ELVE
FLOW an **ELVESYS** brand

Elveflow User Guide

SDK SOFTWARE DEVELOPMENT KIT

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

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
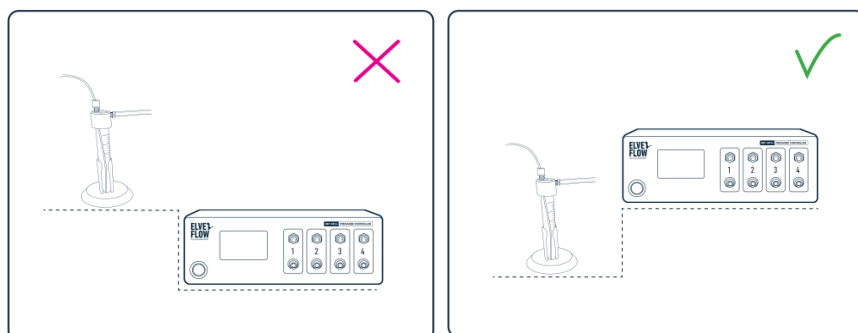
Important remarks

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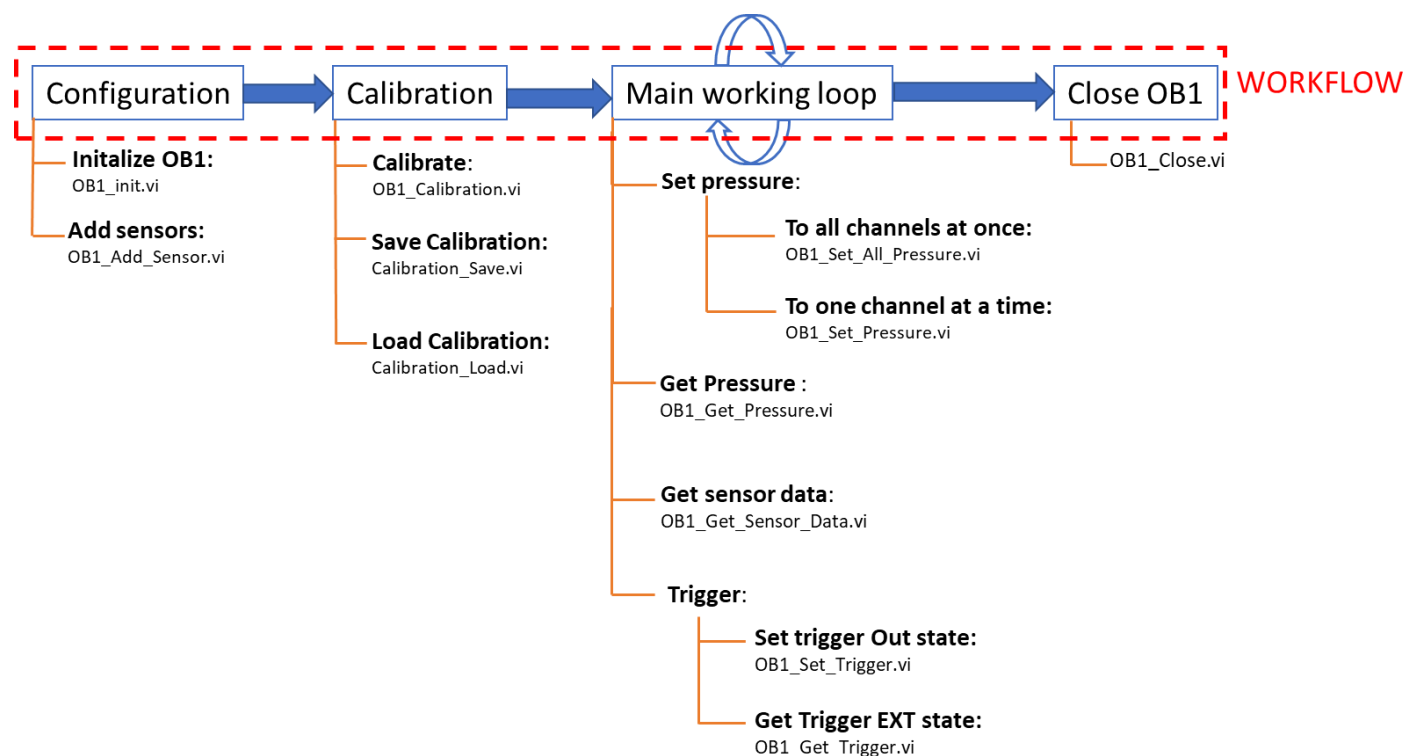
Figure 1 *Typical workflow of a custom OB1 program representing the different types of the OB1 SDK VIs*

