User Manual

English

HBM LabVIEW Driver

Version 4.0

HBM

Hottinger Brüel & Kjaer GmbH
Im Tiefen See 45
D-64239 Darmstadt
Tel. +49 6151 803-0
Fax +49 6151 803-9100
Email: info@hbm.com
Internet: www.hbm.com

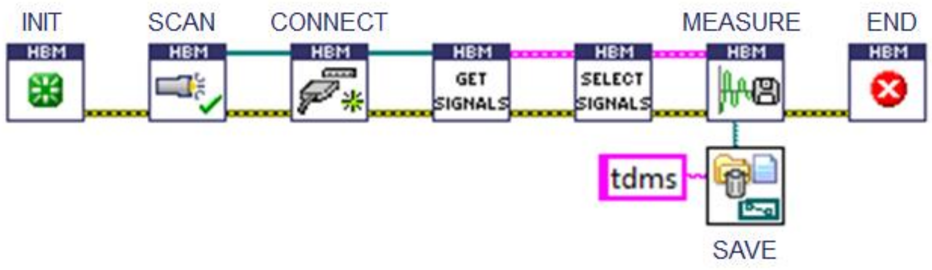
DVS: A04175-4.0
HBM public
01.2021

1 Contents

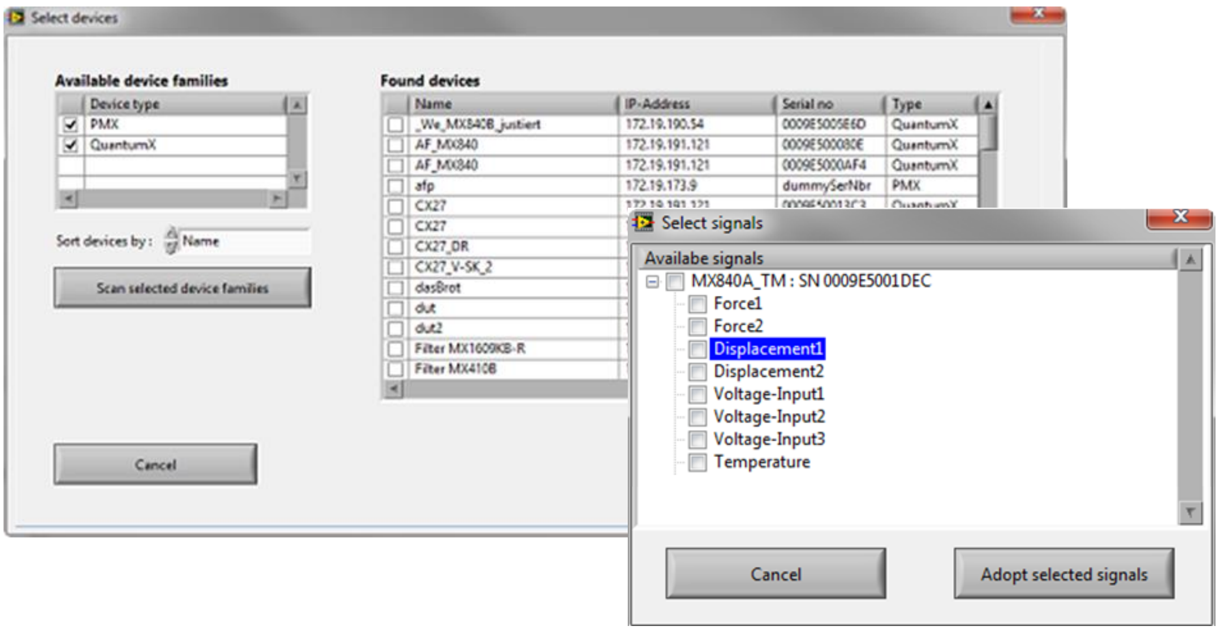
2 Abstract

The HBM LabVIEW Driver provides a comfortable and easy integration of the HBM data acquisition systems QuantumX / SomatXR, PMX and MGCplus in LabVIEW. The driver offers powerful VIs.

With only eight VIs a complete data acquisition from several channels or devices can be realized:



VIs with ready to use graphical user interfaces can be used for device scan and channel selection:



Available device families

	Device type
✓	PMX
✓	QuantumX

Found devices

	Name	IP-Address	Serial no	Type
☐	_We_MX840B_justiert	172.19.190.54	0009E5005E6D	QuantumX
☐	AF_MX840	172.19.191.121	0009E500080E	QuantumX
☐	AF_MX840	172.19.191.121	0009E5000AF4	QuantumX
☐	afp	172.19.173.9	dummySerNbr	PMX
☐	CX27	172.19.191.121	0009E50013C3	QuantumX
☐	CX27			
☐	CX27_DR			
☐	CX27_V-SK_2			
☐	dasBrot			
☐	dut			
☐	dut2			
☐	Filter MX1609KB-R			
☐	Filter MX410B			

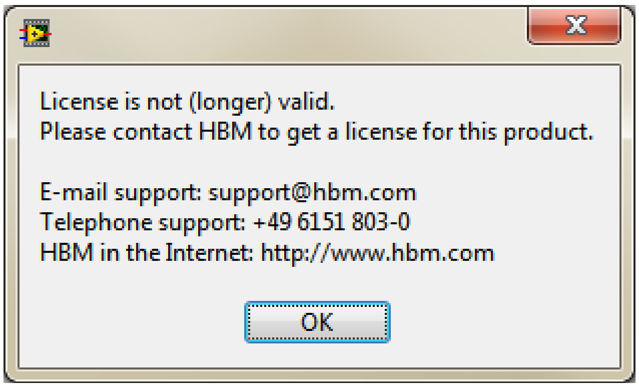
Availabe signals

- ☐ MX840A_TM : SN 0009E5001DEC
  - ☐ Force1
  - ☐ Force2
  - ☐ Displacement1
  - ☐ Displacement2
  - ☐ Voltage-Input1
  - ☐ Voltage-Input2
  - ☐ Voltage-Input3
  - ☐ Temperature

Common device commands, e.g., device scan, channel selection, connect, start or stop measurement are realized by common VIs. So main parts of your application can be reused if you change the data acquisition system.
Documented examples provide a quick start.

3 License

The HBM LabVIEW Driver is protected by a license file.
Please contact your local HBM sales representative to purchase a driver license. HBM will send you a permanent license file which will replace the expired evaluation license file within the folder
LabVIEWXXXX\user.lib\HBM LabVIEW Driver\Dlls.
Your own LabVIEW executables do not need a license file to run. You may distribute your executables together with the VIs from HBM LabVIEW Driver to your customers.
Please refer also the HBM License Conditions for Software which is available in the Windows start menu after installation of the driver.
The HBM LabVIEW Driver comes with a 30 days evaluation license. After that time, it is no longer possible to use the VIs within the LabVIEW IDE.



License is not (longer) valid.
Please contact HBM to get a license for this product.

E-mail support: support@hbm.com
Telephone support: +49 6151 803-0
HBM in the Internet: http://www.hbm.com

4 Technical support

If you require any help or further information, please contact your local HBM sales representative or one of the local support hotlines:

https://www.hbm.com/contact/

HBM on the Internet:

http://www.hbm.com

5 Additional Resources

The technical support team helps customers in many ways. Meanwhile there are various VIs available, that were provided to support certain customer requests. Since these VIs were created for very special demands, they are not installed under the user.lib but are available as an extra zip file which you can find in the same directory as the setup of the HBM LabVIEW Driver.

These VIs can be used to get a further understanding of the usage of the underlying HBM Common API and might already contain the solution or at least a hint to solve specific problems, that are not covered by the VIs that are included in the setup of the HBM LabVIEW Driver.

6 Prerequisites

To use the HBM LabVIEW Driver your system has to meet following requirements:

- LabVIEW 2012 or higher *
- .NET Framework 4.0
- QuantumX Firmware: 4.0.24 or higher
- PMX Firmware: 2.0 or higher
- MGCplus: CP42 Firmware: 4.74 or higher; CP22 Firmware: 4.44 or higher; (CP32 is not supported)

* To use the HBM LabVIEW Driver with LabVIEW 2012 you have to create or extend the file LabVIEW.exe.config within the same folder where LabVIEW.exe is located with the following content:

```
<configuration>
  <startup useLegacyV2RuntimeActivationPolicy="true">
    <supportedRuntime version="v4.0.30319"/>
  </startup>
</configuration>
```

To distribute your program, that uses VIs of the HBM LabVIEW Driver, you need to:

- Install .NET framework 4.0 on the target system
- Copy ALL FILES AND DIRECTORIES (***except your License file!**) below "LabVIEW XXX/user.lib/HBM LabVIEW Driver/DLLs/" into the data directory of your application
- Modify the Firewall (e.g., by executing "Firewall__scan_allowed_for_all.bat" which is located in the directory "…\LabVIEWXXXX\user.lib\HBM LabVIEW Driver\DLLs")
  - For the scan to work, the incoming UDP ports 31416 and 31417 need to be opened for the application.
  - Since Windows® 7 this can be done by commandline
    - `netsh advfirewall firewall add rule name="AppName" direction=in action=allow enable=yes profile=any localport=31416,31417 protocol=UDP edge=yes program="AppExe"`
  - For Windows® XP this can be done by commandline
    - `netsh firewall add allowedprogram name="AppName" mode=ENABLE scope=ALL profile=ALL program="AppExe"`

(*) **Attention: Please do not distribute your License file by mistake!!!**

7 Supported Features Overview

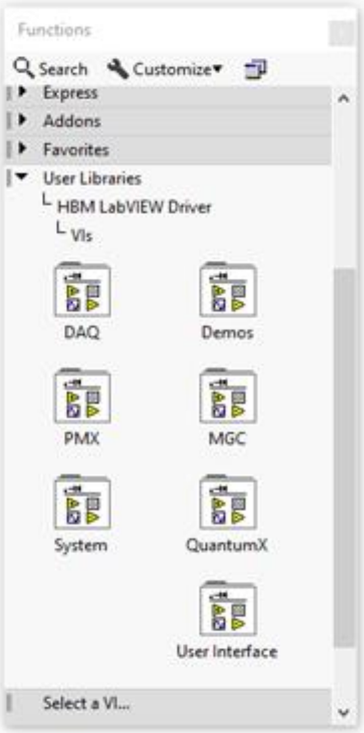
The HBM LabVIEW Driver is based on the HBM Common API. However not all features of the Common API find their equivalent in any of the VIs of this driver.
The first versions of this driver focus on measuring – and do not include VIs that support sensor parameterization for example. Even so, you may use the Common API within your VIs to realize further functionality. None of the VIs of the HBM LabVIEW driver is protected (except Init.vi) and you can use them as examples or templates for your own VIs.

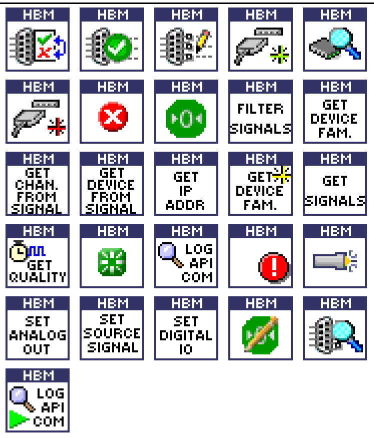
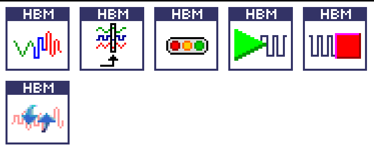
	Common API			**LabVIEW Driver**		
DAQ System	QuantumX /SomatXR	PMX	MGCplus	QuantumX /SomatXR	PMX	MGCplus
Device Scan	✓	✓	✓ (5)	✓	✓	✓ (5)
Measurement configuration	✓	✓	✓	✓ (2)	✓ (2)	✓ (2)
Sensor configuration	✓	✓	✓	(3)	(3)	(3)
Analog In DAQ	✓	✓	✓	✓	✓	✓
Analog Out (direct setting)	✓	✓	✓	✓	✓	✓
Analog Out (channel routing)	✓	✓	✓	✓	✓	✓
Digital In/Out DAQ	✓	✓	✓	✓	✓	✓
Digital Out (direct setting)	✓	✓	✓	✓	✓	✓
CAN DAQ	✓	✓ (4)	✓	✓	✓ (4)	
CAN Raw DAQ	✓	(1)	(1)	✓	(1)	(1)
Optical In DAQ	✓	(1)	(1)	✓	(1)	(1)

(1) Not supported by hardware
(2) Only sample rate and filter frequency can be set by VI
(3) There are no dedicated VIs for sensor configuration (nevertheless it is possible to setup sensor settings by using the API directly – please check the UseCommonApiToSetup-ScalingAndSensor.vi to get an idea how this can be done or check the additional resources mentioned in chapter 5)
(4) Only via CODESYS/calculated channels
(5) Only for MGCplus with CP52

8 LabVIEW Driver VIs Overview

After successful installation of the HBM LabVIEW Driver, there will be a new entry within the function palette of LabVIEW (User Libraries – HBM LabVIEW Driver), containing two directories:
The DLLs directory contains all necessary libraries, the help file, and the license file. If you plan to build LabVIEW executables you have to assert that all files and directories are added to your project (assert to remove your license file from your distribution!).
The VIs directory contains several subdirectories in which you find the following groups:



	**Group System** VIs in this group are used to: • Scan the LAN adapter for devices • Connect and disconnect devices • View signal settings • Activate, change, and assign signal settings • Configurate filters • Set digital and analog outputs • Route signals to analog outputs • Set zero offsets All of these VIs are working with all devices as long as the device is able to support the function (e.g. a MGC-CP42 device cannot be scanned).
	**Group DAQ** This group contains all VIs that are useful for data acquisition. There are VIs that support: • Preparing a continuous measurement • Starting and stopping a continuous measurement • Getting continuous measurement values • Getting single measurement values without running a continuous measurement.
	Group User Interface VIs in this group can be used in an interactive way to: • Scan for certain devices • Connect certain devices • Select signals that you want to use • Measure certain signals • Set digital and analog outputs They all have their own user interfaces that allow an interactive operation.

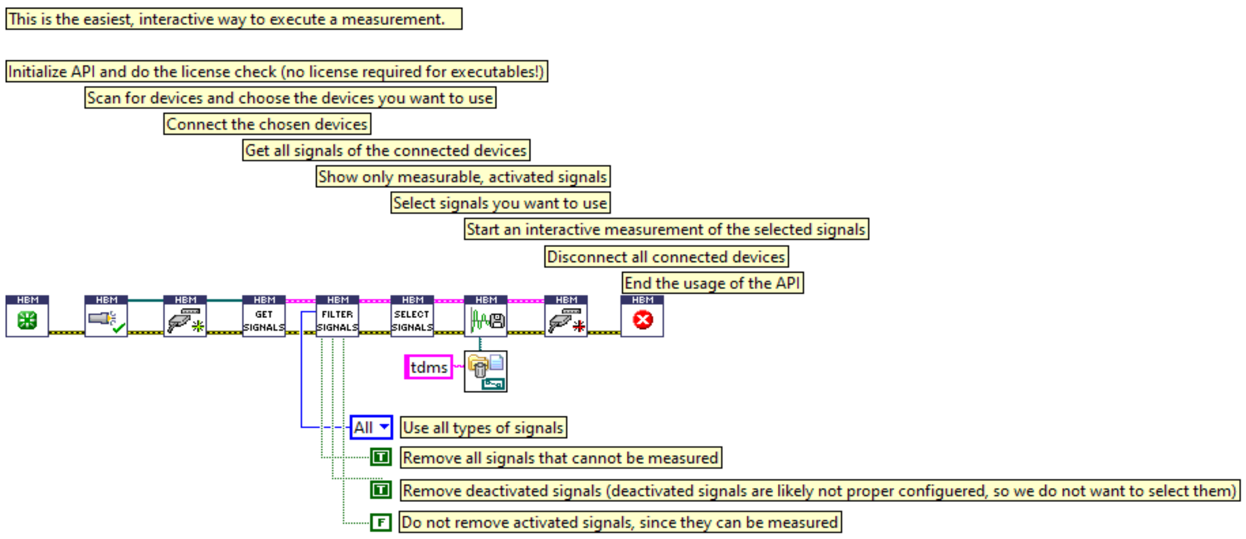
HBM QX; HBM STOP BLINK; HBM START BLINK	**Group QuantumX** This group contains VIs that realize additional features (like start and stop blinking at a certain connector) of QuantumX devices. Additional features are not covered by the common functionality that is implemented for all device types. These VIs can only be used with QuantumX devices!
HBM ACT. TEDS; HBM ASSIGN LIMIT SWITCH; HBM CLEAR PEAK VALUES; HBM PMX; HBM GET DEVICE STATUS; HBM GET DIGITAL OUT MASK; HBM GET LIMIT SWITCH; HBM GET MAX. VALUE; HBM GET MIN VALUE; HBM GET PEAK TO PEAK VALUE; HBM HOLD PEAK VALUES; HBM LOAD PARAM SET; HBM OPEN PMX BROW.; HBM READ PARAM SET; HBM SEND COM.	**Group PMX** This group contains VIs that realize additional features (like loading a certain parameter set or activating TEDs at a certain connector) of PMX devices. Additional features are not covered by the common functionality that is implemented for all device types. These VIs can only be used with PMX devices!
HBM MGC; HBM SEND COM.	**Group MGC** This group contains VIs that realize additional features (like sending a low-level command) of MGC devices. Additional features are not covered by the common functionality that is implemented for all device types. These VIs can only be used with MGC devices!
HBM DEMO	**Group Demo** VIs in this group cover various examples that show how to setup measurements, get measurement values, use additional features of devices or setting digital and analog outputs to certain values. The group contains following VIs: • InteractiveDemo.vi • GetSingleMeasurementDemo.vi • Demo.vi (continuous measurement) • ProgramControlledDemo.vi • MGCProgramControlledDemo.vi • PMXProgramControlledDemo.vi • QuantumXProgramControlledDemo.vi • SettingAnalogOutDemo.vi • SettingDigitalOutDemo.vi • PMXAdditionalFeaturesDemo.vi • ActivateChannelDemo.vi • BuildApplicationDemo.vi • ZeroSomeSignalsBeforeMeasurementDemo.vi • UseCommonApiToSetupScalingAndSensorDemo.vi

9 Examples (Group Demo)

The HBM LabVIEW Driver comes with several demo VIs (located within the Demo folder), that show how to use the VIs and at the same time illustrate the necessary workflow.

