User Manual

English

HBM LabVIEW Driver

HBM

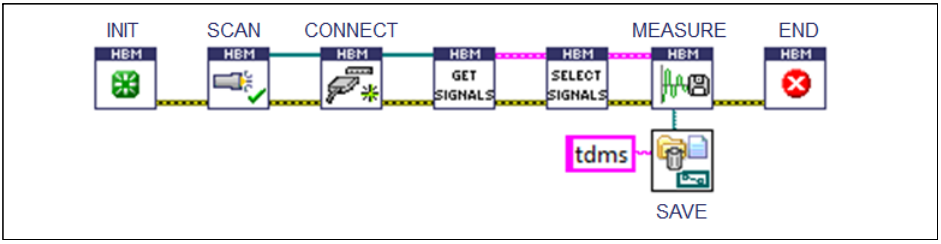
Hottinger Baldwin Messtechnik GmbH
Im Tiefen See 45
D-64239 Darmstadt
Tel. +49 6151 803-0
Fax +49 6151 803-9100
info@hbm.com
www.hbm.com

Mat.:
DVS: A4175-2.0 HBM: public
11.2016

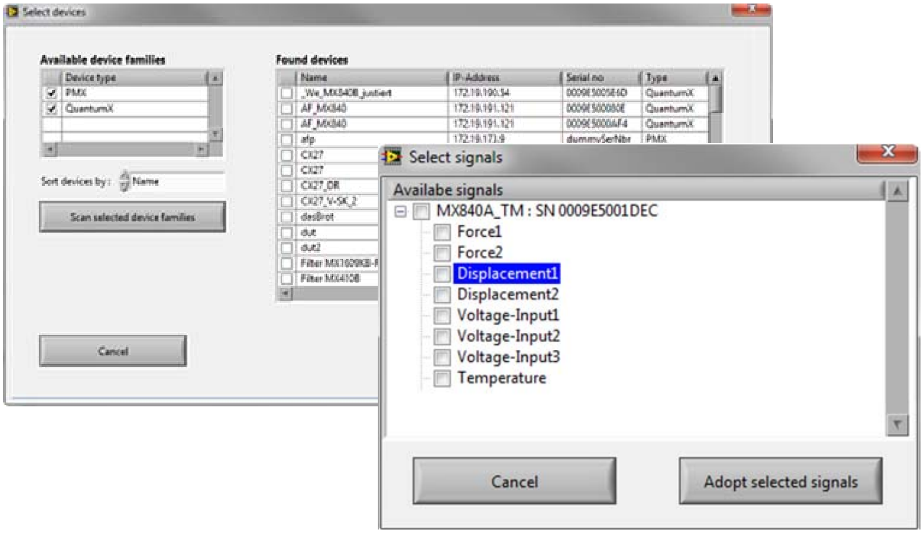
1 Abstract

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:



	Device type
✓	PMX
✓	QuantumX

	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			
☐	CX27			
☐	CX27_DR			
☐	CX27_V-SK_2			
☐	desBrot			
☐	dut			
☐	dut2			
☐	Filter MX1609KB-F			
☐	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.

2 License

The HBM LabVIEW Driver is protected by a license file.

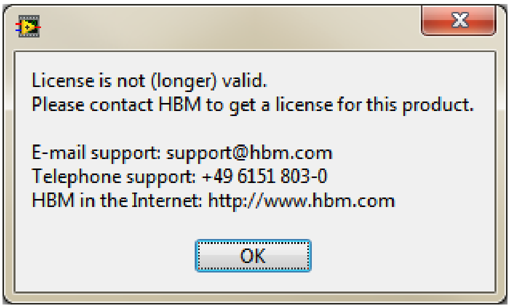
Please contact your local HBM sales representative to purchase a driver license. HBM will send you a permanent license file which will replace the expired evaluation license file within the folder

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

3 Technical support

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

http://www.hbm.com/en/contact/worldwide

HBM on the Internet:

http://www.hbm.com

4 Requirements

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

- LabVIEW 2012 or higher
- .NET Framework 4.0
- QuantumX/SomatXR: Firmware 4.0.24 or higher
- PMX: Firmware 2.0 or higher
- MGCplus with CP42: Firmware 4.74 or higher
  MGCplus with CP22: Firmware 4.44 or higher
  (MGCplus with 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 a program which uses Vis of the HBM Lab-VIEW Driver, you need to:

- Install the .NET framework 4.0 on the target system.
- Copy the following HBM DLLs together with your pro-gram to the target system:
  - Hbm.Api.Common.dll, Hbm.Api.Scan.dll, Hbm.Api.SensorDB.dll, Hbm.Api.Logging.dll, Hbm.Api.Utils.dll, HBM_Streaming.dll,

HBM_Scan.dll, Gibraltar.Agent.dll, NLog.dll, VistaDB.5.NET40.dll.

- DLLs for the HBM-devices you want to use with your program:
  - for PMX: Hbm.Api.Pmx.dll,
  - for QuantumX/SomatXR: Hbm.Api.QuantumX.dll, HBM_QX_Framework.dll,
  - for MGCplus: Hbm.Api.Mgc.dll.

- Copy the file “DeviceDriver.plugins” (this XML file defines the device drivers to be used) to the target system.
- Modify the Firewall, e.g. by executing “Firewall__scan_allowed_for_all.bat” located in the directory “…\LabVIEWXXXX\user.lib\HBM LabVIEW Driver\DLLs”.

For the scan to work, the incoming UDP ports 31.416 and 31.417 need to be opened for the application. Normally the system will ask whether the application should be allowed to use the network, but this will allow all inbound communication for the application. Therefore we recommend to use one of the following:

With Windows® 7 and higher use:

```
netsh advfirewall firewall add rule name="AppName"
direction=in action=allow enable=yes profile=any
localport=31416,31417 protocol=UDP edge=yes
program="AppExe"
```

With Windows® XP use:

```
netsh firewall add allowedprogram name="AppName"
mode=ENABLE scope=ALL profile=ALL
program="AppExe"
```

5 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 version of this driver focus on measuring – and does 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	✓	✓	(1)	✓	✓	1)
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	✓	✓	✓	✓	✓	✓

	Common API			**LabVIEW Driver**		
Digital Out (direct setting)	✓	✓	✓	✓	✓	✓
CAN DAQ	✓	✓ 4)	✓		✓ 4)	

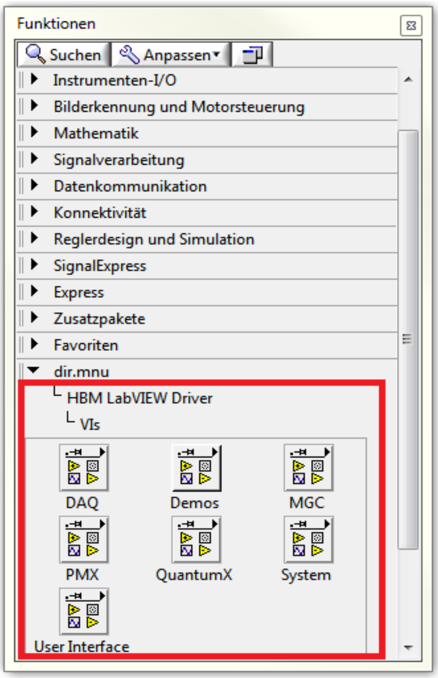
1) Not supported by hardware
2) Only sample rate and filter frequency can be set by VI
3) There are no dedicated VIs for sensor configuration (nevertheless it is possible to setup sensor settings by using the API directly)
4) Only via CODESYS/calculated channels

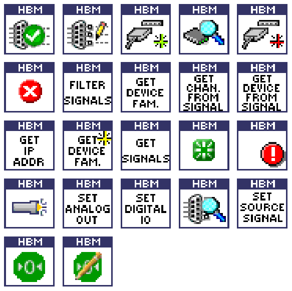
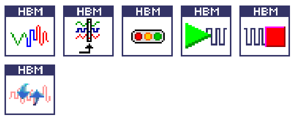
6 LabVIEW Driver VIs Overview

After successful installation of the HBM LabVIEW Driver, there will be a new entry within the function palette of LabVIEW (dir.mnu – 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 DLLs are added to your project.

