attoCONTROL

Electronics & Software Control Units

User Manual

AMC100 Motion Controller

attocube
WITTENSTEIN Group

Products: AMC100, MOVE Software

Table of Contents

I. Notes on This Manual

I.1. Purpose and Availability

This user manual applies to the AMC100, also referred to as "device" in this document.

The user manual explains the installation and operation of the device. It contains instructions for the appropriate use and the maintenance of the device.

- Read this manual before setting up and using the device.
- Always keep this manual easily accessible for all device users.

I.2. Symbols and Conventions

I.2.a. Warning Notes

Warning
This is warning information about hazards that can cause death or severe injuries if the suitable precautions are not taken.

Caution
This is warning information about hazards that can cause injuries if the suitable precautions are not taken.

Note
This is warning information about possible property damages or function restrictions.

Tip
This note provides additional information to simplify your work.

I.2.b. Symbols

For the continuing safety of the operator of this equipment and the protection of the equipment itself, the operator must take notice of the warning symbols and notes throughout this manual and - where applicable - on the device itself.

The following safety symbols are used in this manual:

Hot surface! May cause injury when touched.

General hazard! An instruction which draws attention to the risks of damage to the device, process, or surrounding.

Risk of electric shock! High voltages present. May cause injury or death when touched.

Clarification of an instruction or additional information.

I.2.c. Information Markup

The following design conventions are used in this document to improve traceability:

- Names of organizational elements like folders, files, screens, options etc. are marked by "double quotes".
- Instruction steps which are part of a sequence are displayed with leading ordinal number.
- Instruction steps which are variable or alternative are displayed with leading circle.
- Results of instruction parts are displayed with a leading arrow.
- References to parts of graphics are displayed in **bold** cardinal numbers.
- Software code or program text is displayed in `special font`.
- Variables and wildcards are displayed in *italic font*.
- Keys and buttons are marked by [square brackets].
- References to internal document parts are displayed in orange font.

II. Declarations

II.1. Declaration of Conformity

For Customers in Europe

This equipment has been tested and found to comply with the EC Directives 2014/30/EU "EMC Directive" and 2014/35/EU "Low Voltage Directive".

Compliance was demonstrated by conformance to the following specifications which have been listed in the Official Journal of the European Communities:

- Safety EN61326-1: 2013
- EMC EN61326-1: 2013

For Customers in the USA

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules and meets all requirements of the Canadian Interference-Causing Equipment Standard ICES-003 for digital apparatus. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/T.V. technician for help.

Changes or modifications to the AMC100 not explicitly approved by attocube could void the user's authority to operate the equipment.

II.2. Waste Electrical and Electronic Equipment (WEEE) Directive

Compliance

As required by the Waste Electrical and Electronic Equipment (WEEE) Directive of the European Community and the corresponding national laws, attocube offers all end users within the European Union (EU) the possibility to return "end of life" units without incurring disposal charges.

This offer is valid for attocube's electrical and electronic equipment:

- sold after August 13th 2005,
- marked correspondingly with the crossed out "wheelie bin" logo (see logo to the left),
- sold to a company or institute within the EU,
- currently owned by a company or institute within the EU,
- still complete, not disassembled, and not contaminated.

As the WEEE directive applies to self- contained operational electrical and electronic products, this "end of life" take back service does not refer to other attocube products, such as

- pure OEM products, that means assemblies to be built into a unit by the user (e. g. OEM electronic drivers),
- components,
- mechanics and optics,
- left over parts of units disassembled by the user (PCBs, housings etc.).

If you wish to return an attocube unit for waste recovery, please contact attocube or your nearest dealer for further information.

Waste Treatment on Your Own Responsibility

If you do not return an "end of life" unit to attocube systems, you must hand it to a company specialized in waste recovery. Do not dispose of the unit in a litter bin or at a public waste disposal site.

Ecological Background

It is well known that WEEE pollutes the environment by releasing toxic products during decomposition. The aim of the European RoHS directive is to reduce the content of toxic substances in electronic products in the future.

The intent of the WEEE directive is to enforce the recycling of WEEE. A controlled recycling of end of life products will thereby avoid negative impacts on the environment.

III. Safety Information

III.1. Important Warnings

Warning
Risk of electric shock!

Inappropriate handling of the device may cause death, severe injury or material damage.

- Never remove the device's protective covers or attempt any repair or adjustment!
- Immediately shut off the device, disconnect the mains supply and contact attocube in case of any suspected malfunction!
- Never connect any cabling to the electronics when the outputs are enabled!
- Be careful not to create short-circuits on the connectors or anywhere in the cabling!

Caution
General hazard!

The device's operation under inappropriate conditions may lead to injury or material damage.

- Do not operate the device outside its dedicated supply voltage or environmental limits as specified in IV.4.
- Do not operate the device unless you are properly trained in the use and handling of mains powered electrical equipment.

Note
Servicing and maintenance is only allowed to persons with explicit authorization of attocube. There are no user serviceable parts on the device. Modified or open electronics are no longer covered by attocube's warranty.

- Do not open the device.
- For servicing and repair always contact attocube.

Note
Unauthorized updates can lead to a permanent malfunction and are not covered by attocube's warranty.

- Do not update the firmware of the device without authorization of attocube.

III.2. Labels on the Device

The following labels can be found on the device.

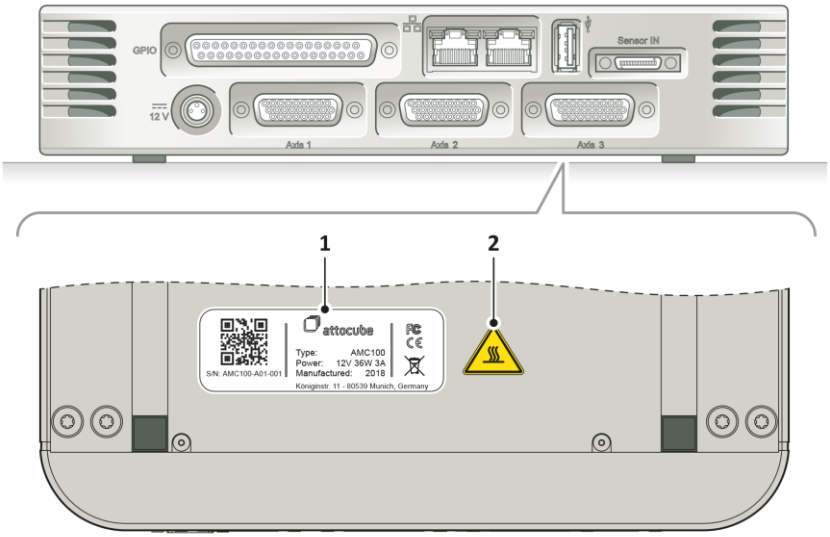
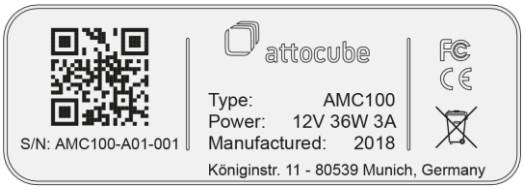


Figure 1 Labels on the device

No.	Label	Explanation	
1		AMC100 type label	
		QR-Code	Link to device support
		S/N	Serial number
		Type	Product name
		Power	Rated power
		Manufactured	Year of manufacture
		FCC	FCC conformity
		CE	CE conformity
		Crossed out wheelie bin	Disposal advice for EU
2		Warning label **Hot surface!**	

IV. Device Description

IV.1. Intended Use

The AMC100 is a controller for the simultaneous operation of up to three positioners out of attocube's ECx series of industrial line Positioners. Depending on the customer's choice, the set of positioners controlled can be an arbitrary combination of one to three dimensional, linear, rotary or goniometric positioners with open-loop or closed-loop feedback.

The ACM100 can be operated via PC using the MOVE software or a webserver interface. Additionally, a *.DLL library is delivered with each device, enabling the integration of the controller's functionalities into customized software routines. A /PRO version with extended feature set is also available, as well as a /IO Feature activating various Trigger In- and Output Options to choose from.

The typical field of application are user steered or automated processes in laboratory and industrial use.

IV.2. Scope of Delivery

The following components are part of the delivery:

- Positioning controller AMC100
- Power cable with mains adapter (suitable for use in UK, Europe or USA)

Note
The unit is shipped with appropriate power cables for usage in the UK, Europe or the USA. When shipped to other territories, the appropriate power plug has to be provided by the user.

- USB-to-Ethernet adapter
- Ethernet cable
- USB flash drive with MOVE software, dynamic link library and documentation

IV.3. Technical Specification

Parameter	Value
Dimensions	220 x 220 x 45 mm
Weight	2 kg
Power input	12 VDC, 3 A
Overvoltage Category	II
Pollution Degree	2

IV.4. Operation Requirements

Caution

If the AMC100 is used in a manner not specified in this manual or by attocube, the protection provided by the AMC100 may be impaired!

Caution

Inappropriate working conditions!

The device's operation under inappropriate conditions may lead to injury or material damage.

- Do not operate the device outdoor.
- Do not operate the device outside its dedicated voltage supply and environmental limits.

Caution

Wrong cabling!

Inadequate equipment may cause electric shocks or fire.

- Only use power supplies and cables provided by attocube!

The following environmental limits have to be observed when operating the device with the Power Supply provided by attocube.

Parameter	Value
Supply voltage	90 V to 240 V AC, +/- 10 %
Line frequency	50 Hz to 60 Hz
Operational area	Indoor use only
Maximum altitude	2000 m
Minimum temperature	5 °C
Maximum temperature	40 °C
Relative humidity at about 30 °C	< 80 % (non-condensing)

Note

The contact with chemicals, humidity or dirt may damage the device.

