National Instruments© LabVIEW™ Programming Manual

1. Introduction

The royalty free National Instruments© LabVIEW™ drivers represent a proprietary interface specifically developed for silicon systems devices. This document provides an explanation of the functions and methods available to application developers. Any software code examples given in this document are for information only and free to use in custom applications.

National Instruments© LabVIEW™ 8.6 to 2016 (32-Bit) and 2009 to 2016 (64-Bit) for Microsoft® Windows™ and Linux® are supported. Please always utilize the most recent version of the driver for the best compatibility and performance. All drivers can be downloaded from http://siliconsystems.at.

If the DHCP-assigned IP address of a device is not known, the royalty free silicon systems Device Manager can be used to find all devices in the local area network. In addition, the software shows various properties, the health status and calibration validity of each device. It can be downloaded from http://siliconsystems.at/support.php?download.

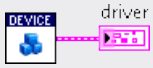
2. Common Device Functionality

The functions provided by the silicon systems driver package can be divided into two subsets: device-specific functions (e.g. reading the temperature of temperature monitor device) and common functions (e.g. retrieving the status of a device or finding available devices in the local area network). In this section the latter functions are explained.

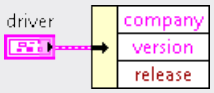
2.1. Get Driver Information

The VI *Device Get Driver* can be used to obtain driver information. It does not expect any arguments and returns a cluster with the following fields: *company* (development company), *version* (driver version) or *release* (driver release date, TimeStamp).

In the following example this information is requested.



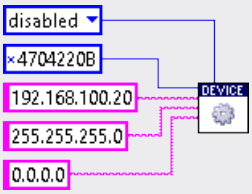
The method *Unbundle By Name* can be utilized to access the fields of the returned cluster as indicated in the example below.



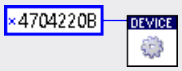
2.2. Configure Network Settings

Each device has the default IP address 192.168.1.1, the subnet mask 255.255.255.0, the gateway 0.0.0.0 and DHCP auto-configuration is enabled. If the network settings of a device shall be modified, e.g. DHCP auto-configuration needs to be disabled, VI *Device Configure* may be used. Apart from the required serial number of the device, the DHCP status, the host IP address, the subnet mask and the gateway IP address are optional. In order to restore the default configuration, the VI can be called without these optional arguments. The device needs to reside in the local area network to be able to receive UDP broadcast datagrams.

In the following example for device with serial number 0x4704220B DHCP auto-configuration is disabled, the host IP address is set to 192.168.100.20, the subnet mask is set to 255.255.255.0 and the gateway IP address is not utilized and set to 0.0.0.0. If the device is only accessed from within the local area network, the gateway is not required to be configured.



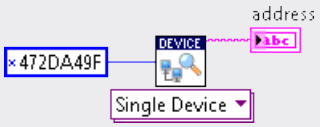
Below the configuration of the device with serial number 0x4704220B is reset.



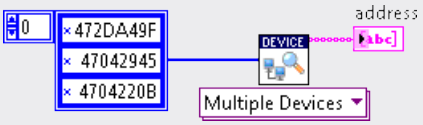
2.3. Finding Devices in the Local Area Network

If the IP address of the desired silicon systems devices are unknown, in particular when the IP addresses are automatically assigned by a DHCP server, the *Device Find* polymorphic VI may be utilized. If the *All Devices* instance is called, an array containing IP addresses of all devices in the local area network (LAN) is returned. The default time-out is 1000 ms and cannot be modified.

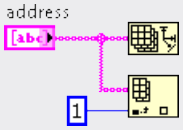
If only the device with a certain serial number shall be looked up, the *Single Device* instance can be called with one argument representing the serial number (unsigned integer). In case of success the IP address of the found device is returned.



If more devices shall be looked up at the same time, an array containing multiple serial numbers can be passed to the VI. An array containing IP addresses of all found devices is returned. The order of IP addresses corresponds to the order of the serial numbers passed to the VI.

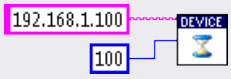


The number of elements of the returned array can be determined with function *Array Size*. An element of the array can be accessed with function *Index Array* as illustrated in the following example.



2.4. Set Device Time-out

When a command or request is sent to a device, a confirmation or response is expected within a certain time. If no response was received, a time-out error occurs. This maximum waiting time is determined by the time-out value which is 1000 ms by default and can be set (in ms) with the aid of VI *Device Set Timeout*. At least a time-out value of 1 ms is expected but, in general, values lower than 100 ms are not recommended. Apart from the new time-out value the device IP address or the DNS name has to be passed to the VI.
In the following example the time-out of device configured with IP address 192.168.1.100 is set to 100 ms.

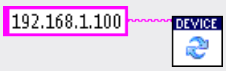


2.5. Get Device Time-out

The currently configured device time-out can be retrieved by the use of VI *Device Get Timeout*. Only the device IP address or the DNS name has to be passed to the VI.
In the following example the time-out of device configured with IP address 192.168.1.100 is retrieved.

2.6. Reset Device

Any device can be reset with the VI *Device Reset*. The effect is the same as disconnecting and reconnecting the device from the Power over Ethernet (PoE) switch. This may be helpful when the network is being reconfigured (assignment of fixed IP addresses, etc.).
Apart from the device IP address or the DNS name the VI does not expect any arguments and is utilized as demonstrated below.

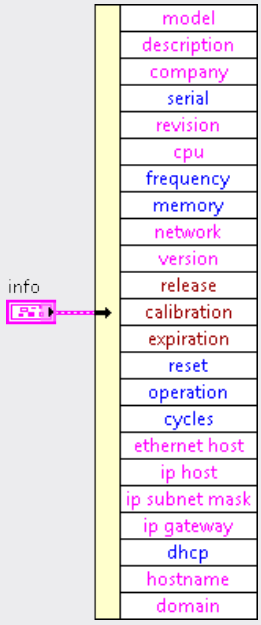


After the device is reset it takes several seconds until it reinitializes itself and gets a new IP address assigned by the DHCP server. Until then the device cannot be accessed.