The VIs directory contains a number of subdirectories in which you find the following groups:



 HBM FILTER SIGNALS GET DEVICE FAM. GET CHAN. FROM SIGNAL GET DEVICE FROM SIGNAL GET IP ADDR GET DEVICE FAM. GET SIGNALS SET ANALOG OUT SET DIGITAL IO SET SOURCE SIGNAL	**Group System** VIs in this group are used to: - Scan the LAN adapter for devices - Connect and disconnect devices - View signal settings - Change and assign signal settings - 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 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/SomatXR 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/SomatXR 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.

7 Examples

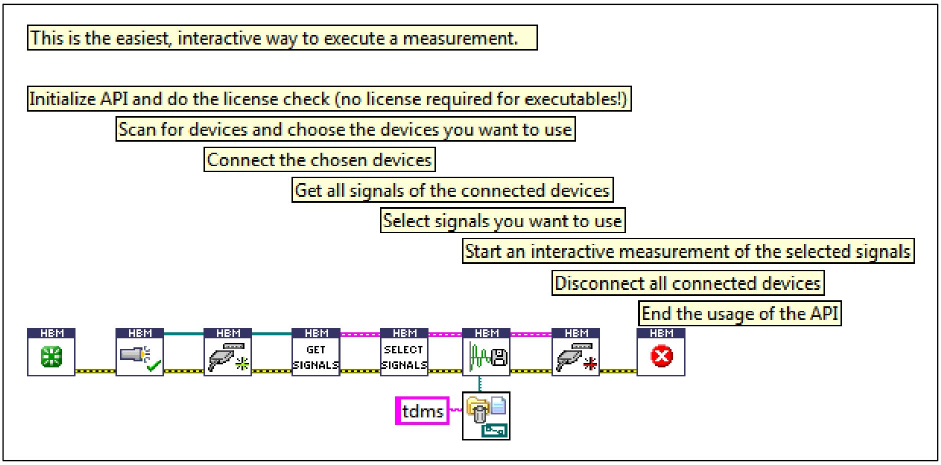
The HBM LabVIEW Driver comes with a number of examples (located within the Demo folder), that show how to use the VIs and at the same time illustrate the necessary workflow.

7.1 Group Demo

7.1.1 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

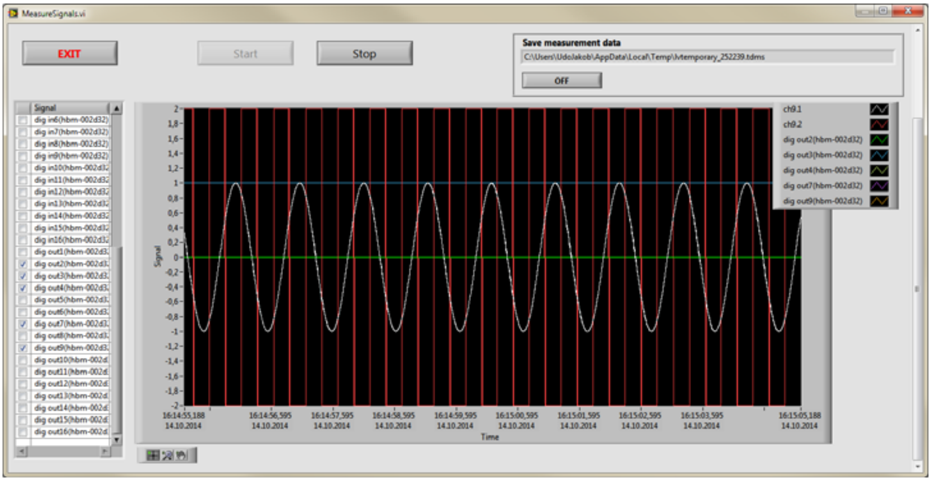
Get all signals of the connected devices

Select signals you want to use

Start an interactive measurement of the selected signals

Disconnect all connected devices

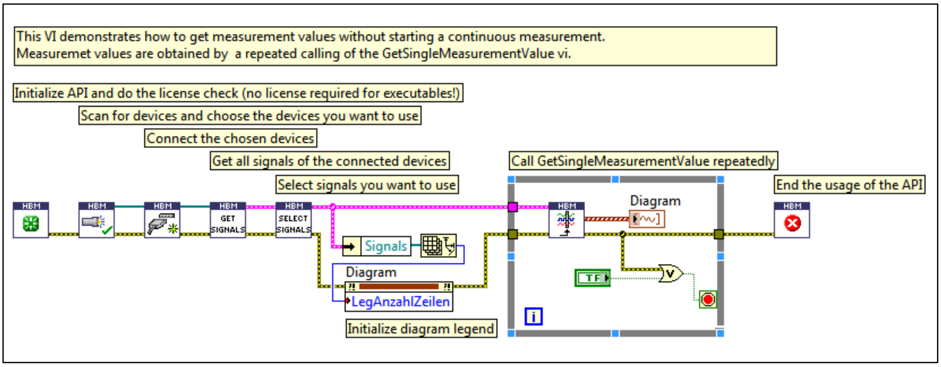
End the usage of the API



7.1.2 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.vi.



This VI demonstrates how to get measurement values without starting a continuous measurement.
Measuremet values are obtained by a repeated calling of the GetSingleMeasurementValue vi.

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

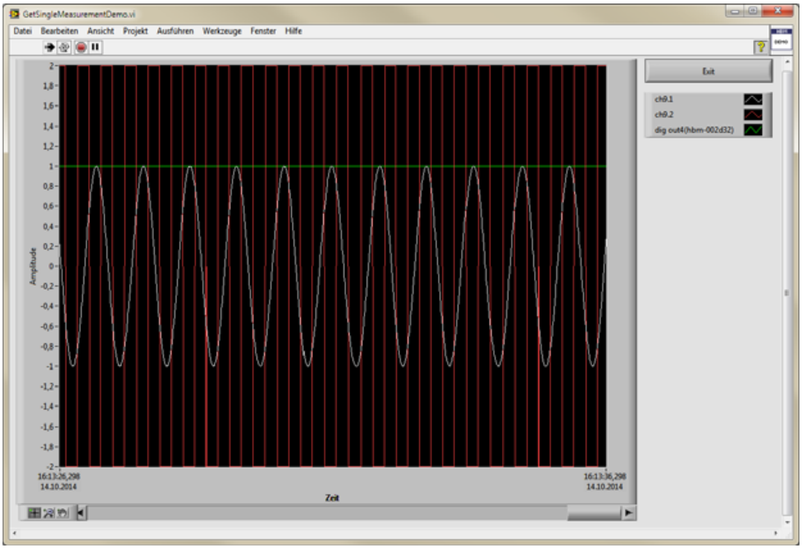
Get all signals of the connected devices