- Do not expose the device to corrosive agents or excessive moisture, heat or dust.
- To clean the AMC100 only use clean and dry cloths

V. Device Setup

V.1. Unpacking and Positioning

How

1. Carefully unpack the controller and the accessories.
2. Inspect the controller and the accessories for any damage.
   - No damage detected: continue with step 3.
   - Damage detected: contact attocube.
3. Place all components on a flat and clean surface.

Note
Inadequate positioning may lead to malfunctions or damage the device.

- Position the device in such a way that the operation of the power supply plug is not impeded.
- Do not obscure the ventilation slots. Make sure that proper airflow is maintained to the unit.

V.2. Connecting

V.2.a. Connector Panel

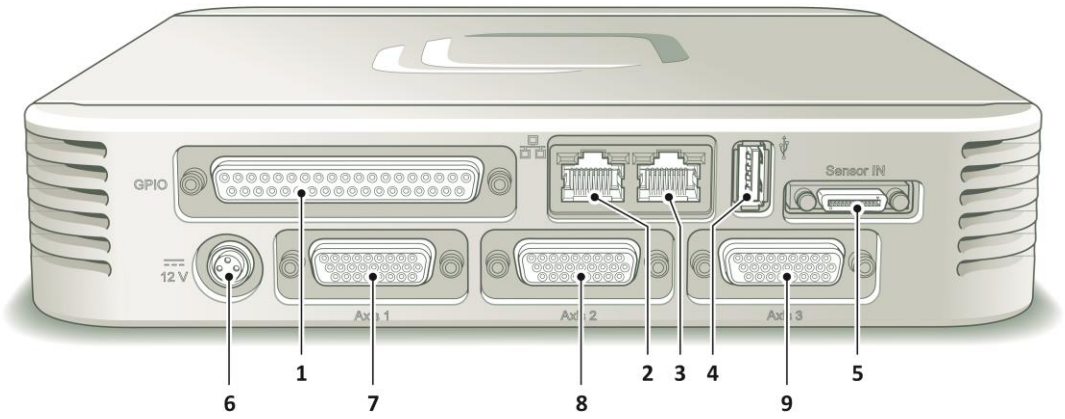


Figure 2 Connector panel

1	GPIO (General Purpose Input/Output)
2	Ethernet uplink
3	Ethernet downlink (only in /PRO version)
4	USB host
5	Interferometric sensor input
6	Power (12 VDC)
7	Axis 1
8	Axis 2
9	Axis 3

Tip
The second Ethernet Port is activated through the /PRO feature upgrade. It can be used to avoid having to connect multiple cables to one PC or having to use a router by instead simply daisy-chaining multiple AMCs (set to static IPs).

V.2.b. Connecting to Voltage Supply

Caution
Wrong cabling!

Inadequate equipment may cause electric shocks or fire.

- Always use power supply cables provided by attocube!

How

1. Connect the device to the supply mains via the power cable and the device's voltage supply socket (Figure 2/**6**).

Figure 3 Operator panel

2. Push the power button at the device's front side (Figure 3/**1**).

Tip
The sockets for outgoing voltage are insulated and do not conduct power unless there are positioners connected.

V.2.c. Connecting to PC via USB

System Requirements

The following requirements have to be met by the PC.

Parameter	Value
Processor design	x86 or x64
USB interface	USB 2.0
Operating System	Microsoft Windows XP® or higher

Connection Procedure

How

1. Plug in the USB connector of the USB-to-Ethernet adapter into the USB socket at your PC.
2. Check if the driver installation of the USB-to-Ethernet adapter starts automatically.
   - Driver installation starts automatically: continue with step 5.
   - Driver installation does not start automatically: continue with step 3.

3. On your PC, open the Windows explorer and navigate to the USB-to-Ethernet adapter's folder (Figure 4) in case a TP-LINK Adapter is used. If another Adapter is used, get the driver from the respective Manufacturers Homepage or the USB flash drive or CD, if provided by the Manufacturer.

Figure 4 Windows explorer, folder of USB-to-Ethernet adapter

4. Double-click the file ending on ".exe".
   → The driver installation is executed.
5. Plug in the connector of the Ethernet cable into the corresponding socket at the USB-to-Ethernet adapter.
6. Plug in the connector of the Ethernet cable into the device's Ethernet socket (Figure 2/**2**).

V.2.d. Connecting to Network

Note
If several controllers shall be connected to the PC an Ethernet switch must be interposed between the USB-to-Ethernet adapter and the devices. An Ethernet switch is not contained in the scope of delivery.

What AMC100 devices can be integrated into LANs. In this case the IP address of the device has to be adapted manually for network integration.

Where Webserver application, "Configuration" screen, "Networking" section

How

1. To open the webserver application, follow the instructions in VIII.1.
2. To open the "Configuration" screen, click [Configuration] in the header of the webserver application.
3. To adapt the IP address, follow the instructions in X.3.a.
4. Close the web browser.

V.2.e. Connecting Positioner

Caution
Wrong connection!

Inadequate connections may cause injury and are likely to damage the device or interfere with an appropriate functioning.

- When connecting the device to customer hardware, carefully take note of the warnings and specifications given in XII.1!

Room temperature use

The positioners for room temperature use are delivered with the following positioner control cables:

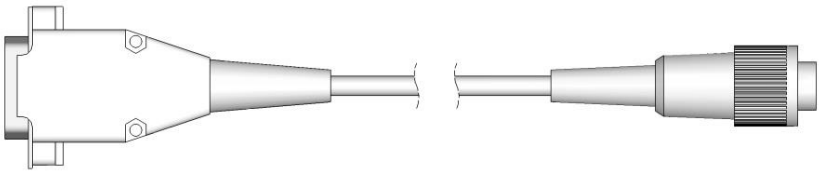


Figure 5 Positioner control cable for room temperature use

Vacuum use The positioners for vacuum use are delivered with the following positioner control cables.

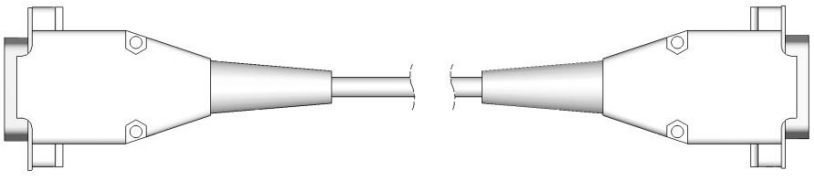


Figure 6 Positioner control cable for connection to vacuum feedthrough

> **Tip**
> attocube optionally provides specific vacuum feedthroughs for one or multiple axes. Contact attocube for more information.

Each cable has a D-sub 26 HD connector on the controller's side.

How

1. Plug in the positioner control cable into the positioner's connector or the vacuum feedthrough's socket, accordingly.

> **Tip**
> Use the test adapter to directly connect a positioner control cable to a positioner for vacuum use.

2. Plug in the D-Sub connector of the positioner control cable into one of the corresponding sockets at the device (Figure 2/**7**, **8**, **9**).

V.2.f. Disconnecting Positioner

> **Caution**
> Risk of electric shock
>
> Contact with power guiding connectors can cause injuries or material damage.
>
> - Always deactivate an axis before physically disconnecting the corresponding positioner!

A positioner can be disconnected at any time in the process, e.g. for changing positioners to be controlled by the device.

How

1. To deactivate an axis, follow the instructions in VII.8.a (for the MOVE software) or VIII.4.a (for the webserver application), respectively.
2. To disconnect a positioner, unplug the positioner control cable from your PC or from the positioner's socket.

V.3. Installing the Software MOVE

How

1. Connect the USB flash drive contained in the scope of delivery to your PC.
2. On your PC, open the Windows explorer and navigate to the USB flash drive's folder.
3. Copy the folder "MOVE" to the directory of your choice on your hard drive.
   → On the operating system Windows 10® the software is ready to use.

If you're using an operating system older than Windows 10® a few additional steps might be necessary:

4. Navigate to the MOVE folder on your hard drive (Figure 7) and double-click the file named MOVE.

Figure 7 Windows explorer, folder of the MOVE software

   → A dialog window appears, prompting you to download and install the .NET framework if the installed version is older than version 4.6.

5. Follow the instructions to download and install the .NET framework.
   → The software is ready to use when the .NET framework is installed.

VI. Device Start Up

How

1. To start the device, push the power button (Figure 3/**1**).
2. Start up your PC.

> **Tip**
> According to your preferences you can work with the AMC100 in one of two possible ways:
>
> - Using the MOVE software (see VII)
> - Using the webserver application (see VIII)

VII. Operation with MOVE Software

VII.1. Starting MOVE Software

How

- On your PC, double-click the MOVE icon.
  → The software starts.

VII.2. Restoring Former Axis Selection

What When starting the software, MOVE automatically checks the connected networks for devices (identified by SN) that were also connected when the software ran the last time. If so, the "Restore Selection" dialog opens.

The "Restore Selection" dialog allows you to comfortably restore the last axis selection, i.e.

- the axes to be controlled.
- the names of the axes connected to the device.

Where Dialog appearing just after software start

> **Tip**
> Make use of the "Restore selection" option if you repeatedly use the axes in the same setting, e.g. a contstant experimental arrangement.

Figure 8 "Restore Selection" dialog

How

- To restore your last axis selection, click [OK] (Figure 8/**1**).
- To discard your last axis selection, click [Cancel] (Figure 8/**2**).

VII.3. "Find Devices" Screen

The "Find Devices" screen is the start screen of the actual MOVE software.

VII.3.a. Overview

The "Find Devices" screen displays the controller devices connected to the respective network. Each controller is displayed in a separate tile.