The main working loop can be simplified with a new set of functions to launch a remote control and monitoring loop. Start the loop calling OB1_Start_Remote_Measuing.vi and stop the loop calling OB1_Stop_Remote_Measuing.vi. You can access the device while this loop is running with the set of remote VIs. Do not access the device with non remote VIs while you are still in remote mode. This remote feature also allows users to easily configure PID loops between devices.
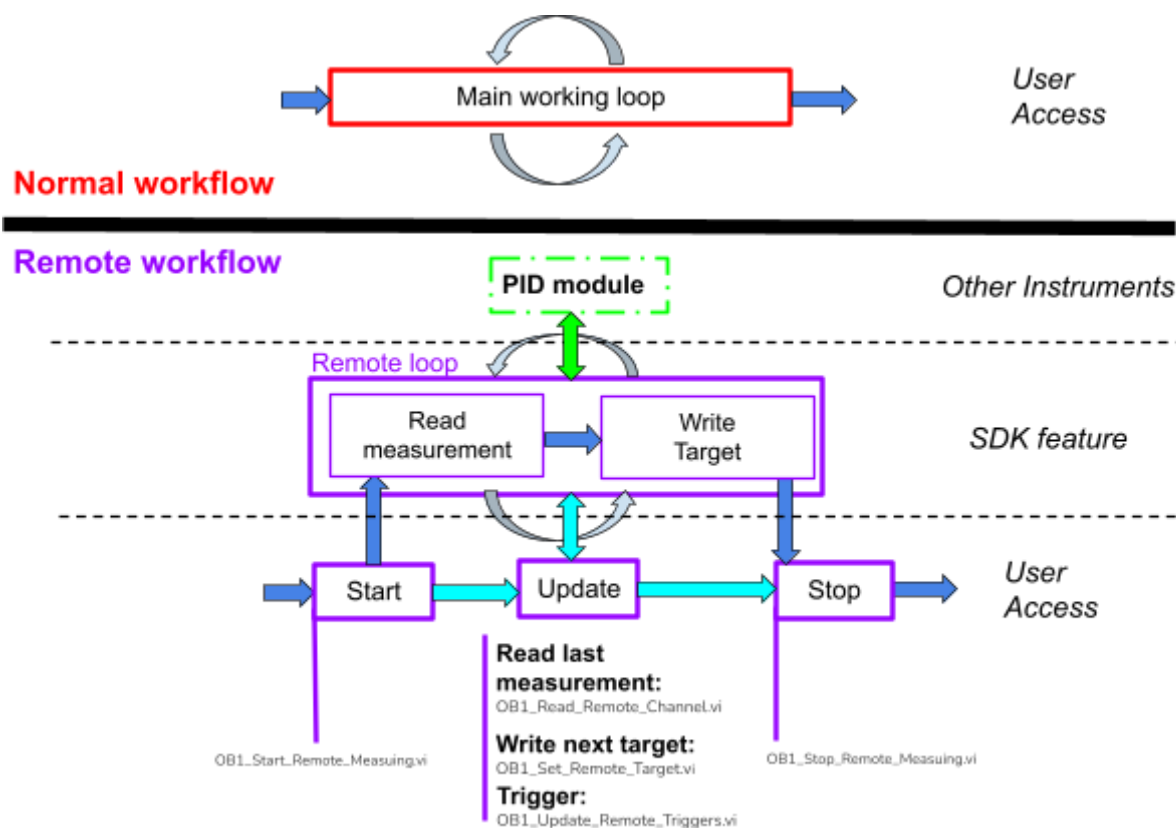
Figure 2 Difference between the normal workflow and the remote workflow

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	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. 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

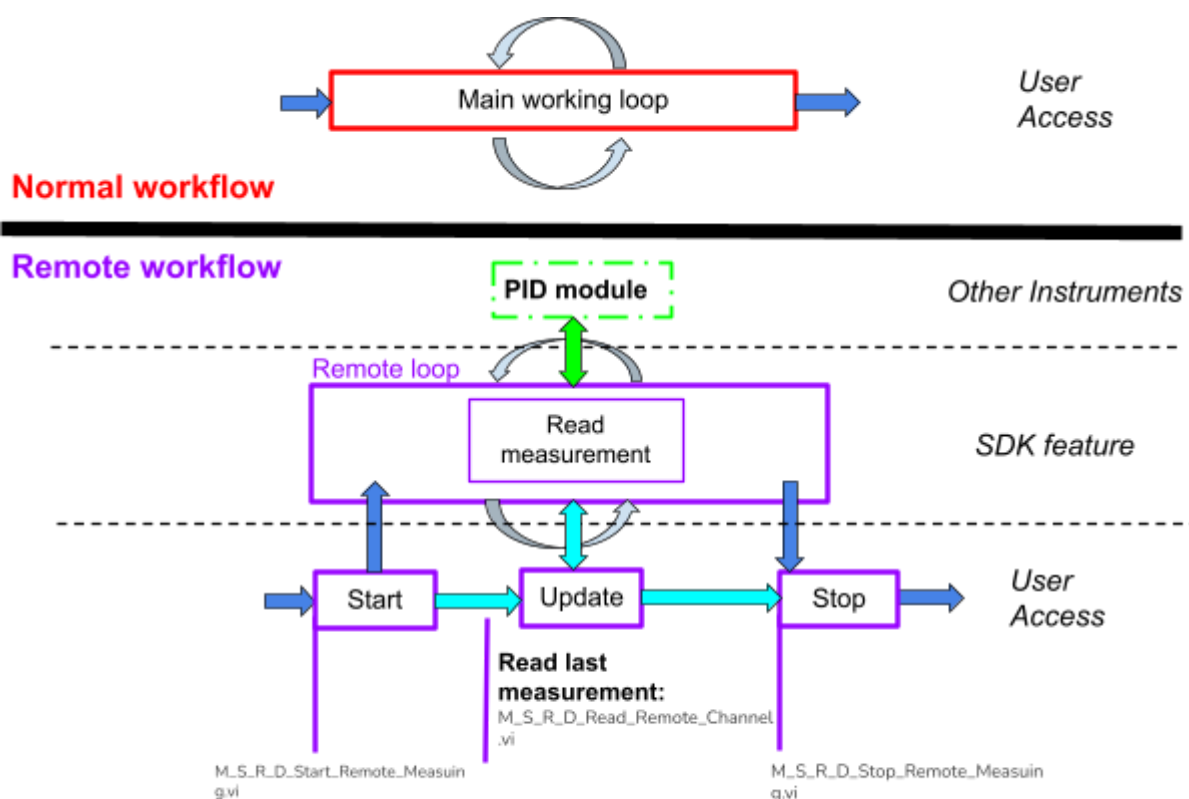|		
		OB1 version, or no digital sensor is detected, a pop-up will inform the user. The Custom_Sensor_Voltage in can be set from 5 to 25V and is used to set a voltage for custom analog sensors (works only for OB1 from 2020 and after).
Calibration		
OB1 cluster (in) Ref num to Slide error in — LVFLOW OB1 CALIB — OB1 cluster (out) Calibration error out	OB1_Calibration.vi	Launch a new OB1 calibration and return the calibration array. Ref num to Slide indicates the progress of the calibration. Once the calibration is done, a cluster of calibration data is generated as an output to use for pressure control. Before Calibration, ensure that ALL channels are properly closed with adequate caps.
Path Calibration error in — LVFLOW CALIB SAVE — error out	Calibration_Save.vi	Saves the actual calibration to the desired path. The function prompts the user to choose a path if no path is specified.
Path error in — LVFLOW CALIB LOAD — Calibration file load ? error out	Calibration_Load.vi	Load the calibration file located at Path and returns the calibration parameters in the Calibration cluster. The function asks the user to choose the path if Path is not valid, empty or not a path. The function indicates if the file was found
Operation		
calib cluster OB1 cluster (in) Pressure Channel_1_to_4 error in — LVFLOW OB1 SET Pressure — OB1 cluster (out) error out	OB1_Set_Pressure.vi	Set the desired value of pressure in the desired channel. Must use the Calibration cluster and the OB1 cluster for the setting of pressure to work properly.
calib cluster OB1 cluster (in) Pressure Array error in — LVFLOW OB1 SET All Pressure — OB1 cluster (out) error out	OB1_Set_All_Pressur e.vi	Works similarly as the vi "OB1_Set_Pressure.vi" except that it sets all the target values of pressure at once using an array as input. This vi needs the calibration and OB1 clusters.
Calibration OB1 cluster (in) Channel_1_to_4 Acquire_Data error in — LVFLOW OB1 GET PRESS. — OB1 cluster (out) Pressure error in	OB1_Get_Pressure.vi	Read the pressure of a selected channel. As with get-sensor-data, if Aquire_Data is TRUE, values of all regulators and analog sensors are read at once. Thus, to save computational time, you can set the value on FALSE for the other channels and iterations.
OB1 cluster (in) Channel_1_to_4 Acquire_Data error in — LVFLOW OB1 GET SENS DATA — OB1 cluster (out) Sens_Data error out	OB1_Get_Sensor_Da ta.vi	Read the sensor data on the requested channel. This function only converts data acquired in these units: flow rate: µl/min, pressure: mbar "Acquire_Data" works as described in the above description of OB1_Get_Pressure.vi. For Digital Sensors, this parameter has no impact NB: For Digital Flow Sensor, if the connection is lost, the OB1 will be reset and the returned value will be zero.