2.7. Acquisition of Device Information

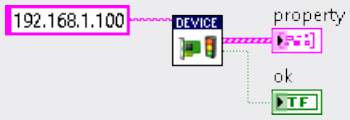
With the VI *Device Get Info* a variety of information can be requested from the device. Apart from the device IP address or the DNS name the VI does not expect any arguments and returns a cluster with the following fields: *model* (device model), *description* (functionality description), *company* (development company), *serial* (serial number), *revision* (hardware revision), *cpu* (CPU type), *frequency* (CPU frequency, in Hz), *memory* (random-access memory, in Bytes), *network* (network adapter), *version* (firmware version), *release* (firmware release date, TimeStamp), *calibration* (calibration date, TimeStamp), *expiration* (expiration of calibration date, TimeStamp), *reset* (time elapsed since reset or power on, in seconds), *operation* (total operation time, in seconds), *cycles* (number of power or reset cycles), *ethernet host* (Ethernet address), *ip host* (IP address), *ip subnet mask* (subnet mask), *ip gateway* (gateway IP address), *dhcp* (DHCP status), *hostname* (hostname for DNS look-up), *domain* (domain name). If the calibration or expiration of calibration dates is not applicable to the requested device, these fields are set to 'YYYY/MM/DD'.
In the following example this information is requested from device configured with IP address 192.168.1.100.

The method *Unbundle By Name* can be utilized to access the fields of the returned cluster as indicated in the example below.

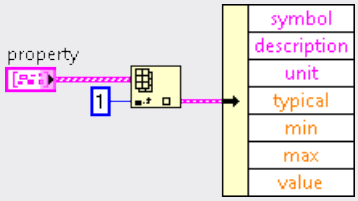


2.8. Acquisition of Device Status

Every device continuously monitors several parameters like supply voltages or the board temperature in order to ensure proper operation within specified conditions. These parameters can be retrieved with VI *Device Get Status* which does not expect any arguments but the device IP address or the DNS name and returns a *property* array of clusters and an *ok* flag. The *ok* flag indicates if all parameters are within their limits. The *property* cluster will be explained later on.
In the next example the status is requested from the device configured with IP address 192.168.1.100.



The number of elements of array *property* can be determined with function *Array Size*. An element of the array can be accessed with function *Index Array* as illustrated in the following example where the second cluster is retrieved.



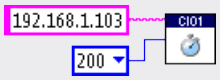
The *property* cluster has the following fields: *symbol* (commonly used symbol), *description* (description of the parameter), *unit* (SI unit), *typical* (typical value, NaN if not applicable), *min* (minimum value, NaN if not applicable), *max* (maximum value, NaN if not applicable), *value* (current value).

3. CIO1 Octal Current Monitor

The *CIO1* device is a versatile and easy-to-use current monitor. With eight inputs, it can be used with any industrial transducer with the current output ranging from 4 mA to 20 mA. The ultra-low noise, the high resolution and the outstanding accuracy make it ideal for industrial applications as well as for scientific experiments. The channels are multiplexed, amplified, conditioned and sampled by the high-performance 24-Bit delta-sigma A/D converter.

3.1. Set Sampling Frequency

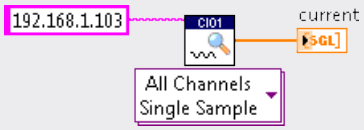
The sampling frequency of the A/D converter can be set with VI *CIO1 Set Frequency*. All eight channels are sampled one after the other at the specified rate. Valid values are 6, 12, 25, 50, 100, 200, 400, 800, 1500 and 3000 Hz. By default the sampling frequency is 6 Hz to ensure lowest noise suitable for most applications.
This is illustrated in the following example where the frequency of the device at IP address 192.168.1.103 is set to 200 Hz.



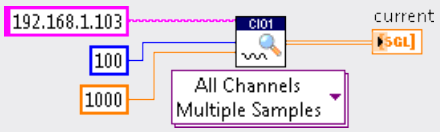
The sampling rate should not be set higher than necessary in order to keep the measurement noise as low as possible. Please refer to the data sheet for more details.

3.2. Measure Current

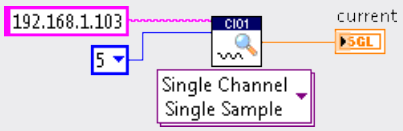
The VI *CIO1 Measure* is utilized to acquire one or more samples from one or more channels (in A). Various instances do exist to suit the demands. If a single sample of all eight channels shall be acquired instance *All Channels, Single Sample* is called which returns an array with currents of all eight channels which is illustrated below.



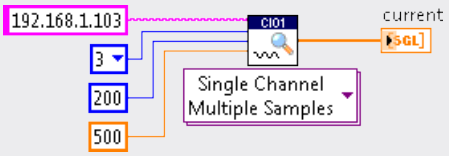
In order to acquire more samples from all eight channels instance *All Channels, Multiple Samples* has to be utilized. The device IP address or the DNS name, the number of samples (from 1 to 10^6) and the frequency (from 6 Hz to the sampling frequency set with VI *CIO1 Set Frequency*) is passed and a two-dimensional array is returned.
The example below acquires 100 samples for each channel of the device at IP address 192.168.1.103 at a sampling frequency of 1000 Hz.



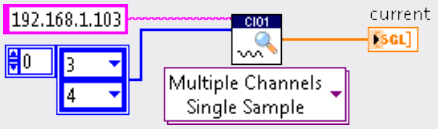
If only one channel is to be sampled, its number (from 1 to 8) and the device IP address or the DNS name are passed to instance *Single Channel, Single Sample*.
In the following example channel 5 of the device at IP address 192.168.1.103 is sampled.



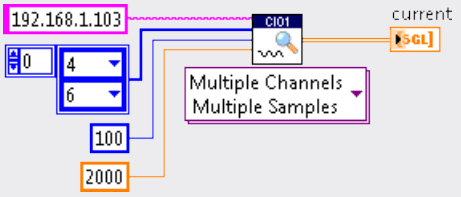
If more than one samples shall be acquired from one channel instance *Single Channel, Multiple Samples* expects four arguments: the device IP address or the DNS name, the channel number, the number of samples (from 1 to 10^6) and the sampling frequency (from 6 to the sampling frequency set with VI *CIO1 Set Frequency*).
In the next example 200 samples from channel 3 of the device at IP address 192.168.1.103 are acquired at 500 Hz.



If more than one channel shall be sampled the device IP address or the DNS name and an array with the channel numbers must be passed to instance *Multiple Channels, Single Sample*.
In the following example samples of channels 3 and 4 of the device at IP address 192.168.1.103 are acquired.



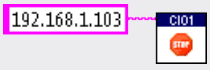
Moreover the acquisition of several samples from several channels is feasible. The expected arguments for instance *Multiple Channels, Multiple Samples* are the device IP address or the DNS name, an array with the channel numbers, the number of samples (from 1 to 10^6) and the sampling frequency (from 6 to the sampling frequency set with VI *CIO1 Set Frequency*).
In the next example at first 100 samples from channels 4 and 6 of the device at IP address 192.168.1.103 are acquired at 2000 Hz.



It is possible to make the instances *All Channels, Multiple Samples* and *Multiple Channels, Multiple Samples* transpose the returned array automatically. The appropriate flag has to be passed correspondingly.