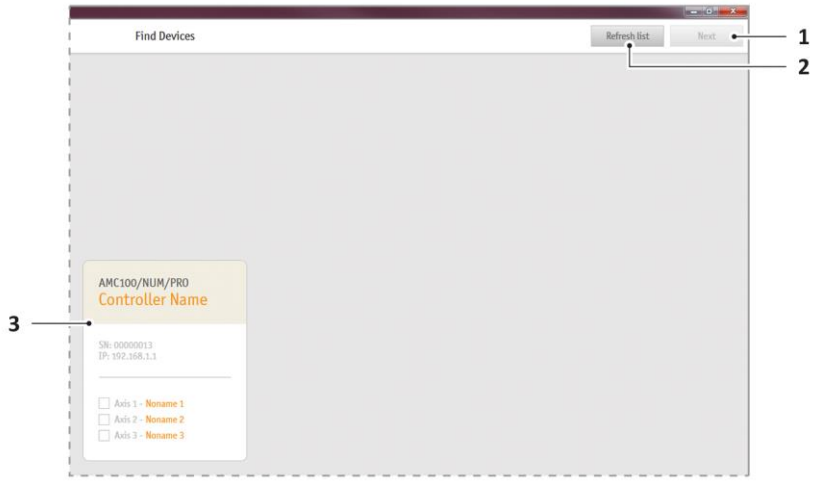


Figure 9 "Find Devices" screen, detail

1 [Next]
2 [Refresh list]
3 Controller tile

- To refresh the list of controllers, click [Refresh list] in the header of the "Find Devices" screen (Figure 9/**2**).

VII.3.b. Controller Tile

The controller tile

- displays several controller and connection properties.
- allows you to label the controller and the axes (see VII.4).
- allows you to select the axes to be controlled (see VII.5).

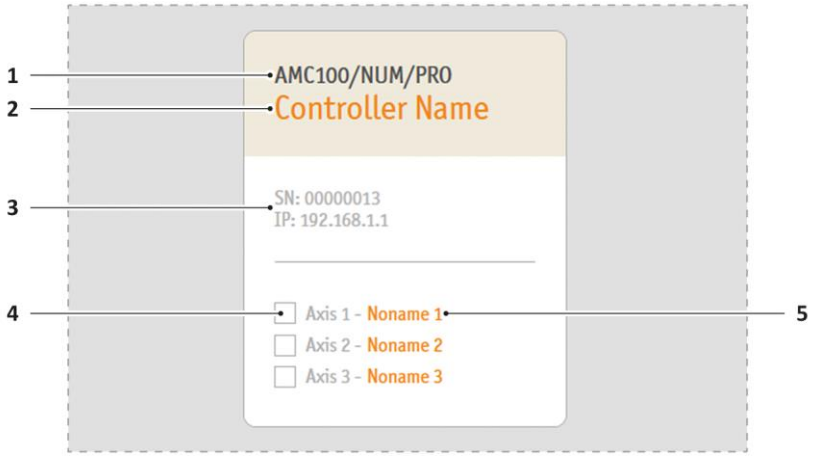


Figure 10 Controller tile

1 Controller type
2 Controller name field
3 Controller's SN and IP address
4 Axis selection checkbox
5 Axis name field

Tip
Figure 10 represents a device with activated /PRO feature. The /PRO feature is available through a separate upgrade. Please contact attocube for further information.

VII.4. Labeling Entities

The "Find Devices" screen allows you to make some comfortable adjustments resulting in an increased transparency, especially when working with the same experimental arrangements over larger periods of time or with varying personnel.

What You can

- assign a name to the controller (see VII.4.a).
- assign names to the axes (see VII.4.b).

Where "Find Devices" screen, controller tile

VII.4.a. Labeling Controller

How

1. To rename the controller, click the controller name field (Figure 10/**2**).
2. Type in the controller name of your choice.

- To save your changes, click [OK].
- To discard your changes, click [Cancel].

VII.4.b. Labeling Axes

> **Tip**
> Make use of the axis labeling option to secure transparency of the axis' assignments (e.g. to the positioning dimension) over time and personnel changes.

How

1. To rename an axis, click the respective axis name field (Figure 10/**5**).
2. Type in the axis name of your choice.

- To save your changes, click [OK].
- To discard your changes, click [Cancel].

VII.5. Selecting Axes

What To configure and control axes you first have to select them.

Where "Find Devices" screen, controller tile

How

- To select an axis, check the respective checkbox (Figure 10/**4**).
  → The axis will be available for configuration and control (see VII.6).

> **Tip**
> To select and use axes from more than one Controller simultaneously the /PRO feature must be activated on all selected Controllers.

VII.6. "Operation" Screen

The "Operation" screen is the screen where the actual positioning process takes place.

What Open the "Operation" screen for positioning.

Where "Find Devices" screen

How

- To open the "Operation" screen, click [Next] (Figure 9/**1**) in the header of the "Find Devices" screen.

> **Tip**
> [Next] is not available unless you have selected at least one axis.

VII.6.a. Overview

The "Operation" screen displays the selected axes each in a separate tile.

> **Tip**
> The headers of the active axes are displayed orange. The headers of the inactive axes are displayed gray.

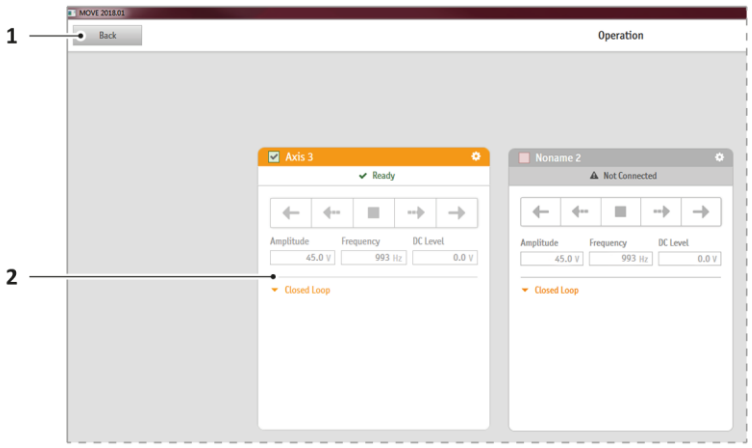


Figure 11 "Operation" screen, detail

1 [Back]
2 Axis tile

- To go back to the "Find Devices" screen click [Back] (Figure 11/**1**) in the header of the "Operation" screen.

VII.6.b. Axis Tile, Operation View

The operation view of the axis tile is the default view on the axis parameters. It

- displays the axis' name and status.
- provides a link to the axis configuration (see VII.7).
- allows you to perform positioning (see VII.8).

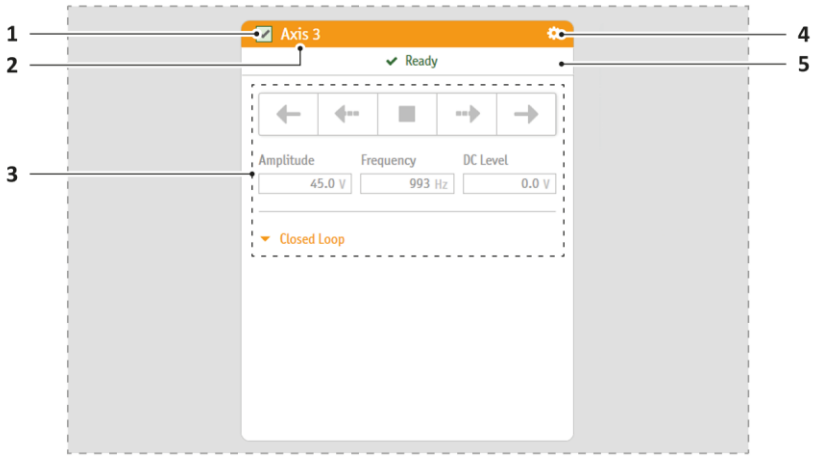


Figure 12 Axis tile, operation view

1 Axis activation checkbox
2 Axis name
3 Axis control field
4 Configuration link
5 Axis status bar

VII.6.c. Axis Tile, Configuration View

- To open the configuration view of the axis tile, click on the configuration link (Figure 12/**4**) in the header of the tile's operation view.

The configuration view of the axis tile

- allows you to specify the positioner type.
- allows you to specify whether the positioner's actuator stops on end of the travel range.
- displays several controller and connection properties.

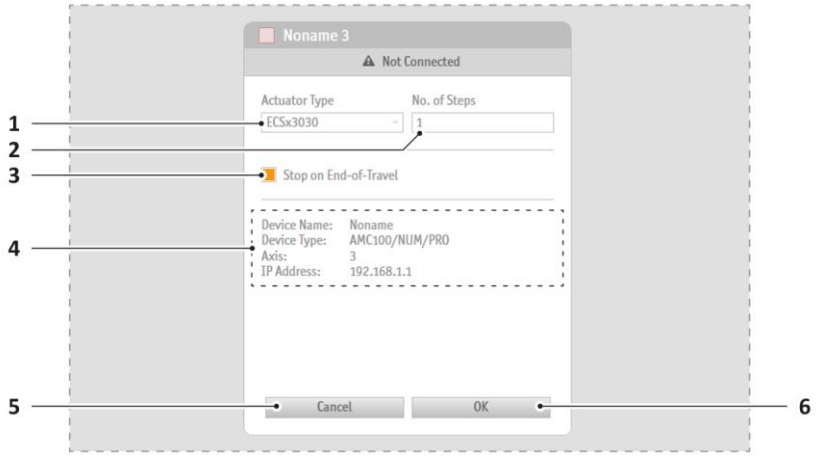


Figure 13 Axis tile, configuration view

1 Actuator type selection field
2 Number of steps setting field
3 Stop on end of travel checkbox
4 Controller properties
5 [Cancel]
6 [OK]

VII.7. Configuring Axis

VII.7.a. Checking Axis Status

The axis status bar (Figure 12/**5**) indicates the current status of the axis. The following states are possible:

Status message	Meaning
Not Connected	No positioner connected
Output Disabled	Positioner connected but axis disabled
Ready	Axis enabled and Positioner waiting for order
Moving	Positioner moving
In Target Range	Positioner is within specified target range around target position (and keeps regulating on the target position)
End of Travel	Positioner reached insurmountable obstacle

VII.7.b. Specifying Positioner Type

What In order to "tell" the device e.g. whether movements are linear or rotational etc. you always have to specify the positioner type after connecting a positioner.