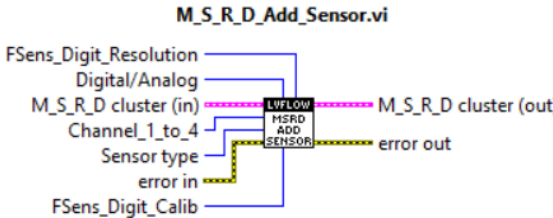|		
OB1 cluster (in), Trigger, error in (no error) → LVFLOW OB1 SET TRIGGER → OB1 cluster (out), error out	OB1_Set_Trigger.vi	Set the trigger Out (EXT) of the OB1 0=>Low(0V) 1=>High(3.3V for OB1 MK3+, 5V for OB1 MK4)
OB1 cluster (in), error in → LVFLOW OB1 GET TRIGGER → OB1 cluster (out), Trigger_out, error out	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)
Remote Operation		
calib cluster, OB1 cluster (in), error in (no error) → LVFLOW OB1 START → OB1 cluster (out), error out	OB1_Start_Remote_Measuing.vi	Start a control & monitoring loop in the background, which automatically reads all sensors and regulators. No direct call to the OB1 can be made until the Stop measuring vi is called. Until then only vi accessing this loop (read channel, set target, triggers) are recommended. Only 10 OB1 can be remotely controlled at the same time
OB1 cluster (in), Channel (1-4), error in (no error) → LVFLOW OB1 READ CHAN → OB1 cluster (out), Regulator, Sensor, error out	OB1_Read_Remote_Channel.vi	Read the measured regulator and sensor values from the background control & monitoring loop
OB1 cluster (in), Target, Channel (1-4), error in (no error) → LVFLOW OB1 SET TARGET → OB1 cluster (out), error out	OB1_Set_Remote_Target.vi	Change, in the running control & monitoring loop, either the regulator target or the target of the PID module if a PID loop is configured. Warning: the target units may vary. If the channel has a flow sensor **and** the PID is running in the loop, then the target will be a flow. Otherwise the target is a pressure
OB1 cluster (in), Trigger in, error in (no error) → LVFLOW OB1 TRIGGER → OB1 cluster (out), Trigger out, error out	OB1_Update_Remote_Triggers.vi	Set the input trigger and get the output trigger of the OB1 in the running control & monitoring loop.
OB1 cluster (in), error in (no error) → LVFLOW OB1 STOP → OB1 cluster (out), error out	OB1_Stop_Remote_Measuing.vi	Stop the background control & monitoring loop
Close OB1 resource		
OB1 cluster (in), error in → LVFLOW OB1 CLOSE → OB1 cluster (out), error out	OB1_Close.vi	Close the communication with the OB1 defined by its appropriate cluster.
Special use (advanced features)		
OB1 cluster (in), error in → LVFLOW OB1 RESET → OB1 cluster (out), error out	OB1_Reset.vi	Warning: advanced feature. Reset OB1 communication for pressure and flow.
OB1 cluster (in), Channel_1_to_4, error in (no error) → LVFLOW RESET DIGITAL SENSOR → OB1 cluster (out), error out	OB1_Reset_Sensor.vi	Warning: advanced feature. Reset digital sensor communication from the selected channel. Select again resolution and calibration type (H2O/Isopro).
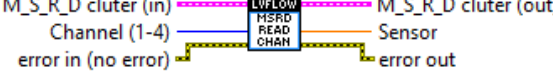
MSRD
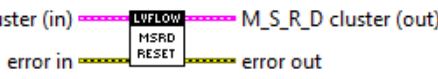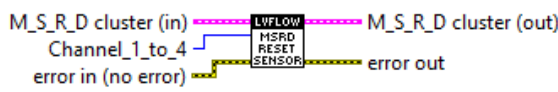
All the available vi for the programming of a customized LabVIEW program are used in the VI "_M_S_R_D_Example.vi" contained in the LLB library "ElveflowLLB.llb".