3.3. Stop Measurement

If VI *CIO1 Measure* shall be terminated VI *CIO1 Stop* can be called which is typically done from another thread or different computer.
Apart from the IP address of the device or the DNS name the VI does not expect any arguments and is utilized as demonstrated below.

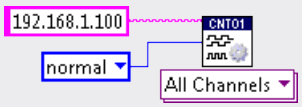


4. CNT01 Quad Quadrature Encoder

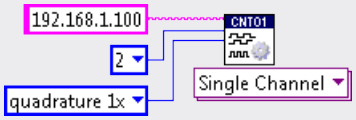
The *CNT01* device is a versatile and easy-to-use quad absolute counter and quadrature encoder. Every counter supports the up/down counting mode to detect impulses of an arbitrary clock source as well as the quadrature encoder mode which is usually utilized to count the revolutions per minute of a spinning shaft or motor.

4.1. Set Counting Mode

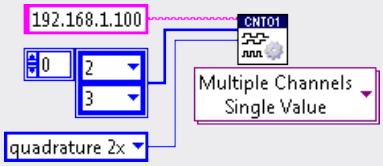
In order to change the counting mode of one or more channels VI *CNT01 Set Mode* is called. The following counting modes are supported: normal (direction and clock), 1x quadrature encoder, 2x quadrature encoder and 4x quadrature encoder. Refer to the data sheet for more information. Various instances do exist to suit the demands.
In the following example instance *All Channels* is utilized to configure all channels of the device at IP address 192.168.1.100 for normal counting mode.



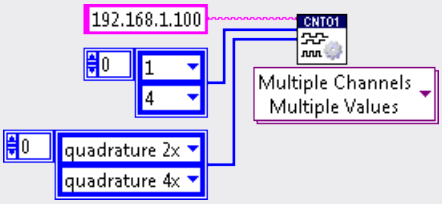
If the counting mode of only one channel is to be changed the instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel number (from 1 to 4) and the new counting mode.
In the next example channel 2 of the device at IP address 192.168.1.100 is configured for 1x quadrature counting.



Additionally the instance *Multiple Channels, Single Value* can be utilized to configure several channels. In that case the VI expects the device IP address or the DNS name, an array with the channel numbers and the new counting mode.
Below channels 2 and 3 of the device at IP address 192.168.1.100 are configured for 2x quadrature counting.



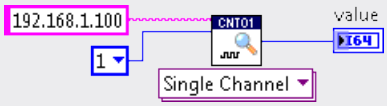
Moreover the instance *Multiple Channels, Multiple Values* is able to configure several channels for different counting modes at once. In that case the VI expects the device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new counting modes.
In the following example channels 1 and 4 of the device at IP address 192.168.1.100 are configured for 2x quadrature and 4x quadrature counting respectively.



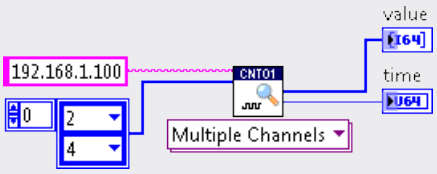
4.2. Read from Channels

The VI *CNT01 Read* is utilized to read the counter value of one or more channels. Various instances do exist to suit the demands. If the counter values of all four channels shall be acquired the instance *All Channels* is called and returns an array with counter values and the relative timestamp (in ms) which can be utilized for precise frequency or rotational speed measurements.
This is demonstrated in the next example where all channels of the device at IP address 192.168.1.100 are read out. Here the timestamp is discarded.

If only the counter value of one channel is to be read out, the channel number (from 1 to 4) is passed to the instance *Single Channel*. The relative timestamp is also discarded in this case.
Channel 1 of the device at IP address 192.168.1.100 is read out as shown in the example below.

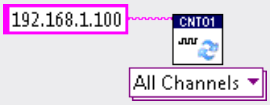


If more than one channel shall be read out an array with the channel numbers must be passed to the instance *Multiple Channels*. As in the previous examples, the relative timestamp is returned in addition to the counter values.
In the following example channels 2 and 4 of the device at IP address 192.168.1.100 are sampled and the timestamp is utilized this time.

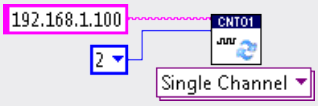


4.3. Clear Channels

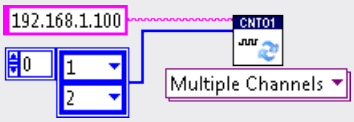
The VI *CNT01 Clear* is utilized to reset the counter value of one or more channels to zero. If the counter values of all four channels shall be reset the instance *All Channels* is called which is demonstrated in the example below.



If the counter value of only one channel is to be reset the channel number (from 1 to 4) is passed to the instance *Single Channel*.
In the next example channel 2 of the device at IP address 192.168.1.100 is reset.

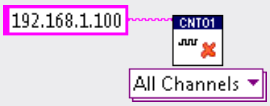


Additionally instance *Multiple Channels* can be utilized to reset the counter value of several channels to zero. In that case the device IP address or the DNS name and an array with the channel numbers is passed.
In the example below channels 1 and 2 of the device at IP address 192.168.1.100 are reset.

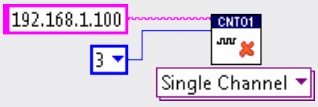


4.4. Disable Channels

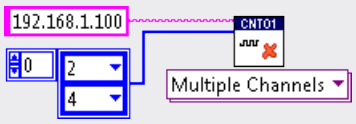
In order to disable one or more channels VI *CNT01 Disable* is utilized. If all four channels shall be disabled the instance *All Channels* is utilized which is illustrated in the example below.



If only one channel shall be disabled the device IP address or the DNS name and the channel number (from 1 to 4) is passed to instance *Single Channel*.
In the next example channel 3 of the device at IP address 192.168.1.100 is disabled.

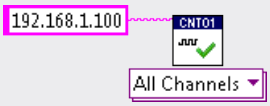


Additionally instance *Multiple Channels* can be utilized to disable several channels at once. In that case the device IP address or the DNS name and an array with the channel numbers is passed.
In the example below channels 2 and 4 of the device at IP address 192.168.1.100 are disabled.

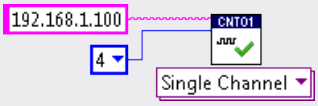


4.5. Enable Channels

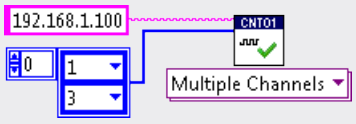
One or more channels are enabled by the use of VI *CNT01 Enable*. If all four channels shall be enabled the instance *All Channels* is utilized which is illustrated in the example below.