9.1 InteractiveDemo.vi

This VI demonstrates a very easy, interactive way to execute a measurement with a minimum number of involved VIs. Many of the VIs of the group User Interface were used. Notice that these VIs work for all supported device types!


This is the easiest, interactive way to execute a measurement.

Initialize API and do the license check (no license required for executables!)

Scan for devices and choose the devices you want to use

Connect the chosen devices

Get all signals of the connected devices

Show only measurable, activated signals

Select signals you want to use

Start an interactive measurement of the selected signals

Disconnect all connected devices

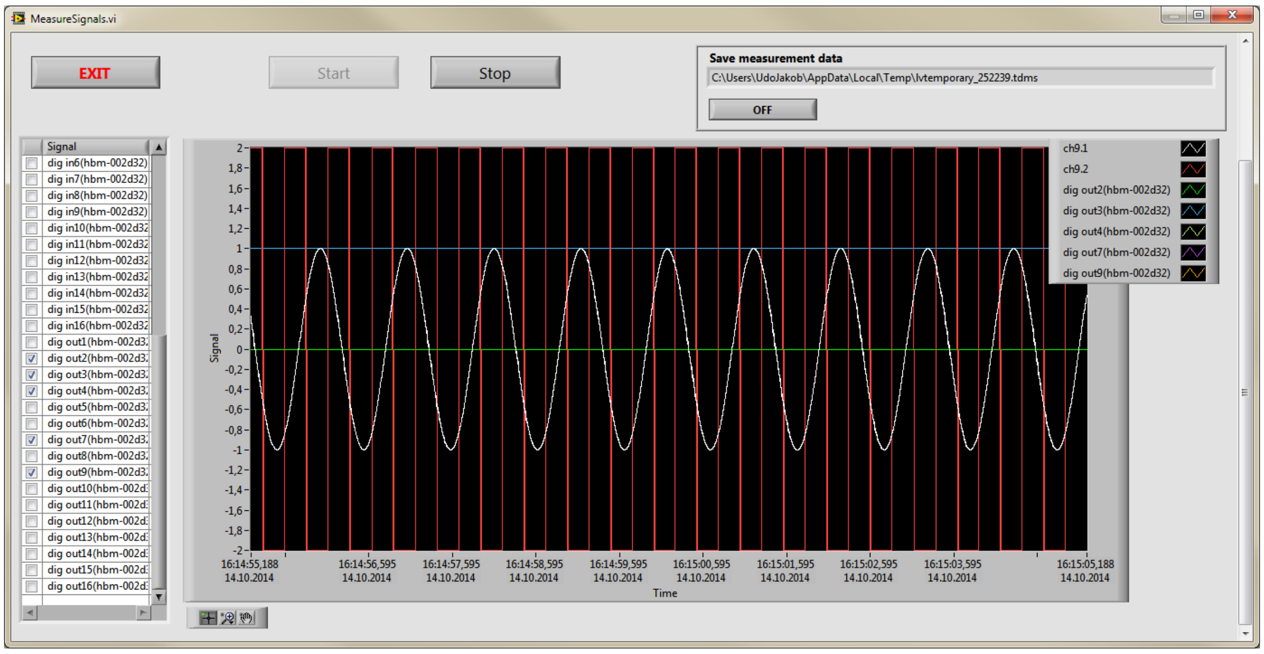
End the usage of the API

All: Use all types of signals

T: Remove all signals that cannot be measured

T: Remove deactivated signals (deactivated signals are likely not proper configuered, so we do not want to select them)

F: Do not remove activated signals, since they can be measured


MeasureSignals.vi

EXIT | Start | Stop

Save measurement data: C:\Users\UdoJakob\AppData\Local\Temp\lvtemporary_252239.tdms

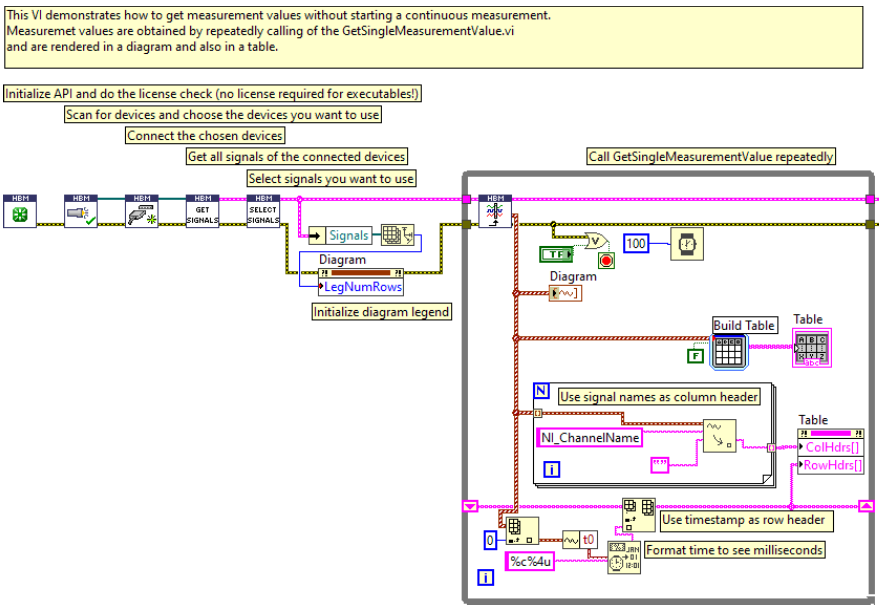
OFF

Signal: dig in6(hbm-002d32), dig in7(hbm-002d32), dig in8(hbm-002d32), dig in9(hbm-002d32), dig in10(hbm-002d32), dig in11(hbm-002d32), dig in12(hbm-002d32), dig in13(hbm-002d32), dig in14(hbm-002d32), dig in15(hbm-002d32), dig in16(hbm-002d32), dig out1(hbm-002d32), ✓ dig out2(hbm-002d32), ✓ dig out3(hbm-002d32), ✓ dig out4(hbm-002d32), dig out5(hbm-002d32), dig out6(hbm-002d32), ✓ dig out7(hbm-002d32), dig out8(hbm-002d32), ✓ dig out9(hbm-002d32), dig out10(hbm-002d32), dig out11(hbm-002d32), dig out12(hbm-002d32), dig out13(hbm-002d32), dig out14(hbm-002d32), dig out15(hbm-002d32), dig out16(hbm-002d32)

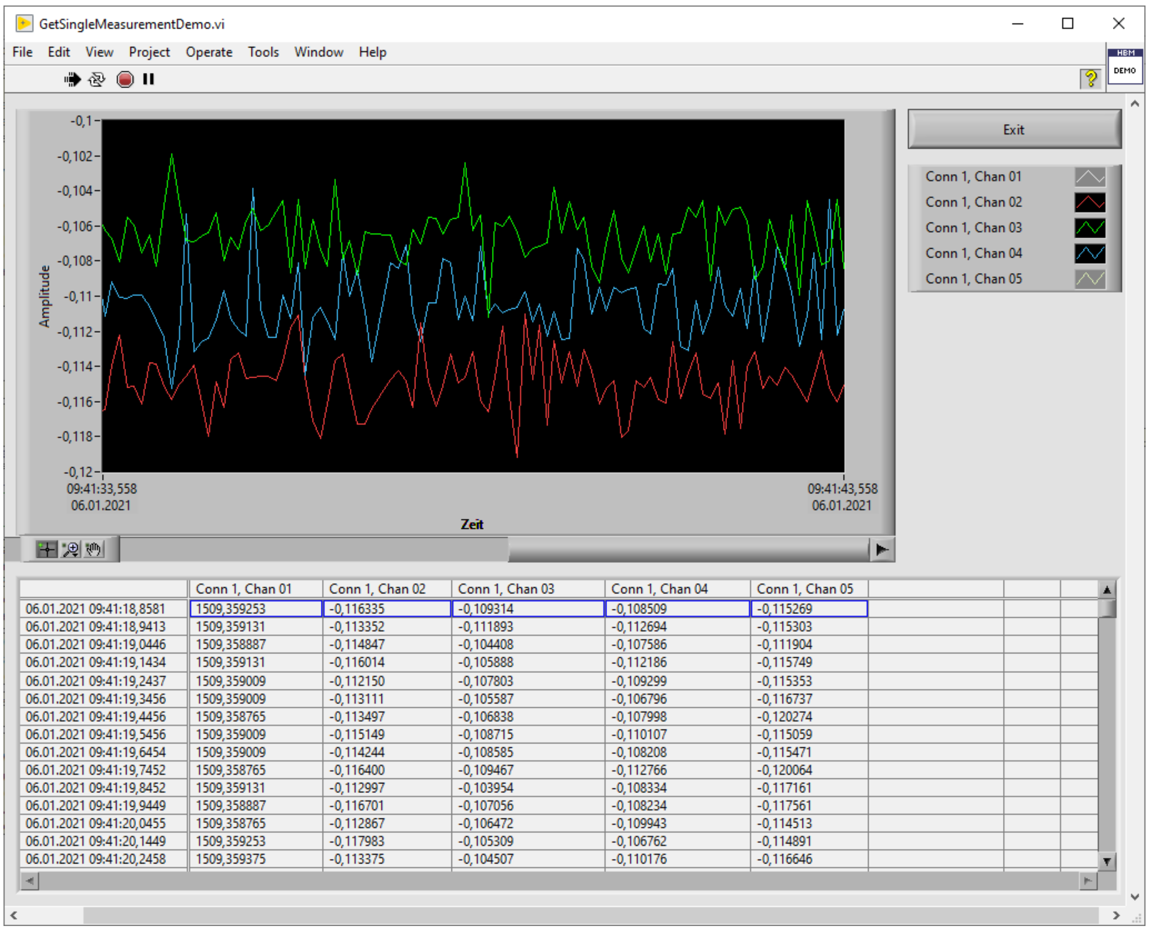
Legend: ch9.1, ch9.2, dig out2(hbm-002d32), dig out3(hbm-002d32), dig out4(hbm-002d32), dig out7(hbm-002d32), dig out9(hbm-002d32)

9.2 GetSingleMeasurementDemo.vi

This VI demonstrates how to get measurement values without starting a continuous measurement.
Measurement values are obtained by a repeated call of the GetSingleMeasurementValue.
Measurement values will be rendered in a diagram and in a table.


This VI demonstrates how to get measurement values without starting a continuous measurement.
Measuremet values are obtained by repeatedly calling of the GetSingleMeasurementValue.vi
and are rendered in a diagram and also in a table.

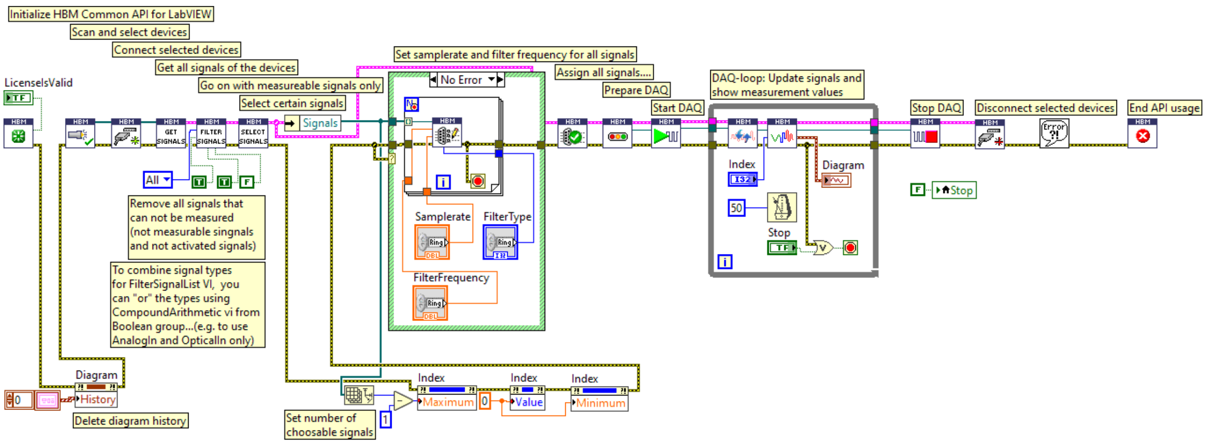
Initialize API and do the license check (no license required for executables!)
Scan for devices and choose the devices you want to use
Connect the chosen devices
Get all signals of the connected devices
Select signals you want to use
Call GetSingleMeasurementValue repeatedly
Disconnect chosen devices
End the usage of the API
Signals
Diagram
LegNumRows
Initialize diagram legend
100
Diagram
Build Table
Table
Use signal names as column header
NI_ChannelName
Table
ColHdrs[]
RowHdrs[]
Use timestamp as row header
Format time to see milliseconds
%c%4u
t0

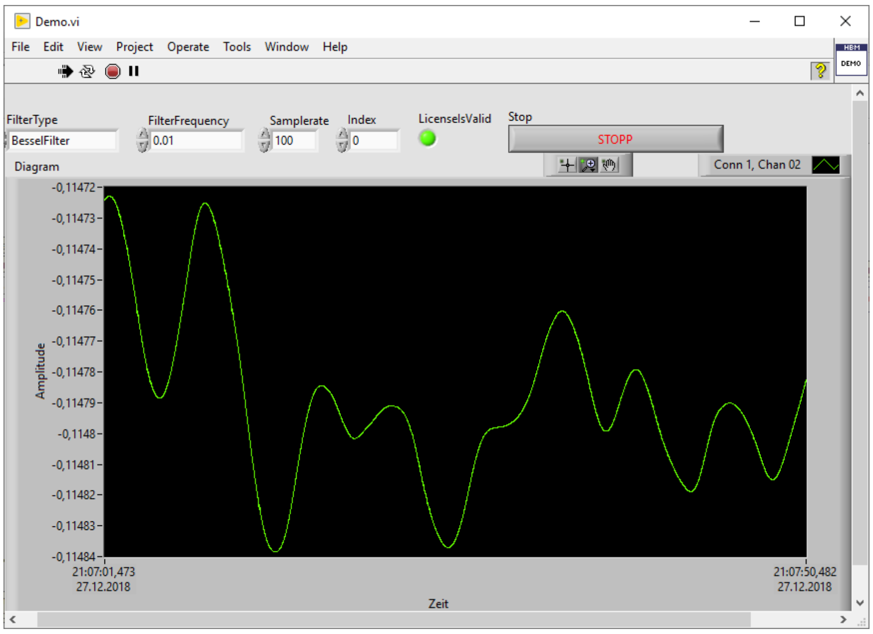


	Conn 1, Chan 01	Conn 1, Chan 02	Conn 1, Chan 03	Conn 1, Chan 04	Conn 1, Chan 05
06.01.2021 09:41:18,8581	1509,359253	-0,116335	-0,109314	-0,108509	-0,115269
06.01.2021 09:41:18,9413	1509,359131	-0,113352	-0,111893	-0,112694	-0,115303
06.01.2021 09:41:19,0446	1509,358887	-0,114847	-0,104408	-0,107586	-0,111904
06.01.2021 09:41:19,1434	1509,359131	-0,116014	-0,105888	-0,112186	-0,115749
06.01.2021 09:41:19,2437	1509,359009	-0,112150	-0,107803	-0,109299	-0,115353
06.01.2021 09:41:19,3456	1509,359009	-0,113111	-0,105587	-0,106796	-0,116737
06.01.2021 09:41:19,4456	1509,358765	-0,113497	-0,106838	-0,107998	-0,120274
06.01.2021 09:41:19,5456	1509,359009	-0,115149	-0,108715	-0,110107	-0,115059
06.01.2021 09:41:19,6454	1509,359009	-0,114244	-0,108585	-0,108208	-0,115471
06.01.2021 09:41:19,7452	1509,358765	-0,116400	-0,109467	-0,112766	-0,120064
06.01.2021 09:41:19,8452	1509,359131	-0,112997	-0,103954	-0,108334	-0,117161
06.01.2021 09:41:19,9449	1509,358887	-0,116701	-0,107056	-0,108234	-0,117561
06.01.2021 09:41:20,0455	1509,358765	-0,112867	-0,106472	-0,109943	-0,114513
06.01.2021 09:41:20,1449	1509,359253	-0,117983	-0,105309	-0,106762	-0,114891
06.01.2021 09:41:20,2458	1509,359375	-0,113375	-0,104507	-0,110176	-0,116646

9.3 Demo.vi

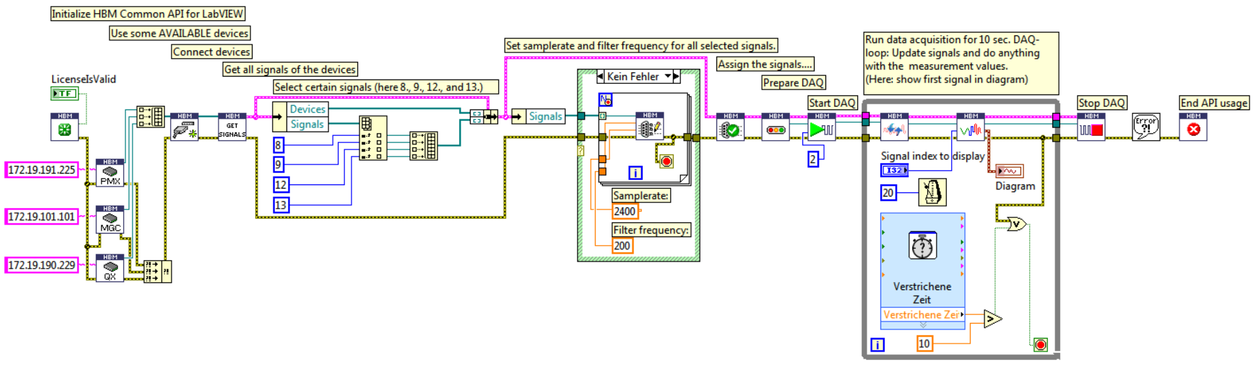
This VI demonstrates how to setup a measurement with certain signals, sample rates and filter frequencies. It also shows how to use the VIs that are required to execute a continuous measurement.