Select signals you want to use

Call GetSingleMeasurementValue repeatedly

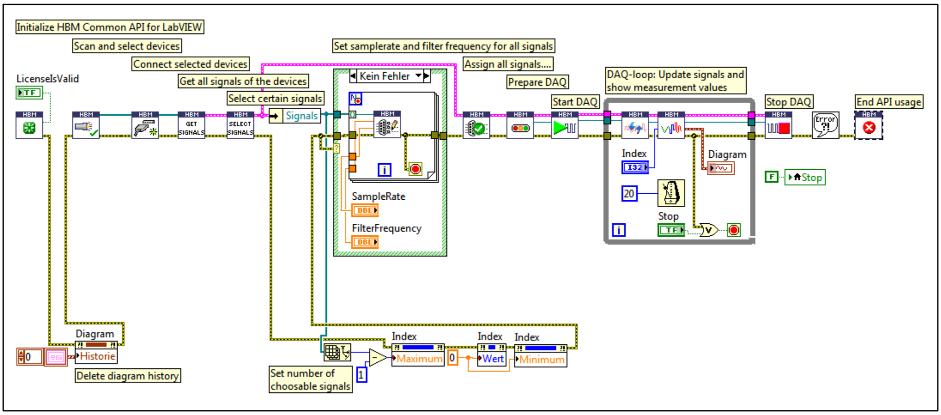
End the usage of the API

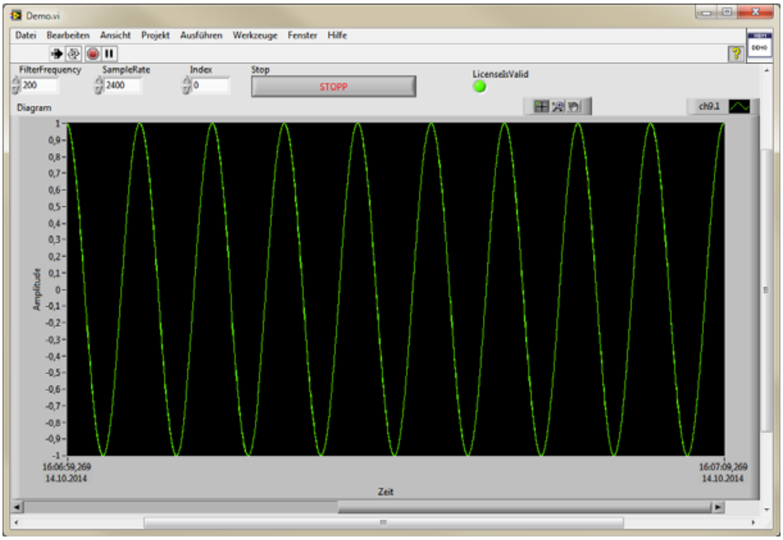
Initialize diagram legend


GetSingleMeasurementDemo.vi
Datei Bearbeiten Ansicht Projekt Ausführen Werkzeuge Fenster Hilfe
Exit
ch9.1
ch9.2
dig out4(hbm-002d32)
Amplitude
Zeit
16:13:26,298
14.10.2014
16:13:36,298
14.10.2014

7.1.3 Demo.vi

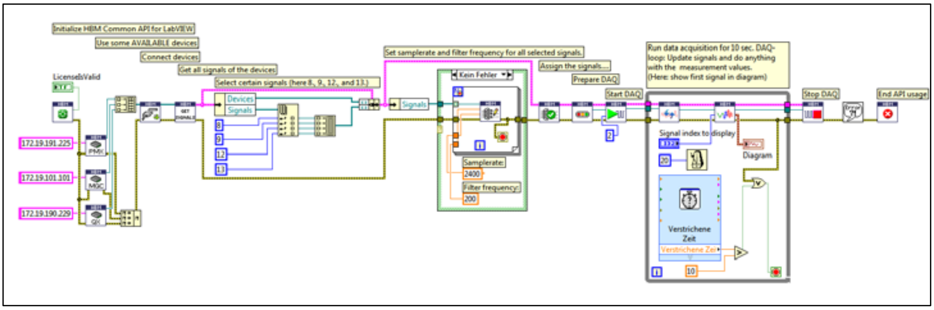
This VI demonstrates how to setup a measurement with certain signals, sample rates and filter frequencies. It also shows how to use the VIs that are required to execute a continuous measurement.

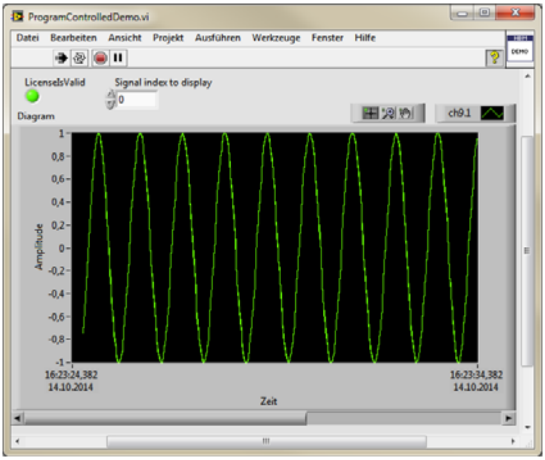




7.1.4 ProgramControlledDemo.vi

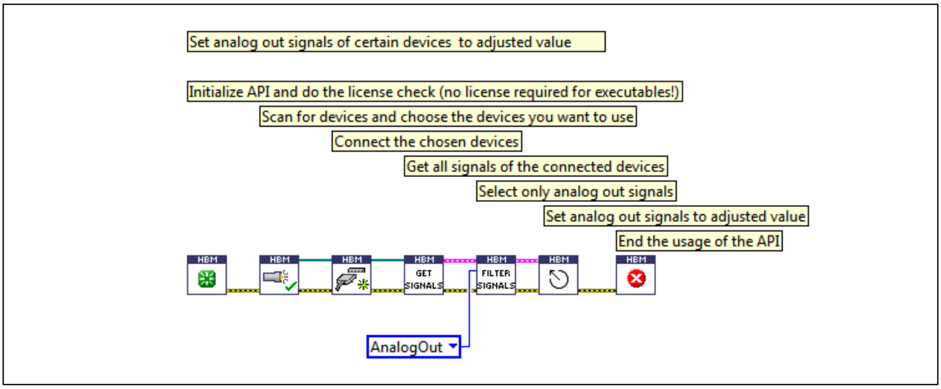
This VI shows how to setup and execute a measurement without any user input. Since most people do not use 3 different DAQ devices at once, you probably have to adapt this demo. Please remove all device types that you do not want to use and adapt “BuildArray.vi“ to the new number of used devices. The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

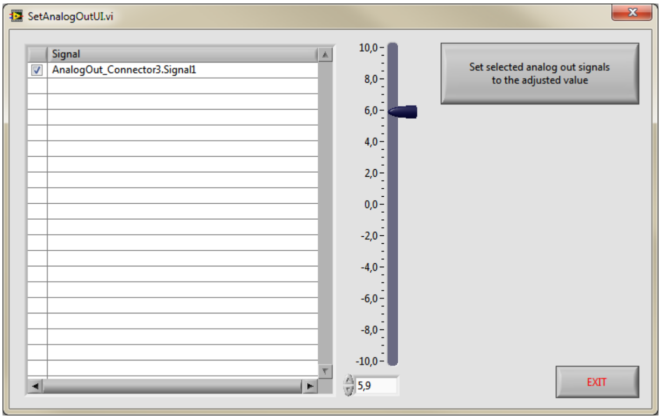

Initialize HBM Common API for LabVIEW
Use some AVAILABLE devices
Connect devices
Get all signals of the devices
Select certain signals (here 8, 9, 12, and 13.)
LicenseIsValid
172.19.191.225
172.19.101.101
172.19.190.229
PMX
MGC
QX
Devices
Signals
8
9
12
13
Set samplerate and filter frequency for all selected signals.
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
10
Stop DAQ
End API usage



7.1.5 SettingAnalogOutDemo.vi

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

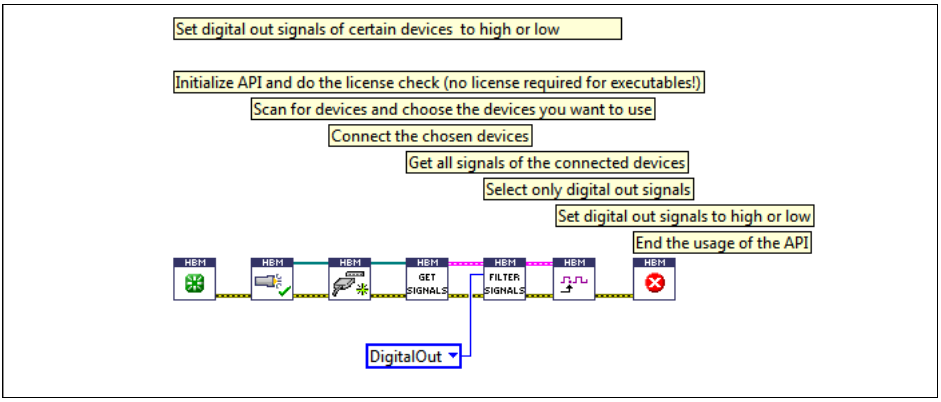

Set analog out signals of certain devices to adjusted value
Initialize API and do the license check (no license required for executables!)
Scan for devices and choose the devices you want to use
Connect the chosen devices
Get all signals of the connected devices
Select only analog out signals
Set analog out signals to adjusted value
End the usage of the API
GET SIGNALS
FILTER SIGNALS
AnalogOut