> **Tip**
> Per default an ECSx3030 positioner is selected.

Where "Operation" screen, axis tile's configuration view

How

1. Click on the arrow of the actuator type selection field (Figure 13/**1**).
   → A drop-down list with all positioner types available for the device appears.
2. Click onto the type of positioner connected to the device at the axis' position.

VII.7.c. Setting Number of Steps

What You can set the number of steps that are implemented with each stepwise move.

> **Note**
> The number of steps setting is only available when the /PRO feature is activated.

Where "Operation" screen, axis tile's configuration view

How

- Type in the desired value into the number of steps setting field (Figure 13/ **2**).

VII.7.d. Stopping on End of Travel

What AMC100 allows you to set a deactivation of the positioner's attempt to travel in case of any physical obstacles. Provided the option is activated, the actuator is automatically turned off when facing such obstacles, which normally would be the end of the travel range.

However, there are cases imaginable where the positioner has to overcome physical obstacles as part of the experimental arrangement. For such arrangements the option should be deactivated.

Note
The End of travel detection as well as the “stop on end of travel” Option is only available when the /PRO feature is activated.

Where "Operation" screen, axis tile’s configuration view

Tip
The option "stop on end of travel" by default is activated.

How

- To deactivate the option "stop on end of travel", uncheck the stop on end of travel checkbox (Figure 13/**3**).
- To activate the deactivated option "stop on end of travel", check the stop on end of travel checkbox (Figure 13/**3**).

VII.7.e. Checking the Device Properties

What The configuration view of the axis tile gives some information on the device’s properties (Figure 13/**3**). The following information is displayed:

Parameter	Content
Device Name	Device name as specified (see VII.4.a)
Device Type	Device type and featured functions
AXIS	Device’s axis, the positioner is connected to
IP Address	IP address of the AMC100 as specified (see V.2.d)

VII.7.f. Saving or Discarding Configurations

What To bring your changes to the axis‘ configurations into effect you have to save the changes made.

Where "Operation" screen, axis tile’s configuration view

How

- To save your changes, click [OK] (Figure 13/**6**).
  → The axis tile’s operation view opens.
- To discard your changes, click [Cancel] (Figure 13/**5**).
  → The axis tile’s operation view opens.

VII.8. Positioning

With the AMC100, positioners of the ECx series can be driven in both open-loop and closed-loop mode.

Note
Position values are specified in micrometers or millidegrees, respectively.

VII.8.a. Activating/Deactivating Axis

Caution
Risk of electric shock

Contact with power guiding connectors can cause injuries or material damage.

- Always deactivate an axis before physically disconnecting the corresponding positioner!
- Always deactivate all axes before shutting off the device.

What

You have to manually activate an axis before being able to control the respective positioner with the AMC100.

Where

"Operation" screen, axis tile's operation view

Tip
The axis by default is deactivated.

How

- To activate an axis, check the axis activation checkbox (Figure 12/**1**).
- To deactivate an axis, uncheck the axis activation checkbox (Figure 12/**1**).

VII.8.b. Open-Loop Positioning

The open-loop positioning mode operates without position feedback from the positioner.

What

The position on the axis can be varied without any absolute position information. Motion aspects like speed and step size depend on the physical motion parameters you set for the movement.

Tip
For additional information on the motion-related significance of amplitude and frequency, consult the positioner's manual.

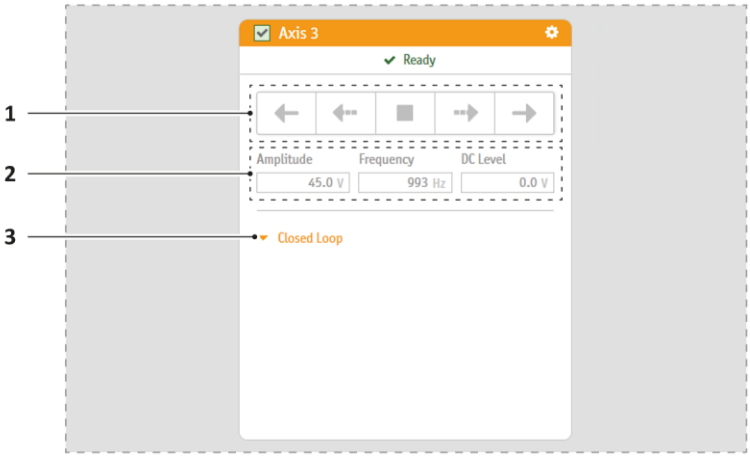


Figure 14 Axis tile, open-loop positioning elements

1 Positioning controls
2 Parameter setting fields
3 [Closed Loop]

Setting Motion Parameters

Where "Operation" screen, axis tile's operation view

How The following fields are available for the setting of the motion parameters (Figure 14/**2**).

Note
The DC level setting is only available when /PRO feature is activated.

Parameter	Value range	Related motion aspect
Amplitude	0 – 45 V	Step size
Frequency	3 – 5000 Hz	Step repetition rate at continuous movement
DC Level	0 – 45 V	Manual fine adjustment of position

Tip
A frequency of 5000 Hz is only available on one axis at one time. Frequencies up to 2000 Hz can be handled on three axes simultaneously.

Note
Typed values have to be confirmed by pressing [Enter]! Unconfirmed values are discarded in favor of the last confirmed value.

- To vary the step size, type in the desired value into the amplitude setting field and press [Enter].
- To vary the step repetition rate, type in the desired value into the frequency setting field and press [Enter].
- To manually adjust the position, type in the desired value into the DC level setting field and press [Enter].

Controlling Movement

Where "Operation" screen, axis tile's operation view

How The following control elements (Figure 14/**1**) are available for movement control.

Control element	Function
	Move continuously in negative direction
	Move stepwise in negative direction
	Stop any movement (including closed-loop positioning movements)
	Move stepwise in positive direction
	Move continuously in positive direction

Tip
The number of steps implemented with any stepwise move can be adapted when /PRO feature is activated (see VII.7.c).

- To initiate or stop a movement along the axis, click the respective control element.

VII.8.c. Closed-Loop Positioning

- To open the closed-loop positioning control elements, click [Closed Loop] (Figure 14/**3**) on the axis tile's operation view.

Note
Closed-loop positioning is only available for positioners with /NUM or /NUM+ encoding.

Refer to the specification sheets of the positioner for information on the motion specifications in closed-loop mode.

The closed-loop positioning mode operates with active position feedback from the positioner.

What Positions on the axis can be analyzed and set with highest precision.

Figure 15 Axis tile, closed-loop positioning elements

1 Position display
2 [Set to zero]
3 [Move in negative direction]
4 Distance setting field
5 [Move in positive direction]
6 [Move to target]
7 Target setting field
8 Reference position display

Position Indication

The position display (Figure 15/**1**) indicates the current position relative to the current zero position.

Tip
The default zero position is the position held at the latest device start up.

Finding Reference Mark

Tip
Encoded positioners provide a physically built in reference mark. Usually this reference mark would be set as zero position by the operator.

The reference position display (Figure 15/**8**) indicates the position of the current zero position relative to the reference mark.

Note
To find the reference mark after a device start up, the positioner has to cross it at least one time. Otherwise the reference position display (Figure 15/**8**) displays "not found".

How

- To find the reference mark, initiate a movement across the entire axis until the reference mark is crossed at least one time.
  → The reference position display (Figure 15/**8**) indicates the position of the current zero position relative to the reference mark.

Setting Zero Position

How

- To set the current position as zero position, click [Set to zero] (Figure 15/**2**).
  → The value in the position display (Figure 15/**1**) changes to "0".
  → The value in the reference position display (Figure 15/**8**) changes to "not found" until the reference mark is crossed the next time.

Moving Defined Distances

Note
Typed values have to be confirmed by pressing [Enter]! Unconfirmed values are discarded in favor of the last confirmed value.

How

1. Into the distance setting field (Figure 15/**4**), type in the desired distance and press [Enter].

- To move in negative direction, click [Move in negative direction] (Figure 15/**3**).
- To move in positive direction, click [Move in positive direction] (Figure 15/**5**).
  → The positioner moves along the defined distance in the defined direction.

Moving to Defined Target

Note
Typed values have to be confirmed by pressing [Enter]! Unconfirmed values are discarded in favor of the last confirmed value.

How

1. Into the target setting field (Figure 15/**7**), type in the desired target coordinates and press [Enter].
2. Click [Move to target] (Figure 15/**6**).
   → The positioner moves to the defined target.

VIII. Operation with Webserver Application

Tip
For background information on the activities of this section consult the corresponding part of the VII.

VIII.1. Starting Webserver Application

How

1. On your PC, open a web browser.
2. Type in the device's IP address into the address line.
   - If the IP address was adapted yet (see V.2.d), use the customized address.
   - If the IP address was not adapted yet, use `192.168.1.1` as IP address.
3. Press [Enter].
   → The webserver application opens.

Note
For each controller a separate browser window must be used.

VIII.2. "Navigation" Screen

The "Navigation" screen is the screen where the actual positioning process takes place. It displays the axes each in a separate tile.

Note
For there's no separate axis selection on the webserver application, the "Navigation" screen always displays three axes.

On the webserver application no entities can be labeled.

What Open the "Navigation" screen for positioning.