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 device reference sensor_type_in Custom_Sensor_Voltage_in (5... error in M_S_R_D cluster (out) error out	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.vi FSens_Digit_Resolution Digital/Analog M_S_R_D cluster (in) Channel_1_to_4 Sensor type error in FSens_Digit_Calib M_S_R_D cluster (out) error out	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.

<table>
<tr><th colspan="3">Operation</th></tr>
<tr>
<td>M_S_R_D cluster (in)
Channel_1_to_4
error in

M_S_R_D cluster (out)
Sens_Data
error out</td>
<td>M_S_R_D_Get_Data.vi</td>
<td>Read the sensor data on the requested channel with a unit of flow rate in µl/min and pressure in mbar.</td>
</tr>
<tr>
<td>M_S_R_D cluster (in)
Channel_1_to_4
ON/OFF
error in

M_S_R_D cluster (out)
error out</td>
<td>M_S_R_D_Set_Filters.vi</td>
<td>Set filter for the corresponding channel.</td>
</tr>
<tr>
<td>M_S_R_D cluster (in)
Set Trigger Output
error in

M_S_R_D cluster (out)
error out</td>
<td>M_S_R_D_Set_Trigger.vi</td>
<td>Update the trigger Output level
Note : only compatible with the latest MSR MCU</td>
</tr>
<tr>
<td>M_S_R_D cluster (in)
error in

M_S_R_D cluster (out)
Get Trigger Input
error out</td>
<td>M_S_R_D_Get_Trigger.vi</td>
<td>Read the trigger Input level
Note : only compatible with the latest MSR MCU</td>
</tr>
<tr><th colspan="3">Remote Operation</th></tr>
<tr>
<td>M_S_R_D cluster (in)
error in (no error)

M_S_R_D cluster (out)
error out</td>
<td>M_S_R_D_Start_Remote_Measuing.vi</td>
<td>Start a monitoring loop running in the background which automatically reads all sensors. No direct call to the MSRD can be made until the Stop measuring function is called. Until then only functions accessing this loop (read channel) are recommended. Only 10 MSRD can be remotely controlled at the same time</td>
</tr>
<tr>
<td>M_S_R_D cluter (in)
Channel (1-4)
error in (no error)

M_S_R_D cluter (out)
Sensor
error out</td>
<td>M_S_R_D_Read_Remote_Channel.vi</td>
<td>Read the measured sensor values from the background monitoring loop.</td>
</tr>
<tr>
<td>M_S_R_D cluster (in)
error in (no error)

M_S_R_D cluster (out)
error out</td>
<td>M_S_R_D_Stop_Remote_Measuing.vi</td>
<td>Stop the background monitoring loop</td>
</tr>
<tr><th colspan="3">Close MSRD resource</th></tr>
</table>

|		
	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 (µL/min), you must first measure the volumetric mass density (g/L).

💡 **New from 3.06.00:** The mainworking loop can be simplified with a new set of functions to launch a remote monitoring loop. Start the loop calling BFS_Start_Remote_Measuing.vi and stop the loop calling BFS_Stop_Remote_Measuing.vi. You can access the device while this loop is running with the set of remote VIs. Do not access the device with non remote VIs while you are still in remote mode. This remote feature also allows users to easily configure PID loops between devices.
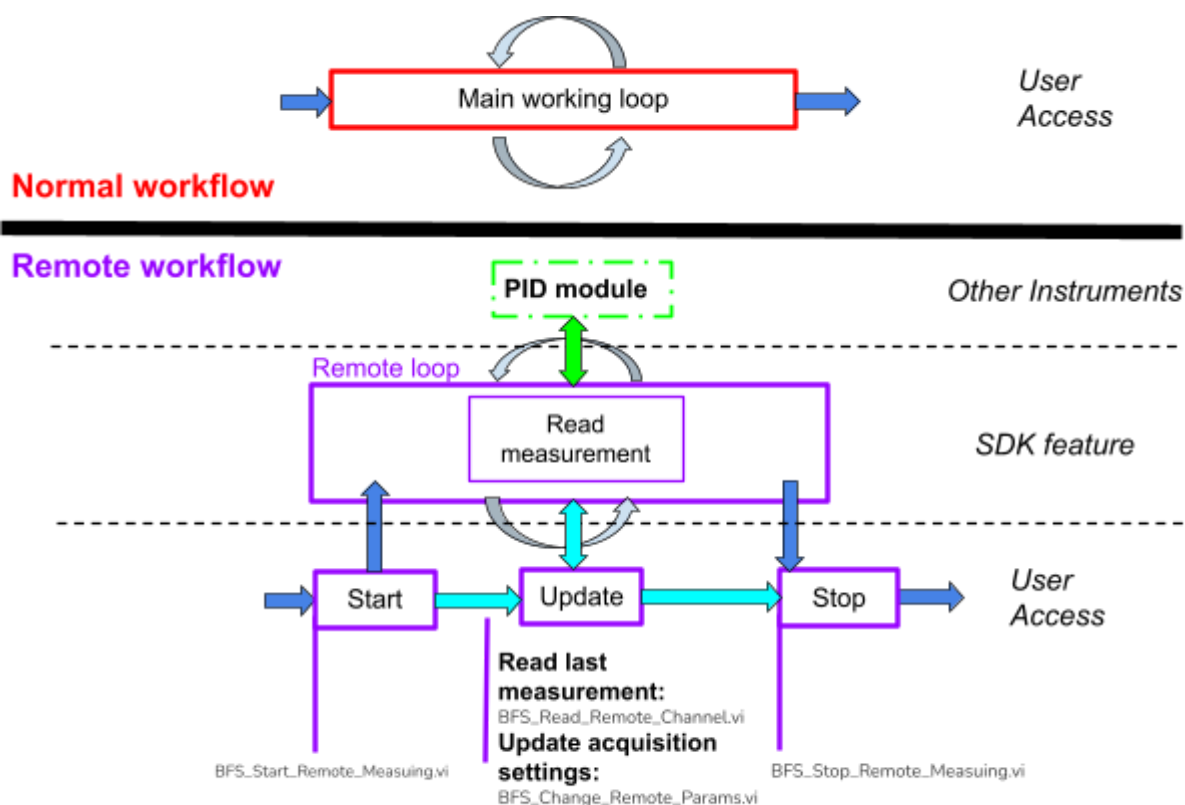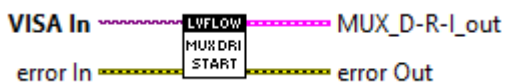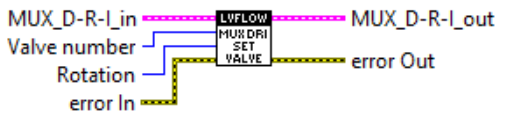