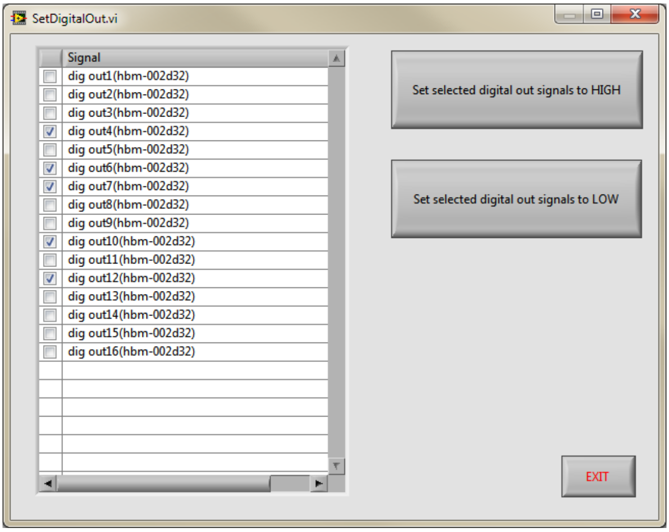

SetAnalogOutUI.vi

	Signal
✓	AnalogOut_Connector3.Signal1

7.1.6 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
End the usage of the API
HBM
GET SIGNALS
FILTER SIGNALS
DigitalOut


SetDigitalOut.vi

	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)

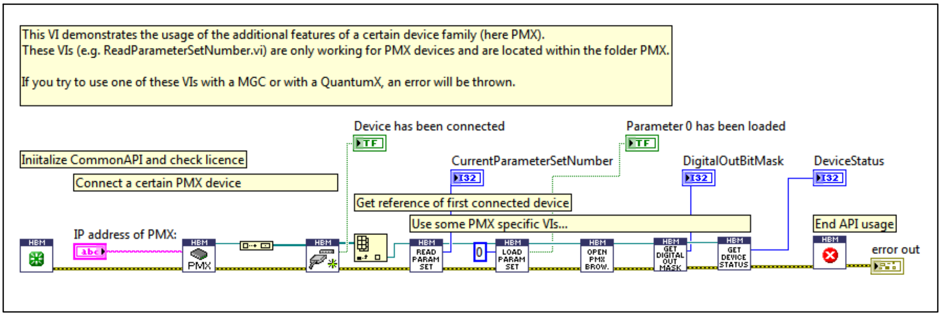
Set selected digital out signals to HIGH
Set selected digital out signals to LOW
EXIT

7.1.7 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.

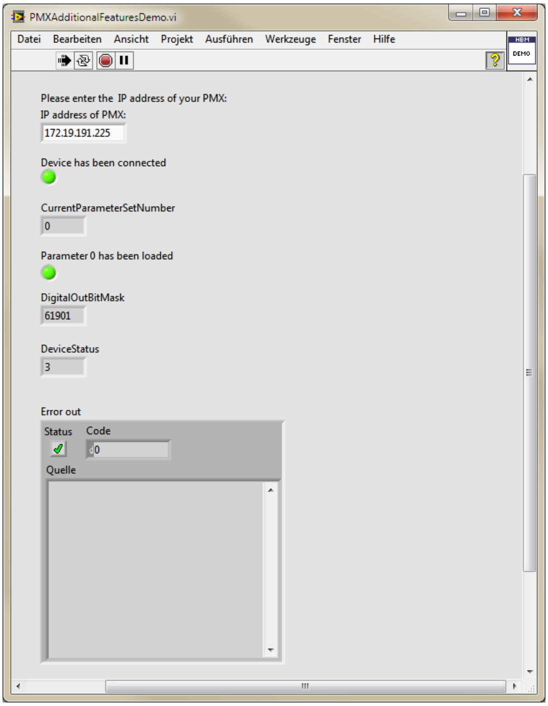
If you try to use one of these VIs with a MGC or with a QuantumX, an error will be thrown.



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
Get reference of first connected device
CurrentParameterSetNumber
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
HBM


PMXAdditionalFeaturesDemo.vi
Datei Bearbeiten Ansicht Projekt Ausführen Werkzeuge Fenster Hilfe
HBM
DEMO

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

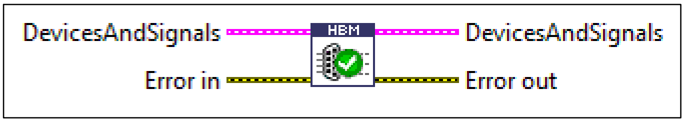
Quelle

8 LabVIEW Driver Details

8.1 Group System

8.1.1 AssignSignals.vi

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

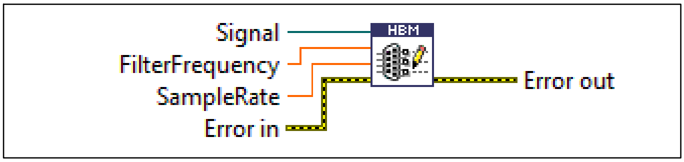


8.1.2 ConfigSignal.vi

Configures 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.

Important

To configure the physical device with these settings, you have to use the AssignSignals VI when the signal is configured.



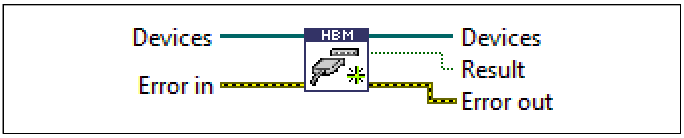
Signal: The signal you want to configure

SampleRate: Sample rate that should be used for measurement

FilterFrequency: Filter frequency to use (filter type - e.g. Bessel or Butterworth will not be changed!)

8.1.3 ConnectDevice.vi

Connects all given devices.

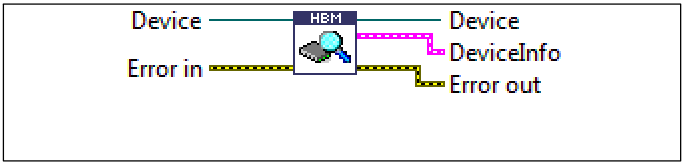


Devices: Array of devices to connect

Result: True if no warning or error occurred during connect

8.1.4 DeviceInfo.vi

Delivers various device properties.

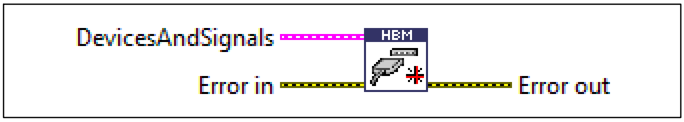


Device: Device whose properties should be read

DeviceInfo: Cluster that contains information about the device

8.1.5 DisconnectDevice.vi

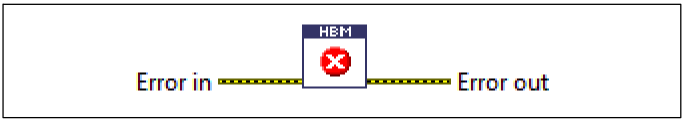
Disconnects all given devices.



Devices: Array of devices to disconnect

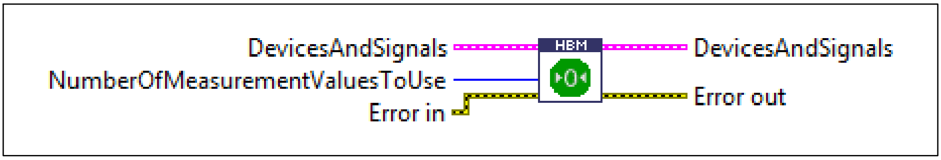
8.1.6 DisposeApi.vi

Use this VI to end the usage of the CommonAPI.



8.1.7 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).