If only one channel shall be enabled the device IP address or the DNS name and the channel number (from 1 to 4) is passed to instance *Single Channel*.
In the next example channel 4 of the device at IP address 192.168.1.100 is enabled.



Additionally instance *Multiple Channels* can be utilized to enable several channels at the same time. In that case the device IP address or the DNS name and an array with the channel numbers is expected.
In the example below channels 1 and 3 of the device at IP address 192.168.1.100 are enabled.

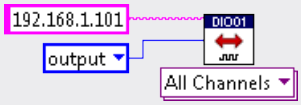


5. DIO01 Octal Bidirectional Digital I/O Device

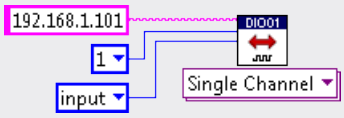
The *DIO01* device is a versatile and easy-to-use octal, bidirectional digital I/O module. Each of the eight channels can individually be configured for input or output. Custom timers, counters, pulse generators, logic analyzers, functional tests or digital communication protocols like the widespread SPI bus system can easily be implemented. Digital control loops and custom serial or parallel protocols can be realized in software and modifications are done much more comfortable compared to equivalent hardware solutions.

5.1. Set Channel Direction

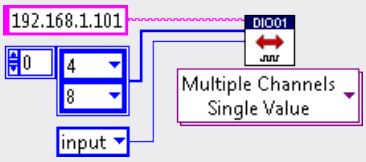
With the aid of VI *DIO01 Set Direction* every channel can individually be configured as input or output. Various instances do exist to suit the demands.
If the direction of all channels shall be changed the device IP address or the DNS name and the new direction is passed to instance *All Channels*. By default all eight channels are configured as input in order to protect any attached peripheral devices.
In the following example all channels of the device at IP address 192.168.1.101 are configured as output.



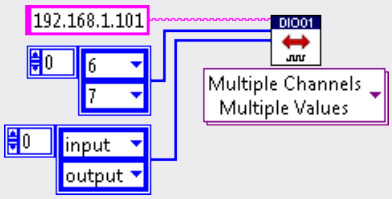
If the direction of only one channel is to be changed the instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel number (from 1 to 8) and the new direction.
In the next example channel 1 of the device at IP address 192.168.1.101 is configured as input.



Additionally the instance *Multiple Channels, Single Value* can be utilized to configure several channels for the same direction. The device IP address or the DNS name, an array with the channel numbers and the new direction are passed to the instance.
Below channels 4 and 8 of the device at IP address 192.168.1.101 are configured as input.



The instance *Multiple Channels, Multiple Values* is able to configure several channels for different directions at the same time. In that case the instance expects the device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new directions.
In the following example channels 6 and 7 of the device at IP address 192.168.1.101 are set to input and output respectively.

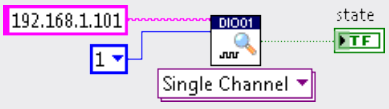


If a channel is switched from input to output direction, its initial logical state is low.

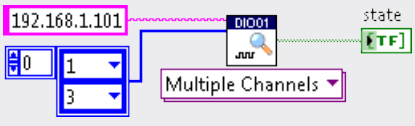
5.2. Read from Channels

The VI *DIO01 Read* is utilized to read the logical state of one or more channels which have been configured as input with VI *DIO01 Set Direction* before. Various instances do exist to suit the demands. If the logical states of all eight channels shall be acquired, instance *All Channels* is utilized which returns an array with logical states of all eight channels.
This is demonstrated in the next example where all channels of the device at IP address 192.168.1.101 are read out.

If only the logical state of one channel is to be read out, the device IP address or the DNS name and the channel number (from 1 to 8) are passed to instance *Single Channel*.
Channel 1 of the device at IP address 192.168.1.101 is read out as shown in the example below.



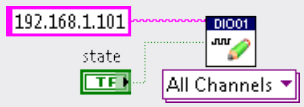
If more than one channel shall be read out the device IP address or the DNS name and an array with the channel numbers must be passed to instance *Multiple Channels*.
In the following example channels 1 and 3 of the device at IP address 192.168.1.101 are sampled.



5.3. Write to Channels

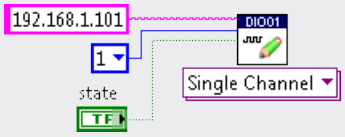
The VI *DIO01 Write* allows to set one or more channels to logical high (*true*) or logical low (*false*) which have been configured as output with VI *DIO01 Set Direction* before. Various instances do exist to suit the demands. If the logical state of all channels shall be changed, the new state is passed to instance *All Channels*.

In the following example all channels of the device at IP address 192.168.1.101 are updated.



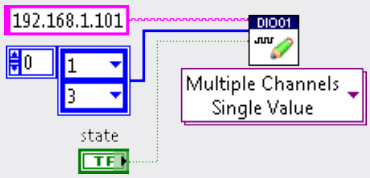
If only one channel is to be updated, the instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel (from 1 to 8) and the new logical state.

In the next example channel 1 of the device at IP address 192.168.1.101 is updated.



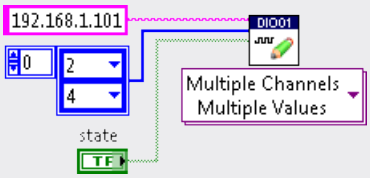
Additionally the instance *Multiple Channels, Single Value* can be utilized to update several channels at once. In that case the instance expects the device IP address or the DNS name, an array with the channel numbers and the new state.

Below channels 1 and 3 of the device at IP address 192.168.1.101 are updated.



The instance *Multiple Channels, Multiple Values* is able to update several channels to different logical states at the same time.The device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new states are passed to the instance.

In the following example channels 2 and 4 of the device at IP address 192.168.1.101 are updated.



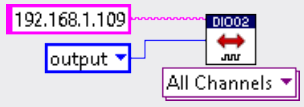
6. DIO02 Octal Bidirectional Digital I/O Device

The *DIO02* device is a versatile and easy-to-use octal, bidirectional digital I/O module which can be interfaced with open collector inputs or outputs. Every of the eight channels can individually be configured for input or output. Custom timers, counters, pulse generators, logic analyzers, functional tests or digital communication protocols can easily be implemented. Digital control loops and custom serial or parallel protocols can be realized in software and modifications are done in a much more comfortable way compared to equivalent hardware solutions.