The table below gives a description of each vi.

Icon	File name	Description
Configuration		

Diagram	VI Name	Description
VISA resource name → BFS INIT → BFS_Out, error in → error out	BFS_Init.vi	Initiate the communication with the BFS sensor and gets the actual sensor configuration (scale)
Operation		
BFS_type_In → BFS GET DENSITY → BFS_type_Out, Density_g_L, error in (no error) → error out	BFS_Get_Density_val.vi	Get the actual volumetric mass density (g/L). This operation is required in order to obtain the flow rate.
BFS_type_In → BFS GET FLOW → BFS_type_Out, Measure Direct, error in (no error) → error out	BFS_Get_Flow_val.vi	Measure the fluid flow in µL/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_type_In → BFS GET TEMP → BFS_type_Out, Temperature_deg_C_, error in (no error) → error out	BFS_Get_Temperature_val.vi	Measure the fluid temperature in °C.
BFS_type_In, Sensor_Filter_0_to_1 → BFS SET FILT → BFS_type_Out, error in (no error) → error out	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_type_In → BFS DO ZEROING → BFS_type_Out, error in → error out	BFS_Do_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.
Remote Operation		
Filter (0,1), BFS cluster (in), Measure density, Measure temperature, error in (no error) → BFS START → BFS cluster (out), error out	BFS_Start_Remote_Measuing.vi	Start a monitoring loop in the background which automatically reads all sensors. No direct call to the BFS should be made until the Stop measuring function is called. Until then only vi accessing this loop (read channel, set params) are recommended. Only 10 BFS can be remotely controlled at the same time
BFS cluster (in), error in (no error) → BFS READ CHAN → BFS cluster (out), Flow (µL/min), Density (g/L), error out, Temperature (°C)	BFS_Read_Remote_Channel.vi	Read the measured sensor values from the background monitoring loop, If the Measure Density ? or Measure Temperature ? parameters are set to FALSE, then the corresponding indicator will return 0.
Measure temperature?, BFS cluster (in), Filter (0,1), Measure density?, error in (no error) → BFS SET PARAM → BFS cluster (out), error out	BFS_Change_Remote_Params.vi	Change the BFS acquisition parameters.
BFS cluster (in), error in (no error) → BFS STOP → BFS cluster (out), error out	BFS_Stop_Remote_Measuing.vi	Stop the background monitoring loop

Close BFS resource		
	BFS_Close.vi	Close the BFS communication and free the resource.

MUX D-R-I (DISTRIBUTION, DISTRIBUTOR, RECIRCULATION or INJECTION)
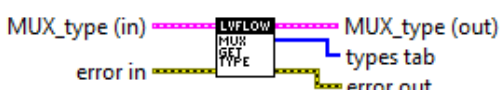
Please see the example file "_MUX_D-R-I_Example.vi" for a standard usage of the available MUX D-R-I VIs. The following table gives a description of the MUX D-R-I VIs.

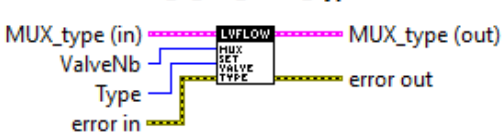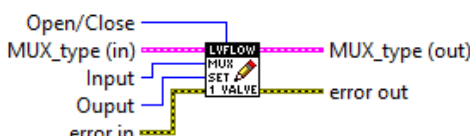| Icon	File name	Function
VISA In — MUX_D-R-I_out / error In — error Out (LVFLOW MUX DRI START)	MUX_D-R-I_Init.vi	Establish connection with MUX Distribution/Distributor/Recirculation/Injection. You must input a VISA reference and select the COM port.
MUX_D-R-I_in / Valve number / Rotation / error In — MUX_D-R-I_out / error Out (LVFLOW MUX DRI SET VALVE)	MUX_D-R-I_SetValve.vi	Switch the MUX Distribution/Distributor/Recirculation/Injection to the desired valve. 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. Rotation indicates the path the valve will follow to select a valve, either shortest, clockwise or counter clockwise.
MUX_D-R-I_in / error In — MUX_D-R-I_out / selected_Valve / error Out (LVFLOW MUX DRI GET VALVE)	MUX_D-R-I_GetValve.vi	Get the current valve number. If the valve is changing, function returns 0.
MUX_D-R-I_in / Action / error In — MUX_D-R-I_out / Answer / error Out (LVFLOW MUX DRI SEND COMM)	MUX_D-R-I_SendCommand.vi	!This function only works for MUX Distribution 12 or Recirculation 6! Get the Serial Number or Home the valve. Remember that Home the valve takes several seconds. Home the valve is necessary as an initialization step before using the valve for a session.
MUX_D-R-I_in / error In — error Out (LVFLOW MUX DRI CLOSE)	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** DevName — MUX_type error in — error out	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 MUX_type error in — error out	Closes the task and releases the allocated resources.
	MUX_G_Set_Trigger.vi MUX_type (in) — MUX_type (out) Trigger error in — error out	Set the trigger Out (EXT) 0=>Low(0V) 1=>High(5V)
	MUX_G_Get_Trigger.vi MUX_type (in) — MUX_type (out) error in — Trigger error out	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** MUX_type (in) — MUX_type (out) error in — types tab error out	Get valve type plugged into your MUX Wire V3
	MUX_G_Set_Valve_Type.vi MUX_type (in) — MUX_type (out) ValveNb Type — error out error in	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 MUX_type (in) — MUX_type (out) MUX_Wire_Valves_type error in — error out	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** MUX_type (in) — MUX_type (out) Valves error in — error out	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** Open/Close MUX_type (in) — MUX_type (out) Input Ouput — error out error in	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