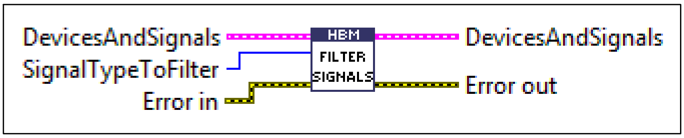


DevicesAndSignals: Array of devices and array of signals. Zeroing is done for all signals within the array. Zeroing can only be executed for signals that deliver valid measurement values!

NumberOfMeasurementValuesToUse: 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).

8.1.8 FilterSignalList.vi

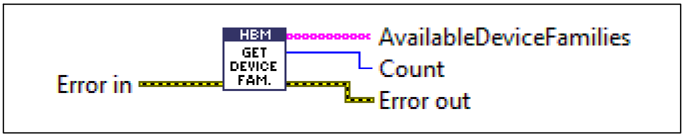
Filters all given signals according to the given signal type.



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.

8.1.9 GetAvailableDeviceFamilyNames.vi

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

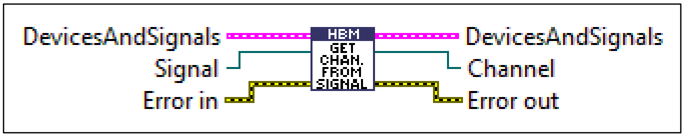


AvailableDevices: Array of available device family names

Count: Number of available device families

8.1.10 GetChannelFromSignal.vi

Finds the channel to which the given signal belongs.

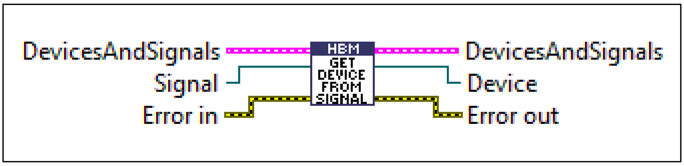


Signal: Signal to check to which channel it belongs

Channel: The channel to which the signal belongs

8.1.11 GetDeviceFromSignal.vi

Finds the device to which the given signal belongs.

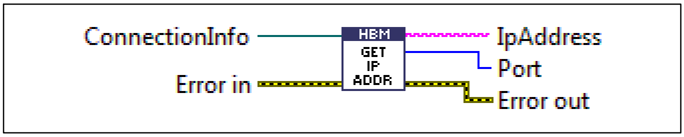


Signal: Signal to check to which device it belongs

Device: The device to which the signal belongs

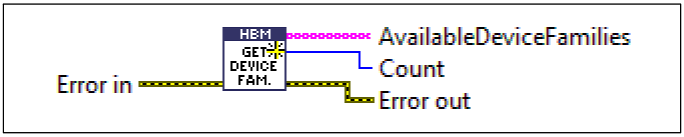
8.1.12 GetIpAddressFromConectionInfo.vi

Get the IP address and the port of a connection info



8.1.13 GetScanableDeviceFamilyNames.vi

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

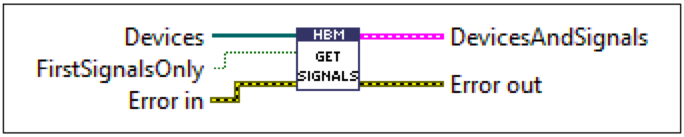


AvailableDeviceFamilies: Array of available device family names

Count: Number of available device families

8.1.14 GetSignals.vi

Delivers (all) signals of the given devices.



Devices: Devices from which to deliver the signals

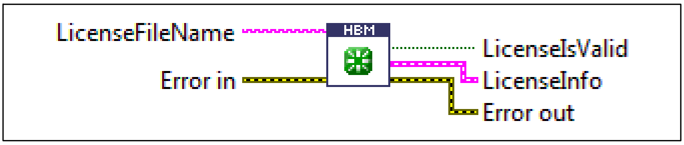
FirstSignalsOnly: Set to true (default) if you only want to get the respective first signals of each channel (QuantumX/SomatXR devices return 2 signals per channel if FirstSignalsOnly is set to false).

DevicesAndSignals: Cluster consisting of an array of devices and an array of all signals of the given devices

8.1.15 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")

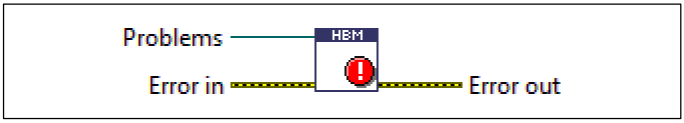
LicenseIsValid: True if given license file is valid or Lab-VIEW runtime is used

LicenseInfo: Information about the license holder (if license is not an evaluation license)

8.1.16 Problems.vi

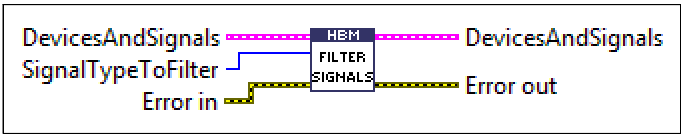
Generates an error, if the problems (e.g. as a result of an assign function) contain any errors.

Generates a warning, if the problems contain any warn-ing but no error.



8.1.17 ScanForDevices.vi

Scans for devices. Repeat this several times to find more devices.



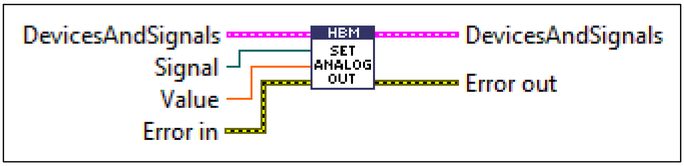
AvailableDeviceFamilies: Array of device family names (e.g. PMX, QuantumX) you want to scan. Leave this connection empty to find all scanable devices of all device families.

SortedBy: Enumeration that defines the property (Name, IpAddress or SerialNumber) that will be used to sort the found devices

Devices: Found devices according to given device family names

8.1.18 SetAnalogOut.vi

Sets the analog out signal to the given value

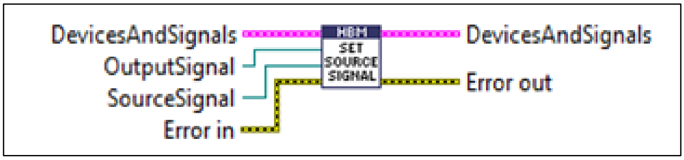


Signal: Signal (has to be of type AnalogOutSignal) to adjust

Value: Value to set

8.1.19 SetAnalogOutSourceSignal.vi

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

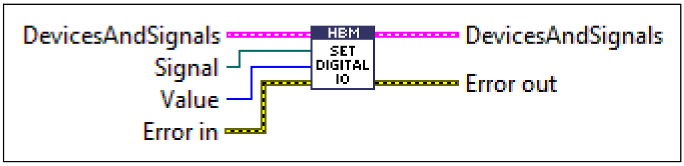


OutputSignal: Analog out signal to configure (has to be of type AnalogOutSignal).

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).

8.1.20 SetDigitalIO.vi

Sets the digital signal according to the given value.

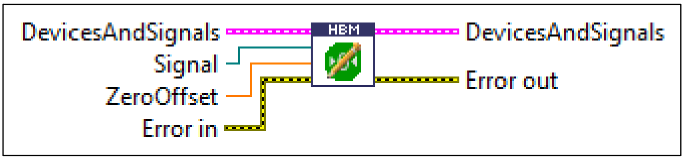


Signal: Signal (has to be of type DigitalSignal) to adjust

Value: DigitalValueType (High or Low) to set

8.1.21 SetZeroOffset.vi

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

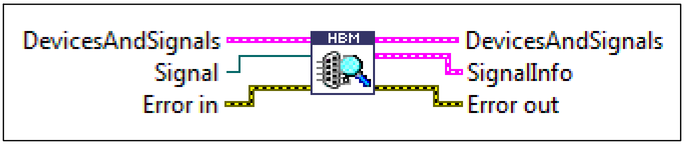


Signal: Analog in signal whose zero offset should be set (notice that actually the channels zero offset will be set and therefore other signals that belong to the same channel are also affected)

ZeroOffset: Zero offset you want to use

8.1.22 SignalInfo.vi

Delivers various signal properties.



