

Elveflow User Guide

SDK SOFTWARE DEVELOPMENT KIT

DOCUMENT REF: UGSDK-LABVIEW-250429



LabVIEW™

LABVIEW GUIDE

Symbols used in this document



Important information. Disregarding this information could increase the risk of damage to the equipment, or the risk of personal injuries.



Helpful information. This information will facilitate the use of the instrument and/or contribute to its optimal performance.



Additional information available on the internet or from your Elveflow representative.

READ THIS MANUAL CAREFULLY BEFORE USING THE SOFTWARE



This manual must be read by every person who is or will be responsible for using the Elveflow software development kit (SDK).

Due to the continual development of the products, the content of this manual may not correspond to the new software. Therefore, we retain the right to make adjustments without prior notification.

Important SDK safety notices:

1. The SDK gives the user complete control over Elveflow products. Beware of pressure limits for containers, chips and other parts of your setup. They might be damaged if the pressure applied is too high.
2. Use a computer with enough power to avoid software freezing.

If these conditions are not RESPECTED, the user is exposed to dangerous situations and the instrument can undergo permanent damage. Elvesys and its partners cannot be held responsible for any damage related to the misuse of the instruments.

Table of contents

Getting started	4
Before starting	5
Important remarks	5
LabVIEW's SDK programming	6
OB1	6
OB1_init.vi	6
OB1_Add_Sensor.vi	7
OB1_Calibration.vi	7
Calibration_Save.vi	7
Calibration_Load.vi	7
OB1_Set_Pressure.vi	7
OB1_Set_Sensor.vi	7
OB1_Get_Channel.vi	7
OB1_Get_All_Channel.vi	7
OB1_Set_Trigger.vi	8
OB1_Get_Trigger.vi	8
OB1_Close.vi	8
OB1_Reset.vi	8
OB1_Reset_Sensor.vi	8
MSRD	9
M_S_R_D_Init.vi	9
M_S_R_D_Add_Sensor	9
M_S_R_D_Get_Data.vi	9
M_S_R_D_Set_Filters.vi	9
M_S_R_D_Set_Trigger.vi	9
M_S_R_D_Get_Trigger.vi	9
M_S_R_D_Close.vi	10
M_S_R_D_Reset.vi	10
M_S_R_D_Reset_Sensor.vi	10
BFS	11
BFS_Init.vi	11
BFS_Get_Density_val.vi	11
BFS_Get_Flow_val.vi	11

BFS_Get_Temperature_val.vi	11
BFS_Set_Filter_val.vi	11
BFS_Do_Zeroing.vi	11
BFS_Close.vi	11
MUX D-R-I (DISTRIBUTION, DISTRIBUTOR, RECIRCULATION or INJECTION)	11
MUX_D-R-I_Init.vi	12
MUX_D-R-I_SetValve.vi	12
MUX_D-R-I_GetValve.vi	12
MUX_D-R-I_SendCommand.vi	12
MUX_D-R-I_Close.vi	12
Other MUX Series	13
Common to all MUX Series	13
MUX WIRE	13
MUX FLOW SWITCH	13
MUX CROSS CHIP	13
Remote PID	14
PID_Add.vi	16
PID_Set_Running.vi	16
PID_Set_Params.vi	16
Quick start examples	17
Quick start	17
Remote mode	19
Appendix	21
Error handling:	21

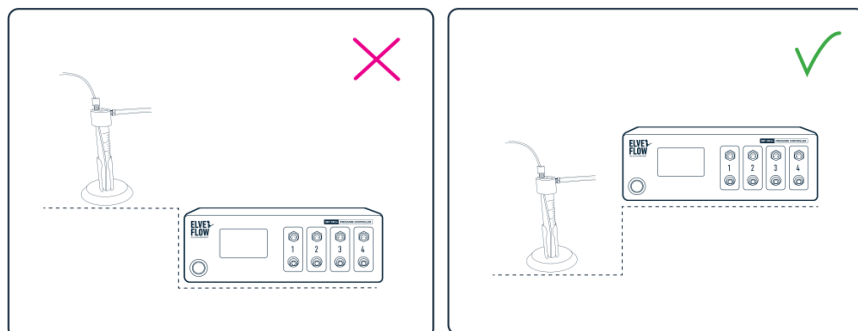
Getting started

Elveflow proposes a standard development kit for LabVIEW, C++, Python and MATLAB

The following sections will guide you through the steps to add a new instrument or sensor, explore its basic and advanced features and use it with other instruments to automate your experiment in LabVIEW.

Before starting

To prevent backflow in the pressure regulator, always place liquid reservoirs under the instrument



Important remarks



For all programming languages:

- If MUX Distribution/Distributor/Recirculation/Injection or BFS are used, FTDI drivers are required (<http://www.ftdichip.com/Drivers/D2XX.htm>). You can find these drivers in the same folder the ESI is installed. Default location would be C:\Program Files (x86)\Elvesys\driver (look for driver_MUX_distAndBFS.exe).
- Do not simultaneously use the ESI software and the SDK, some conflict would occur.