Where Webserver application, any screen

How

- To open the "Navigation" screen, click [Navigation] in the header of the webserver application.

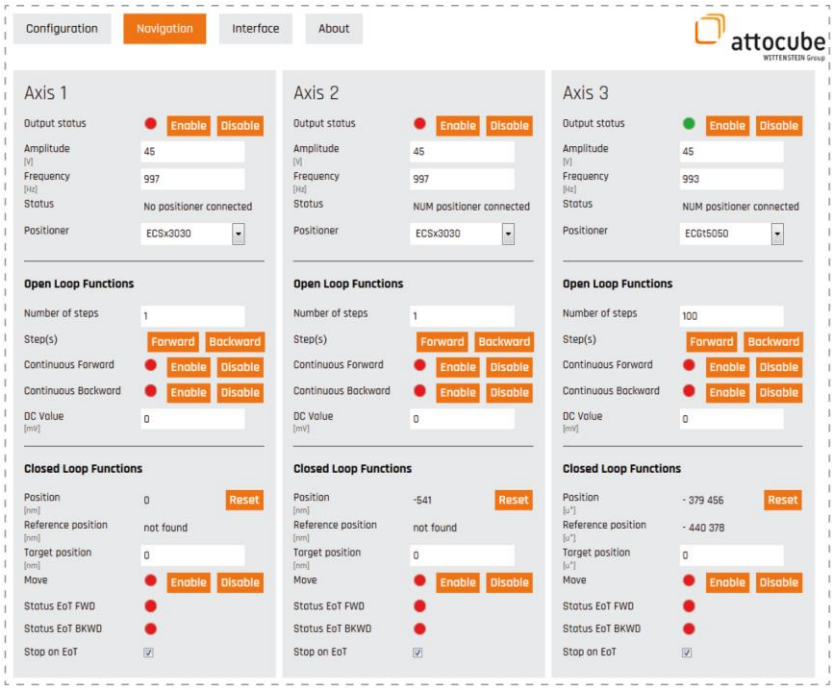


Figure 16 "Navigation" screen

VIII.3. Configuring Axis

What You have to set some general axis parameters prior to the positioning.

Where "Navigation" screen, axis tile's "General" and "Closed Loop Functions" sections

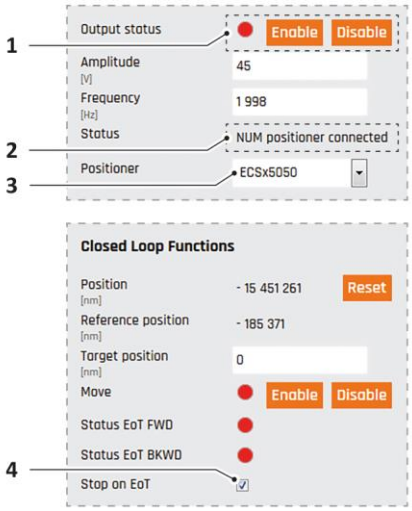


Figure 17 Axis tile, "General" and "Closed Loop Functions" sections

1 Output status control
2 Status display
3 Positioner selection field
4 Stop on EoT checkbox

VIII.3.a. Checking Axis Status

The status display (Figure 17/**2**) indicates the current status of the axis. The following states are possible:

Status message	Meaning
No positioner connected	No positioner connected
NUM positioner connected	Positioner for open-loop and closed-loop positioning connected
OL positioner connected	Positioner for open-loop positioning connected

VIII.3.b. Specifying Positioner Type

How

1. Click on the arrow of the positioner selection field (Figure 17/**3**).
   → A drop-down list with all positioner types available for the device appears.
2. Click onto the type of positioner connected to the device at the axis' position.

VIII.3.c. Stopping on End of Travel

Tip
The End of travel detection as well as the "stop on end of travel" Option is only available when the /PRO feature is activated. If activated, the default value for the "stop on end of travel" function is true.

How

- To deactivate the option "stop on end of travel", uncheck the stop on EoT checkbox (Figure 17/**4**).
- To activate the deactivated option "stop on end of travel", check the stop on EoT checkbox (Figure 17/**4**).

VIII.4. Positioning

Note
Position values are specified in micrometers or millidegrees, respectively.

VIII.4.a. Activating/Deactivating Axis

Caution
Risk of electric shock

Contact with power guiding connectors can cause injuries or material damage.

- Always deactivate an axis before physically disconnecting the corresponding positioner!
- Always deactivate all axes before shutting off the device.

What You have to manually activate an axis before being able to control the respective positioner with the AMC100.

Where "Navigation" screen, axis tile's "General" section

Tip
The axis by default is deactivated.

How

- To activate an axis, click [Enable] on the output status control (Figure 17/**1**).
  → The corresponding indicator light changes to green.
- To deactivate an axis, click [Disable] on the output status control (Figure 17/**1**).
  → The corresponding indicator light changes to red.

VIII.4.b. Open-Loop Positioning

What The position on the axis can be varied without any absolute position information. Motion aspects like speed and step size depend on the physical motion parameters you set for the movement.

Setting Motion Parameters

Where "Navigation" screen, axis tile's "General" section

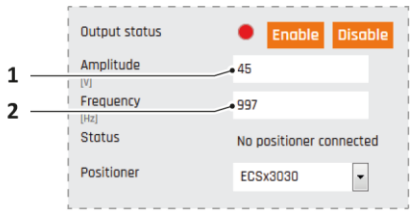


Figure 18 Axis tile, "General" section

1 Amplitude setting field
2 Frequency setting field

How The following fields are available for the setting of the motion parameters.

Parameter	Value range	Related motion aspect
Amplitude	0 – 45 V	Step size
Frequency	3 – 5000 Hz	Step repetition rate at continuous movement

Tip
A frequency of 5000 Hz is only available on one axis at one time. Frequencies up to 2000 Hz can be handled on three axes simultaneously.

- To vary the step size, type in the desired value into the amplitude setting field (Figure 18/**1**) and press [Enter].
- To vary the repetition rate, type in the desired value into the frequency setting field (Figure 18/**2**) and press [Enter].

Controlling Movement

Where "Navigation" screen, axis tile's "Open Loop Functions" section

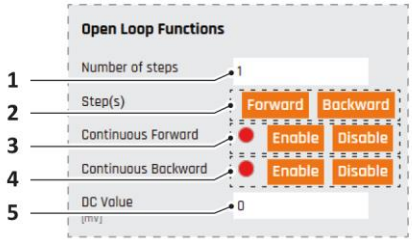


Figure 19 Axis tile, "Open Loop Functions" section

1 Number of steps setting field
2 Stepwise movement control
3 Continuous forward control
4 Continuous backward control
5 DC value setting field

Note
The number of steps setting is only available when /PRO feature is activated.

How

- To vary the number of steps implemented with each stepwise move, type in the desired value into the number of steps setting field (Figure 19/**1**) and press [Enter].
- To initiate a stepwise movement in positive direction, click [Forward] on the stepwise movement control (Figure 19/**2**).
- To initiate a stepwise movement in negative direction, click [Backward] on the stepwise movement control (Figure 19/**2**).
- To initiate a continuous movement in positive direction, click [Enable] on the continuous forward control (Figure 19/**3**).
  → The corresponding indicator light changes to green.
- To stop a continuous movement in positive direction, click [Disable] on the continuous forward control (Figure 19/**3**).
  → The corresponding indicator light changes to red.
- To initiate a continuous movement in negative direction, click [Enable] on the continuous backward control (Figure 19/**4**).
  → The corresponding indicator light changes to green.

- To stop a continuous movement in negative direction, click [Disable] on the continuous backward control (Figure 19/**4**).
  → The corresponding indicator light changes to red.

Note
The DC level setting is only available when /PRO feature is activated.

- To manually adjust the position, type in the desired value into the DC level setting field and press [Enter] (Figure 19/**5**).

VIII.4.c. Closed-Loop Positioning

What Positions on the axis can be analyzed and set with highest precision.

Note
Closed-loop positioning is only available for positioners with /NUM or /NUM+ encoding.

Refer to the specification sheets of the positioner for information on the motion specifications in closed-loop mode.

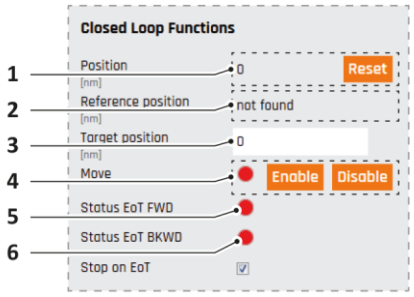


Figure 20 Axis tile, "Closed Loop Functions" section

1 Position display and [Reset]
2 Reference position display
3 Target position setting field
4 Movement control
5 Status EoT FWD indicator
6 Status EoT BKWD indicator

Position Indication

The position display (Figure 20/**1**) indicates the current position relative to the current zero position.

Tip
The default zero position is the position held at the latest device start up.

Finding Reference Mark

The reference position display (Figure 20/**2**) indicates the position of the current zero position relative to the reference mark.

Note
To find the reference mark after a device start up, the positioner has to cross it at least one time. Otherwise the reference position display (Figure 20/**2**) displays "not found".

How

- To find the reference mark, initiate a movement across the entire axis until the reference mark is crossed at least one time.
  → The reference position display (Figure 20/**2**) indicates the position of the current zero position relative to the reference mark.

Setting Zero Position

How

- To set the current position as zero position, click [Reset] (Figure 20/**1**).
  → The value in the position display (Figure 20/**1**) changes to "0".
  → The value in the reference position display (Figure 20/**2**) changes to "not found" until the reference mark is crossed the next time.

Moving to Defined Target

How

1. Into the target position setting field (Figure 20/**3**), type in the desired target coordinates and press [Enter].
2. On the movement control (Figure 20/**4**), click [Enable].
   → The positioner moves to the defined target.