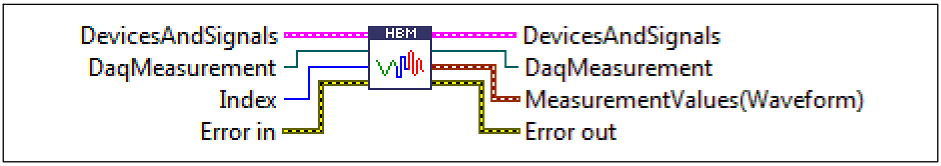
Signal: Signal whose properties should be read

SignalInfo: Cluster that contains information about the signal

8.2 Group DAQ

8.2.1 GetMeasurementValues.vi

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

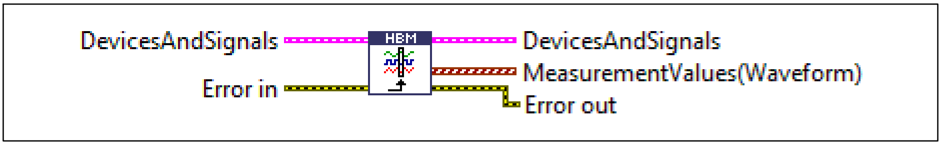


DaqMeasurement: Reference to the DaqMeasurement instance

Index: The index of the signal within the given array of signals from which the measurement values should be delivered

8.2.2 GetSingleMeasurementValues.vi

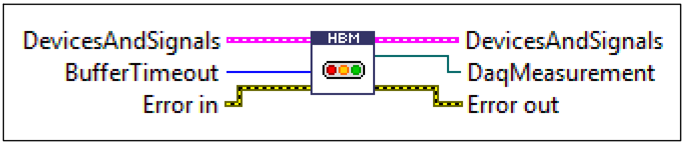
Obtains one single measurement value for each given signal without initializing a continuous measurement. Filter settings will be ignored.



8.2.3 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.



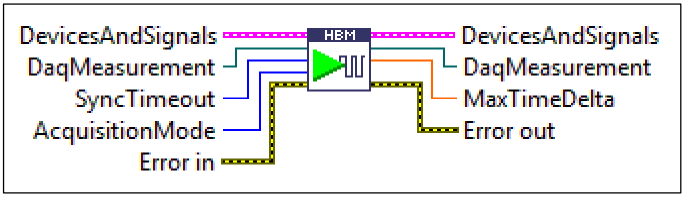
DevicesAndSignals: Array of devices and their signals that have been chosen for measurement

DaqMeasurement: Reference to the DaqMearurement object that handles the measurement

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)

8.2.4 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)

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.

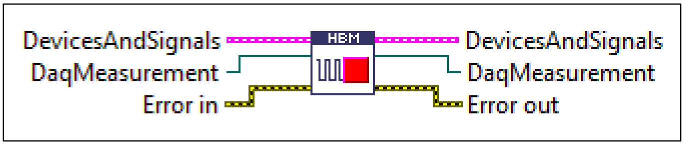
SyncTimeout: Maximum time in ms, that is used to start a synchronized measurement. If it is not possible to start a synchronized measurement within this time, an error will be thrown

AcquisitionMode: Modus of the data acquisition. SoftwareSynchronized = 0, HardwareSynchronized = 1, Unsynchronized = 2

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 taken into account and the function returns with 0.0)

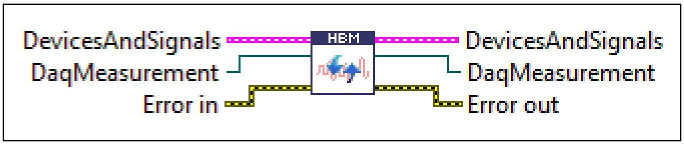
8.2.5 StopDaq.vi

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



8.2.6 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 20 Hz gets the same number of new measurement values as signal_B on device 2 with sample rate 20 Hz!

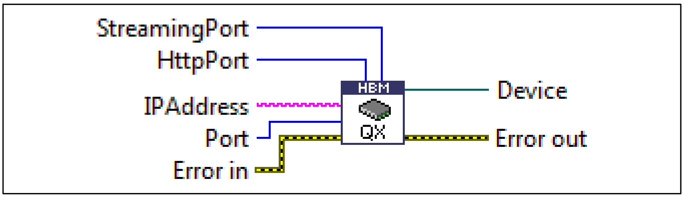
Otherwise (if StartDaq started an unsynchronized measurement), the signals get all measurement values that are accumulated since the last call of this function.

This function also asserts that there is enough allocated memory for the measurement values of each signal.

8.3 Group QuantumX

8.3.1 QuantumX_Device.vi

Generates a new QuantumX/SomatXR device.



IPAddress: IPAddress of the device you want to connect with

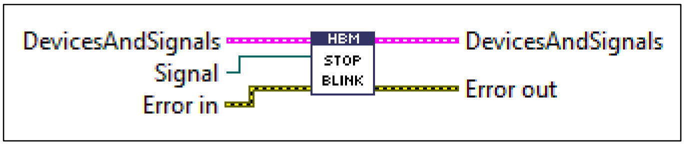
Port: Port of the device you want to connect with (default port is 5001)

HttpPort: Http port of the device (default is 80)

StreamingPort: Port that is used for streaming (default is 7411)

8.3.2 QuantumX_DisableBlinking.vi

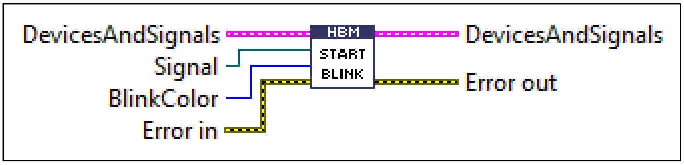
Disables blinking at the connector to which the given signal belongs.



Signal: Signal, whose connector should stop blinking

8.3.3 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

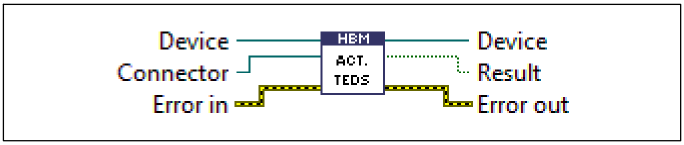
BlinkColor: Blinking color of the LED

8.4 Group PMX

8.4.1 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



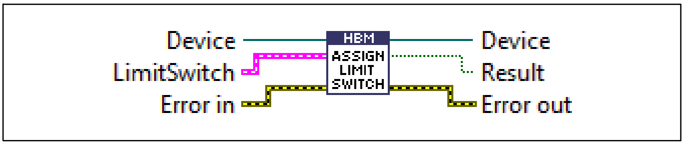
Connector: Connector to which the TEDs is connected

Result: True, if TEDs settings could be loaded and activated

8.4.2 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

LimitSwitchNumber: Number of the limit to assign (1..32)

LimitSwitch:

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

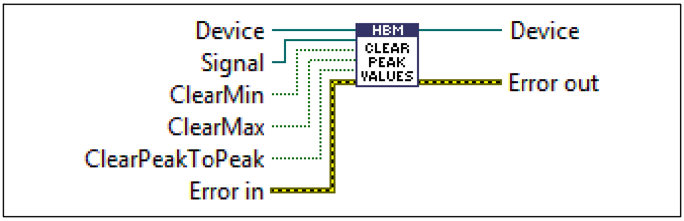
LimitSwitchNumber: Number of the limit switch. This can range from 1 to 32

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

8.4.3 PMX_ClearPeakValues.vi

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



Signal: Signal that holds the peak values

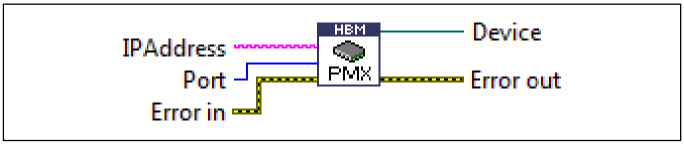
ClearMin: Set to true, if you want to clear the minimum value (default value is true)

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)

8.4.4 PMX_Device.vi

Generates a new PMX device.