If you would like our expert advice on a particular section of your code, please be sure to give us some details about your issue.

Such as

- the SDK function you're using: e.g. OB1_init.vi
- the version of the SDK are you using: e.g. V3.08.01
- your Windows version: Windows 7 32-bit, or Windows 10 64-bit ?
- the environment you're using e.g. e.g. LabVIEW dev 2018.
- the elveflow device(s) you will use this code with (e.g. OB14000452042983 with a MUX-D-42989).
- etc... (add any relevant information you may have).

This will help minimize the turnaround time for understanding what your issue is and what the general fix would look like.

LabVIEW's SDK programming

OB1

All the available VI for the programming of a customized LabVIEW program are used in the VI “_OB1_Example.vi” contained in the LLB library “ElveflowLLB.llb”.

The structure of the main VI you would develop including Elveflow instruments should follow the same workflow as represented in the following figure. Using this workflow, you will start with a **configuration** and a **calibration** before starting to operate the OB1 and its connected sensors. Then, you can perform your instrumentation using the functions represented in the “**main working loop**”.

After finishing, please remember to close the OB1 reference using OB1_Close.vi

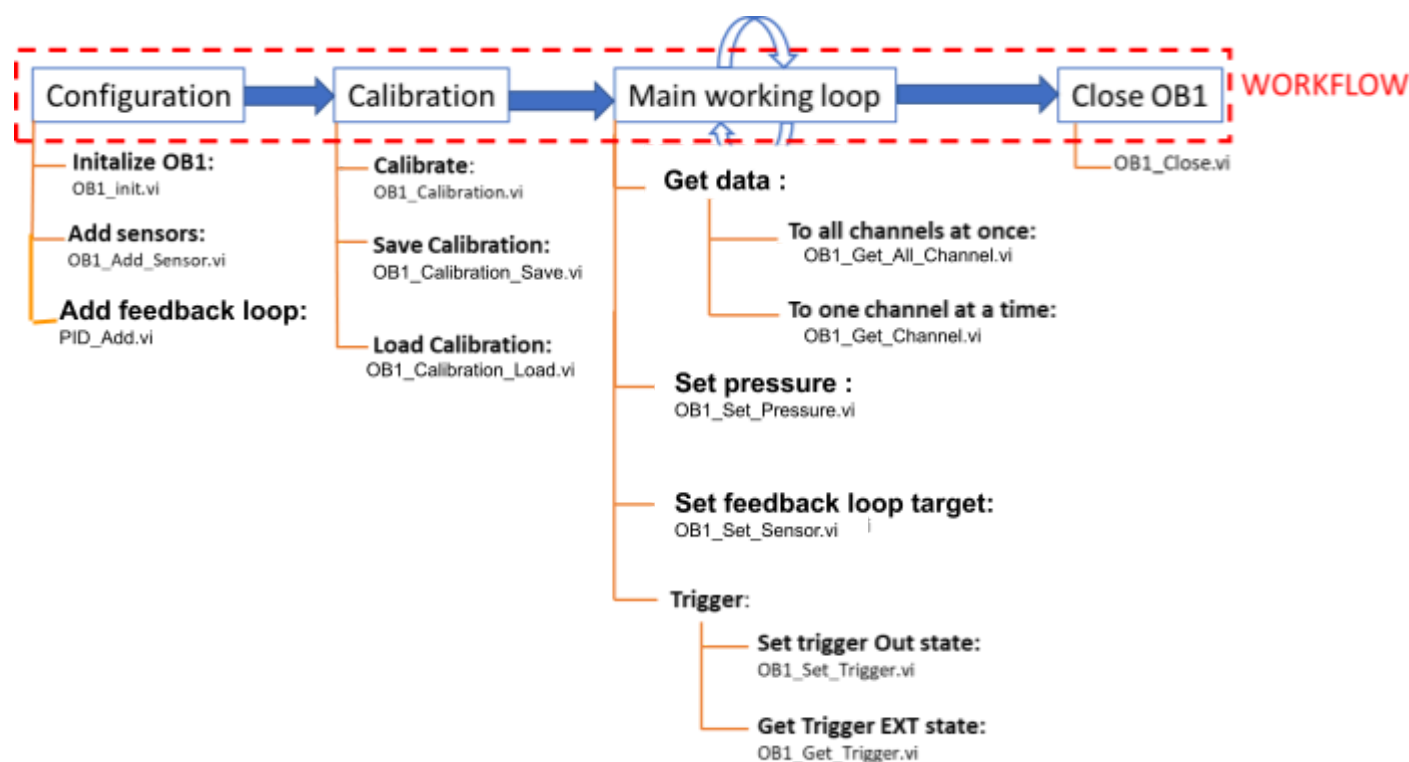
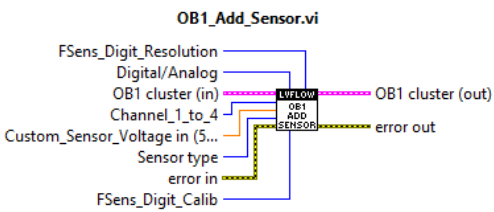