- To stop the movement before the positioner reaches the target, click [Disable] on the movement control (Figure 20/**4**).
  → The positioner stops moving.

Checking Movement Status

The axis tile's "Closed Loop Functions" section holds different elements indicating the movement status of the positioner. The following states are indicated:

Control element	Color	Related movement status
Movement control (Figure 20/**4**)	red	Positioner not moving
	green	Positioner moving
Status EoT FWD indicator (Figure 20/**5**)	red	No obstacle in positive direction detected
	green	Positioner reached insurmountable obstacle in positive direction
Status EoT BKWD indicator (Figure 20/**6**)	red	No obstacle in negative direction detected
	green	Positioner reached insurmountable obstacle in negative direction

IX. Device Shut Off

Tip
To allow an independent functioning of the device (e.g. for automated experimental arrangements), the AMC100 works properly even when the software is closed.

Caution
Risk of electric shock

Contact with power guiding connectors can cause injuries or material damage.

- Always deactivate all axes before shutting off the device.

What

The shutting off procedure implies deactivating all axes, closing the software and switching off the device.

How

1. To deactivate the axes, follow the instructions in VII.8.a (for the MOVE software) or VIII.4.a (for the webserver application), respectively.
2. Close the software application.
   - To close the MOVE software, click on [X] in the upper right corner of the software window.
   - To close the webserver application, click on [X] in the upper right corner of the browser window.
3. To switch off the device, push the power button (Figure 3/**1**).

X. Update, Upgrade and Maintenance

Note
There are no user serviceable parts inside the controller!

All update, upgrade and maintenance procedures are carried out in the webserver application.

X.1. Starting Webserver Application

- To start the webserver application, follow the instructions in VIII.1.

X.2. "About" Screen

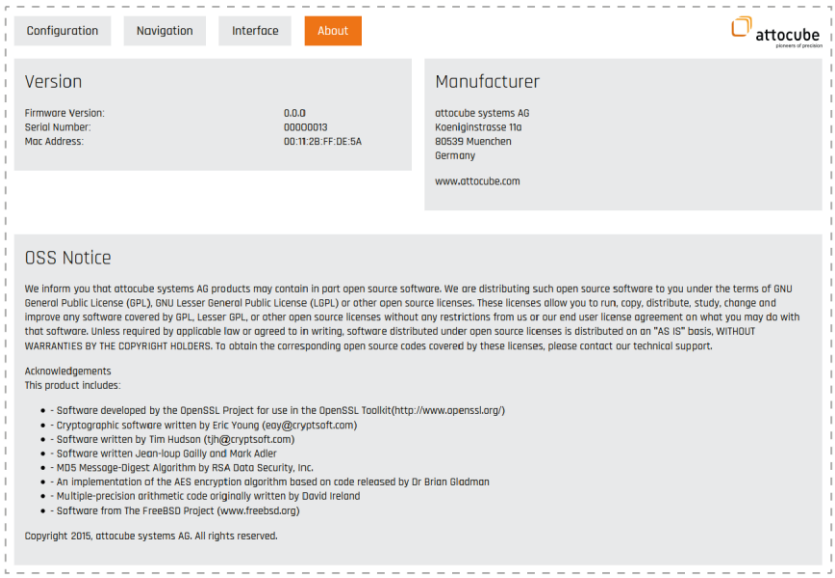
The "About" screen provides you with comprehensive information on

- the device itself and the used software.
- the manufacturer.
- sources of integrated software parts.
- activated licenses.

Note
As the information of the "About" screen may be relevant for servicing, updating, upgrading and maintaining the device or its parts, prepare to retrieve the information and provide it to attocube in case of questions and servicing requests.

- To open the "About" screen, click [About] in the header of the webserver application.

Figure 21 "About" screen, detail



X.3. "Configuration" Screen

On the "Configuration" screen of the webserver application you can

- adapt the network settings.
- update your firmware.
- upgrade your software by activating features.

○ To open the "Configuration" screen, click [Configuration] in the header of the webserver application.

Figure 22 "Configuration" screen

X.3.a. Adapting Network Connection

What If you're not using the device as DHCP server, you can manually adapt the network connection settings.

Where "Configuration" screen, "Networking" section

> **Tip**
> The device by default runs a DHCP server, so the DHCP server checkbox is checked. The default subnet mask is `255.255.255.0`. The preset IP address is `192.168.1.1`.

How

1. Uncheck the DHCP server checkbox.
   → Additional fields are displayed.

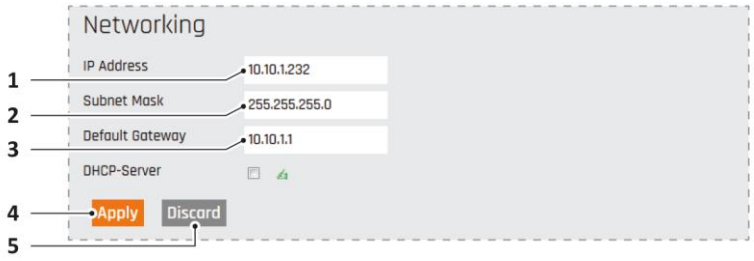


Figure 23 "Configuration" screen, "Networking" section

1 IP address setting field
2 Subnet mask setting field
3 Default gateway setting field
4 [Apply]
5 [Discard]

2. To adapt the network connection, type in the desired information into the corresponding setting fields (Figure 23/**1**, **2**, **3**) according to your network's configuration.

> **Note**
> Make sure the chosen IP is available for your network and is not used already.

- To save your changes, click [Apply] (Figure 23/**4**).
- To discard your changes, click [Discard] (Figure 23/**5**).

X.3.b. Updating Firmware

attocube provides occasional updates in the form of image files. The files are delivered in a zip file. attocube informs you when an update for your device is available.

Note
Unauthorized updates can lead to a permanent malfunction and are not covered by attocube's warranty.

- Always contact attocube for technical support, before updating the firmware of the device.

What The update has to be prepared and initiated manually.

Where Windows Explorer on your PC and "Configuration" screen, "Firmware Update" section

Figure 24 "Configuration" screen, "Firmware Update" section

How

1. Unpack the delivered zip file to the directory of your choice on your hard drive.
2. In the "Firmware Update" section of the "Configuration" screen, click the browse button.
   → An explorer window opens.
3. Navigate to the update file's folder.
4. Select the file ending on ".image" and click [Open].
   → The update file is uploaded automatically.
   → After the successful upload additional buttons are displayed.

- To complete the updating process, click [Install & Reboot].
  → The update is installed and the device is rebooted.

Note
It is essential to the update process that the device is not disconnected during the reboot!

- Do not disconnect the device until a green bar at the top of the web application states the following: `Connection to server restored. You can continue working now!`

- To discard the updating process, click [Discard].

X.3.c. Feature Activation

To increase your performance using the AMC100, attocube provides you with the possibility to activate additional features. License files are delivered for this purpose.

Tip
Contact attocube for available upgrades.

What The upgrade has to be prepared and initiated manually.

Where Windows Explorer on your PC and "Configuration" screen, "Feature Upgrade" section

Figure 25 "Configuration" screen, "Feature Upgrade" section

How

1. Copy the delivered license file to the directory of your choice on your hard drive.
2. In the "Feature Upgrade" section of the "Configuration" screen, click the browse button.
   → An explorer window opens.
3. Navigate to the license file's folder.
4. Select the file ending on ".gpg" and click [Open].
   → The license file is uploaded automatically.
   → After the successful upload additional buttons are displayed.

○ To complete the upgrading process, click [Update & Reboot].
→ The corresponding feature is activated and the device is rebooted.

Note
It is essential to the upgrade process that the device is not disconnected during the reboot!

- Do not disconnect the device until a green bar at the top of the web application states the following: `Connection to server restored. You can continue working now!`

○ To discard the upgrading process, click [Discard].

XI. Troubleshooting

Note
Unauthorized error handling may result in permanent malfunction and is not covered by attocube's warranty.

- Do not take any action not proposed for troubleshooting in this document.
- If problems occur that are not mentioned in this section, contact attocube for help.
- If the problems cannot be solved by the proposed action, contact attocube for help.

Connection Failed

What If no connection between the device and the PC is established, take the following steps until the problem is solved:

How

1. Be sure at least a minute has passed since the device's start up, so the device had enough time to boot. If the problem is not solved, continue with step 2.

Tip
You can tell if the device boots from a short noise produced by its fan, approximately ten seconds after start up.

2. Check if the indicator light next to the Ethernet socket is blinking.
   - The device is connected to the PC via USB-to-Ethernet adapter and the indicator light is not blinking: continue with step 3.
   - Else: continue with step 4.
3. Disconnect the USB-to-Ethernet adapter from your PC and connect it again. If the problem is not solved, continue with step 4.
4. Reboot the device by switching it off and on again at the power button. If the problem is not solved, contact attocube for help.

IP Address Lost

What If you lost the IP address of the device so you cannot access to it via the webserver application, take the following steps until the problem is solved:

How

1. Open the MOVE software and go to the "Find Devices" screen (see VII.3).
   → The IP address of the device is displayed in the respective controller tile (Figure 10/**3**).

No Access via Webserver Application

What If you cannot access to the device using the webserver application with the right IP address, take the following steps until the problem is solved:

How

1. Try opening the webserver application with the latest version of the Microsoft Internet Explorer, Google Chrome or Mozilla Firefox. If the problem is not solved, continue with step 2.
2. Erase the cookies and history of your web browser. If the problem is not solved, contact attocube for help.

Positioner not found

What If you are using older Positioners out of attocubes industrial line, the AMC100 might not be able to detect the Connection between Positioner and AMC and you will not be able to activate the Output on the respective axis. In this case you will get the Status “No positioner connected” in the Webserver and “Not Connected” in the MOVE Software, even if a Positioner is physically connected.