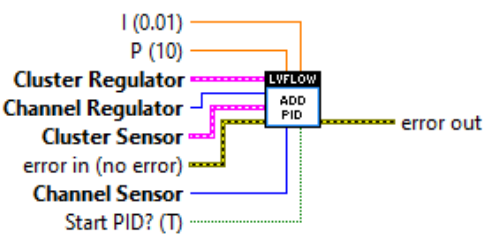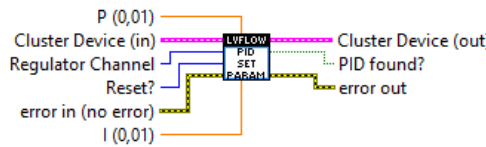|		can then use it without initialization or closing steps.

Remote PID

If you configure any of the compatible instruments (OB1, AF1, MSRD, BFS) in the remote mode, where the acquisition is run 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.
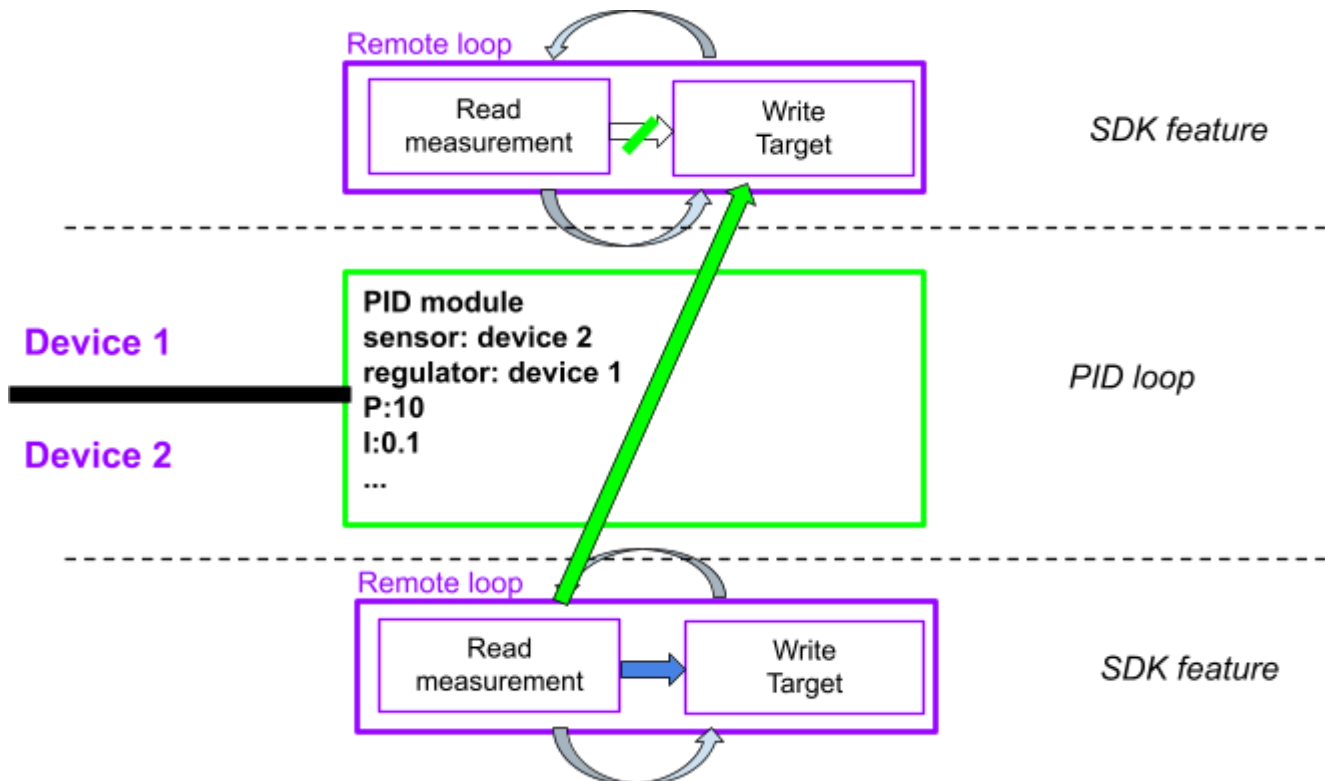
In this figure, we consider the example of two devices both capable of pressure control and sensor reading. These two devices are configured in remote mode and therefore measure the sensor, then update the pressure automatically. By calling add_PID between device 1 and device 2, the measurement from 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		
I (0.01) P (10) **Cluster Regulator** **Channel Regulator** **Cluster Sensor** error in (no error) **Channel Sensor** Start PID? (T) LVFLOW ADD PID error out	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		
Cluster Device (in) Regulator Channel Running error in (no error) LVFLOW PID SET RUN Cluster Device (out) PID found? error out	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.
P (0,01) Cluster Device (in) Regulator Channel Reset? error in (no error) I (0,01) LVFLOW PID SET PARAM Cluster Device (out) PID found? error out	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 MK3+ 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.