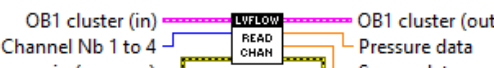
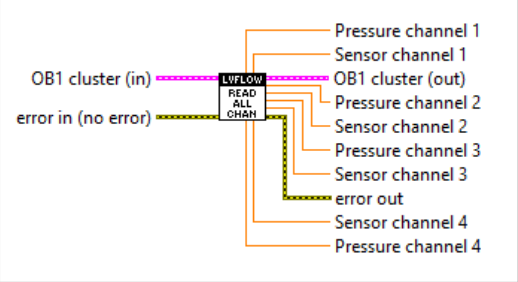
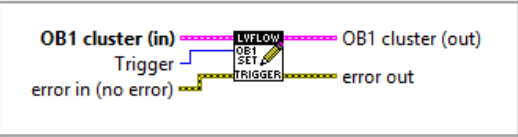
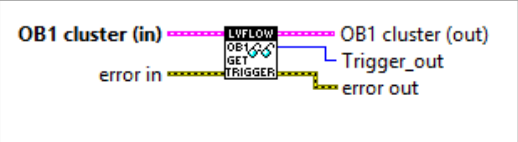
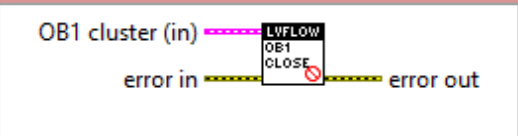

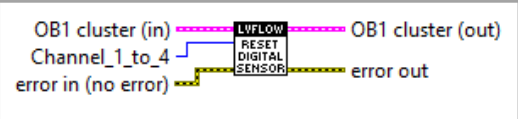


Figure 1 Typical workflow of a custom OB1 program representing the different types of the OB1 SDK VIs

A description of each VI can be found in the table below or by using the “context help” window in LabVIEW.

Icon	File name	Description
Configuration		
	OB1_init.vi	Initialize OB1 with the device reference and the type of regulators to be used. This VI generates an identification cluster of the OB1 to be used in other VIs. Note : use a NI-845X device for the ‘device reference’ input if the device is a MK3 or MK3+, otherwise use a

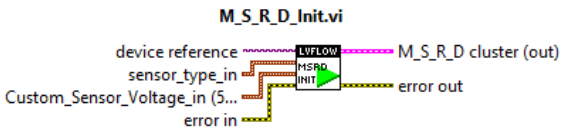
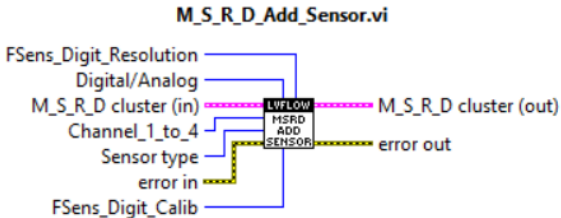
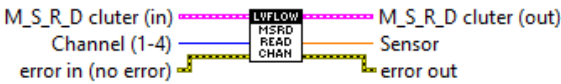
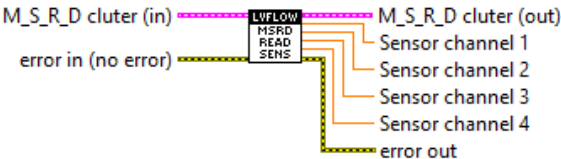

		VISA instr and do not connect the 'regulator type' input' for OB1 MK4 devices
	OB1_Add_Sensor.vi	<p>Add a sensor (flow or pressure) connected to the OB1. You must define the type of sensor (digital or analog), the channel it is connected to, the sensor type (80µL/min.... etc.) and the type of fluid for calibration.</p> <p>For digital sensors, the sensor type is automatically detected. For other sensors, these parameters are not considered. In case the sensor is not compatible with the OB1 version, or no digital sensor is detected, a pop-up will inform the user.</p> <p>The Custom_Sensor_Voltage in can be set from 5 to 25V and is used to set a voltage for custom analog sensors (works only for OB1 from 2020 and after).</p>
Calibration		
	OB1_Calibration.vi	<p>Launch a new OB1 calibration and return the calibration array.</p> <p>Before Calibration, ensure that ALL channels are properly closed with adequate caps.</p>
	OB1_Calibration_Save.vi	<p>Saves the actual calibration to the desired path. The function prompts the user to choose a path if no path is specified.</p>
	OB1_Calibration_Load.vi	<p>Load the calibration file located at Path and returns the OB1 cluster with the new calibration parameters.</p> <p>The function asks the user to choose the path if Path is not valid, empty or not a path.</p>
Operation		
	OB1_Set_Pressure.vi	<p>Set the desired value of pressure in the desired channel.</p>
	OB1_Set_Sensor.vi	<p>If a PID loop has been configured with the regulator channel, then sets the new target of the PID loop to the desired value. Otherwise set the pressure value.</p>
	OB1_Get_Channel.vi	<p>Returns the pressure measurement and possibly the sensor data associated to the given channel number. If no sensor is connected then the sensor data returns 0</p>