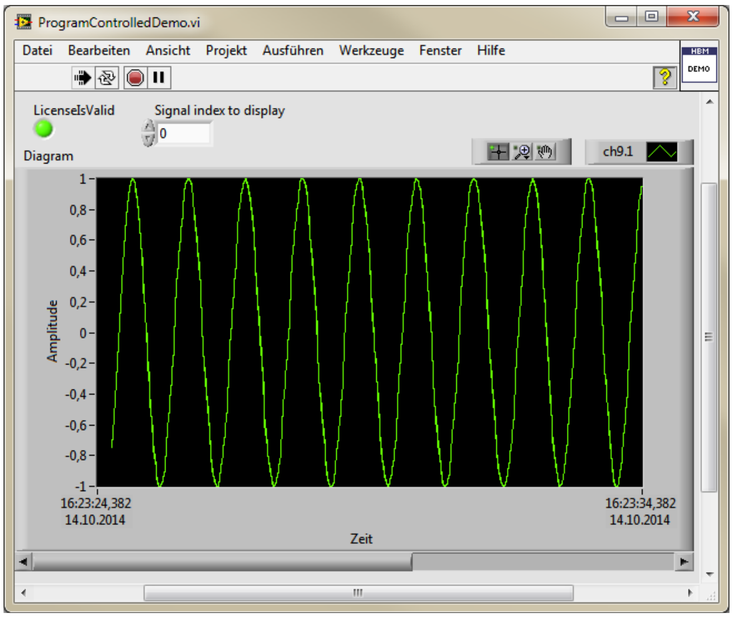

Initialize HBM Common API for LabVIEW
Scan and select devices
Connect selected devices
Get all signals of the devices
Go on with measureable signals only
Select certain signals
Set samplerate and filter frequency for all signals
Assign all signals....
Prepare DAQ
Start DAQ
DAQ-loop: Update signals and show measurement values
Stop DAQ
Disconnect selected devices
End API usage
LicenselsValid
Signals
No Error
All
Remove all signals that can not be measured (not measurable singnals and not activated signals)
To combine signal types for FilterSignalList VI, you can "or" the types using CompoundArithmetic vi from Boolean group...(e.g. to use AnalogIn and OpticalIn only)
Samplerate
FilterType
FilterFrequency
Index
Diagram
Stop
History
Delete diagram history
Set number of choosable signals
Maximum
Value
Minimum



9.4 ProgramControlledDemo.vi

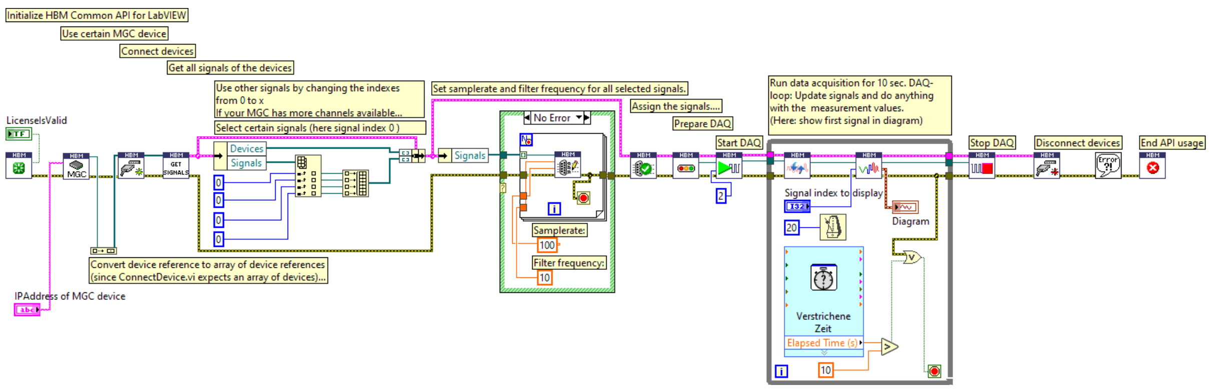
This VI shows how to setup and execute a measurement without any user input. Since most people do not use 3 different DAQ devices at once, you probably have to adapt this demo. Please remove all device types that you do not want to use and adapt "BuildArray.vi" to the new number of used devices. The signal numbers under "Select certain signals (here…)" have also to be adapted to run the demo without an error.

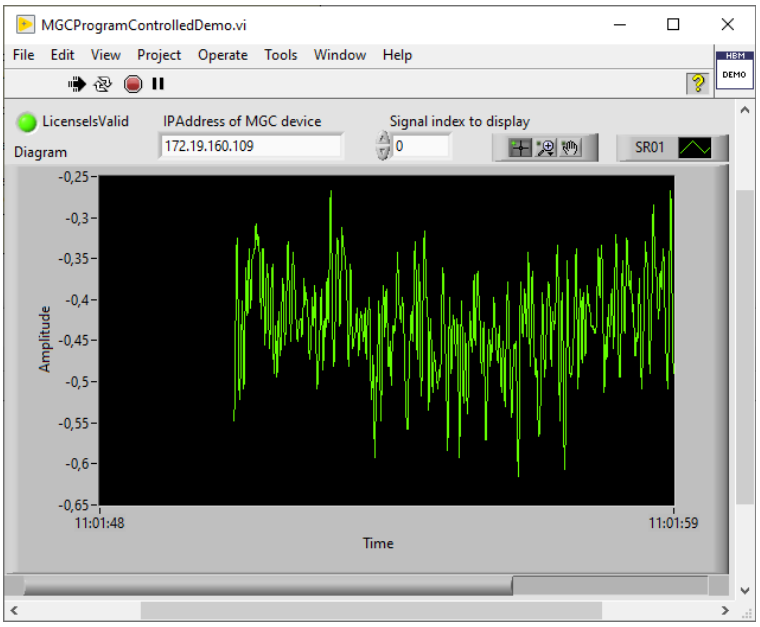

Initialize HBM Common API for LabVIEW
Use some AVAILABLE devices
Connect devices
Get all signals of the devices
Select certain signals (here 8., 9., 12., and 13.)
LicenseIsValid
172.19.191.225
172.19.101.101
172.19.190.229
PMX
MGC
QX
Devices
Signals
8
9
12
13
Signals
Set samplerate and filter frequency for all selected signals.
Kein Fehler
Samplerate:
2400
Filter frequency:
200
Assign the signals....
Prepare DAQ
Start DAQ
2
Run data acquisition for 10 sec. DAQ-loop: Update signals and do anything with the measurement values. (Here: show first signal in diagram)
Signal index to display
20
Diagram
Verstrichene Zeit
Verstrichene Zeit
10
Stop DAQ
End API usage



9.5 MGCProgramControlledDemo.vi

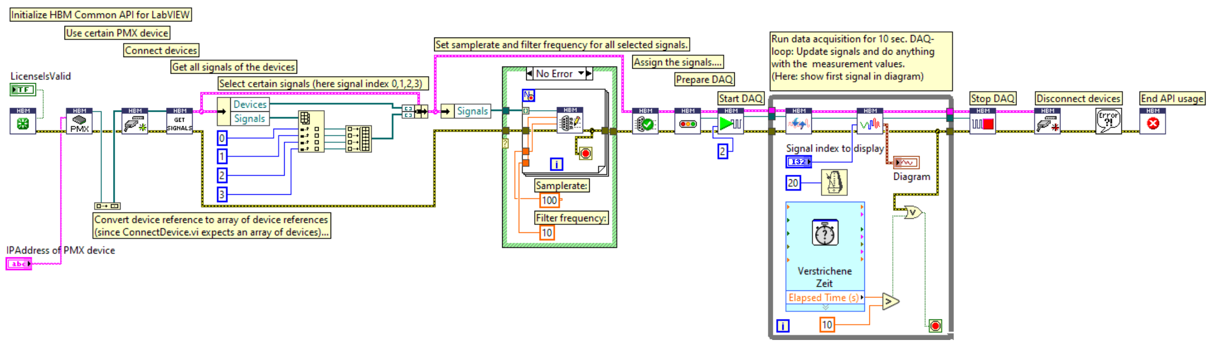
This VI shows how to setup and execute a measurement with an MGC device without any user input. The signal numbers under "Select certain signals (here…)" should be adapted if you want to use more signals than just the very first one.

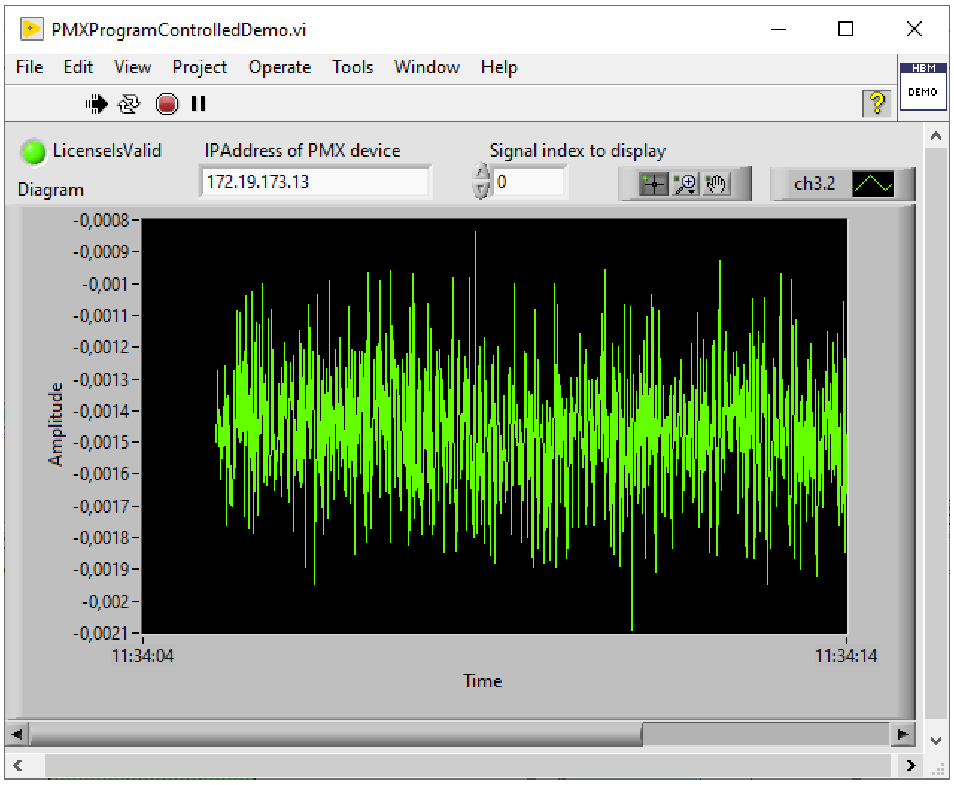

Initialize HBM Common API for LabVIEW
Use certain MGC device
Connect devices
Get all signals of the devices
Use other signals by changing the indexes from 0 to x If your MGC has more channels available...
Select certain signals (here signal index 0)
Set samplerate and filter frequency for all selected signals.
Assign the signals....
Prepare DAQ
Start DAQ
Run data acquisition for 10 sec. DAQ-loop: Update signals and do anything with the measurement values. (Here: show first signal in diagram)
Stop DAQ
Disconnect devices
End API usage
LicenseIsValid
MGC
GET SIGNALS
Devices
Signals
No Error
Samplerate:
100
Filter frequency:
10
Signals
2
Signal index to display
I32
20
Diagram
Verstrichene Zeit
Elapsed Time (s)
10
Convert device reference to array of device references (since ConnectDevice.vi expects an array of devices)...
IPAddress of MGC device



9.6 PMXProgramControlledDemo.vi

This VI shows how to setup and execute a measurement with a PMX device without any user input. The signal numbers under "Select certain signals (here…)" should be adapted if you want to use more signals than just the very first one.
Depending on the amplifiers build into the PMX, it could occur that the signals are not measurable. In this case, you have to adapt the index of the signals to use.


Initialize HBM Common API for LabVIEW
Use certain PMX device
Connect devices
Get all signals of the devices
Select certain signals (here signal index 0,1,2,3)
LicenseIsValid
HBM
PMX
GET SIGNALS
Devices
Signals
0
1
2
3
Convert device reference to array of device references (since ConnectDevice.vi expects an array of devices)...
IPAddress of PMX device
Set samplerate and filter frequency for all selected signals.
No Error
Samplerate:
100
Filter frequency:
10
Assign the signals....
Prepare DAQ
Start DAQ
2
Run data acquisition for 10 sec. DAQ-loop: Update signals and do anything with the measurement values. (Here: show first signal in diagram)
Signal index to display
I32
20
Diagram
Verstrichene Zeit
Elapsed Time (s)
10
Stop DAQ
Disconnect devices
Error ?!
End API usage

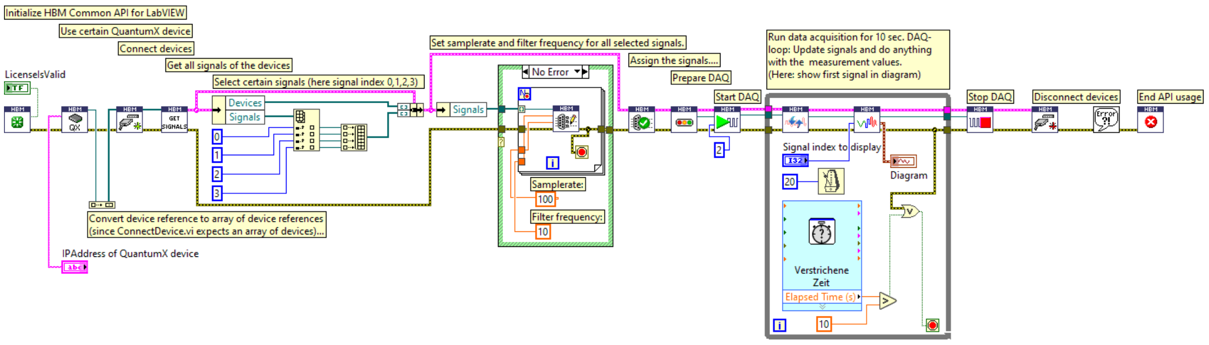


IPAddress of PMX device: 172.19.173.13

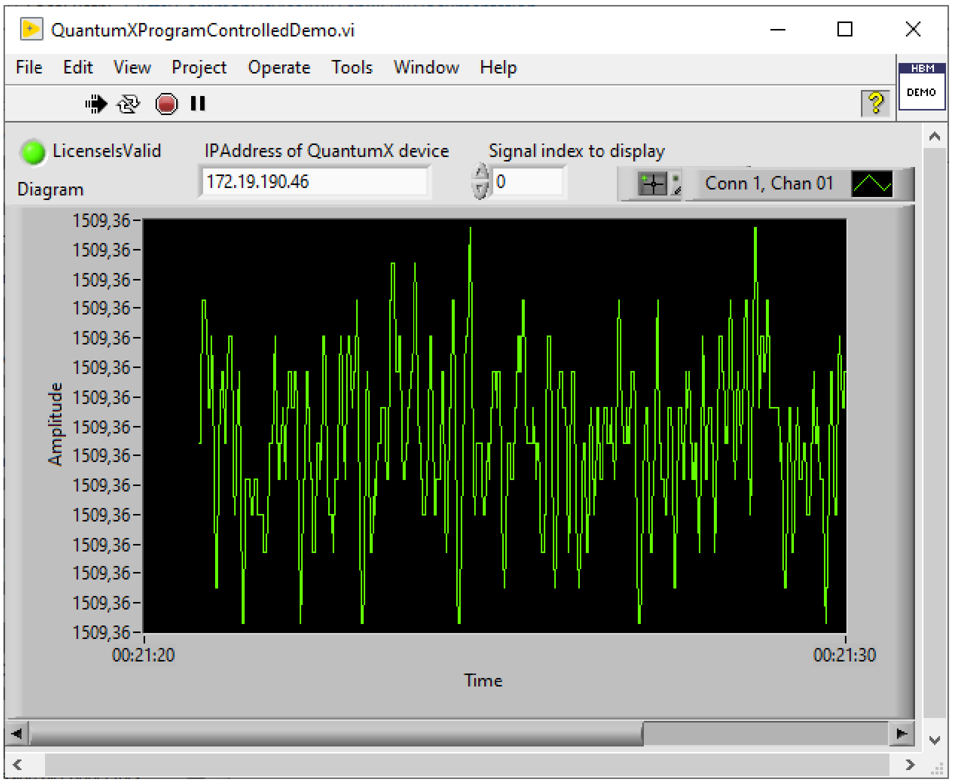
Signal index to display: 0

9.7 QuantumXProgramControlledDemo.vi

This VI shows how to setup and execute a measurement with a QuantumX device without any user input. The signal numbers under “Select certain signals (here…)” should be adapted if you want to use more signals than just the very first one.