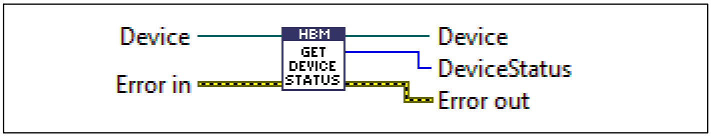


IPAddress: IPAddress of the device you want to connect with

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

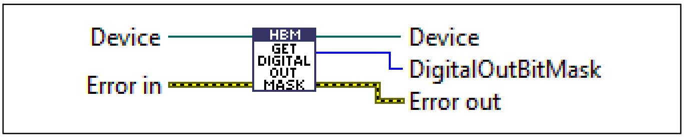
8.4.5 PMX_GetDeviceStatus.vi

Returns the current status of the device.



8.4.6 PMX_GetDigitalOutMask.vi

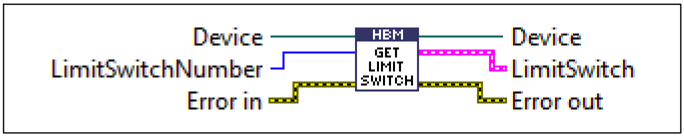
Reads the digital output port of the PMX device.(16 bit)



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

8.4.7 PMX_GetLimitSwitch.vi

Gets the limit switch info of the given limit switch number.



LimitSwitchNumber: Number of the limit to query (1..32)

LimitSwitch:

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

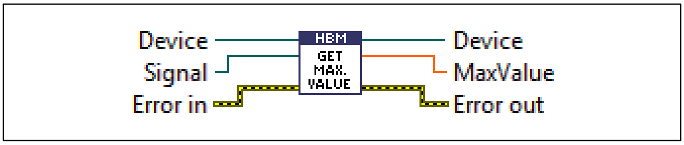
LimitSwitchNumber: Number of the limit switch. This can range from 1 to 32

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

8.4.8 PMX_GetMaxValue.vi

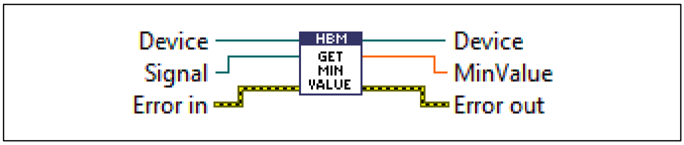
Gets the maximum value of the given signal.



MaxValue: Maximum value of the given signal since last clear

8.4.9 PMX_GetMinValue.vi

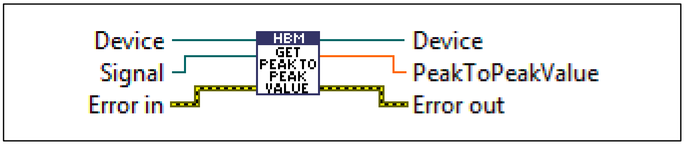
Gets the minimum value of the given signal.



MinValue: Minimum value of the given signal since last clear

8.4.10 PMX_GetPeakToPeakValue.vi

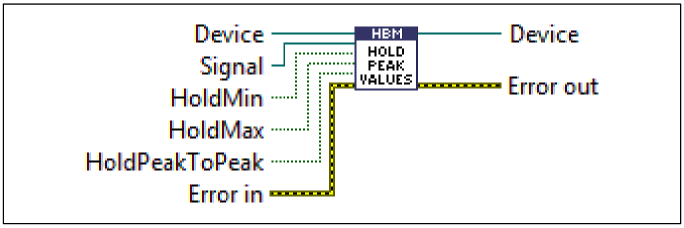
Gets the peak to peak value of the given signal.



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

8.4.11 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)

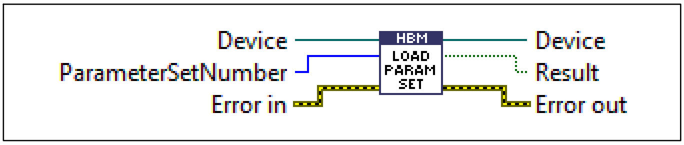
HoldMax: Set to true, if maximum value should be frozen (default value is true)

HoldPeakToPeak: Set to true, if peak to peak value should be frozen (default value is true)

8.4.12 PMX_LoadParameterSet.vi

Loads the given parameter set number.

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



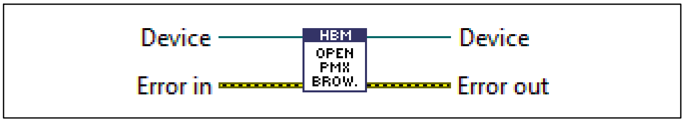
Device: PMX device on which to load the given parameter set number

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

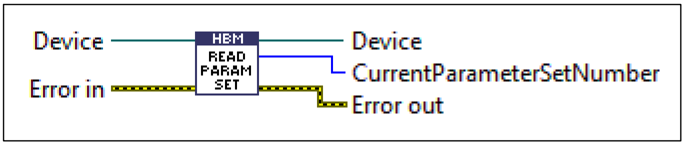
8.4.13 PMX_OpenPMXBrowser.vi

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



8.4.14 PMX_ReadParameterSetNumber.vi

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

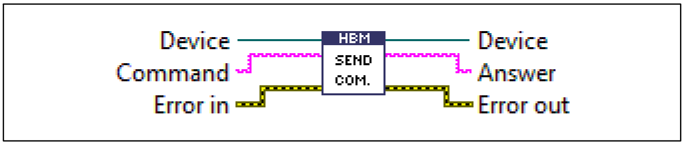


Device: PMX device on which to read the currently loaded parameter set number

CurrentParameterSetNumber: Currently loaded parameter set number (0,1,2,3,..)

8.4.15 PMX_SendCommand.vi

Sends a command to the device and returns its answer.



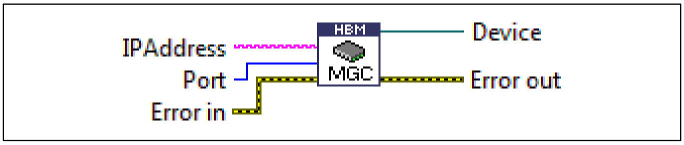
Command: Command to send to the device (see manual of the device for details)

Answer: Answer of the device according to sent command

8.5 Group MGC

8.5.1 MGC_Device.vi

Generates a new MGC device.

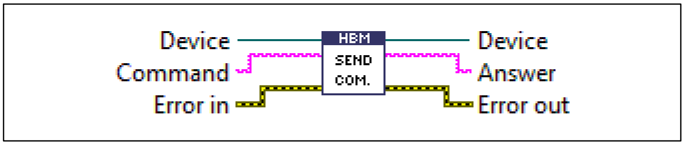


IPAddress: IP address of the device you want to connect with

Port: Port of the device you want to connect with (default port is 7)

8.5.2 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)

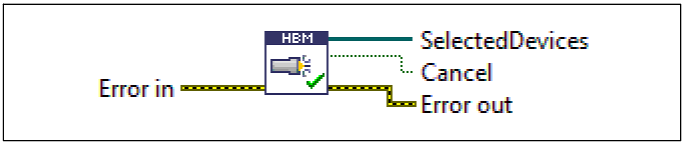
Answer: Answer of the device according to sent command

8.6 Group User Interface

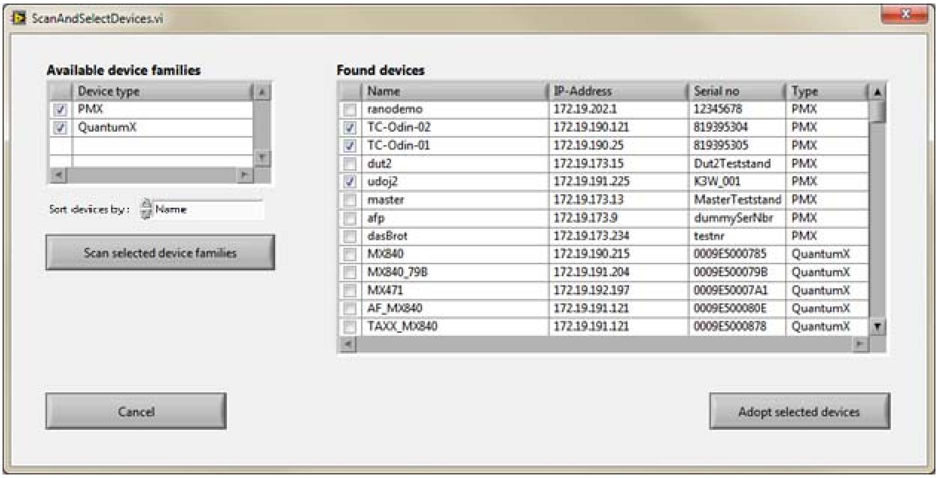
All VIs in this group can be used in an interactive way. They come with a ready to use graphical user interface and can be concatenated to archive maximum functionality with minimal effort.