	OB1_Get_All_Channels.vi	Returns all pressure measurements and all sensor data associated to the device. If no sensor is connected then the sensor data returns 0
	OB1_Set_Trigger.vi	Set the trigger Out (EXT) of the OB1 0=>Low(0V) 1=>High(3.3V for OB1 MK3+, 5V for OB1 MK4)
	OB1_Get_Trigger.vi	Get the state of the trigger IN (INT). If nothing is connected it returns a High state. 0=>Low(0V) 1=>High(3.3V)
Close OB1 resource		
	OB1_Close.vi	Close the communication with the OB1 defined by its appropriate cluster.
Special use (advanced features)		
	OB1_Reset.vi	Warning: advanced feature. Reset OB1 communication for pressure and flow.
	OB1_Reset_Sensor.vi	Warning: advanced feature. Reset digital sensor communication from the selected channel. Select again resolution and calibration type (H2O/Isopro).

MSRD

All the available vi for the programming of a customized LabVIEW program are used in the VI “_M_S_R_D_Example.vi” contained in the LLB library “ElveflowLLB.llb”.

There are four available VI's that can be used when using the sensor reader which is able to read digital sensors. If your sensor reader is new, it is a MSRD. These VI do not work with FSR or old MSR.

Icon	File name	Description
Configuration		
	M_S_R_D_Init.vi	Initiate the communication with the MSR. Sensor type has to be defined here and in the M_S_R_D_Get_Data.vi. This VI generates an identification cluster of the instrument to be used with other VIs. The Custom_Sensor_Voltage_in can be set from 5 to 25V and is used to set a voltage for custom analog sensors.
	M_S_R_D_Add_Sensor	Add a sensor (flow or pressure) connected to the OB1. You must define the type of sensor (digital or analog), the channel it is connected to, the sensor type which has to be the same as for the Init step (80µL/min.... etc.) and the type of fluid for calibration. For digital sensors, the sensor type is automatically detected. For other sensors, these parameters are not considered. In case the sensor is not compatible with the MSRD version, or no digital sensor is detected, a pop-up will inform the user.
Operation		
	M_S_R_D_Get_Channel.vi	Read the sensor data on the requested channel with a unit of flow rate in µL/min and pressure in mbar.
	M_S_R_D_Get_All_Channel.vi	Read the sensor data on all channels with a unit of flow rate in µL/min and pressure in mbar.
	M_S_R_D_Set_Filters.vi	Set filter for the corresponding channel.

	M_S_R_D_Set_Trigger.vi	Update the trigger Output level Note : only compatible with the latest MSR MCU
	M_S_R_D_Get_Trigger.vi	Read the trigger Input level Note : only compatible with the latest MSR MCU
Close MSRD resource		
	M_S_R_D_Close.vi	Close the communication with the sensor reader and free the resources.
Special use (advanced features)		
	M_S_R_D_Reset.vi	Warning: advanced feature. Reset MSRD communication.
	M_S_R_D_Reset_Sensor.vi	Warning: advanced feature. Reset digital sensor communication from the selected channel. Select again resolution and calibration type (H2O/Isopro).



- Sensors connected to channel 1-2 and 3-4 should be the same type, otherwise they will not be considered, and the user will be informed by a prompt message.
- Sensor type has to be declared in the Init and in the Add Sensor step and has to be the same for both.

BFS

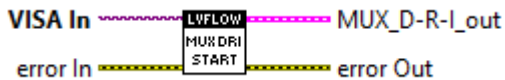
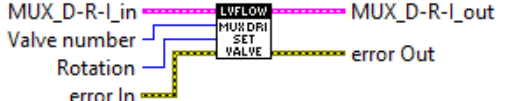



Please see the example file “_BFS_Example.vi” for a standard usage of the available BFS VI's. As with other instruments, there are three steps for programming: Initialization, instrumentation and resource liberation. Please note that for this particular sensor, in order to measure the flow rate ($\mu\text{L}/\text{min}$), you must first measure the volumetric mass density (g/L).

The table below gives a description of each vi.

Icon	File name	Description
Configuration		
	BFS_Init.vi	Initiate the communication with the BFS sensor and sets the actual sensor configuration: <ul style="list-style-type: none"> • The filtering value • If a density measurement is made before each flow rate measurement • If a temperature measurement is made before each flow rate measurement
Operation		
	BFS_Select_meas.vi	Configure the acquisition properties of the BFS : <ul style="list-style-type: none"> • The filtering value • If a density measurement is made before each flow rate measurement • If a temperature measurement is made before each flow rate measurement
	BFS_Read_Channel.vi	Measure the fluid flow in $\mu\text{L}/\text{min}$. You have to measure the density beforehand so that flow measurement works properly. Please ensure that the target fluid is inside the BFS when measuring the density. If you get -inf or +inf, the density wasn't correctly measured.
	BFS_Set_Filter_val.vi	Set the instrument's filter. Default value is "0.1". Maximum filtering value (slow response): 0.000001 Minimum filtering value, no filter (fast response time): 1.
	BFS_Zeroing.vi	Perform zero calibration of the BFS. Ensure that there is no flow when performed; it is advised to use valves. The calibration procedure is finished when the green LED stops blinking.
Close BFS resource		
	BFS_Close.vi	Close the BFS communication and free the resource.