How Please contact the attocube Support Team under support@attocube.com.

Software Crash

To enable an effective error tracking, the MOVE software creates log files. If your software has crashed, please provide attocube with the log file, so latent errors on your specific device can be corrected and the software can be improved with the next version.

By default you can find the latest log file (named "MOVE_*DateAndTime*") in the documents folder of your user account on your hard drive.

XII. System Integration

AMC100 can be integrated with external systems or devices by

- combining it with third party hardware (see XII.1).
- establishing incoming and outgoing trigger connections (see XII.2).
- controlling it with individual software interfaces (see XII.3).

XII.1. Connecting to Third Party Hardware

Caution
General hazard!

Inadequate hardware connections may cause injury and are likely to damage the device or interfere with an appropriate functioning.

- Always contact attocube for technical support, before combining the device with third party hardware.
- Do carefully observe the information in this section when combining the device with third party hardware.

Note
attocube is not liable for any damages resulting from an unauthorized combination of the device with third party hardware. Unauthorized combination with third party hardware is not covered by attocube's warranty.

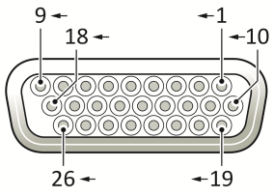
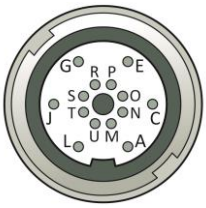
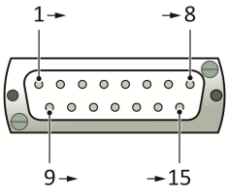
XII.1.a. Cabling Restrictions

For optimal performance, obey the following combination restrictions:

- Do not to connect cabling with a wire resistance > 5 Ω.
- Use EMV housings as enclosure for the D-sub connectors.
- Use extra shielded twisted pair wires for the piezo voltage supply.
- Do not connect any cable > 5 m.

XII.1.b. Pin Assignments

The pins of the device's positioner control cables are assigned as follows:

26 pin D-sub socket (female)	Sensor I/O	Piezo voltage	Positioner control cable (/NUM)	
			14 pin circular socket (female), Room temperature	15 pin D-sub connector (male), Vacuum
				
1, 2		+	T	11
4, 5, 6		-	U	10
10	UB = 5V		S	12
11, 12, 13, 14, 15, 16, 17	GND		E	6
18	Pos-Con		G	5
19	U2-		P	14
20	U2+		R	13
21	U1-		J	4
22	U1+		L	3
23	U0-		O	15
24	U0+		N	1
26	1-wire IO		M	2, 7, 8 or 9

XII.2. Triggering

AMC100 allows you to

- trigger positioning movements by input signals.
- send the axis' positioning information to an external interpreter.

Tip
Trigger Functionalities are activated through the /IO feature which is available as a separate upgrade. Please contact attocube for further information.

XII.2.a. Pin Assignments

AMC100 uses the GPIO socket at the device's connector panel (Figure 2/**1**) for the communication of trigger signals. Each position signal consists of two parts transmitted via different pins. The semantics of the signal depends on the used communication protocol. A third pin communicates error signals.

The pins of the 37 pin D-sub socket are assigned as follows.

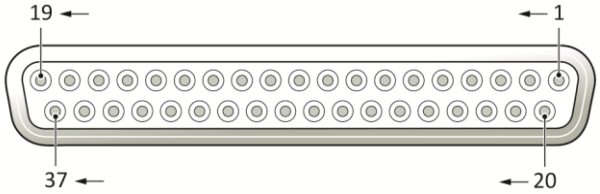


Figure 26 GPIO socket, pin assignment

	Pin	Protocol function		
		AquadB	Trigger	Stepper
Axis 1				
Input	1	A+	UP+	Step-Pulse+
	20	A-	UP-	Step-Pulse-
	2	B+	DOWN+	Direction+
	21	B-	DOWN-	Direction-
	3	Error+	Error+	Error+
	22	GND	GND	GND
Output	4	A+	n/a	
	23	A-		
	5	B+		
	24	B-		
	6	Error+		
	25	GND		

	Pin	Protocol function		
		AquadB	Trigger	Stepper
Axis 2				
Input	7	A+	UP+	Step-Pulse+
	26	A-	UP-	Step-Pulse-
	8	B+	DOWN+	Direction+
	27	B-	DOWN-	Direction-
	9	Error+	Error+	Error+
	28	GND	GND	GND
Output	10	A+	n/a	
	29	A-		
	11	B+		
	30	B-		
	12	Error+		
	31	GND		
Axis 3				
Input	13	A+	UP+	Step-Pulse+
	32	A-	UP-	Step-Pulse-
	14	B+	DOWN+	Direction+
	33	B-	DOWN-	Direction-
	15	Error+	Error+	Error+
	34	GND	GND	GND
Output	16	A+	n/a	
	35	A-		
	17	B+		
	36	B-		
	18	Error+		
	37	GND		

XII.2.b. Trigger Output Modes and Parameters

For outgoing trigger communication, the following communication protocols are supported.

AquadB

According to the logic of AquadB, the signals on lines A and B designate the increment of the position change. The direction of the position change is defined by whether signal A (positive) or B (negative) is leading the signal. AquadB is available in LVTTL and LVDS mode.

The following parameters have to be specified.

Parameter	Value range	Related communication aspect
Clock	40 – 1,280 ns (in steps of 40 ns)	Minimal signal emitting period
Resolution	0.001 – 64.93 nm	Position change resolution

XII.2.c. Trigger Input Modes and Parameters

For incoming trigger communication, the following communication protocols are supported.

AquadB

According to the logic of AquadB, the signals on lines A and B designate the increment of the position change. The direction of the position change is defined by whether signal A (positive) or B (negative) is leading the signal. AquadB is available in LVTTL and LVDS mode. It can only be applied in the closed-loop mode.

The following parameters have to be specified.

Parameter	Value range	Related communication aspect
Loop mode	"open-loop"/ "closed-loop"	Motion valued in metric distance or motion steps
Change/steps per pulse	0 – n nm/ 0 – n steps	Position change resolution

Stepper

According to the logic of Stepper, the signal on line A designates the increment of the position change, while the signal on line B designates the direction of the position change. Stepper is available in LVTTL and LVDS mode and can be applied in open-loop and closed-loop mode.

The following parameters have to be specified.

Parameter	Value range	Related communication aspect
Loop mode	"open-loop"/ "closed-loop"	Motion valued in metric distance or motion steps
Change/steps per pulse	0 – n nm/ 0 – n steps	Position change resolution

Trigger

According to the logic of Trigger, the signal on line A designates the increment of the position change in positive direction, while the signal on line B designates the increment in negative direction. Trigger is available in LVTTL and LVDS mode and can be applied in open-loop and closed-loop mode.

The following parameters have to be specified.

Parameter	Value range	Related communication aspect
Loop mode	"open-loop"/ "closed-loop"	Motion valued in metric distance or motion steps
Change/steps per pulse	0 – n nm/ 0 – n steps	Position change resolution

XII.2.d. "Interface" Screen

On the "Interface" screen the settings for incoming and outgoing communication can be configured each in a separate tile. The tiles are divided in three sections corresponding to the axes.

What Open the "Interface" screen to configure the settings for the axis' incoming and outgoing communication.

Where Webserver application, any screen

How

- To open the "Interface" screen, click [Interface] in the header of the webserver application.

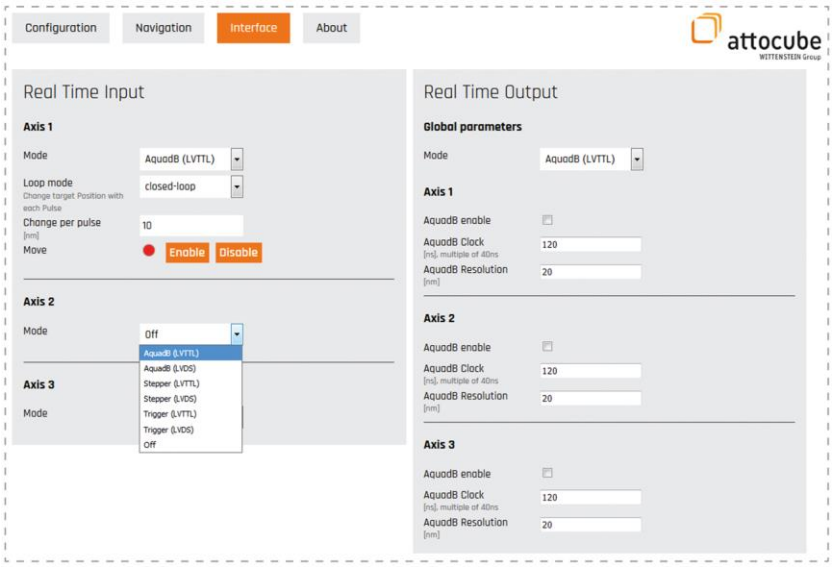


Figure 27 "Interface" screen

XII.2.e. Configuring Real Time Output

What To enable the communication between AMC100 and the external interpreter, you have to choose a communication protocol and set the according output parameters.

> **Tip**
> Regarding the parameters and value ranges related to the specific communication protocol, consult XII.2.b.

Setting Global Output Parameters

Where "Interface" screen, output tile's "Global parameters" section

Figure 28 Output tile, "Global parameters" section

How

1. Click on the arrow of the signal mode selection field.
   → A drop-down list with the available signal modes appears.
2. Click onto the signal mode to be applied.