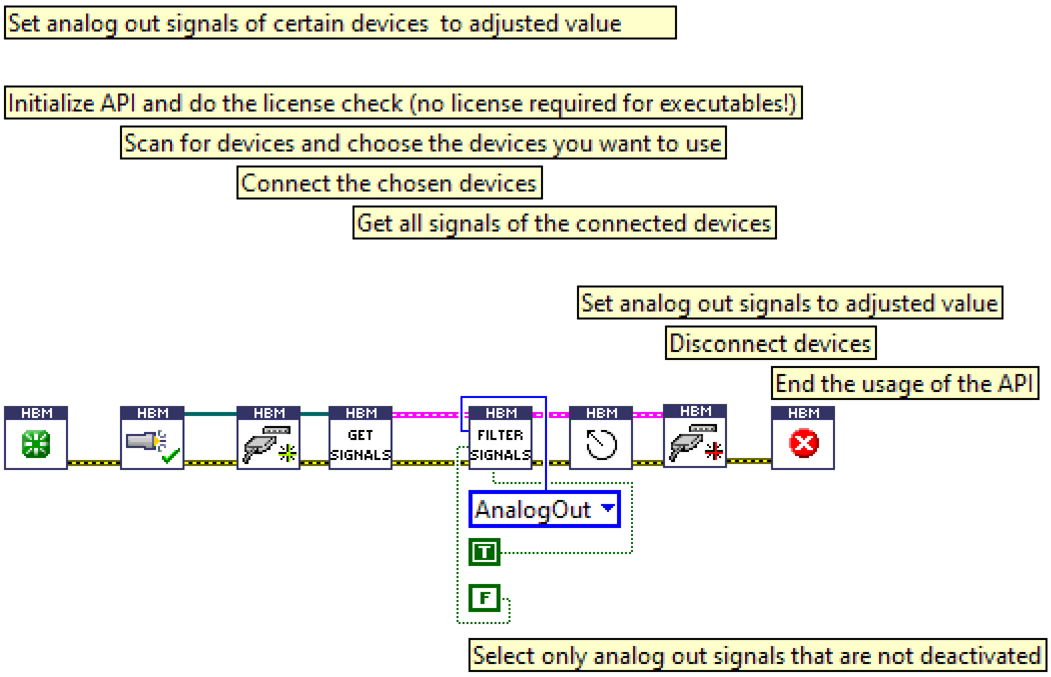


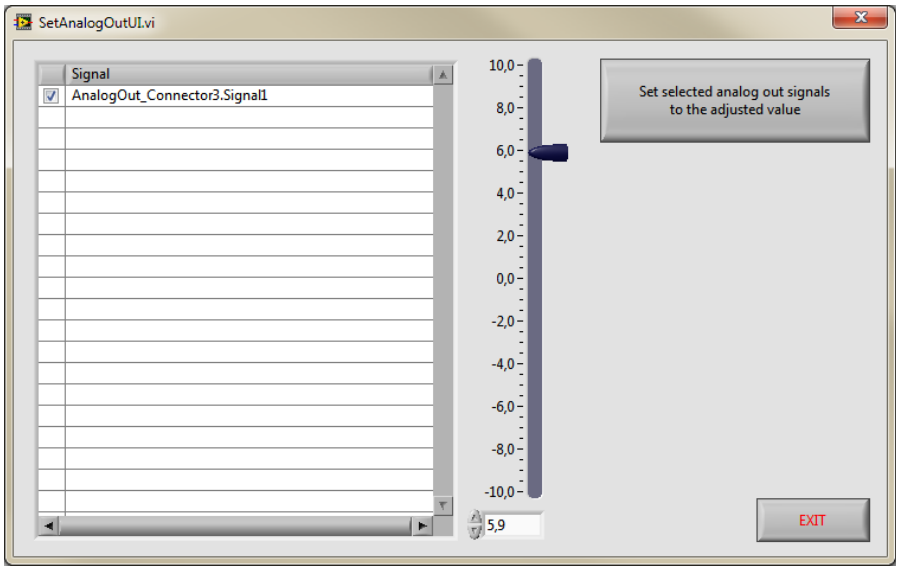
Initialize HBM Common API for LabVIEW
Use certain QuantumX device
Connect devices
Get all signals of the devices
Select certain signals (here signal index 0,1,2,3)
LicensesValid
Devices
Signals
Convert device reference to array of device references (since ConnectDevice.vi expects an array of devices)...
IPAddress of QuantumX device
Set samplerate and filter frequency for all selected signals.
No Error
Samplerate:
100
Filter frequency:
10
Assign the signals....
Prepare DAQ
Start DAQ
Run data acquisition for 10 sec. DAQ-loop: Update signals and do anything with the measurement values. (Here: show first signal in diagram)
Signal index to display
Diagram
Verstrichene Zeit
Elapsed Time (s)
10
Stop DAQ
Disconnect devices
End API usage



9.8 SettingAnalogOutDemo.vi

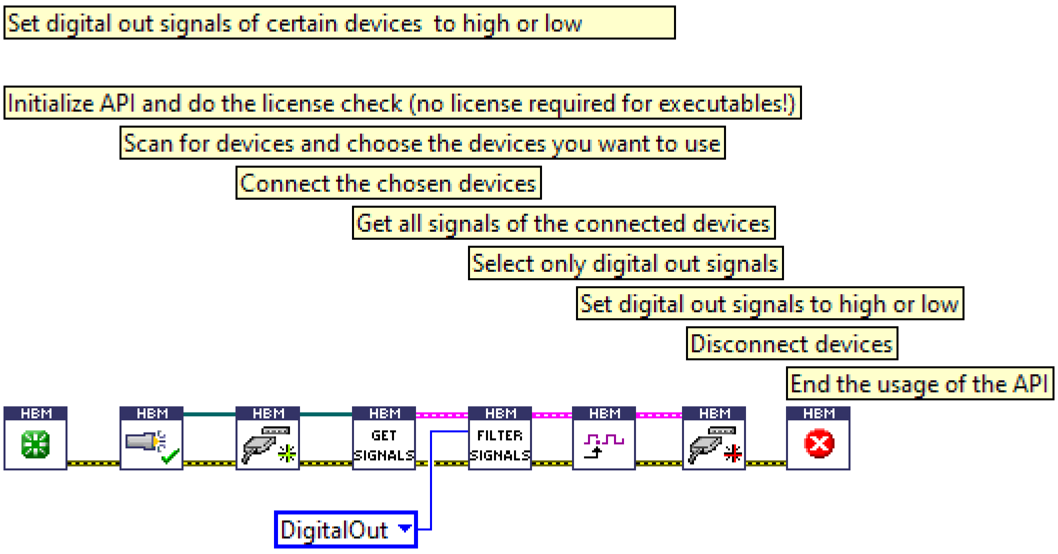
This VI demonstrates how to adjust analog out signals and how to filter certain types of signals (here: analog out signals).

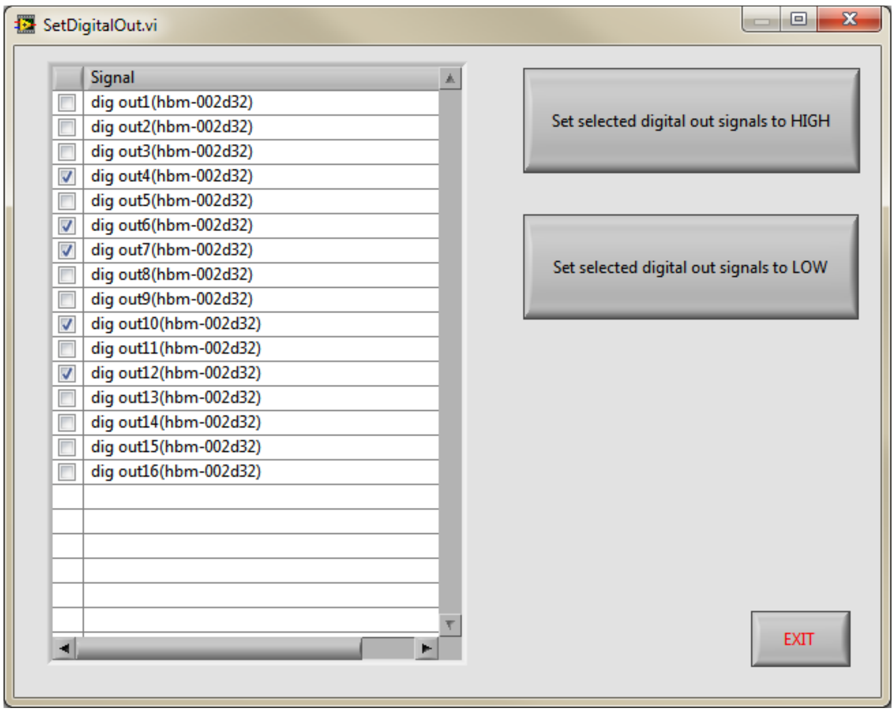

Set analog out signals of certain devices to adjusted value
Initialize API and do the license check (no license required for executables!)
Scan for devices and choose the devices you want to use
Connect the chosen devices
Get all signals of the connected devices
Set analog out signals to adjusted value
Disconnect devices
End the usage of the API
HBM
GET SIGNALS
FILTER SIGNALS
AnalogOut
T
F
Select only analog out signals that are not deactivated


SetAnalogOutUI.vi
Signal
☑ AnalogOut_Connector3.Signal1
10,0
8,0
6,0
4,0
2,0
0,0
-2,0
-4,0
-6,0
-8,0
-10,0
5,9
Set selected analog out signals to the adjusted value
EXIT

9.9 SettingDigitalOutDemo.vi

This VI demonstrates how to adjust digital out signals and how to filter certain types of signals (here: digital out signals).

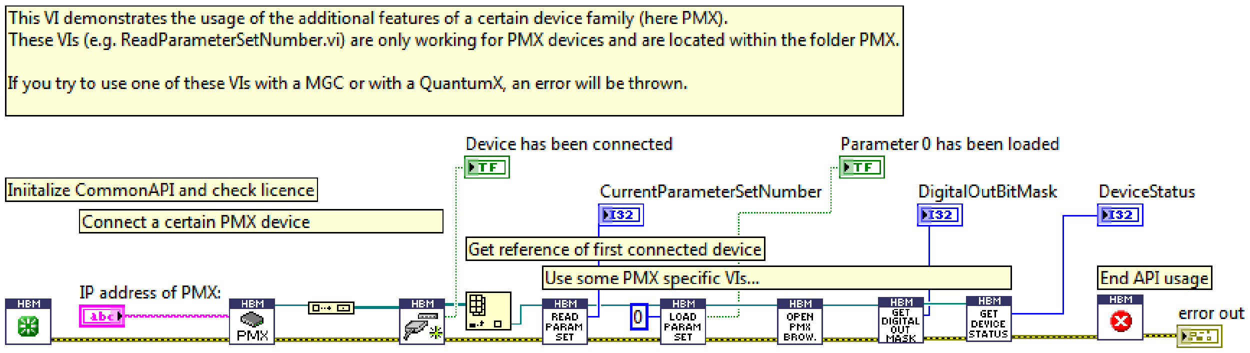

Set digital out signals of certain devices to high or low
Initialize API and do the license check (no license required for executables!)
Scan for devices and choose the devices you want to use
Connect the chosen devices
Get all signals of the connected devices
Select only digital out signals
Set digital out signals to high or low
Disconnect devices
End the usage of the API
HBM
GET SIGNALS
FILTER SIGNALS
DigitalOut


SetDigitalOut.vi
Set selected digital out signals to HIGH
Set selected digital out signals to LOW
EXIT

	Signal
☐	dig out1(hbm-002d32)
☐	dig out2(hbm-002d32)
☐	dig out3(hbm-002d32)
✓	dig out4(hbm-002d32)
☐	dig out5(hbm-002d32)
✓	dig out6(hbm-002d32)
✓	dig out7(hbm-002d32)
☐	dig out8(hbm-002d32)
☐	dig out9(hbm-002d32)
✓	dig out10(hbm-002d32)
☐	dig out11(hbm-002d32)
✓	dig out12(hbm-002d32)
☐	dig out13(hbm-002d32)
☐	dig out14(hbm-002d32)
☐	dig out15(hbm-002d32)
☐	dig out16(hbm-002d32)

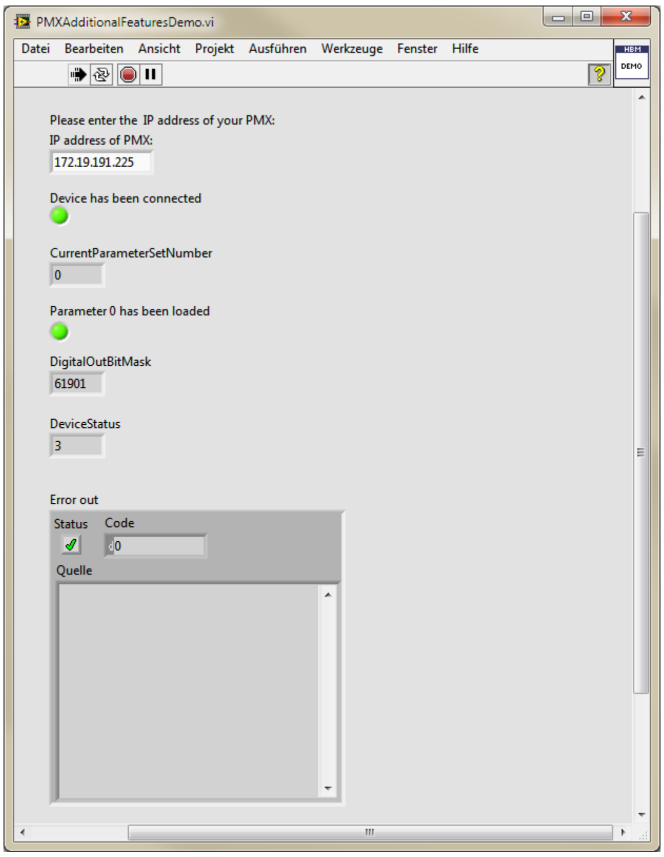
9.10 PMXAdditionalFeaturesDemo.vi

This VI demonstrates the usage of the additional features of a certain device family (here PMX).
These VIs (e.g., ReadParameterSetNumber.vi) are only working for PMX devices and are located within the folder PMX.
If you try to use one of these VIs with a MGC or with a QuantumX, an error will be thrown.


This VI demonstrates the usage of the additional features of a certain device family (here PMX).
These VIs (e.g. ReadParameterSetNumber.vi) are only working for PMX devices and are located within the folder PMX.

If you try to use one of these VIs with a MGC or with a QuantumX, an error will be thrown.