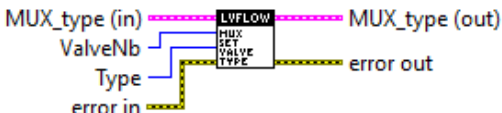
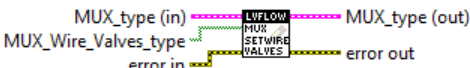

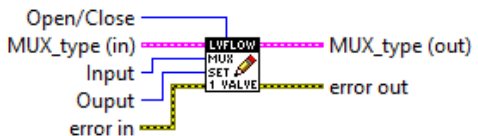
MUX D-R-I (DISTRIBUTION, DISTRIBUTOR, RECIRCULATION or INJECTION)

Please see the example file “_MUX_D-R-I_Example.vi” for a standard usage of the available MUX D-R-I VIs. The following table gives a description of the MUX D-R-I VIs.

Icon	File name	Function
	MUX_D-R-I_Init.vi	Establish connection with MUX Distribution/Distributor/Recirculation/Injection . You must input a VISA reference and select the COM port.
	MUX_D-R-I_SetValve.vi	<p>Switch the MUX Distribution/Distributor/Recirculation/Injection to the desired valve.</p> <p>For MUX Distribution 12, between 1-12. For MUX Distributor (6 or 10 valves), between 1-6 or 1-10. For MUX Recirculation 6 or MUX Injection (6 valves), the two states are 1 or 2.</p> <p>Rotation indicates the path the valve will follow to select a valve, either shortest, clockwise or counter clockwise.</p>
	MUX_D-R-I_GetValve.vi	Get the current valve number. If the valve is changing, function returns 0.
	MUX_D-R-I_SendCommand.vi	<p>!This function only works for MUX Distribution 12 or Recirculation 6!</p> <p>Get the Serial Number or Home the valve. Remember that Home the valve takes several seconds.</p> <p>Home the valve is necessary as an initialization step before using the valve for a session.</p>
	MUX_D-R-I_Close.vi	Close the communication with the MUX Distribution, Distributor, Recirculation or Injection and free the VISA resource.

Other MUX Series

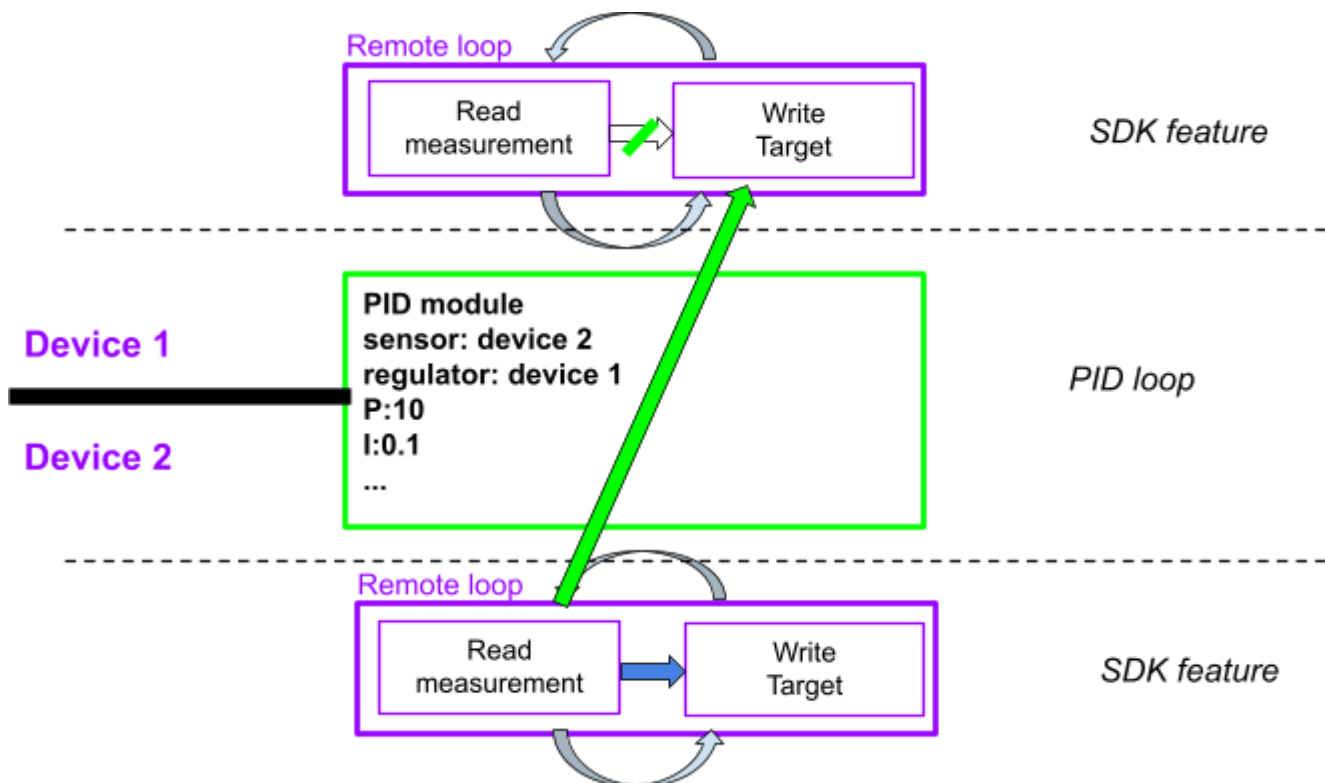
The MUX Series encompasses three instruments: MUX CROSS CHIP, MUX FLOW SWITCH and MUX WIRE. They are grouped together here because they use the same VIs to start and end the communication. The table below gives the description of each VI. The example VI “_Mux_Example.vi” illustrates the usage of all these VIs.

