERR:003) – pumping error or leak. Maximum allowed response time has been exceeded
- (ERR:004) – controller error. Internal error. Controller needs physical restart

### **Control algorithm suggestions**

Each device has a physical operation range  $p_{min}[mbar] \dots p_{max}[mbar]$ , which is limited by software control, so that all the devices in the series have the same performance. However devices do not communicate in mbars, but uses the internal unit range, which relates to the actual pressure on a linear scale. This internal unit range is fixed, but depends on each device calibration. It would be advisable to start communication by requesting these pressure range values and thereafter units can be converted by simple linear interpolation, as shown in the following.

$$p[mbar] = \frac{p_{max}[mbar] - p_{min}[mbar]}{p_{max}[d] - p_{min}[d]} p[d] + p_{min}[mbar]$$

A similar conversion works in the other direction

$$p[d] = \frac{p_{max}[d] - p_{min}[d]}{p_{max}[mbar] - p_{min}[mbar]} p[mbar] + p_{min}[d]$$

Where :-

$p_{min}[d]$

Is the Minimum sensor reading (at 0mbar) in internal units (typically 2800 units)

$p_{max}[d]$

Is the maximum sensor reading (at 500mbar) in internal units (typically 25000 units)

*These values vary from device to device as they are dependent on calibration which is performed for each device during manufacture.*

## 5. Using the .NET library for Mitos Fluika Pressure and Vacuum Pumps

### User interface

For easy and convenient development of your custom control applications, the Fluika .NET library provides graphical user interface components that can be quickly integrated into your program.

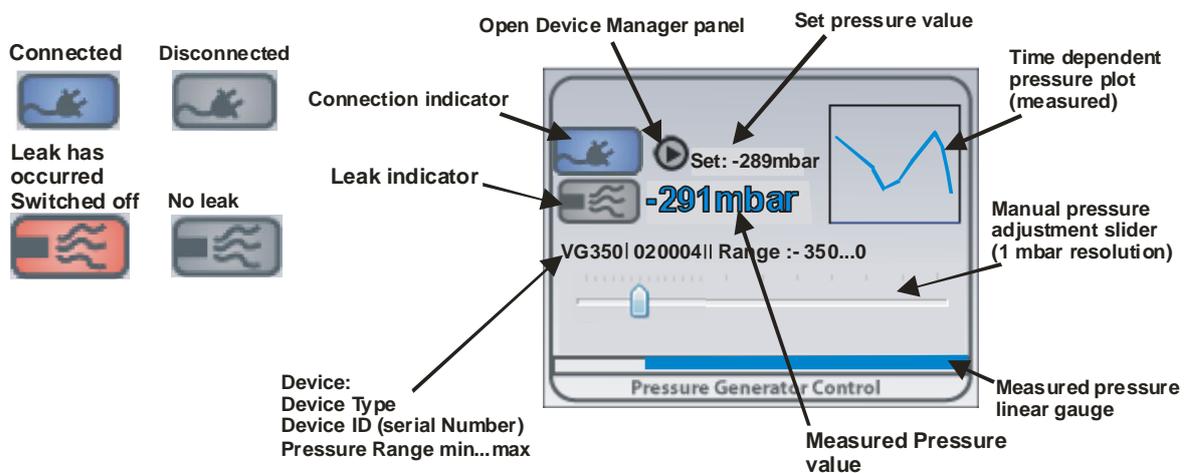
For use in your .NET project, the Fluika .NET library will need to be pasted into your development folder (and distributed with your application). The library contains:

- 'FluikaPG.dll, FluikaDeviceManager.dll'
- 'FDM\_Settings' folder

Add the component '**FluikaPG**' into your toolbox:

Toolbox --> Right mouse click --> "Choose Items ..." from drop down menu -->.NET Framework components -->Browse --> Choose FluikaPG.dll file

Now you can place a graphical control element (type: **FluikaPG.FluikaPGControl**) into your software (Figure 1). Use as many control elements as there are pressure controllers being used in the project. The actual software design can be slightly different depending on whether you use one or multiple controllers at the same time (see later examples).



**Figure 1:** Graphical user element component (size: 215 x 169 pix) containing everything for manual connection and control of the device.

## Programming

Syntax here is given for .NET C++, but the component can also be used with other languages such as C# or Visual Basic. Components and examples have been developed with Visual Studio 2008 and work on both 32- and 64-bit Windows computers.

## Properties

```
int setPressure
```

Sets active pressure in mbar. Resolution is 1 mbar. Vacuum values are with minus sign.

```
String^ searchID
```

Is 6 character long string of device ID (Serial number), which this component shall connect. \* can be used as a wildcard to replace characters, which shall be not considered in search.

```
bool enableAutoConnectByID
```

If automated searching of devices with ID specified in property *searchID* allowed.