Iniitalize CommonAPI and check licence
Connect a certain PMX device
IP address of PMX:
Device has been connected
CurrentParameterSetNumber
Get reference of first connected device
Use some PMX specific VIs...
Parameter 0 has been loaded
DigitalOutBitMask
DeviceStatus
End API usage
error out
READ PARAM SET
LOAD PARAM SET
OPEN PMX BROW.
GET DIGITAL OUT MASK
GET DEVICE STATUS
PMX


PMXAdditionalFeaturesDemo.vi
Datei Bearbeiten Ansicht Projekt Ausführen Werkzeuge Fenster Hilfe
Please enter the IP address of your PMX:
IP address of PMX:
172.19.191.225
Device has been connected
CurrentParameterSetNumber
0
Parameter 0 has been loaded
DigitalOutBitMask
61901
DeviceStatus
3
Error out
Status Code
✓ 0
Quelle

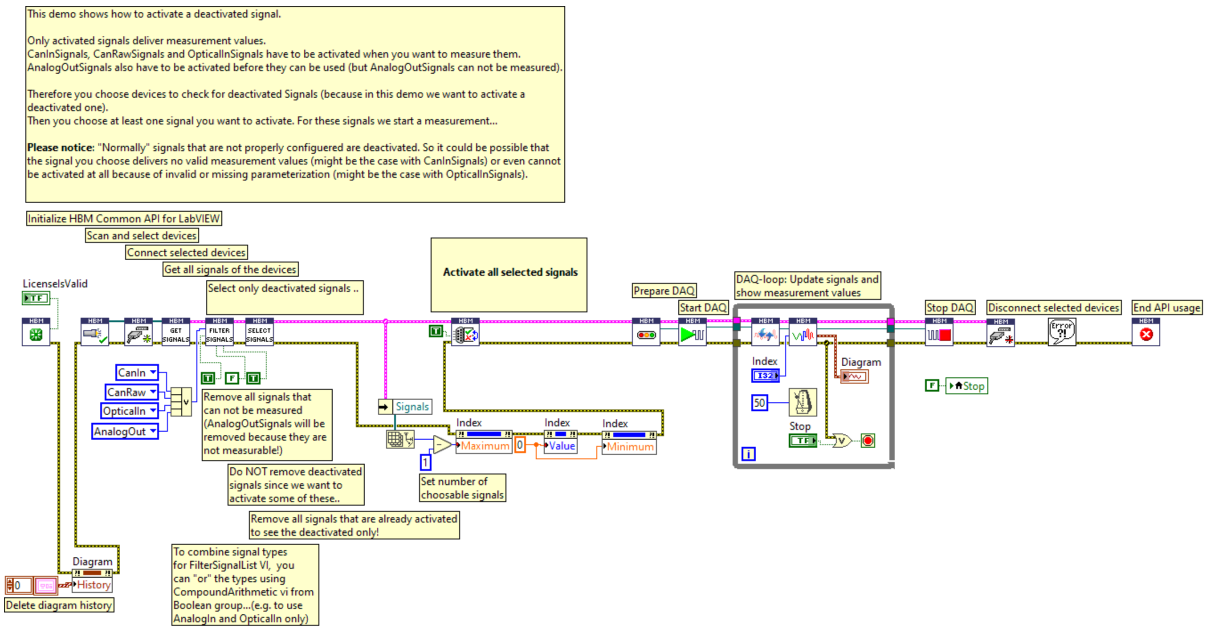
9.11 ActivateChannelDemo.vi

This demo shows how to activate a deactivated signal.

Only activated signals deliver measurement values.
CanInSignals, CanRawSignals and OpticalInSignals have to be activated when you want to measure them.
AnalogOutSignals also have to be activated before they can be used (but AnalogOutSignals cannot be measured).

Therefore, you choose devices to check for deactivated Signals (because in this demo we want to activate a deactivated one).
Then you choose at least one signal you want to activate. For these signals we start a measurement...

Please notice: "Normally" signals that are not properly configured are deactivated. So, it could be possible that the signal you choose delivers no valid measurement values (might be the case with CanInSignals) or even cannot be activated at all because of invalid or missing parameterization (might be the case with OpticalInSignals).



This demo shows how to activate a deactivated signal.

Only activated signals deliver measurement values.
CanInSignals, CanRawSignals and OpticalInSignals have to be activated when you want to measure them.
AnalogOutSignals also have to be activated before they can be used (but AnalogOutSignals can not be measured).

Therefore you choose devices to check for deactivated Signals (because in this demo we want to activate a deactivated one).
Then you choose at least one signal you want to activate. For these signals we start a measurement...

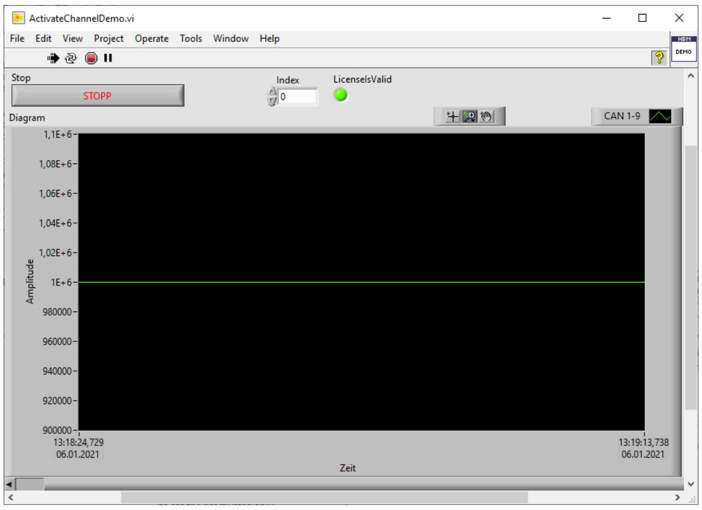
Please notice: "Normally" signals that are not properly configuered are deactivated. So it could be possible that the signal you choose delivers no valid measurement values (might be the case with CanInSignals) or even cannot be activated at all because of invalid or missing parameterization (might be the case with OpticalInSignals).

Remove all signals that can not be measured (AnalogOutSignals will be removed because they are not measurable!)

Do NOT remove deactivated signals since we want to activate some of these..

Remove all signals that are already activated to see the deactivated only!

To combine signal types for FilterSignalList VI, you can "or" the types using CompoundArithmetic vi from Boolean group...(e.g. to use AnalogIn and OpticalIn only)



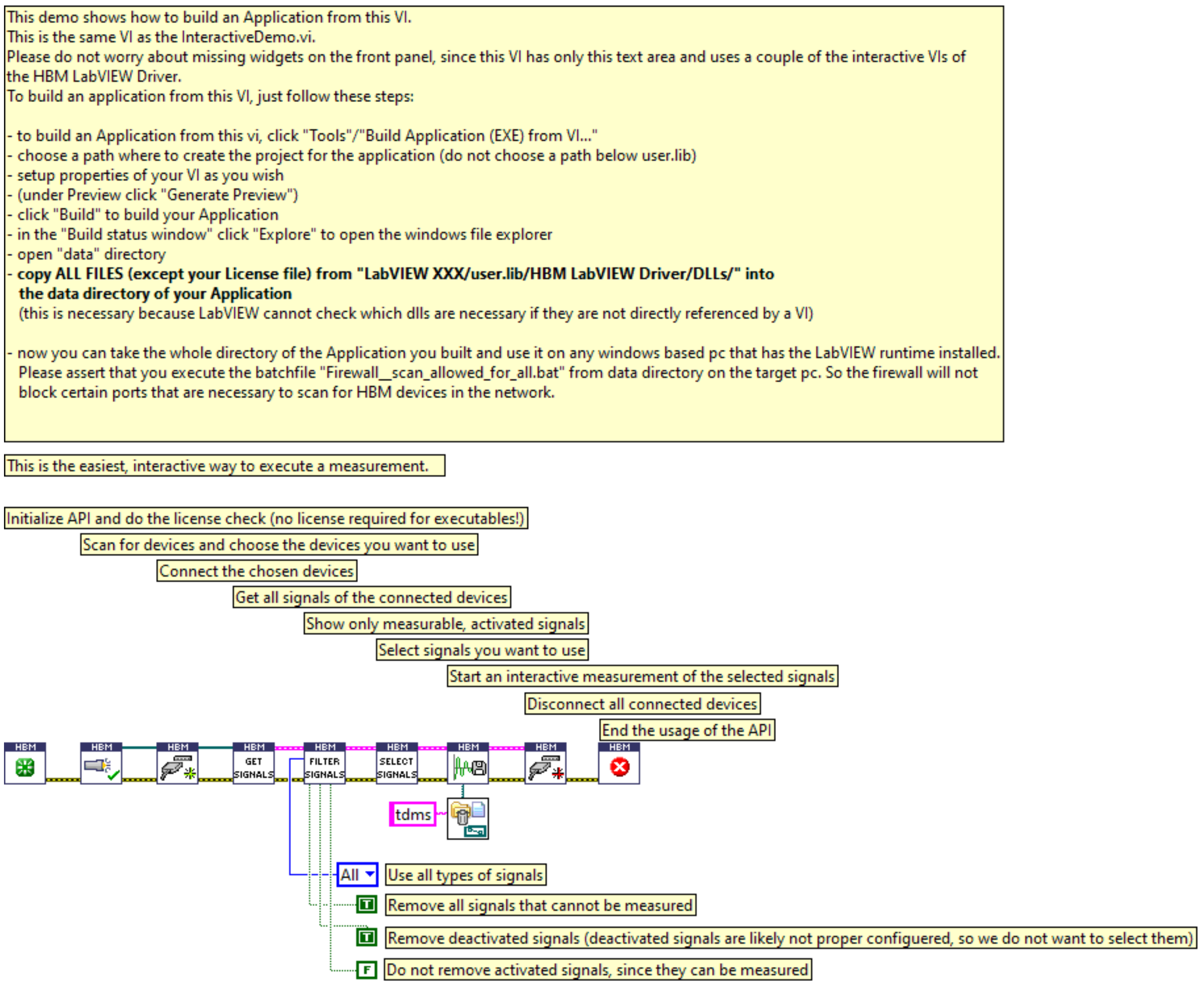
Here we activated a CAN channel that has not been configured correctly before. Because of that, the measurement values are all invalid (delivering 1000000 as value).

9.12 BuildApplicationDemo.vi

This demo shows how to build an Application from this VI. This is the same VI as the InteractiveDemo.vi.
Please do not worry about missing widgets on the front panel, since this VI has only this text area and uses a couple of the interactive VIs of the HBM LabVIEW Driver.
To build an application from this VI, just follow these steps:

- To build an Application from this vi, click "Tools"/"Build Application (EXE) from VI..."
- Choose a path where to create the project for the application (do not choose a path below user.lib)
- Setup properties of your VI as you wish
- (under Preview click "Generate Preview")
- Click "Build" to build your Application
- In the "Build status window" click "Explore" to open the windows file explorer
- Open "data" directory
- Copy ALL FILES (except your License file) from "LabVIEW XXX/user.lib/HBM LabVIEW Driver/DLLs/" into the data directory of your Application (this is necessary because LabVIEW cannot check which Dlls are necessary if they are not directly referenced by a VI)

Now you can take the whole directory of the Application you built and use it on any windows-based pc that has the LabVIEW runtime installed.
Please assert that you execute the batch file "Firewall__scan_allowed_for_all.bat" from data directory on the target pc. So, the firewall will not block certain ports that are necessary to scan for HBM devices in the network.


This demo shows how to build an Application from this VI.
This is the same VI as the InteractiveDemo.vi.
Please do not worry about missing widgets on the front panel, since this VI has only this text area and uses a couple of the interactive VIs of the HBM LabVIEW Driver.
To build an application from this VI, just follow these steps:

- to build an Application from this vi, click "Tools"/"Build Application (EXE) from VI..."
- choose a path where to create the project for the application (do not choose a path below user.lib)
- setup properties of your VI as you wish
- (under Preview click "Generate Preview")
- click "Build" to build your Application
- in the "Build status window" click "Explore" to open the windows file explorer
- open "data" directory
- **copy ALL FILES (except your License file) from "LabVIEW XXX/user.lib/HBM LabVIEW Driver/DLLs/" into the data directory of your Application**
  (this is necessary because LabVIEW cannot check which dlls are necessary if they are not directly referenced by a VI)

- now you can take the whole directory of the Application you built and use it on any windows based pc that has the LabVIEW runtime installed.
  Please assert that you execute the batchfile "Firewall__scan_allowed_for_all.bat" from data directory on the target pc. So the firewall will not block certain ports that are necessary to scan for HBM devices in the network.

This is the easiest, interactive way to execute a measurement.

Initialize API and do the license check (no license required for executables!)
Scan for devices and choose the devices you want to use
Connect the chosen devices
Get all signals of the connected devices
Show only measurable, activated signals
Select signals you want to use
Start an interactive measurement of the selected signals
Disconnect all connected devices
End the usage of the API

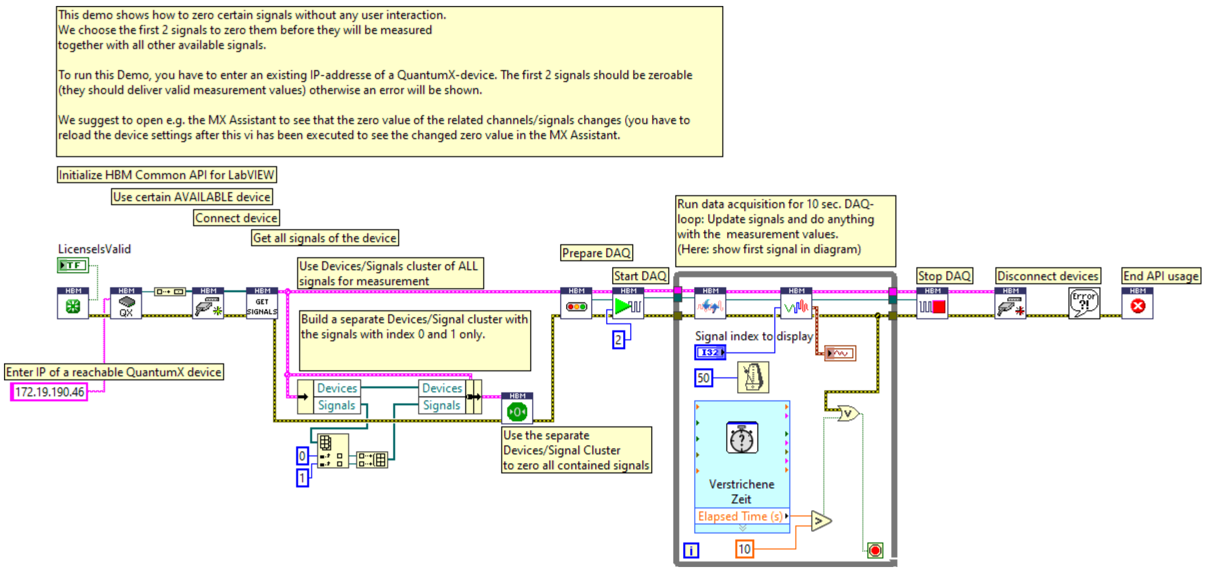
All — Use all types of signals
T — Remove all signals that cannot be measured
T — Remove deactivated signals (deactivated signals are likely not proper configuered, so we do not want to select them)
F — Do not remove activated signals, since they can be measured

9.13 ZeroSomeSignalsBeforeMeasurementDemo.vi

This demo shows how to zero certain signals without any user interaction.
We choose the first 2 signals to zero them before they will be measured
together with all other available signals.

To run this Demo, you have to enter an existing IP-address of a QuantumX-device. The first 2 signals should be zero-able (they should deliver valid measurement values) otherwise an error will be shown.

We suggest opening e.g., the MX Assistant to see that the zero value of the related channels/signals changes (you have to reload the device settings after this vi has been executed to see the changed zero value in the MX Assistant).



This demo shows how to zero certain signals without any user interaction.
We choose the first 2 signals to zero them before they will be measured
together with all other available signals.

To run this Demo, you have to enter an existing IP-addresse of a QuantumX-device. The first 2 signals should be zeroable
(they should deliver valid measurement values) otherwise an error will be shown.

We suggest to open e.g. the MX Assistant to see that the zero value of the related channels/signals changes (you have to
reload the device settings after this vi has been executed to see the changed zero value in the MX Assistant.

Run data acquisition for 10 sec. DAQ-loop: Update signals and do anything with the measurement values.
(Here: show first signal in diagram)

Use Devices/Signals cluster of ALL signals for measurement

Build a separate Devices/Signal cluster with the signals with index 0 and 1 only.

Use the separate Devices/Signal Cluster to zero all contained signals

9.14 UseCommonApiToSetupScalingAndSensorDemo.vi

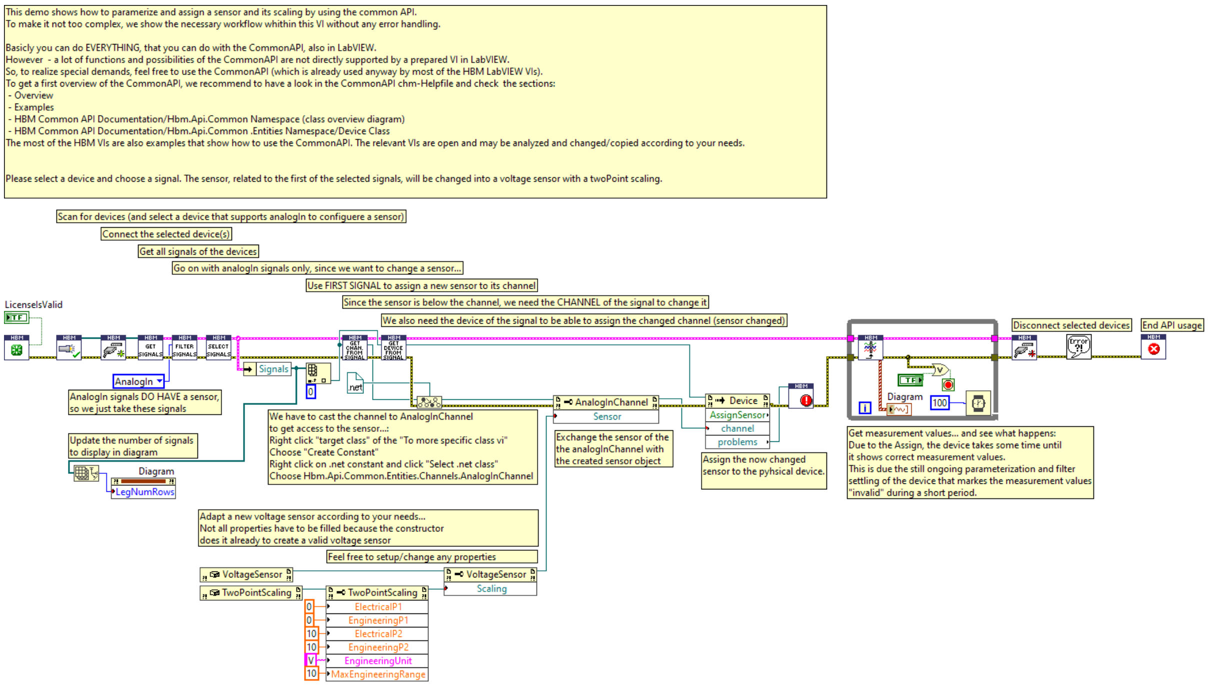
This demo shows how to parameterize and assign a sensor and its scaling by using the common API.
To make it not too complex, we show the necessary workflow within this VI without any error handling.