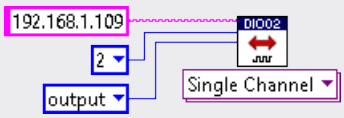
6.1. Set Channel Direction

With the aid of VI *DIO02 Set Direction* every channel can individually be configured as input or output. Various instances do exist to suit the demands. If the direction of all channels shall be changed the device IP address or the DNS name and the new direction is passed to instance *All Channels*. By default all eight channels are configured as input in order to protect any attached peripheral devices.

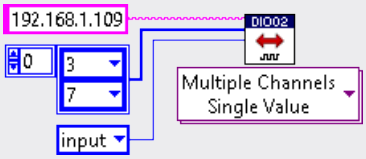
In the following example all channels of the device at IP address 192.168.1.109 are configured as output.



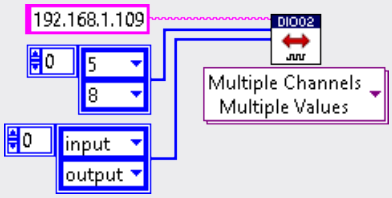
If the direction of only one channel is to be changed the instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel number (from 1 to 8) and the new direction.
In the next example channel 2 of the device at IP address 192.168.1.109 is configured as output.



Additionally the instance *Multiple Channels, Single Value* can be utilized to configure several channels for the same direction. The device IP address or the DNS name, an array with the channel numbers and the new direction are passed to the instance.
Below channels 3 and 7 of the device at IP address 192.168.1.109 are configured as input.



The instance *Multiple Channels, Multiple Values* is able to configure several channels for different directions at the same time. In that case the instance expects the device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new directions.
In the following example channels 5 and 8 of the device at IP address 192.168.1.109 are set to input and output respectively.

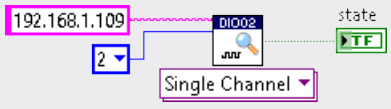


If a channel is switched from input to output direction, its initial logical state is low.

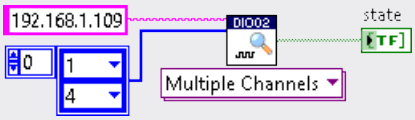
6.2. Read from Channels

The VI *DIO02 Read* is utilized to read the logical state of one or more channels which have been configured as input with VI *DIO02 Set Direction* before. Various instances do exist to suit the demands. If the logical states of all eight channels shall be acquired, instance *All Channels* is utilized which returns an array with logical states of all eight channels.
This is demonstrated in the next example where all channels of the device at IP address 192.168.1.109 are read out.

If only the logical state of one channel is to be read out, the device IP address or the DNS name and the channel number (from 1 to 8) are passed to instance *Single Channel*.
Channel 2 of the device at IP address 192.168.1.109 is read out as shown in the example below.



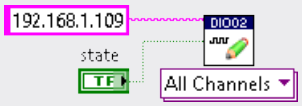
If more than one channel shall be read out the device IP address or the DNS name and an array with the channel numbers must be passed to instance *Multiple Channels*.
In the following example channels 1 and 4 of the device at IP address 192.168.1.109 are sampled.



6.3. Write to Channels

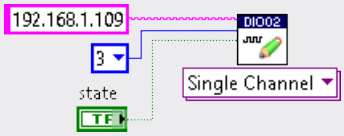
The VI *DIO02 Write* allows to set one or more channels to logical high (*true*) or logical low (*false*) which have been configured as output with VI *DIO01 Set Direction* before. Various instances do exist to suit the demands. If the logical state of all channels shall be changed, the new state is passed to instance *All Channels*.

In the following example all channels of the device at IP address 192.168.1.109 are updated.



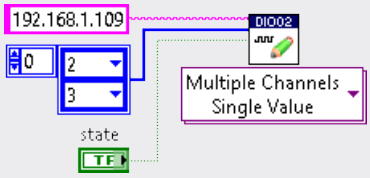
If only one channel is to be updated, the instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel (from 1 to 8) and the new logical state.

In the next example channel 3 of the device at IP address 192.168.1.109 is updated.



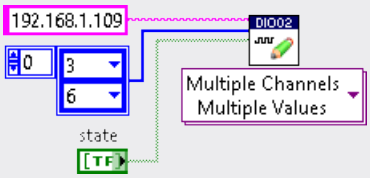
Additionally the instance *Multiple Channels, Single Value* can be utilized to update several channels at once. In that case the instance expects the device IP address or the DNS name, an array with the channel numbers and the new state.

Below channels 2 and 3 of the device at IP address 192.168.1.109 are updated.



The instance *Multiple Channels, Multiple Values* is able to update several channels to different logical states at the same time.The device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new states are passed to the instance.

In the following example channels 3 and 6 of the device at IP address 192.168.1.109 are updated.

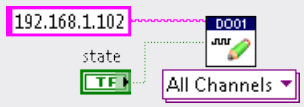


7. DO01 Octal Digital Output Buffer

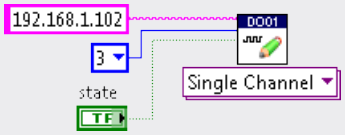
The *DO01* device is a versatile and easy-to-use octal high-current digital output module. Each of the eight channels can individually be set to logical low or high. Depending on the voltage rating of the connected transducers, an external power supply is needed to supply them. Electro-mechanical relays, electrical or electro-pneumatic valves, long transmission lines, digital transducers or even DC motors can directly be interfaced with the *DO01* device.

7.1. Write to Channels

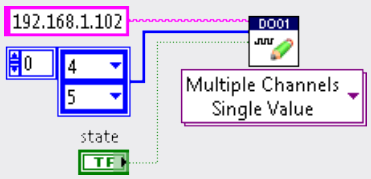
The VI *DO01 Write* allows to set one or more channels to logical high (*true*, supply voltage) or logical low (*false*, no voltage). By default the state of all eight channels is logical low in order to keep the outputs unpowered. Various instances do exist to suit the demands. If the logical state of all channels shall be changed, the device IP address or the DNS name and the new state are passed to instance *All Channels* which is shown in the example below.



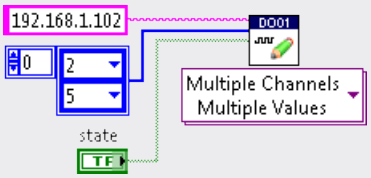
If only one channel is to be updated the instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel (from 1 to 8) and the new logical state.
In the next example channel 3 of the device at IP address 192.168.1.102 is updated.



Additionally instance *Multiple Channels, Single Value* can be utilized to update several channels to the same new state. In that case the instance expects the device IP address or the DNS name, an array with the channel numbers and the new state.
Below channels 4 and 5 of the device at IP address 192.168.1.102 are updated.