## Methods

```
void activate()
```

Activates FLUIKA device system. This function shall be called first before others. It initializes the component.

```
int getPressure()
```

Returns latest measured pressure value in mbar. Resolution is 1 mbar. Vacuum values are with minus sign.

```
String^ getLastError()
```

Returns code and message of the last error. Error codes are

- "[0] No Error" If there is no error
- "[1] Data communication error" If data communication error has occurred
- "[3] Sensor error" If sensor error has occurred (device restart is required)

```
array<array<String^>>^ getDeviceList()
```

Returns the latest list of connected devices. This list is a string array with size 256 x 5. Each of 256 lines represents on COM port counted from 1 to 256. Each line has 5 String values:

- First is by port number (1 to 256) as a string.
- Second port status, which can have 4 different states:
  - "OFFLINE" No device connected to this port.
  - "BUSY" Port could not be opened. Either hardware problem or port is already opened by same or another program.
  - "NOREPLY" Device do not reply as fluika device. Either connected device is not fluika device, or hardware failure has occurred.
  - "FLUIKA" Device is recognized as FLUIKA device.

<ul style="list-style-type: none"> <li>• Third is deviceID (SN: 6-digits), if it is fluika device, otherwise empty string</li> <li>• Fourth is device type (eg. "PG500")</li> <li>• Fifth is device alias name (<i>this is becoming obsolete!</i>)</li> </ul>
<pre>int getResponseTime()</pre> <p>Returns the latest response time. It is updated before the ResponseTimeReceived event is called. If pressure is set either by the function setPressure or by slider than time is counted between setting the value and until it is reached. Unit is ms.</p>
<pre>int previousSetPressure()</pre> <p>Returns the previous set pressure value. Not the current, but one before it.</p>
<pre>void calibrationSettings(array&lt;int&gt;^ value)</pre> <p>Loads a set of calibration constants into device. This feature is currently under development. But will allow in future tuning device performance without changing the firmware.</p>
<pre>void setLeakTime(int value)</pre> <p>Sets leak time. This is maximum continuous allowed pumping time before pumps shuts off. Time is given in seconds. Minimum 1, maximum 100 seconds. Default is 10s. This is to protect pressure generator from continuous empty run, which may shorten its lifetime.</p>
<pre>void deviceOff()</pre> <p>Switches off all pneumatic components in device. It minimizes power consumption and pressure</p> <p>Returns to atmospheric pressure, 0.</p>
<pre>void pumpingOff()</pre> <p>Switches off pumping, but does not release valves. Pressure returns slowly to 0 due to leaks.</p>
<pre>int getMinPressure()</pre> <p>Gets minimum pressure value of currently connected device, value in mbar</p>
<pre>int getMaxPressure()</pre> <p>Gets maximum pressure value of currently connected device, value in mbar</p>
<pre>String^ deviceID()</pre> <p>Gets device ID (Serial number) of currently connected device</p>
<pre>String^ deviceType()</pre> <p>Gets device type of currently connected device. Eg. "PG500" or "VG350"</p>
<pre>String^ devicePort()</pre> <p>Gets COM port of currently connected device</p>
<pre>bool isConnected()</pre> <p>Is device connected currently</p>
<pre>void updateDeviceList()</pre> <p>Gets device ID (Serial number) of currently connected device</p>
<pre>void connectDevice(int portNr)</pre>

Connect to a device with following COM port number
<code>void disconnectDevice ()</code>
Disconnect device

## Events

BecameConnected
This event is called when device becomes connected
BecameDisconnected
This event is called when device becomes disconnected, either by programmed disconnection or by physical loss of connection (eg. cable was pulled out)
PressureChanged
This event is called when set pressure value has been changed, either through code or user interface
PressureReadingUpdated
This event is called when read pressure value has been updated
ErrorOccured
This event is called when error occurred
LeakOccured
This event is called when leak occurred. When leak occurs, pump is switching off, until pressure value is set again
UpdatedDeviceList
This event is called when device list becomes updated
ResponseTimeReceived
This event is called when set pressure value has been reached and response time has been determined

## **Example**

The table below describes a simple example with one pressure controller. Place the control component (**fluikaPGControl1**) on your form (**Form1**). Take form Load event and insert the following code:

```
private: System::Void Form1_Load(System::Object^ sender,
System::EventArgs^ e)
{
    fluikaPGControl1->activate(); //Activate component
    fluikaPGControl1->searchID="*****"; //Connect to any device
    fluikaPGControl1->enableAutoConnectByID=true; //Allow
    autoconnect
}
```

This program will automatically connect to any pressure controller that is connected to the computer. You can also connect the device physically to a USB port and then disconnect and connect again. The program will automatically detect the presence of the device and establish a connection with it.

If you have more than one device in your project, you will need to use the 'FluikaInstrument' component to manage connectivity of all devices. If this component is not used, there is risk that multiple components will access the same ports simultaneously, which can lead to malfunction.



**The Dolomite Centre Ltd.**

Unit 1, Anglian Business Park, Royston,  
Hertfordshire, SG8 5TW, United Kingdom

**T:** +44 (0)1763 242491

**F:** +44 (0)1763 246125

**E:** [info@dolomite-microfluidics.com](mailto:info@dolomite-microfluidics.com)

**W:** [www.dolomite-microfluidics.com](http://www.dolomite-microfluidics.com)

**Dolomite Microfluidics**

29 Albion Place  
Charlestown, MA 02129

**F:** 617 848 1211

**F:** 617 500 0136

**E:** [salesus@dolomite-microfluidics.com](mailto:salesus@dolomite-microfluidics.com)

**W:** [www.dolomite-microfluidics.com](http://www.dolomite-microfluidics.com)