Basically, you can do EVERYTHING, that you can do with the CommonAPI, also in LabVIEW. However - a lot of functions and possibilities of the CommonAPI are not directly supported by a prepared VI in LabVIEW.
So, to realize special demands, feel free to use the CommonAPI (which is already used anyway by most of the HBM LabVIEW VIs).
To get a first overview of the CommonAPI, we recommend having a look in the CommonAPI chm-Helpfile and check the sections:

- Overview
- Examples
- HBM Common API Documentation/Hbm.Api.Common Namespace (class overview diagram)
- HBM Common API Documentation/Hbm.Api.Common.Entities Namespace/Device Class

The most of the HBM VIs are also examples that show how to use the CommonAPI. The relevant VIs are open and can be analyzed and changed/copied according to your needs.

Please select a device and choose a signal. The sensor, related to the first of the selected signals, will be changed into a voltage sensor with a two Point scaling.



This demo shows how to paramerize and assign a sensor and its scaling by using the common API.
To make it not too complex, we show the necessary workflow whithin this VI without any error handling.

Basicly you can do EVERYTHING, that you can do with the CommonAPI, also in LabVIEW.
However - a lot of functions and possibilities of the CommonAPI are not directly supported by a prepared VI in LabVIEW.
So, to realize special demands, feel free to use the CommonAPI (which is already used anyway by most of the HBM LabVIEW VIs).
To get a first overview of the CommonAPI, we recommend to have a look in the CommonAPI chm-Helpfile and check the sections:
- Overview
- Examples
- HBM Common API Documentation/Hbm.Api.Common Namespace (class overview diagram)
- HBM Common API Documentation/Hbm.Api.Common .Entities Namespace/Device Class
The most of the HBM VIs are also examples that show how to use the CommonAPI. The relevant VIs are open and may be analyzed and changed/copied according to your needs.

Please select a device and choose a signal. The sensor, related to the first of the selected signals, will be changed into a voltage sensor with a twoPoint scaling.

Scan for devices (and select a device that supports analogIn to configuere a sensor)

Connect the selected device(s)

Get all signals of the devices

Go on with analogIn signals only, since we want to change a sensor...

Use FIRST SIGNAL to assign a new sensor to its channel

Since the sensor is below the channel, we need the CHANNEL of the signal to change it

We also need the device of the signal to be able to assign the changed channel (sensor changed)

AnalogIn signals DO HAVE a sensor,
so we just take these signals

Update the number of signals
to display in diagram

We have to cast the channel to AnalogInChannel
to get access to the sensor...:
Right click "target class" of the "To more specific class vi"
Choose "Create Constant"
Right click on .net constant and click "Select .net class"
Choose Hbm.Api.Common.Entities.Channels.AnalogInChannel

Exchange the sensor of the
the analogInChannel with
the created sensor object

Assign the now changed
sensor to the pyhsical device.

Get measurement values... and see what happens:
Due to the Assign, the device takes some time until
it shows correct measurement values.
This is due the still ongoing parameterization and filter
settling of the device that markes the measurement values
"invalid" during a short period.

Disconnect selected devices

End API usage

Adapt a new voltage sensor according to your needs...
Not all properties have to be filled because the constructor
does it already to create a valid voltage sensor

Feel free to setup/change any properties

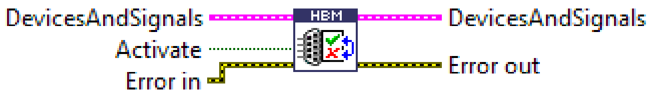
10 LabVIEW Driver Details

10.1 Group System

10.1.1 ActivateSignals.vi

Activates or deactivates the given signals (according to the Activate setting). Only activated signals can be used for a measurement!
Supported signal types are: CanInSignal, CanRawSignal, FbgSignal (optical), and AnalogOutSignal.
All signals that can be activated or deactivated will be treated. Signal types that do not support activation/deactivation will be ignored.
It is not necessary to execute AssignSignal.vi afterwards since the signals will be activated/deactivated on the device directly. Error out delivers problems that occurred during activation/deactivation (e.g., due to no longer connected device).

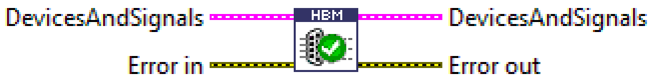
Please notice: Activating signals that are not properly configured will cause problems!



DevicesAndSignals	Cluster consisting of an array of devices and an array of signals.
Activate	Set to true to activate a signal (actually a channel), false to deactivate.

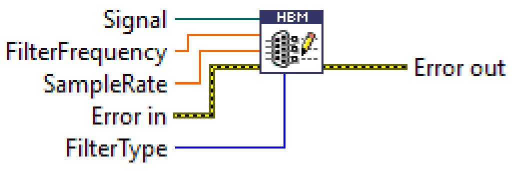
10.1.2 AssignSignals.vi

Assigns the settings of the given signals to the physical signal of the according device.



10.1.3 ConfigSignal.vi

Configurates the signal according to given parameters. If the signal does not support a filter frequency (e.g., a virtual signal like a calculated channel) NO error will be created here.
Remarks: To configure the physical device with these settings, you have to use the AssignSignals VI when the signal is configured.

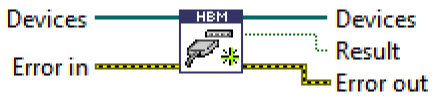


Signal	The signal you want to configurate
SampleRate	Sample rate that should be used for measurement
FilterFrequency	Filter frequency to use
FilterType	FilterType to use. PresentFilter is the default setting, meaning that the currently active FilterType as set on the device will be used

Please notice: Not all FilterTypes and SampleRate/FilterFrequency combinations are supported by all devices. If you choose an invalid one, an error will be delivered when you assign the settings to the device via AssignSignal.vi.

10.1.4 ConnectDevice.vi

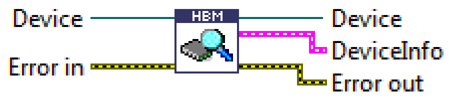
Connects all given devices.



Devices	Array of devices to connect
Result	True if no warning or error occurred during connect

10.1.5 DeviceInfo.vi

Delivers various device properties.

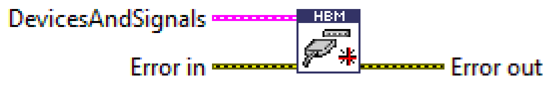


Device	Device whose properties should be read
DeviceInfo	Cluster that contains information about the device

The DeviceInfo cluster contains the following information:
FamilyName, Name, Model, SerialNo, FirmwareVersion, HardwareVersion, ProtectionType, IsConnected, IsReadingDaqValues, IsUnsupportedModel, Timeout, SyncMode

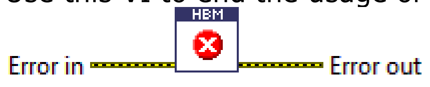
10.1.6 DisconnectDevice.vi

Disconnects all given devices.



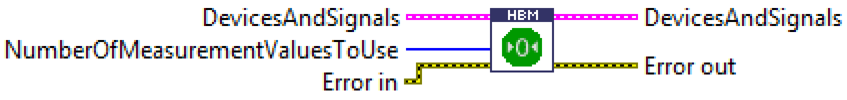
10.1.7 DisposeApi.vi

Use this VI to end the usage of the CommonAPI.



10.1.8 ExecuteZeroing.vi

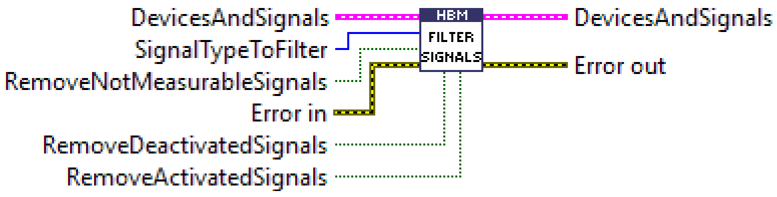
Executes a zero balancing for all signals (notice that actually the channels zero offset will be set and therefore other signals that belong to the same channel are also affected).



DevicesAndSignals	Array of devices and array of signals. Zeroing is done for all signals within the array. Zeroing can only be executed for signals that deliver valid measurement values!
NumberOfMeasurement ValuesToUse	Defines how many measurement values of each signal will be used to calculate an average mean value that will be used as zero offset for the signals (default value is 1).

10.1.9 FilterSignalList.vi

Filters all given signals according to the given signal type and further properties.

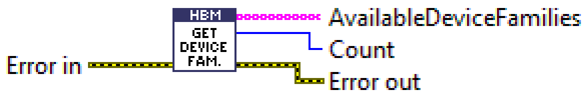


SignalTypeToFilter	Signal type (you may add different signal types, e.g.: DigitalOut + AnalogOut) you want to filter. After execution the devices and signals output contains these signals only. Default value is "All" - so all signal types will pass this VI.
RemoveNotMeasurableSignals	Default is false. Signals that cannot be measured pass the VI. Set this to true to use only measurable signals (e.g., no AnalogOutSignals).
RemoveDeactivatedSignals	Default is true. Deactivated signals are likely not proper configured.
RemoveActivatedSignals	Default is false. Set to true, to let only deactivated signals pass.

ATTENTION PLEASE: You have to close the reference (.net - CloseReference.vi) of Signals (of the DevicesAndSignals cluster output). Otherwise, you will get a memory leak after using this vi very often.

10.1.10 GetAvailableDeviceFamilyNames.vi

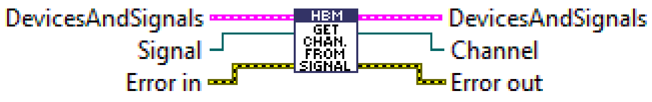
Determines a list of all available device families (device drivers).



AvailableDevices	Array of available device family names
Count	Number of available device families

10.1.11 GetChannelFromSignal.vi

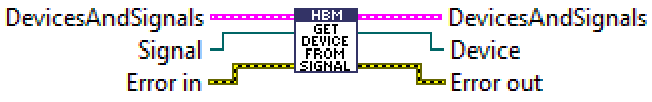
Finds the channel to which the given signal belongs.



Signal	Signal to check to which channel it belongs
Channel	The channel to which the signal belongs

10.1.12 GetDeviceFromSignal.vi

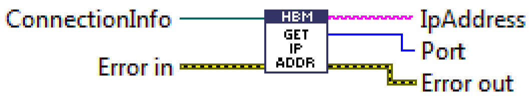
Finds the device to which the given signal belongs.



Signal	Signal to check to which device it belongs
Device	The device to which the signal belongs

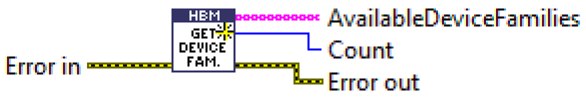
10.1.13 GetIpAddressFromConectionInfo.vi

Get the IP address and the port of a connection info.



10.1.14 GetScanableDeviceFamilyNames.vi

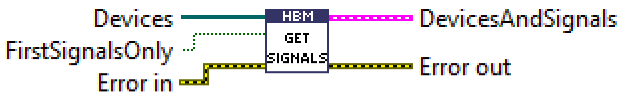
Determines a list of all available device families (device drivers).



AvailableDeviceFamilies	Array of available device family names
Count	Number of available device families

10.1.15 GetSignals.vi

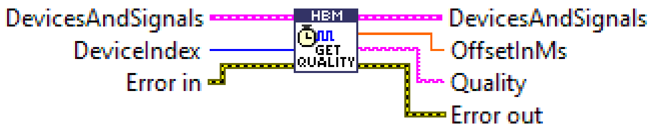
Delivers (all) signals of the given devices.



Devices	Devices from which to deliver the signals
FirstSignalsOnly	Set to true (default) if you only want the get the respective first signals of each channel (QuantumX devices return 2 signals per channel if FirstSignalsOnly is set to false).
DevicesAndSignals	Cluster consisting of an array of devices and an array of all signals of the given devices

10.1.16 GetSynchronizationQuality.vi

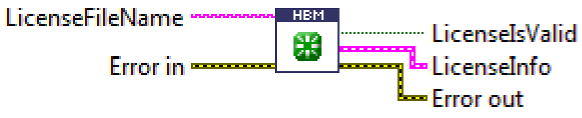
Returns the offset (timespan between synchronization time source and time of device) in ms and a TimeSource dependent string with further information about the quality of the synchronization.
For NtpTimeSource this string includes the parameters (comma seperated): "remote, refid, st, t, when, poll, reach, delay, offset, jitter".
For PtpTimeSource this string includes the following parmeters (comma seperated): "GrandmasterID, SyncMode, TimeScaleFlags, UtcOffset, UtcOffsetUsed, MasterOffset".



DevicesIndex	Index of the device to get the time source quality information from.
OffsetInMs	Offset to time source in milliseconds, double.MaxValue if offset cannot be determined
Quality	TimeSource type dependent information about the synchronization quality.

10.1.17 Init.vi

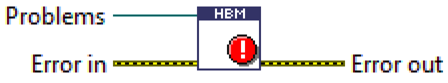
Initializes the underlying common API.
If this VI runs within LabVIEW IDE the given license file will be verified.



LicenseFileName	Full path and filename of the license file. If this string is empty the license file is searched within the DLL directory (search pattern is "*.license").
LicenseIsValid	True if given license file is valid or LabVIEW runtime is used.
LicenseInfo	Information about the license holder (if license is not an evaluation license).

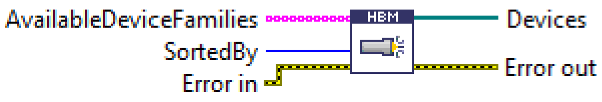
10.1.18 Problems.vi

Generates an error, if the problems (e.g., because of an assign function) contain any errors. Generates a warning if the problems contain any warning but no error.



10.1.19 ScanForDevices.vi

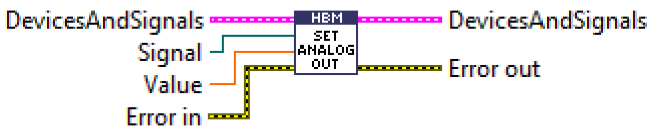
Scans for devices. Repeat this several times to find more devices.
Notice: Typically, you have to wait 6 seconds after the execution of the Init.vi until all devices that support the scan mechanism are collected and will be returned by this VI. So, if you want to use the Scan.vi only once (e.g., within a non-interactive workflow), you have to assert that 6 seconds passed since executing the Init.vi to be sure to get all available devices.



AvailableDeviceFamilies	Array of device family names (e.g., PMX, QuantumX) you want to scan. Leave this connection empty to find all scannable devices of all device families.
SortedBy	Enumeration that defines the property (Name, IpAddress or SerialNumber) that will be used to sort the found devices.
Devices	Found and sorted devices according to given device family names and sorting settings.

10.1.20 SetAnalogOut.vi

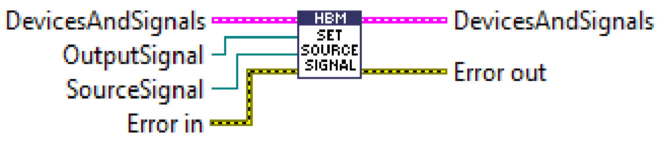
Sets the analog out signal to the given value.



Signal	Signal (has to be of type AnalogOutSignal) to adjust
Value	Value to set

10.1.21 SetAnalogOutSourceSignal.vi

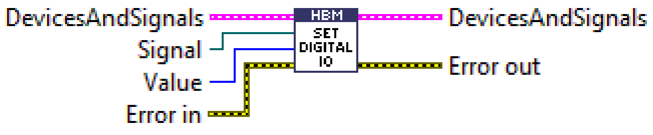
Sets the source signal for an analog out signal to the given source.



OutputSignal	Analog out signal to configure (has to be of type AnalogOutSignal).
SourceSignal	Signal whose measurement values should be transformed into an output voltage at the physical connector of the given output signal (according to the output scaling settings of the output signal).

10.1.22 SetDigitalIO.vi

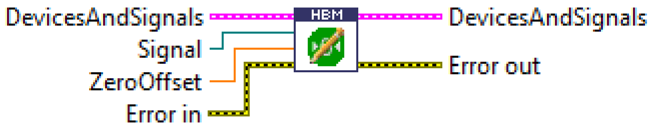
Sets the digital signal according to the given value.