Instrument	Icon and VI name	Description
Common to all MUX Series	MUX_G_Init.vi 	Initializes the instrument using the device name and returns the identification cluster. Use a VISA instr for the 'device reference' input if the device is a MUX Wire V3, otherwise Use a NI-845X device
	MUX_G_Close.vi 	Closes the task and releases the allocated resources.
	MUX_G_Set_Trigger.vi 	Set the trigger Out (EXT) 0=>Low(0V) 1=>High(5V)
	MUX_G_Get_Trigger.vi 	Get the state of the trigger In (INT). If nothing is connected it returns a High state. 0=>Low(0V) 1=>High(5V)
MUX WIRE	MUX_G_Get_Valve_Type.vi 	Get valve type plugged into your MUX Wire V3
	MUX_G_Set_Valve_Type.vi 	This function is available for MUX Wire V3 using custom Valves or Valve V2. Valve V3 type are automatically recognized by the MUX ValveNB (MUX port where your valve is plugged) Type (Type of valve plugged)
	MUX_G_Wire_Set_Valve_Array.vi 	Set the valve array of the MUX Wire. The Valve array is a 1x16 matrix of booleans representing the valves connected to the instrument (TRUE for open and FALSE for close).
MUX FLOW SWITCH	MUX_G_Set_Valve_Array.vi 	Set the valve array of the instrument. Valve array here is a matrix of 4x4 booleans that control the internal valves. An ON value opens the corresponding internal valve and lets the fluid flow.
MUX CROSS CHIP	MUX_G_Set_Valve.vi 	Set the state of one valve of the instrument using the Input and Output parameters. These parameters correspond to the fluidic inputs and outputs. This function has the particularity to open and close the communication channel on each call. You

		can then use it without initialization or closing steps.
--	--	--

PID

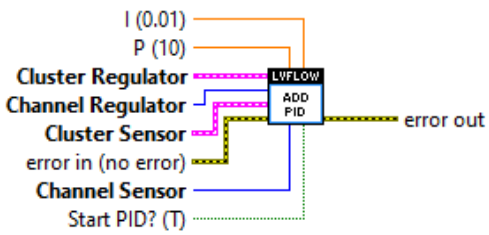

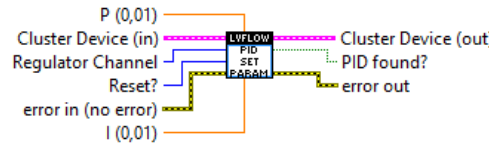
If you configure any of the compatible instruments (OB1, AF1, MSRD, BFS), because the acquisition runs autonomously, you will also be able to configure, start and stop PID loops between these instruments without having to write the code itself of the loop. A fully working flow regulation can be started with an OB1 and an MFS with a single function call. Subsequent modification of the PID target is achieved in the remote loop of the device controlling the pressure/flow. A PID loop can be started on a single remote loop if the device can regulate the pressure/flow and a sensor is also connected to it.



In this figure, we consider the example of two devices both capable of pressure control and sensor reading. These two devices are configured in remote mode and therefore measure the sensor, then update the pressure automatically. By calling `add_PID` between device 1 and device 2, the measurement from the sensor of device 2 is broadcasted to the remote loop of device 1. A PID loop is then continuously running across the two remote loops and can be stopped, modified or reset on demand.



PI or PID? The current SDK only allows PI parameters at the moment. Users who need to use the derivative term can use their own PID loop instead.

Icon	File name	Description
Configuration		
	PID_Add.vi	Add a PID loop between a regulator and a sensor. The PID loop can later be called with the device hosting the regulator coupled with its channel (if the device has more than 1). Only works when using the remote measurement vi.
Operation		
	PID_Set_Running.vi	Adjust the running status of a PID loop. The PID loop is chosen based on the input device hosting the regulator coupled with the regulator channel (if the device has more than 1). Only works when using the remote measurement vi.
	PID_Set_Params.vi	Adjust the PID parameters of a PID loop. The PID loop is chosen based on the input device hosting the regulator coupled with the regulator channel (if the device has more than 1). Only works when using the remote measurement vi.