Quick start

1) In your SDK folder, go to "LabVIEW" folder. There should be a file named "ElveflowLLB.llb"

2) Double click on the file. It will open the following window:
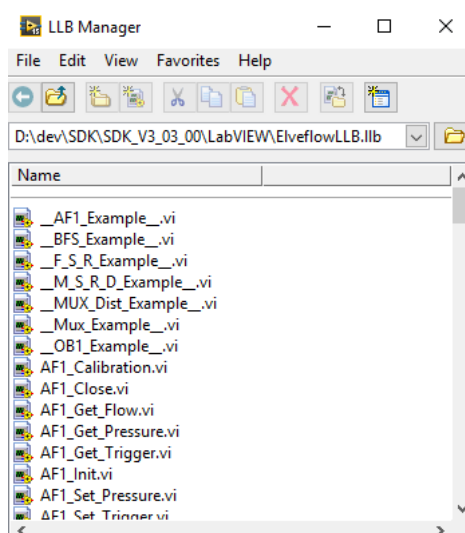
3) Double click on the example you want to run. Here we will open __OB1_Example__.vi

4) If warnings appear, ignore them. You should now have the following window opened:
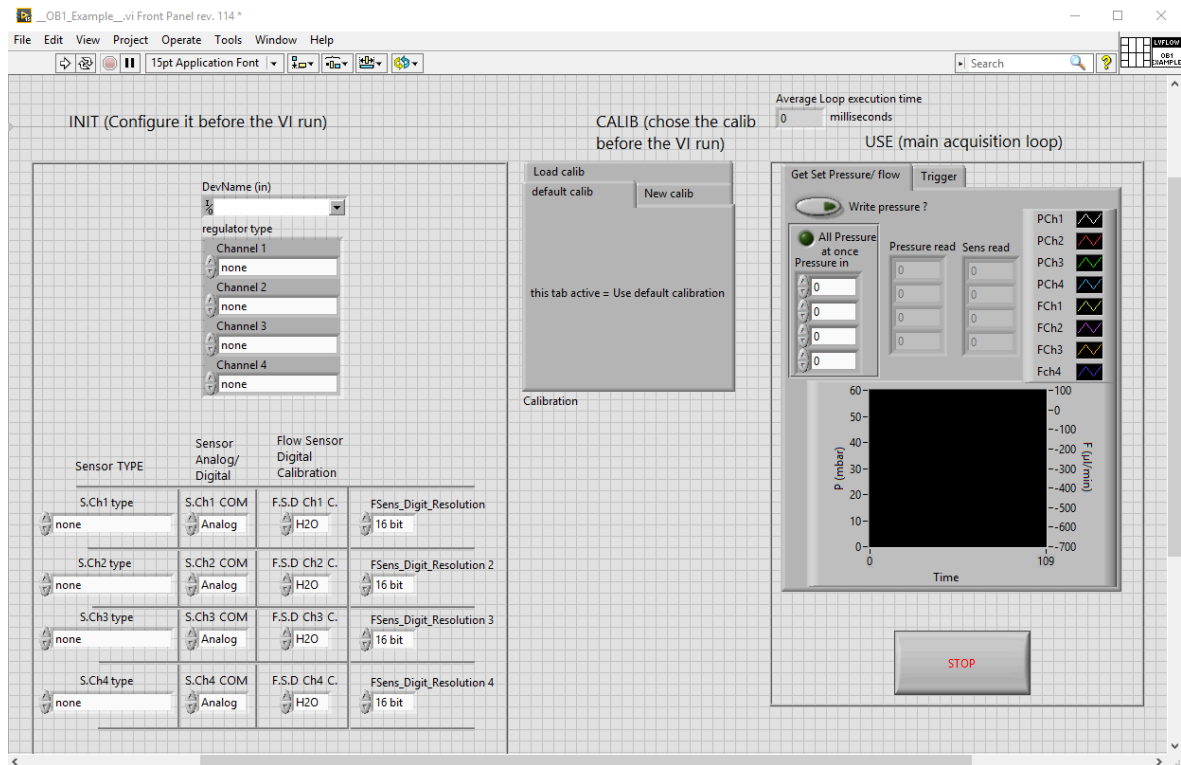
5) To test the example, be sure that your OB1 is connected to the computer and turned on. Connect also all the flow sensors you want to use.

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)

- S.ChX type (with X=1 to 4)

- S.ChX COM (with X=1 to 4)

- FSens_Digit_ResolutionX (with X=1 to 4)