Signal	Signal (has to be of type DigitalSignal) to adjust
Value	DigitalValueType (High or Low) to set

10.1.23 SetZeroOffset.vi

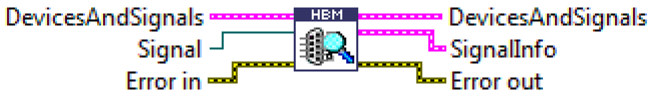
Sets the zero offset of the given analog in signal to the given value.



Signal	Analog in signal whose zero offset should be set (notice that actually the channels zero offset will be set and therefore other signals that belong to the same channel are also affected).
ZeroOffset	Zero offset you want to use.

10.1.24 SignalInfo.vi

Delivers various signal properties.



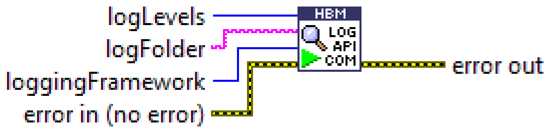
Signal	Signal whose properties should be read
SignalInfo	Cluster that contains information about the signal

The SignalInfo cluster contains the following information:
Name, SynchronMode, SampleRate, FilterFrequency, UpdatedValueCount, Values, Timestamps, States, BufferOverrunOccurred, EngineeringUnit, ChannelName.

10.1.25 StartLogging.vi

This VI logs the actions of the HBM Common API, which is used by this LabVIEW driver. It can help to find problems, related to errors or issues that occur when using the VIs of this driver.

Using this VI will slow down your program. Please do not use it, until HBM support asks you to.

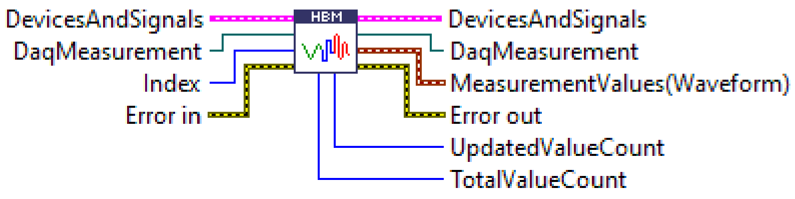


LoggingFramework	Framework that is used for logging (NLog or Loupe is supported)
LogLevel	Determines which actions will be logged. Select "All" for full logging.
LogFolder	Directory name, where the logfile will be created.

10.2 Group DAQ

10.2.1 GetMeasurementValues.vi

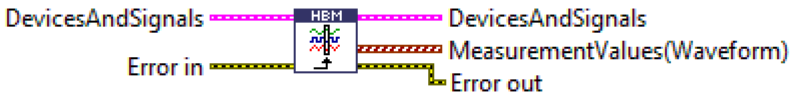
Delivers measurement values in a standard LabVIEW format (first timestamp of the measurement values in utc time format, the samplerate in 1/samplerate in Hz, and an array of measurement values).



DaqMeasurement	Reference to the DaqMeasurement instance
Index	The index of the signal within the given array of signals from which the measurement values should be delivered.
UpdatedValueCount	Number of updated values by the last execution of UpdateMeasurementValues.vi.
TotalValueCount	Total number of measurement values during the running data acquisition for the signal with the given index.

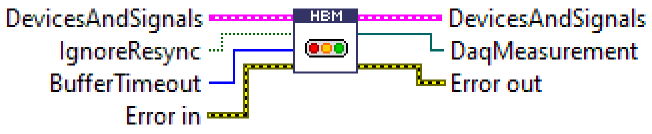
10.2.2 GetSingleMeasurementValues.vi

Obtains one single measurement value for each given signal without initializing a continuous measurement. Filter settings will be ignored.



10.2.3 PrepareDAQ.vi

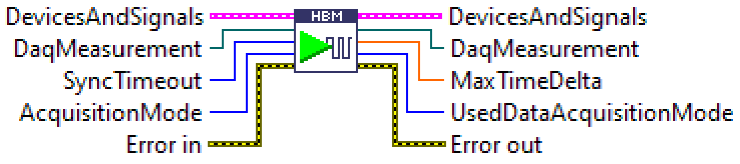
Prepares a continuous measurement.
If any signal does not belong to the given devices or the device does not support the demanded sample rate, an error will be thrown.



DevicesAndSignals	Array of devices and their signals that have been chosen for measurement
IgnoreResync	Determines if re-synchronization of a time source during a running data acquisition should be ignored. At present this is only possible for streaming devices (QuantumX family). Set to false if data acquisition should be stopped in case of a detected re-synchronization (the default is true - so if you do not connect it, resync will always be ignored)
BufferTimeout	Buffer timeout in milliseconds (default is 3000ms). Used to calculate size of internal circular buffer. At least there is a buffer of 1000 values for each signal. Normally the buffer size is (bufferTimeout/1000) * sample rate of the signal, prior added to the measurement (E.g.: Signal.SampleRate=1200Hz, bufferTimeOut=1000ms => size of internal circular buffer is 1200 entries)
DaqMeasurement	Reference to the DaqMearurement object that handles the measurement

10.2.4 StartDaq.vi

Starts (synchronized) measurement of all signals that were added to the measurement.

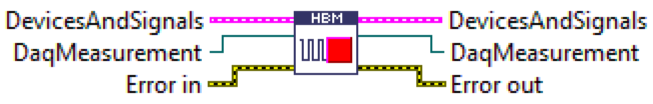


DevicesAndSignal	Devices and Signals that are used for the measurement (just passed through).
SyncTimeout	Maximum time in ms, that is used to start a synchronized measurement. If it is not possible to start a synchronized measurement within this time, an error will be thrown.
AcquisitionMode	Modus of the data acquisition. SoftwareSynchronized = 0 HardwareSynchronized = 1 Unsynchronized = 2 Auto = 3 Use SoftwareSynchronized, if first timestamp of all signals should be as close as possible to each other. Therefore, the devices have to be synchronized! Use HardwareSynchrnonized, if the devices are synchronized by hardware (cable) and deliver a common first timestamp for each signal. Use Unsynchronized to start a measurement without a common first timestamp for all signals (in that case syncTimeOut is not considered and the function returns with 0.0) Use Auto mode to let the API decide which mode should be used. The CommonAPI tries to use the most suitable mode - if devices are not synchronous to each other, unsynchronized mode will be chosen.

MaxTimeDelta	Maximum difference between time stamps of the first measurement values. 0 means, that the first time stamp of each signal has the same value. UsedDataAcquisitionMode: Returns the used DataAcquisitionMode which has been used to start the latest measurement. If the latest measurement has been started with DataAcquisitionMode.Auto, the used data acquisition mode will be the real mode that has been chosen (Unsynchronized, HardwareSynchronized or TimestampSynchronized).

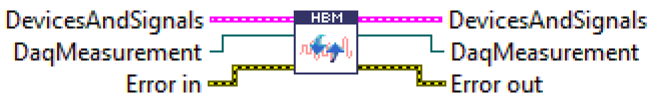
10.2.5 StopDaq.vi

Stops data acquisition of all devices that take part in the measurement



10.2.6 UpdateMeasurementValues.vi

Updates the signals measurement values. Call this function periodically during a measurement.

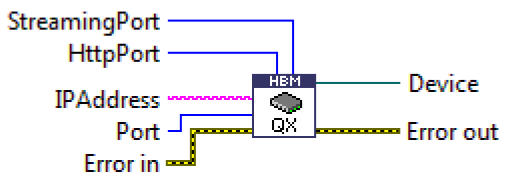


This function updates all measurement values of all signals that take part in measuring.
In case of a synchronized start of data acquisition, it asserts that each signal with same sample rate (distributed to various devices) gets the same number of new measuring values. E.g., signal_A on device 1 with sample rate 20Hz gets the same number of new measurement values as signal_B on device 2 with sample rate 20 Hz!
Otherwise (if StartDaq started an unsynchronized measurement, the signals get all measurement values that are accumulated since the last call of this function.
This function also asserts that there is enough allocated memory for the measurement values of each signal.

10.3 Group QuantumX

10.3.1 QuantumX_Device.vi

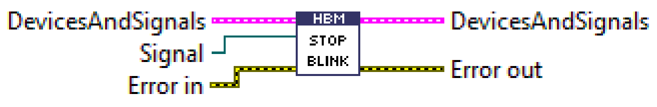
Generates a new QuantumX device.



IPAddress	IPAddress of the device you want to connect with
Port	Port of the device you want to connect with (default port is 5001)
HttpPort	Http port of the device (default is 80)
StreamingPort	Port that is used for streaming (default is 7411)

10.3.2 QuantumX_DisableBlinking.vi

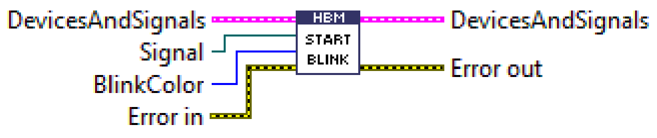
Disables blinking at the connector to which the given signal belongs.



Signal	Signal, whose connector should stop blinking.

10.3.3 QuantumX_EnableBlinking.vi

Enables blinking at the connector to which the given signal belongs.

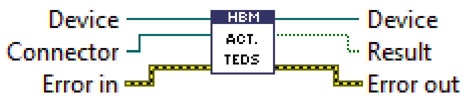


Signal	Signal, whose connector should start blinking in the given color.
BlinkColor	Blinking color of the LED.

10.4 Group PMX

10.4.1 PMX_ActivateTEDs.vi

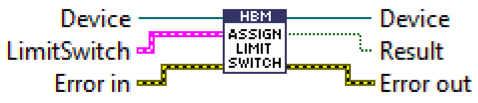
Loads and activates TEDs settings at given connector.
On success, the sensor object of the given connector will be replaced by an updated version according to TEDs settings



Connector	Connector to which the TEDs is connected
Result	True, if TEDs settings could be loaded and activated

10.4.2 PMX_AssignLimitSwitch.vi

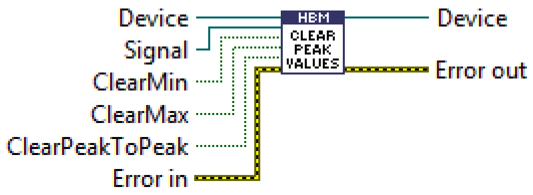
Assigns LimitSwitch settings to the given device
If a limit switch with an already in use LimitSwitchNumber is assigned, that existing limit switch will be overwritten.



Result	True, if assignment was successful, otherwise false
LimitSwitch	Limit switch to assign
	LimitSwitch cluster contains the following element:
LimitSwitchNumber	Number of the limit to assign (1 ...32)
Enabled	Determines if limit switch operation is activated.
Hysteresis	Hysteresis value. Dependents on OperationDirection, it may also be used to define the band span.
IgnoreMeasurementValueStatus	Determines if the status of the measurement value is ignored during evaluation of limit switch status.
InputSignal	Signal whose measurement value is used to evaluate the limit switch.
InvertResetBehaviour	If true, the defined ResetBehavior is inverted.
Limit	Limit value. Dependents on OperationDirection, it may also be used to define the lower band value.
OperatingDirection	Operation direction of the limit switch.
ResetBehaviorMask	Reset behavior. Binary mask which is ANDed with all digital inputs. The reset behavior can be inverted with InvertResetBehavior. Default setting is 0.

10.4.3 PMX_ClearPeakValues.vi

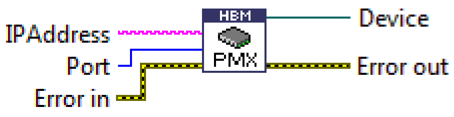
Clears the peak values (min, max, peak to peak) of the given signal.



Signal	Signal that holds the peak values
ClearMin	Set to true, if you want to clear the minimum value (default value is true)
ClearMax	Set to true, if you want to clear the maximum value (default value is true)
ClearPeakToPeak	Set to true, if you want to clear the peak-to-peak value (default value is true)

10.4.4 PMX_Device.vi

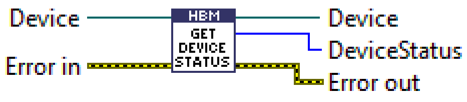
Generates a new PMX device.



IPAddress	IPAddress of the device you want to connect with
Port	Port of the device you want to connect with (default port is 55000)

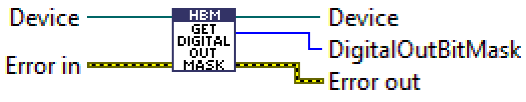
10.4.5 PMX_GetDeviceStatus.vi

Returns the current status of the device



10.4.6 PMX_GetDigitalOutMask.vi

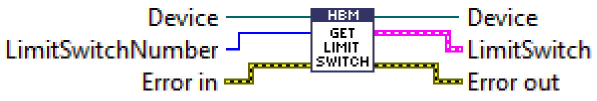
Reads the digital output port of the PMX device. (16 bit)



DigitalOutBitMask	16 Bit digital output (0=off, 1=on)

10.4.7 PMX_GetLimitSwitch.vi

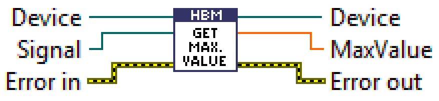
Gets the limit switch info of the given limit switch number



LimitSwitchNumber	Number of the limit to query (1 ...32)
LimitSwitch	Limit switch
	LimitSwitch cluster contains the following element:
LimitSwitchNumber	Number of the limit switch. This can range from 1 to 32.
Enabled	Determines if limit switch operation is activated.
Hysteresis	Hysteresis value. Dependents on OperationDirection, it may also be used to define the band span.
IgnoreMeasurementValueStatus	Determines if the status of the measurement value is ignored during evaluation of limit switch status.
InputSignal	Signal whose measurement value is used to evaluate the limit switch.
InvertResetBehaviour	If true, the defined ResetBehavior is inverted.
Limit	Limit value. Dependents on OperationDirection, it may also be used to define the lower band value.
OperatingDirection	Operation direction of the limit switch.
ResetBehaviorMask	Reset behavior. Binary mask which is ANDed with all digital inputs. The reset behavior can be inverted with InvertResetBehavior. Default setting is 0.

10.4.8 PMX_GetMaxValue.vi

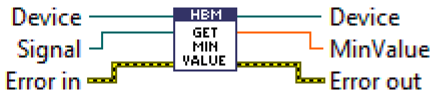
Gets the maximum value of the given signal



MaxValue	Maximum value of the given signal since last clear

10.4.9 PMX_GetMinValue.vi

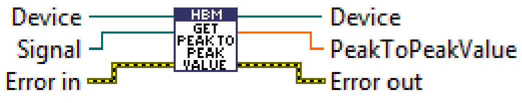
Gets the minimum value of the given signal



MinValue	Minimum value of the given signal since last clear

10.4.10 PMX_GetPeakToPeakValue.vi

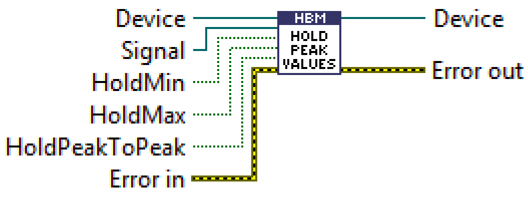
Gets the peak-to-peak value of the given signal



PeakToPeakValue	Peak to peak value of the given signal since last clear

10.4.11 PMX_HoldPeakValues.vi

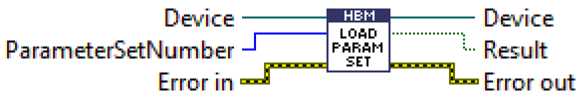
Holds or enables peak value function (min, max, peak to peak) of the given signal.



Signal	Signal for which peak should be enabled/disabled
HoldMin	Set to true if minimum value should be frozen (default value is true)
HoldMax	Set to true if maximum value should be frozen (default value is true)
HoldPeakToPeak	Set to true if peak to peak value should be frozen (default value is true)

10.4.12 PMX_LoadParameterSet.vi

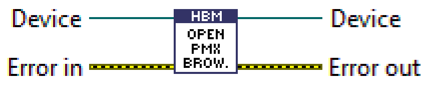
Loads the given parameter set number.
If the parameter set exists and has been loaded the result is true, else false.



Device	PMX device on which to load the given parameter set number
ParametersetNumber	Parameter set that should be loaded (0,1,2,3...). -1 means: Load factory setup into currently active parameter set and activate it.
Result	True if parameter set exists and has been loaded

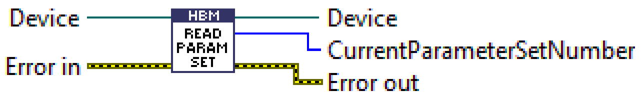
10.4.13 PMX_OpenPMXBrowser.vi

Opens the default web browser to show the configuration panel for the given PMX device



10.4.14 PMX_ReadParameterSetNumber.vi

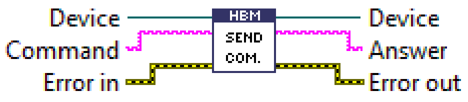
Reads the currently loaded parameter set number (0,1,2,3...).



Device	PMX device on which to read the currently loaded parameter set number
CurrentParametersetNumber	Currently loaded parameter set number (0, 1, 2, 3,..)