Quick start examples

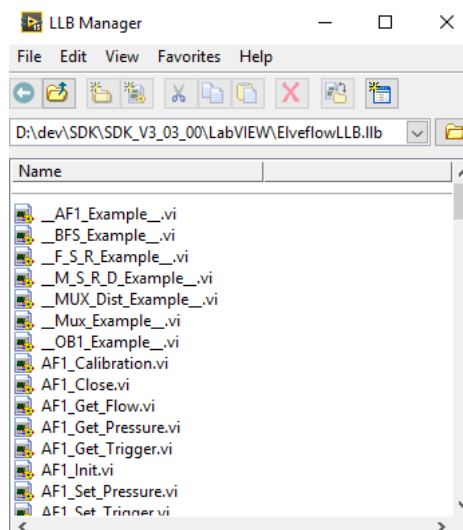
This section is here to guide you on how to use and modify the examples in each language. First of all, unzip the SDK file to have your uncompressed SDK folder.

All explanations described here are only for OB1 examples, but the principle is the same for other examples/instruments. We will consider for this quick start that we are using an OB1 MK4 with two regulators 0-200 mbar on channel 1 and 2, one regulator -1-1 bar on channel 3 and one regulator 0-8 bar on channel 4. On this OB1 we have a 1000 µL/min digital flow sensor that we want to use with H2O calibration and 16 bits resolution connected on channel 1 and a 1 bar pressure sensor connected on channel 3.

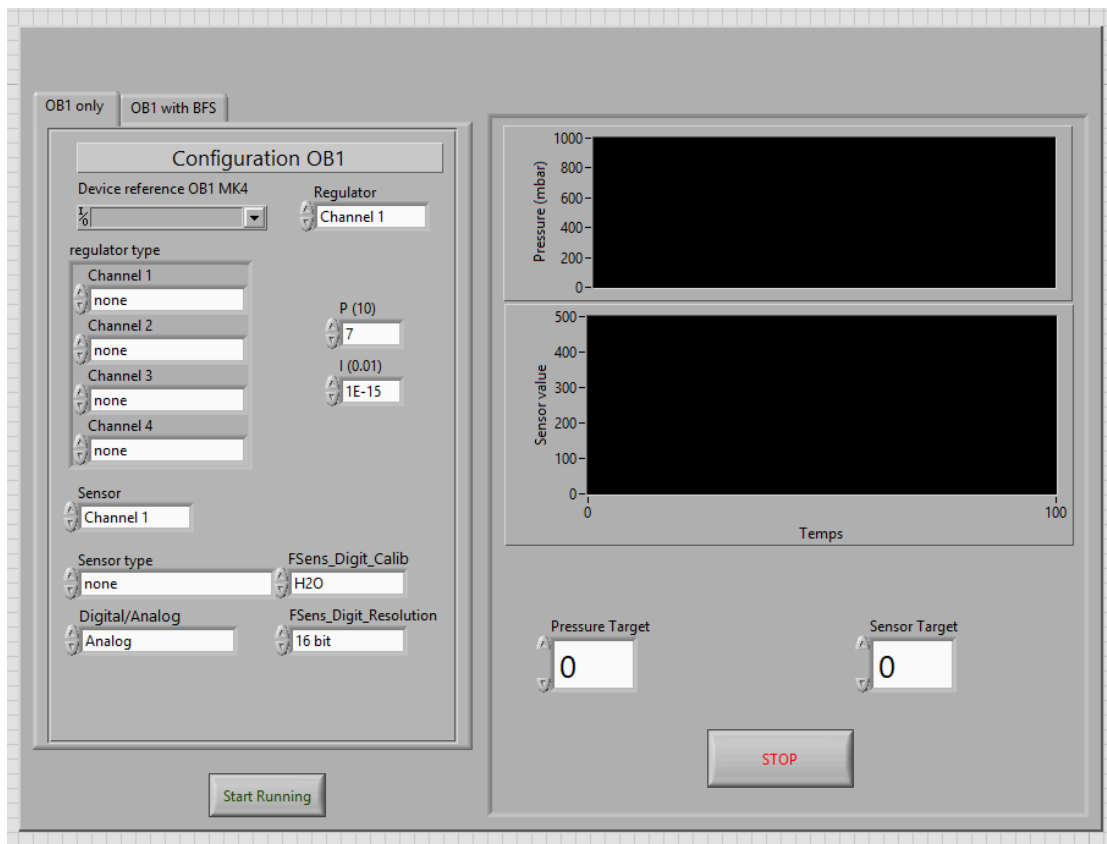
Furthermore, we will try a feedback loop between the flow sensor connected on sensor 1 and the first regulator channel.