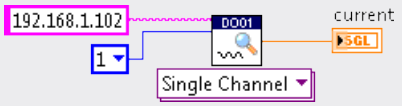
The instance *Multiple Channels, Multiple Values* is able to update several channels to different logical states at the same time. The device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new states are passed to the instance.
In the following example channels 2 and 5 of the device at IP address 192.168.1.102 are updated.



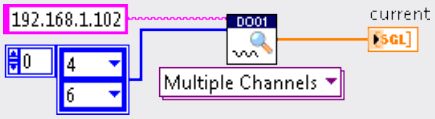
7.2. Measure Current

The *DO01* device is able to measure the current (in A) through all eight channels to provide useful feedback information. With the aid of VI *DO01 Measure* the current of one or more channels can be read out. If the currents of all eight channels shall be acquired the instance *All Channels* must be utilized which returns an array with currents of all eight channels.
In the next example the currents of all channels of the device at IP address 192.168.1.102 are measured.

If only the current of one channel is to be acquired, the device IP address or the DNS name and the channel number (from 1 to 8) are passed to instance *Single Channel*.
Channel 1 of the device at IP address 192.168.1.102 is read out as shown in the example below.



If more than one channel shall be read out the device IP address or the DNS name and an array with the channel numbers must be passed to instance *Multiple Channels*.
In the following example channels 4 and 6 of the device at IP address 192.168.1.102 are sampled.

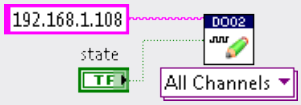


8. DO02 Octal SPST Relay Module

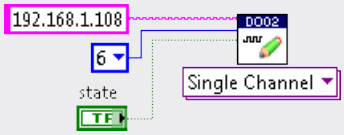
The *DO02* device is a octal easy-to-use high-voltage high-current relay module. Each channel can be enabled or disabled individually. Electrical or electro-pneumatic valves, heaters or AC / DC motors can be controlled with the *DO02* device.

8.1. Write to Channels

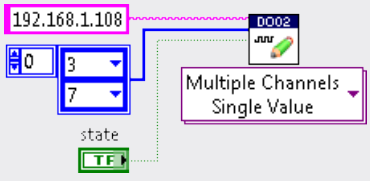
The VI *DO02 Write* allows to enable (*true)* or disable (*false*) one or more relays. By default the state of all relays is disabled in order to keep the attached components unpowered, if applicable. Various instances do exist to suit the demands. If the state of all relays shall be changed, the device IP address or the DNS name and the new state are passed to instance *All Channels* which is shown in the example below.



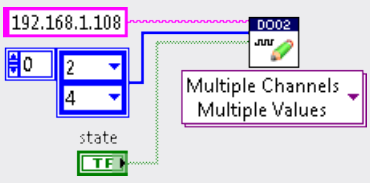
If only one relay is to be updated the instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel (from 1 to 8) and the new state.
In the next example channel 6 of the device at IP address 192.168.1.108 is updated.



Additionally instance *Multiple Channels, Single Value* can be utilized to update several relays to the same new state. In that case the instance expects the device IP address or the DNS name, an array with the channel numbers and the new state.
Below channels 3 and 7 of the device at IP address 192.168.1.108 are updated.



The instance *Multiple Channels, Multiple Values* is able to update several relays to different states at the same time. The device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new states are passed to the instance.
In the following example channels 2 and 4 of the device at IP address 192.168.1.108 are updated.

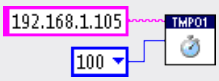


9. TMP01 Octal Thermocouple Monitor

The *TMP01* device is a versatile and easy-to-use temperature monitor. With eight inputs, it can be used with nearly any thermocouple type. The device was designed to meet the demands of scientific or industrial applications where the high temperature range, ultra-low noise and high resolution are important concerns.

9.1. Set Sampling Frequency

The sampling frequency of the A/D converter can be set with VI *TMP01 Set Frequency*. All eight channels are sampled one after the other at the specified rate. The VI expects the device IP address or the DNS name and the new sampling frequency. Valid values are 6, 12, 25, 50, 100, 200, 400, 800, 1500 and 3000 Hz. By default the sampling frequency is 6 Hz to ensure lowest noise which is suitable for most applications.
This is illustrated in the following example where the frequency of the device at IP address 192.168.1.105 is set to 100 Hz.

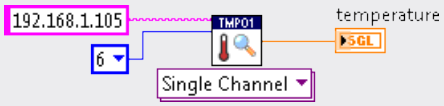


The sampling rate should not be set higher than necessary in order to keep the measurement noise as low as possible. Please refer to the data sheet for more details.

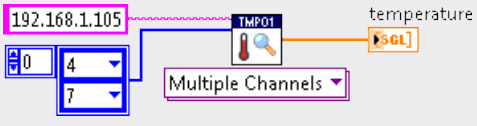
9.2. Measure Temperature

The VI *TMP01 Measure* is utilized to acquire one or more samples (in K) from one or more channels. Various instances do exist to suit the demands. If all eight channels shall be sampled instance *All Channels* is be called without any arguments but the device IP address or the DNS name and returns an array with temperatures of all eight channels which is demonstrated in the example below.

If only one channel is to be sampled, its number (from 1 to 8) has to be passed to instance *Single Channel*.
In the example below channel 6 of the device at IP address 192.168.1.105 is read out.

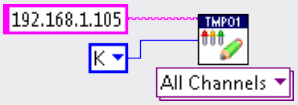


If more than one channel shall be sampled the device IP address or the DNS name and an array with the channel numbers must be passed to instance *Multiple Channels*.
In the following example channels 4 and 7 of the device at IP address 192.168.1.105 are sampled.

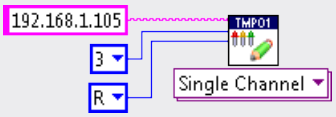


9.3. Set Temperature Sensor

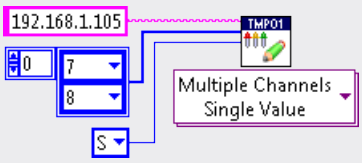
With the use of VI *TMP01 Set Sensor* every channel can individually be configured to be used with a different type of thermocouple. If all channels of a device shall be utilized with the same type of thermocouple, the new type is passed to instance *All Channels*. The following thermocouple types are supported: B, C, E, J, K, M, N, P, R, S and T. Additionally custom thermocouples with arbitrary curves can be utilized when thermocouple type *custom* is configured. In that case the VI returns the measured voltage (from -12 mV to +79 mV). By default all channels are disabled in order to avoid unintentional misconfiguration.
In the following example all channels of the device at IP address 192.168.1.105 are set to thermocouple type K.