10.4.15 PMX_SendCommand.vi

Sends a command to the device and returns its answer.

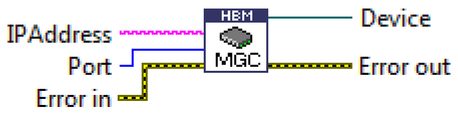


Command	Command to send to the device (see manual of the device for details)
Answer	Answer of the device according to sent command

10.5 Group MGC

10.5.1 MGC_Device.vi

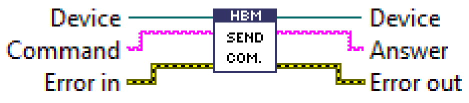
Generates a new MGC device.



IPAddress	IP address of the device you want to connect with
Port	Port of the device you want to connect with (default port is 7)

10.5.2 MGC_SendCommand.vi

Sends a command to the device and returns its answer.



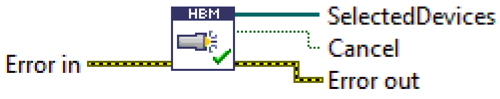
Command	Command to send to the device (see manual of the device for details)
Answer	Answer of the device according to sent command

10.6 Group User Interface

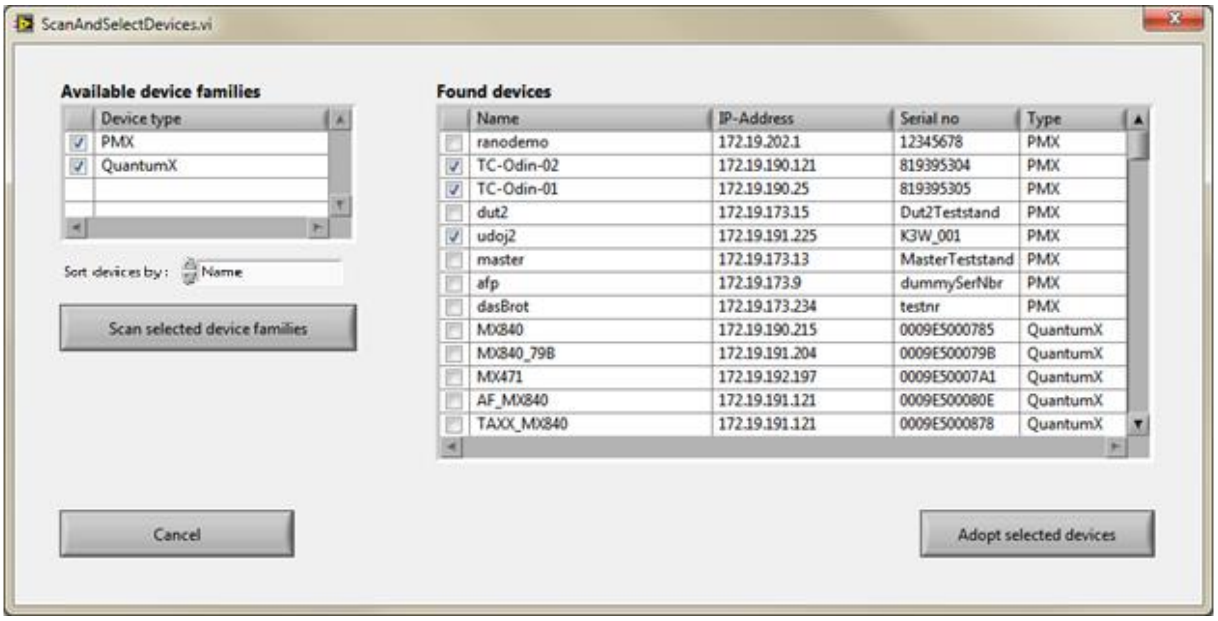
All VIs in this group can be used in an interactive way. They come with a ready to use graphical user interface and can be concatenated to archive maximum functionality with minimal effort.

10.6.1 ScanAndSelectDevices.vi

Scan the network for devices of certain device families (e.g. PMX or QuantumX) and select devices you want to use.



SelectedDevices	Array of selected devices



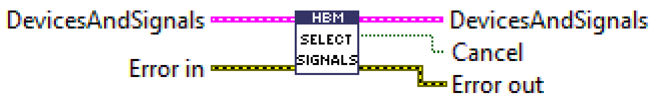
	Device type
✓	PMX
✓	QuantumX

Sort devices by: Name

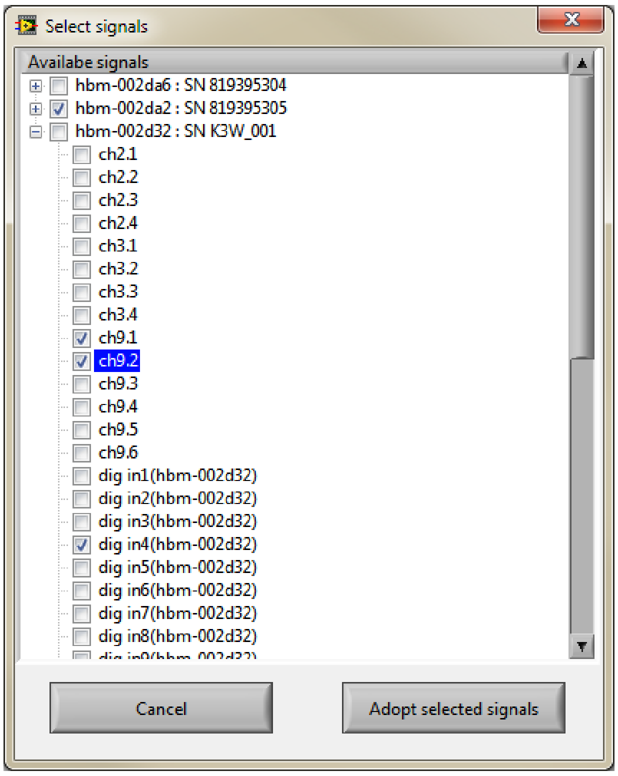
	Name	IP-Address	Serial no	Type
	ranodemo	172.19.202.1	12345678	PMX
✓	TC-Odin-02	172.19.190.121	819395304	PMX
✓	TC-Odin-01	172.19.190.25	819395305	PMX
	dut2	172.19.173.15	Dut2Teststand	PMX
✓	udoj2	172.19.191.225	K3W_001	PMX
	master	172.19.173.13	MasterTeststand	PMX
	afp	172.19.173.9	dummySerNbr	PMX
	dasBrot	172.19.173.234	testnr	PMX
	MX840	172.19.190.215	0009E5000785	QuantumX
	MX840_79B	172.19.191.204	0009E500079B	QuantumX
	MX471	172.19.192.197	0009E50007A1	QuantumX
	AF_MX840	172.19.191.121	0009E500080E	QuantumX
	TAXX_MX840	172.19.191.121	0009E5000878	QuantumX

10.6.2 SelectSignals.vi

Select certain signals that should be used later (e.g. for measuring).



DevicesAndSignals (Input)	Cluster consisting of an array of devices and an array of signals.
DevicesAndSignals (Output)	Cluster consisting of an array of devices and an array of SELECTED signals.



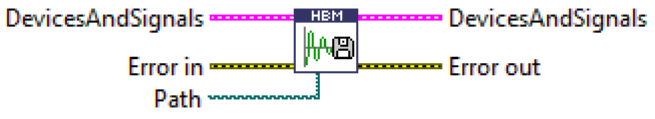
Select signals

Availabe signals

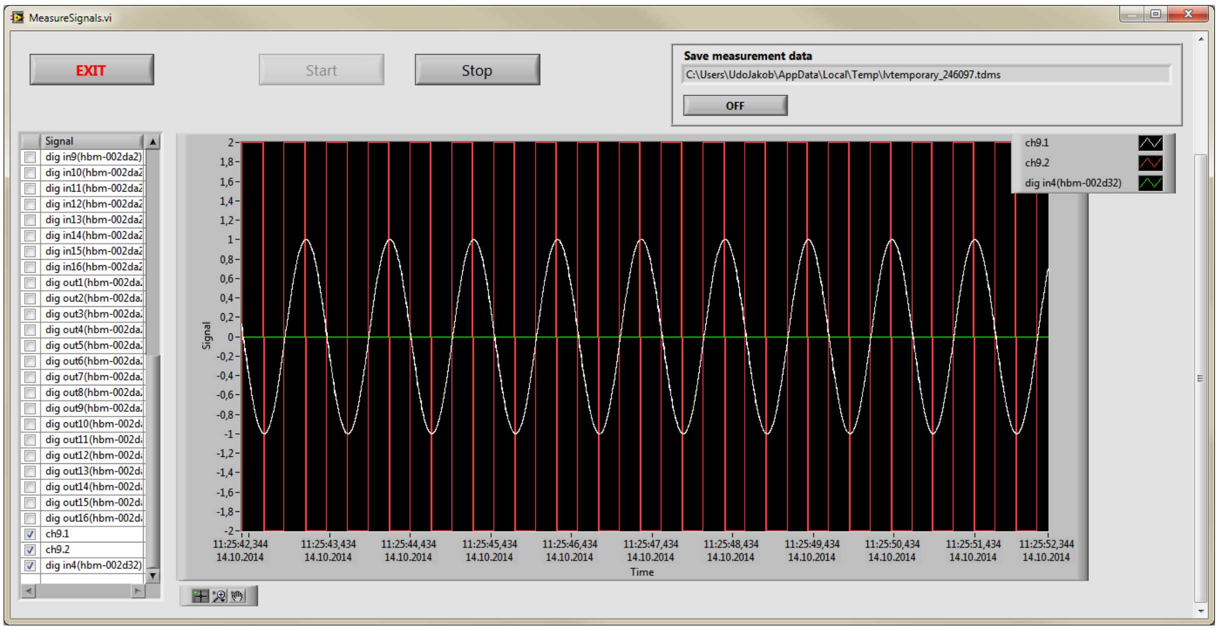
- [ ] hbm-002da6 : SN 819395304
- [✓] hbm-002da2 : SN 819395305
- [ ] hbm-002d32 : SN K3W_001
  - [ ] ch2.1
  - [ ] ch2.2
  - [ ] ch2.3
  - [ ] ch2.4
  - [ ] ch3.1
  - [ ] ch3.2
  - [ ] ch3.3
  - [ ] ch3.4
  - [✓] ch9.1
  - [✓] ch9.2
  - [ ] ch9.3
  - [ ] ch9.4
  - [ ] ch9.5
  - [ ] ch9.6
  - [ ] dig in1(hbm-002d32)
  - [ ] dig in2(hbm-002d32)
  - [ ] dig in3(hbm-002d32)
  - [✓] dig in4(hbm-002d32)
  - [ ] dig in5(hbm-002d32)
  - [ ] dig in6(hbm-002d32)
  - [ ] dig in7(hbm-002d32)
  - [ ] dig in8(hbm-002d32)

10.6.3 MeasureSignals.vi

Use this VI to interactively choose signals you want to measure and visualize or save.



DevicesAndSignals	Cluster consisting of an array of devices and an array of all signals you want to measure
Path	Complete path and name of the file to save the measurement values in.

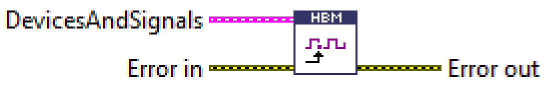


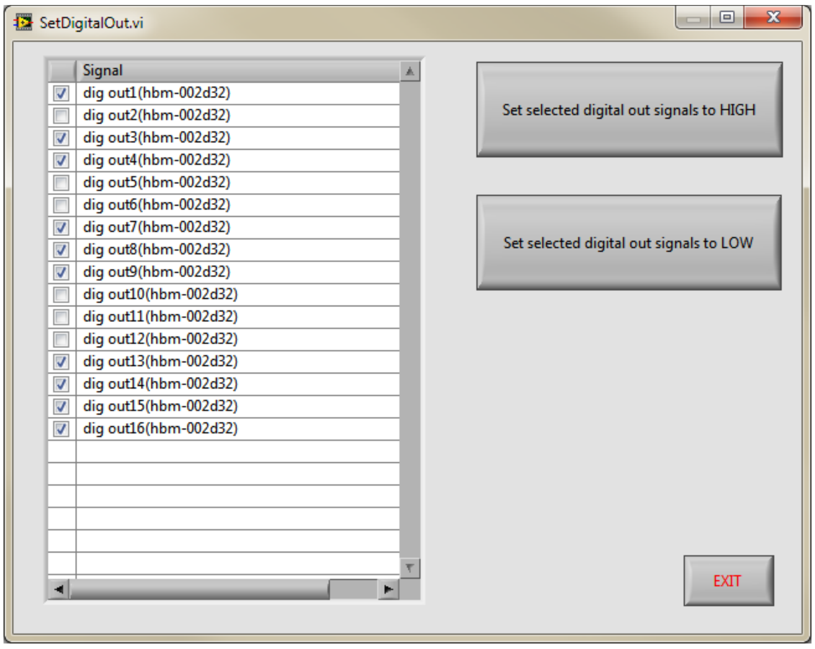
Save measurement data
C:\Users\UdoJakob\AppData\Local\Temp\lvtemporary_246097.tdms

	Signal
☐	dig in9(hbm-002da2)
☐	dig in10(hbm-002da2
☐	dig in11(hbm-002da2
☐	dig in12(hbm-002da2
☐	dig in13(hbm-002da2
☐	dig in14(hbm-002da2
☐	dig in15(hbm-002da2
☐	dig in16(hbm-002da2
☐	dig out1(hbm-002da
☐	dig out2(hbm-002da
☐	dig out3(hbm-002da
☐	dig out4(hbm-002da
☐	dig out5(hbm-002da
☐	dig out6(hbm-002da
☐	dig out7(hbm-002da
☐	dig out8(hbm-002da
☐	dig out9(hbm-002da
☐	dig out10(hbm-002d
☐	dig out11(hbm-002d
☐	dig out12(hbm-002d
☐	dig out13(hbm-002d
☐	dig out14(hbm-002d
☐	dig out15(hbm-002d
☐	dig out16(hbm-002d
✓	ch9.1
✓	ch9.2
✓	dig in4(hbm-002d32)

10.6.4 SetDigitalOut.vi

Use this VI to set selected digital out signals to high or low.

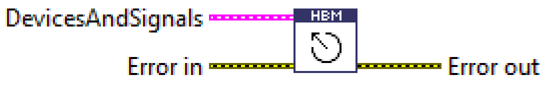


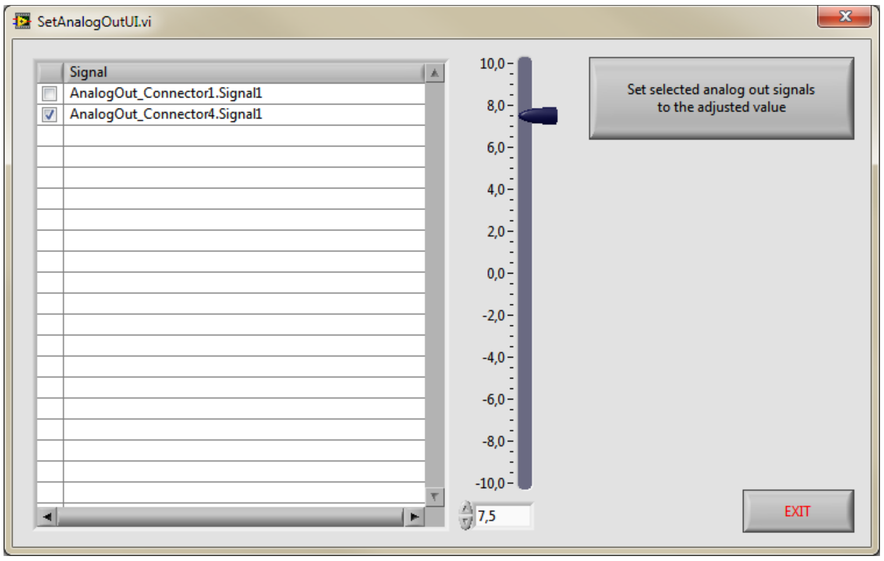

SetDigitalOut.vi
Set selected digital out signals to HIGH
Set selected digital out signals to LOW
EXIT

	Signal
✓	dig out1(hbm-002d32)
☐	dig out2(hbm-002d32)
✓	dig out3(hbm-002d32)
✓	dig out4(hbm-002d32)
☐	dig out5(hbm-002d32)
☐	dig out6(hbm-002d32)
✓	dig out7(hbm-002d32)
✓	dig out8(hbm-002d32)
✓	dig out9(hbm-002d32)
☐	dig out10(hbm-002d32)
☐	dig out11(hbm-002d32)
☐	dig out12(hbm-002d32)
✓	dig out13(hbm-002d32)
✓	dig out14(hbm-002d32)
✓	dig out15(hbm-002d32)
✓	dig out16(hbm-002d32)

10.6.5 SetAnalogOutUI.vi

Use this VI to set selected analog out signals to a certain value.





	Signal
☐	AnalogOut_Connector1.Signal1
☑	AnalogOut_Connector4.Signal1

www.hbm.com

HBM Test and Measurement
Tel. +49 6151 803-0
Fax +49 6151 803-9100
info@hbm.com

measure and predict with confidence

HBM