Quick start

- 1) In your SDK folder, go to the "LabVIEW" folder. There should be a file named "ElveflowLLB.llb"
- 2) Double click on the file. It will open the following window:

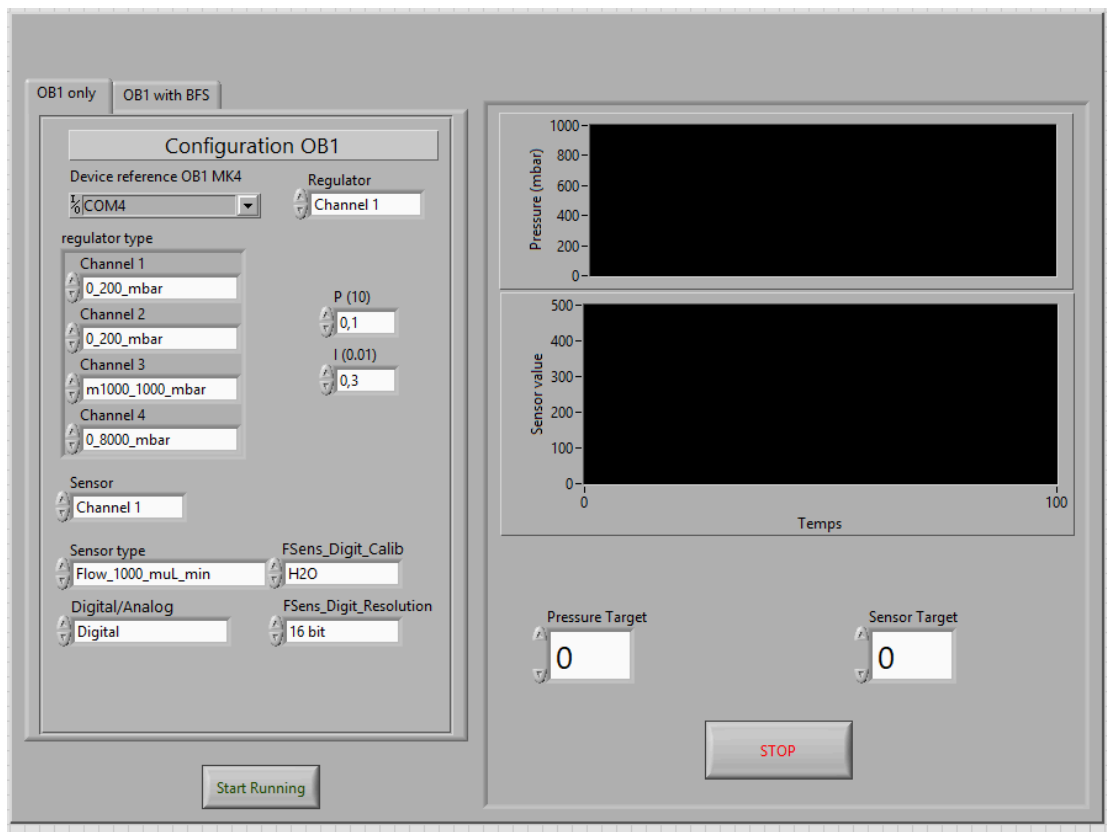


- 3) Double click on the example you want to run. Here we will open __OB1_Example__.vi
- 4) If warnings appear, ignore them. You should now have the following window opened:



- 5) To test the example, be sure that your OB1 is connected to the computer and turned on. Connect also all the flow sensors you want to use.
- 6) Considering the OB1 used you need to modify the following elements prior to running the VI:
 - DevName (in) (a list will appear with connected)
 - regulator type (Channel 1 to 4) [Note : filling those settings is not necessary for OB1 MK4 devices]
 - S.ChX type (with X=1 to 4)
 - S.ChX COM (with X=1 to 4)
 - FSens_Digit_ResolutionX (with X=1 to 4)
 - PI parameters

Considering the OB1 used for this example (described at the beginning of the section) the modified VI should be configured as follows:



- 7) Then the example is ready to be launched and will output pressure readings and sensor readings in the graph and tables.
- 8) When VI is running, to change pressure, modify the value “**Pressure Target**”. When you want to directly input a flow rate target that the OB1 will control via a feedback loop, modify the value “**Sensor Target**”

Now the example should run, for more details please refer to the block diagram and User Guide.

Appendix

Error handling:

All functions return an error code. If this code is 0 no error occurs. Other values indicate that an error occurs. Some personalized errors were added.

Error code:	Signification:
-8000	No Digital Sensor found
-8001	No pressure sensor compatible with OB1 MK3
-8002	No Digital pressure sensor compatible with OB1 MK3+
-8003	No Digital Flow sensor compatible with OB1 MK3
-8004	No IPA config for this sensor
-8005	Sensor not compatible
-8006	No Instrument with selected ID
-8007	Wrong MUX device
-8008	Only available for MUX Wire V3 devices
-8009	Type 1,2 and 3 are reserved for V3 valves. If you are using custom or older valves please use 4,5 and 6 types
-8030	No communication with OB1
-8031	No communication with BFS
-8032	No communication with MSRD
-8033	OB1 remote loop has not been executed
-8034	BFS remote loop has not been executed
-8035	MSRD remote loop has not been executed

Other errors can be found in the LabVIEW error user guide. (<http://www.ni.com/pdf/manuals/321551a.pdf>)