Considering the OB1 used for this example (described at the beginning of the section) the modified VI should be configured as follows:
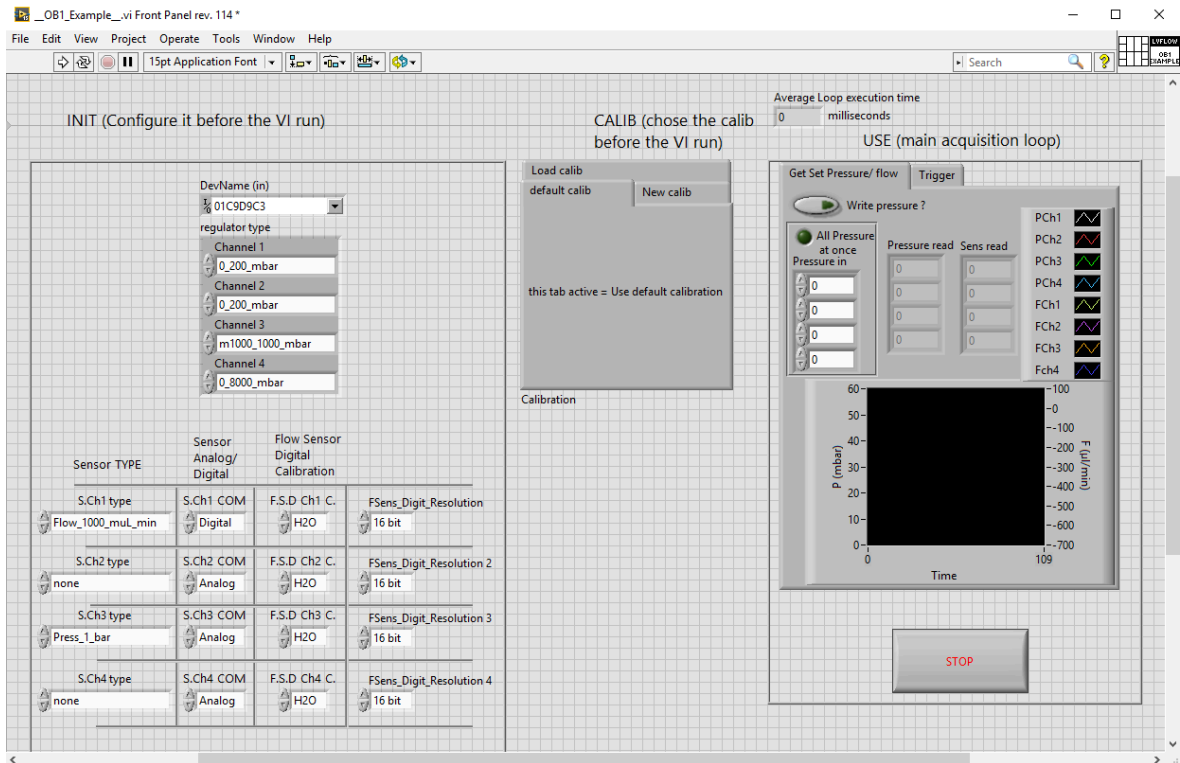
7) On the middle tab you can choose to perform calibration, load an existing calibration or use default calibration. For this example we will only use default calibration. Remember that calibration files generated through ESI cannot be used with SDK. Only SDK generated calibration files can be loaded using SDK.

8) Then the example is ready to be launched and will output pressure readings and sensor readings in the graph and tables.

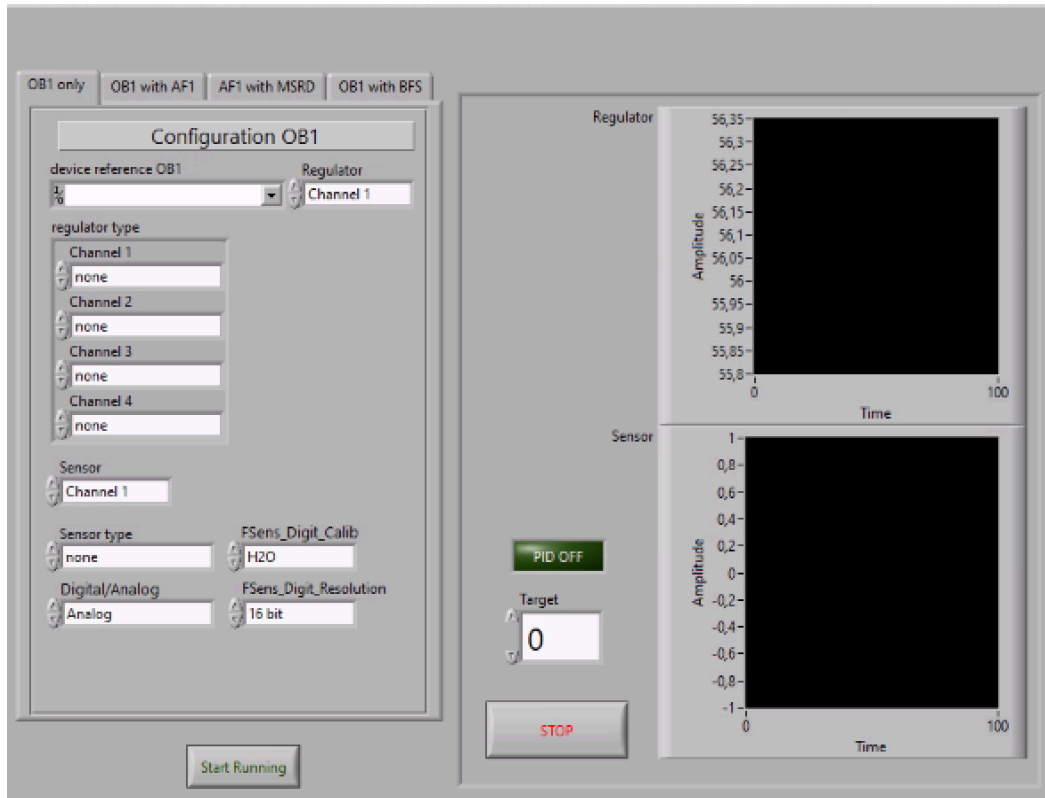9) When VI is running, to change pressure, modify values from this table:
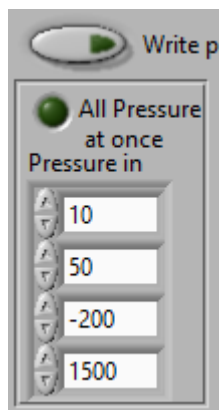
And click on "Write pressure?" button to write and unclick it.

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

Remote mode

10) Go back to the llb menu. Now we will open __Remote_PID_Examples__.vi

11) Fill in the left panel the information about your OB1 regulator and sensor you want to configure in a PID loop, then press 'Start Running' at the bottom.

12) You can now activate the PID with the green indicator at the center. Change the Target to observe the behavior of the loop.

13) Once satisfied with the process, you can press 'stop' and look at the code itself.

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 with AF1
-8006	No Instrument with selected ID

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