8.6.1 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



Available device families

	Device type
✓	PMX
✓	QuantumX

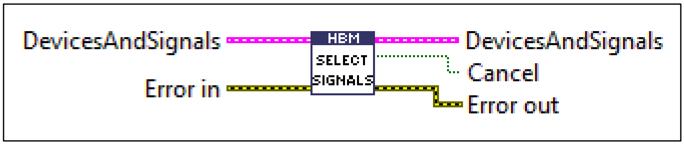
Sort devices by : Name

Found devices

	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

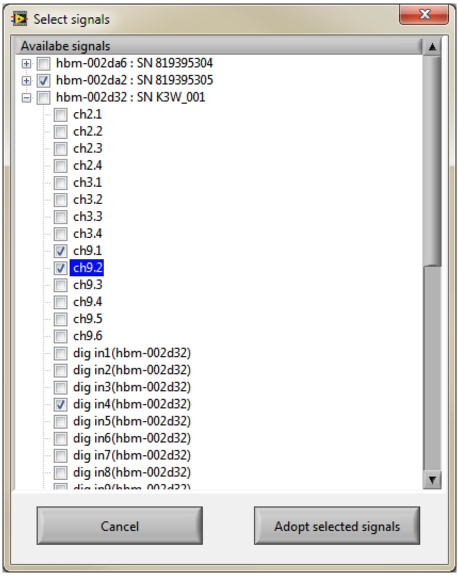
8.6.2 SelectSignals.vi

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

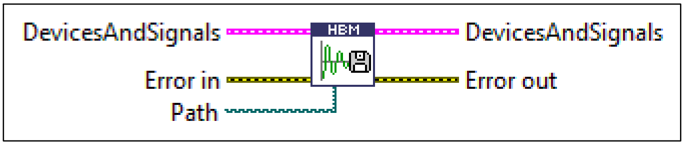

Select signals

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)

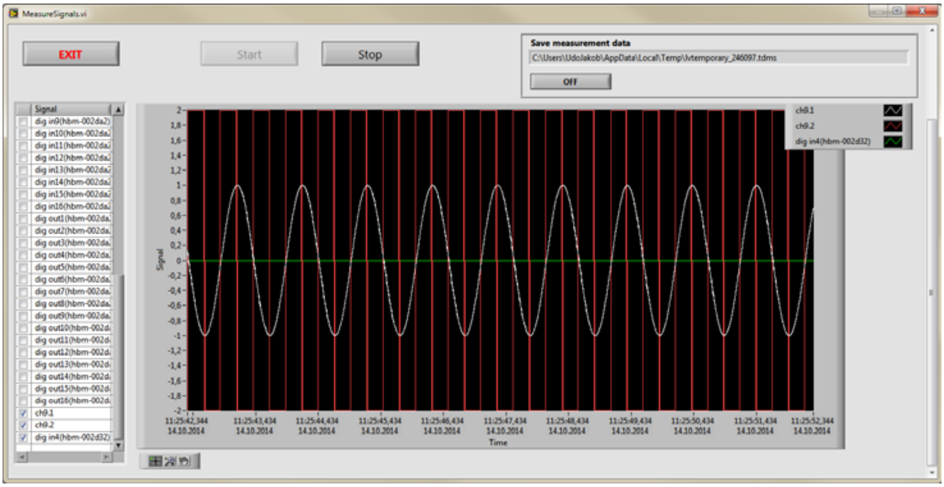
8.6.3 MeasureSignals.vi

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



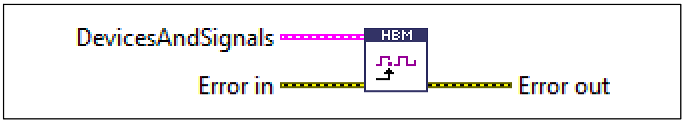
Save measurement data

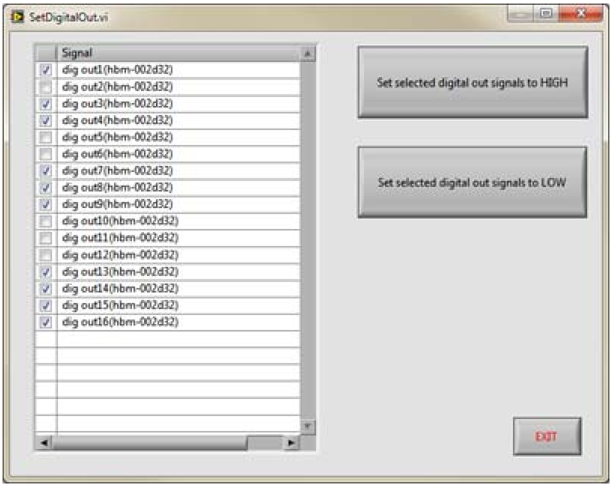
C:\Users\UdoJakob\AppData\Local\Temp\lvtemporary_246097.tdms

	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)

8.6.4 SetDigitalOut.vi

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



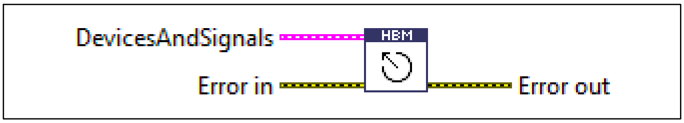

SetDigitalOut.vi

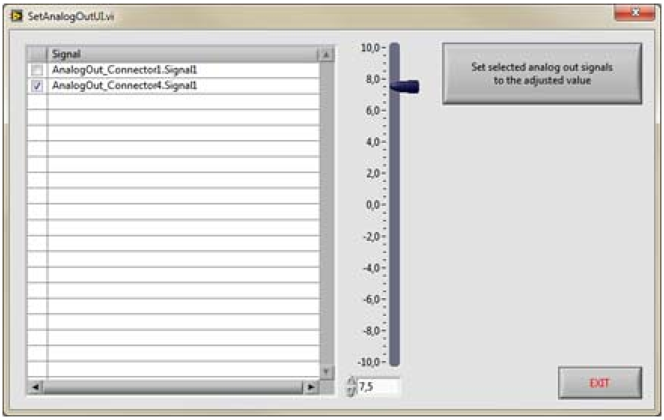
	Signal
✓	dig out1(hbm-002d32)
	dig out2(hbm-002d32)
✓	dig out3(hbm-002d32)
✓	dig out4(hbm-002d32)
	dig out5(hbm-002d32)
	dig out6(hbm-002d32)
✓	dig out7(hbm-002d32)
✓	dig out8(hbm-002d32)
✓	dig out9(hbm-002d32)
	dig out10(hbm-002d32)
	dig out11(hbm-002d32)
	dig out12(hbm-002d32)
✓	dig out13(hbm-002d32)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

Set selected digital out signals to HIGH
Set selected digital out signals to LOW
EXIT

8.6.5 SetAnalogOutUI.vi

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




SetAnalogOutUI.vi

	Signal
☐	AnalogOut_Connector1.Signal1
✓	AnalogOut_Connector4.Signal1

www.hbm.com

A4175-2.0 HBM: public

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

measure and predict with confidence

HBM