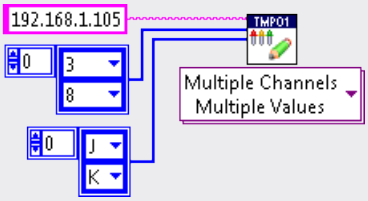
If one or more channels are not utilized, these channels should be configured as *none* which increases the effective sampling rate of the remaining channels. If only one channel is to be configured instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel number (from 1 to 8) and the thermocouple type.
In the next example channel 3 of the device at IP address 192.168.1.105 is configured for thermocouple type R.



Additionally instance *Multiple Channels, Single Value* can be utilized to configure several channels for the same thermocouple type. The device IP address or the DNS name and an array with the channel numbers and the new thermocouple type has to be passed to the instance.
Below channels 7 and 8 of the device at IP address 192.168.1.105 are configured for thermocouple type S.



The instance *Multiple Channels, Multiple Values* is able to configure several channels to be utilized with different thermocouple types at the same time. In that case the device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new thermocouple types.
In the next example channels 3 and 8 of the device at IP address 192.168.1.105 are set to thermocouple type J and K respectively.

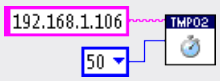


10. TMP02 Quad RTD Monitor

The *TMP02* device is a versatile and easy-to-use temperature monitor. With four inputs, it can be used with Platinum resistors, temperature diodes or NTC thermistors. The device was designed to meet the demands of scientific or industrial applications where the high temperature range, ultra-low noise and high resolution are important concerns.

10.1. Set Sampling Frequency

The sampling frequency of the A/D converter can be set with VI *TMP02 Set Frequency*. All four channels are sampled one after the other at the specified rate. The VI expects the device IP address or the DNS name and the new sampling frequency. Valid values are 6, 12, 25, 50, 100, 200, 400, 800, 1500 and 3000 Hz. By default the sampling frequency is 6 Hz to ensure lowest noise which is suitable for most applications.
This is illustrated in the following example where the frequency of the device at IP address 192.168.1.106 is set to 50 Hz.

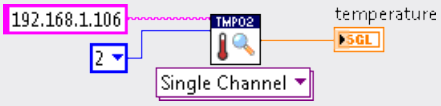


The sampling rate should not be set higher than necessary in order to keep the measurement noise as low as possible. Please refer to the data sheet for more details.

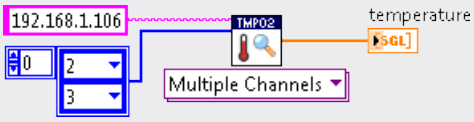
10.2. Measure Temperature

The VI *TMP02 Measure* is utilized to acquire one or more samples (in K) from one or more channels. Various instances do exist to suit the demands. If all four channels shall be sampled instance *All Channels* is be called without any arguments but the device IP address or the DNS name and returns an array with temperatures of all four channels which is demonstrated in the example below.

If only one channel is to be sampled, its number (from 1 to 4) has to be passed to instance *Single Channel*.
In the example below channel 2 of the device at IP address 192.168.1.106 is read out.

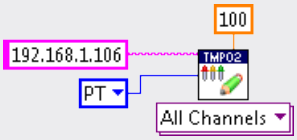


If more than one channel shall be sampled the device IP address or the DNS name and an array with the channel numbers must be passed to instance *Multiple Channels*.
In the following example channels 2 and 3 of the device at IP address 192.168.1.106 are sampled.

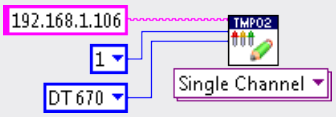


10.3. Set Temperature Sensor

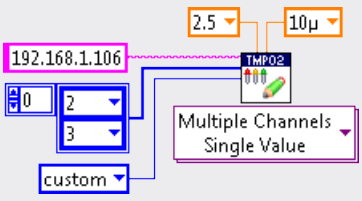
With the use of VI *TMP02 Set Sensor* every channel can individually be configured to be used with a different type of sensor. If all channels of a device shall be utilized with the same type of sensor, the new type is passed to instance *All Channels*. The following sensor types are supported: PT, DT 470, DT 670, KTY 81-1, KTY 81-2, KTY 82-1, KTY 82-2, KTY 83-1 and KTY 84-1. Additionally custom sensors with arbitrary curves can be utilized when sensor type *custom* is configured. In that case the excitation current (10 µA or 1 mA) and the voltage range (0.5 V, 1.0 V, 2.5 V or 5.0 V) need to be configured and the VI returns the measured voltage. By default all channels are disabled in order to avoid unintentional misconfiguration.
In the following example all channels of the device at IP address 192.168.1.106 are set to PT100 resistors.



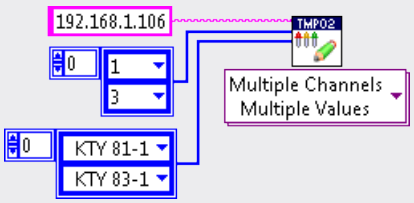
If one or more channels are not utilized, these channels should be configured as *none* which increases the effective sampling rate of the remaining channels. If only one channel is to be configured instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel number (from 1 to 4) and the sensor type.
In the next example channel 1 of the device at IP address 192.168.1.106 is configured for DT 670 temperature diode.



Additionally instance *Multiple Channels, Single Value* can be utilized to configure several channels for the same sensor type. The device IP address or the DNS name and an array with the channel numbers and the new sensor type has to be passed to the instance.
Below channels 2 and 3 of the device at IP address 192.168.1.106 are configured for custom sensor with excitation current of 10 µA and the voltage range of 2.5 V.



The instance *Multiple Channels, Multiple Values* is able to configure several channels to be utilized with different sensor types at the same time. In that case the device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new sensor types.
In the next example channels 1 and 3 of the device at IP address 192.168.1.106 are set to thermistors KTY 81-1 and KTY 83-1 respectively.

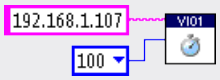


11. VI01 Octal Voltage Monitor

The *VI01* device is a versatile and easy-to-use voltage monitor. With eight inputs, it can be used with any industrial transducer with the voltage output ranging from -10 V to +10 V. The ultra-low noise, the high resolution and the outstanding accuracy make it ideal for industrial applications as well as for scientific experiments. The channels are multiplexed, amplified, conditioned and sampled by the high-performance 24-Bit delta-sigma A/D converter.

11.1. Set Sampling Frequency

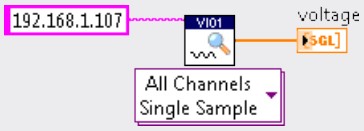
The sampling frequency of the A/D converter can be set with VI *VI01 Set Frequency*. All eight channels are sampled one after the other at the specified rate. Valid values are 6, 12, 25, 50, 100, 200, 400, 800, 1500 and 3000 Hz. By default the sampling frequency is 6 Hz to ensure lowest noise suitable for most applications.
This is illustrated in the following example where the frequency of the device at IP address 192.168.1.107 is set to 100 Hz.



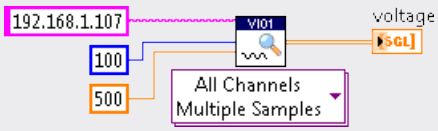
The sampling rate should not be set higher than necessary in order to keep the measurement noise as low as possible. Please refer to the data sheet for more details.

11.2. Measure Voltage

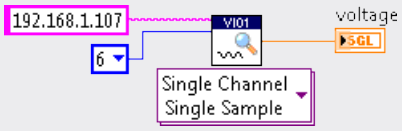
The VI *VI01 Measure* is utilized to acquire one or more samples from one or more channels (in V). Various instances do exist to suit the demands. If a single sample of all eight channels shall be acquired instance *All Channels, Single Sample* is called which returns an array with voltages of all eight channels which is illustrated below.



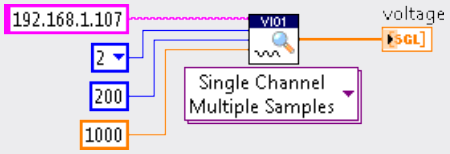
In order to acquire more samples from all eight channels instance *All Channels, Multiple Samples* has to be utilized. The device IP address or the DNS name, the number of samples (from 1 to 10^6) and the frequency (from 6 Hz to the sampling frequency set with VI *VI01 Set Frequency*) is passed and a two-dimensional array is returned.
The example below acquires 100 samples for each channel of the device at IP address 192.168.1.107 at a sampling frequency of 500 Hz.



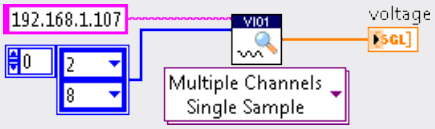
If only one channel is to be sampled, its number (from 1 to 8) and the device IP address or the DNS name are passed to instance *Single Channel, Single Sample*.
In the following example channel 6 of the device at IP address 192.168.1.107 is sampled.



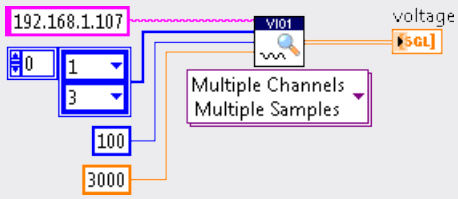
If more than one samples shall be acquired from one channel instance *Single Channel, Multiple Samples* expects four arguments: the device IP address or the DNS name, the channel number, the number of samples (from 1 to 10^6) and the sampling frequency (from 6 to the sampling frequency set with VI *VI01 Set Frequency*).
In the next example 200 samples from channel 2 of the device at IP address 192.168.1.107 are acquired at 1000 Hz.



If more than one channel shall be sampled the device IP address or the DNS name and an array with the channel numbers must be passed to instance *Multiple Channels, Single Sample*.
In the following example samples of channels 2 and 8 of the device at IP address 192.168.1.107 are acquired.



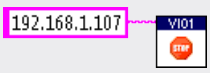
Moreover the acquisition of several samples from several channels is feasible. The expected arguments for instance *Multiple Channels, Multiple Samples* are the device IP address or the DNS name, an array with the channel numbers, the number of samples (from 1 to 10^6) and the sampling frequency (from 6 to the sampling frequency set with VI *VI01 Set Frequency*).
In the next example at first 100 samples from channels 1 and 3 of the device at IP address 192.168.1.107 are acquired at 3000 Hz.



It is possible to make the instances *All Channels, Multiple Samples* and *Multiple Channels, Multiple Samples* transpose the returned array automatically. The appropriate flag has to be passed correspondingly.

11.3. Stop Measurement

If VI *VI01 Measure* shall be terminated VI *VI01 Stop* can be called which is typically done from another thread or different computer.
Apart from the IP address of the device or the DNS name the VI does not expect any arguments and is utilized as demonstrated below.



12. VO01 Octal Voltage Output Device

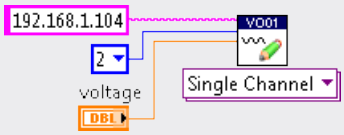
The *VO01* device is a versatile and easy-to-use voltage output device. With eight outputs, it can be used with any analog industrial interface with voltage input ranging from -10 V to +10 V. The ultra-low noise, the high resolution and the outstanding accuracy make it ideal for industrial applications as well as for scientific experiments.

12.1. Control the Output Voltage

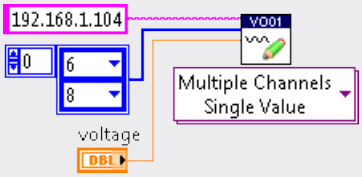
In order to control the output voltage of one or more channels VI *VO01 Control* is utilized. If all eight channels of a device shall be modified the device IP address or the DNS name and the new voltage (from -10 V to +10 V) is passed to instance *All Channels*. By default all channels are set to 0 V to protect any attached peripheral devices.

In the following example the utilization is demonstrated.

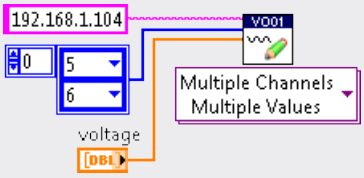
If only one channel is to be modified instance *Single Channel* is called with three arguments: the device IP address or the DNS name, the channel (from 1 to 8) and the voltage (from -10 V to +10 V).
In the following example channel 2 of the device at IP address 192.168.1.107 is modified.



Additionally instance *Multiple Channels, Single Value* can be utilized to set several channels to a new voltage at the same time. In that case the instance expects the device IP address or the DNS name, an array with the channel numbers and the new voltage.
Below channels 6 and 8 of the device at IP address 192.168.1.107 are modified.



The instance *Multiple Channels, Multiple Values* is able to change the voltages of several channels to different values at once. The instance expects the device IP address or the DNS name, an array with the channel numbers and an array of the same size with the new voltages.
In the following example channels 5 and 6 of the device at IP address 192.168.1.107 are updated.



13. Error Handling

If one of VIs described in that document could not be executed properly an error is returned which shall be evaluated by the caller of the method. The reason of the failure can be deduced from the error code and the error message. Please refer to the LabVIEW™ programming manual for more information on error handling techniques.
If for example the voltage of channel 9 of device *VI01* shall be measured, an *invalid parameter* error is triggered since only eight channels are available. If the error out pin of the VI is left unconnected a dialog window with an error message will appear.