Setting Axis Output Parameters

Where "Interface" screen, output tile's "Axis" section

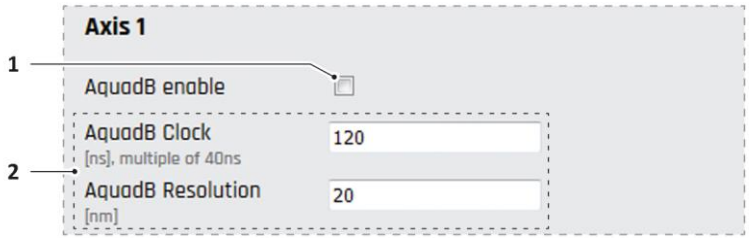


Figure 29 Output tile, "Axis" section

1 Output enable checkbox
2 Output parameter setting fields

How

- To enable the communication of positioning information, check the output enable checkbox (Figure 29/**1**).
- To set the communication parameters, type in the desired values into the output parameter setting fields (Figure 29/**2**).

Saving or Discarding Configurations

Where "Interface" screen, output tile

How

- To save your changes, click [Apply].
- To discard your changes, click [Discard].

XII.2.f. Configuring Real Time Input

What To allow the AMC100 the interpretation and implementation of incoming positioning commands, you first have to set the according communication parameters.

> **Tip**
> Regarding the parameters and value ranges related to the specific communication protocol, consult XII.2.c.

Where "Interface" screen, input tile's "Axis" section

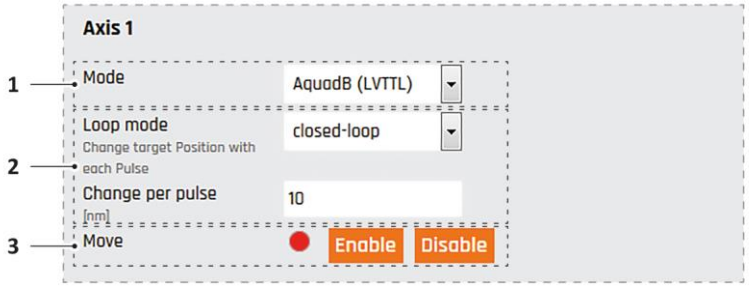


Figure 30 Input tile, "Axis" section

1 Signal mode selection field
2 Input parameter setting fields
3 Input activation control

> **Note**
> To allow you a comfortable adjustment of the parameters, the real time input is not applied until it is activated explicitly.

Setting the Axis Signal Mode

How

1. Click on the arrow of the signal mode selection field (Figure 30/**1**).
   → A drop-down list with the available signal modes appears.
2. Click onto the signal mode to be applied.

Setting the Axis Input Parameters

How

- To set the communication parameters, type in the desired values into the input parameter setting fields (Figure 30/**2**).

Disabling Real Time Input

How

1. Click on the arrow of the signal mode selection field (Figure 30/**1**).
   → A drop-down list with the available signal modes appears.
2. Click onto "Off".
   → The real time input is disabled for this axis.

Activating/Deactivating Real Time Input

How

- To activate real time input, click [Enable] on the input activation control (Figure 30/**3**).
  → The corresponding indicator light changes to green.
- To deactivate real time input, click [Disable] on the input activation control (Figure 30/**3**).
  → The corresponding indicator light changes to red.

Saving or Discarding Configurations

Where "Interface" screen, input tile

How

- To save your changes, click [Apply].
- To discard your changes, click [Discard].

XII.3. Software Interfaces

You can integrate the AMC100 into complex automated processes via individual software interfaces. For that purpose, a dynamic link library (DLL) is contained on the USB flash drive included in the scope of delivery.

The DLL can be used in various programming languages like C, LabVIEW or Matlab. For LabVIEW a dedicated set of VIs including an example VI is provided.

The following sections provide information on methods, commands and parameters to be used for calling up the DLL functions with the respective language.

XII.3.a. Calling Up Functions via JSON-RPC

The AMC100 allows platform-independent communication using JSON-RPC via TCP/IP. When using JSON-RPC, the following conventions apply.

Note
Part of the conventions mentioned below are specific for the handling of attocube devices and are not necessarily applicable in other contexts.

Communication Structure

The client sends a request message to the AMC100's IP address. This message contains basic information as the method name, the necessary parameters and a message ID.

The AMC100 always answers to a request with a response message to the client's IP address. This message contains the requested information, an error code (by default "0") and the message ID of the corresponding request message.

Markup Conventions

Messages are composed of information parts belonging to different categories (protocol, method, parameter, message ID). Each of these parts is itself composed of the category name and one or more values.

- Entire messages are framed by curly braces.
- Different information parts are separated by a comma.
- The name and the values of an information category are separated by a colon.
- Message parts framed by superscript double quotes are strings. Message parts not framed by superscript double quotes are numbers or Boolean values.
- The name of the information category "parameters" is shortened as "params".
- The values of the category parameters are framed by square brackets.
- Values of different parameters are separated by a comma.

Request Message Structure

A request message is structured according to the following example:

Note
The order of the request message parts is mandatory.

{"jsonrpc": "2.0", "method": *method name*, "params": [*parameter 1, parameter 2, … , parameter n*], "id": *call id*}

Category	Name	Values
Protocol version	"jsonrpc"	"2.0"
Method	"method"	String as defined in XII.3.d
Parameters	"params"	Values as defined in XII.3.d
Message ID	"id"	Unique number

Response Message Structure

A response message is structured according to the following example.

`{"jsonrpc": "2.0", "result": [`*return value 1*`, …, `*return value n*`], "id": `*call id*`}`

Category	Name	Values
Protocol version	"jsonrpc"	"2.0"
Return values	"result"	Error codes and return values as defined in XII.3.d
Message ID	"id"	Unique number of the corresponding request message

The error code is the first of the result values. By default it is "0" (no error).

Example

The following example shows how to generate the JSON-RPC method call to set the amplitude of axis 1 to 45 V. XII.3.d delivers the input values for method and params that are necessary to create the call. It also gives information on the values contained in the corresponding response message.

JSON Method(s) and Parameters		
Set	method	`com.attocube.amc.control.setControlAmplitude`
	params	axis number, amplitude in mV
	result	error number

Note
If you use JSON-RCP for method calling, you have to provide the message ID for every request.

The resulting request message looks as follows:

```
{"jsonrpc": "2.0", "method":
"com.attocube.amc.control.setControlAmplitude", "params":
[0,45000], "id": 1}
```

The resulting response message of the successful operation looks as follows:

```
{"jsonrpc":"2.0","result":[0],"id":1}
```

XII.3.b. Calling Up Functions via C-DLL

attocube provides a C-DLL that can be used to handle all TCP/IP communication by means of JSON-RPC.

The following message structure demonstrates how to generate a call to set the amplitude of an axis. XII.3.d delivers the input values that are necessary to create the call.

C-DLL
`AMC_controlAmplitude(`*Int32 deviceHandle,Int32 axis,Int32* amplitude,Bln32 set*`)`

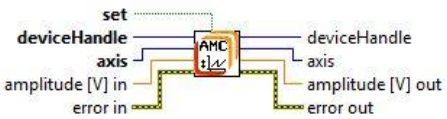
Note
The value for "Bln32 set" must be set to "true" to call up a setting function.

XII.3.c. Calling Up Functions via LabVIEW

For LabVIEW a complete set of SubVIs is provided. The SubVIs are using the native LabVIEW TCP read and write VIs so there are no dependencies to external DLLs anymore. However, “Legacys VIs” are also provided, calling the C-DLL from within LabVIEW in case still needed.

Tip
The provided LabVIEW Project includes an example VI mimicking the Web Apps behavior as a reference on how to use the SubVIs.

The following message structure demonstrates how to generate a call to set the amplitude of an axis. XII.3.d delivers the input values that are necessary to create the call.

LabVIEW		
`AMC_controlAmplitude.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	amplitude	amplitude in V
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	amplitude	set amplitude in V
	error	error message

XII.3.d. AMC100 Functions

The following functions are available for controlling the AMC100.

Note
In the following line-up, necessary input parameters are marked by a superposed asterisk.

Note
When creating commands, the axes have to be numerated from "0" (axis 1) to "2" (axis 3).

Connect

This function initializes and connects a selected device.

LabVIEW		
AMC_Connect.vi		
		
In	deviceAddress	IP of the connected Device
	error	error message
Out	deviceHandle*	device handle
	error	error message
C-DLL call		
AMC_Connect(*const char *deviceAddress, Int32* deviceHandle*)		

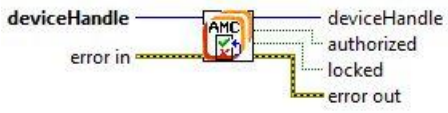
Close

This function closes the connection to a device.

LabVIEW		
AMC_Close.vi		
		
In	deviceHandle*	device handle
	error	error message
Out	error	error message
C-DLL call		
AMC_Close(*Int32 deviceHandle*)		

getLockStatus

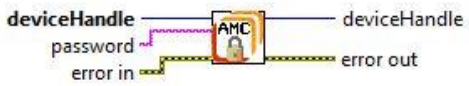
This function gets information whether the device is locked and if access is authorized.

LabVIEW		
`AMC_getLockStatus.vi`		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	authorized	indicates if access is granted
	locked	indicates if the device is locked
	error	error message
C-DLL call		
`AMC_getLockStatus` (*Int32 deviceHandle, Bln32* locked, Bln32* authorized*)		
JSON method(s) and parameters		
get	method	`getLockStatus`
	params	
	result	lock status, authorization status

lock

This function locks the device, so the calling of functions is only possible with valid password.

Note
If the device is locked and the access is not authorized, you need to execute the grantAccess function and enter the correct password prior to running any other VI with the established device handle connection.

LabVIEW		
AMC_lock.vi		
		
In	deviceHandle*	device handle
	password	password for locking the Device
	error	error message
Out	deviceHandle*	device handle
	error	error message
C-DLL call		
AMC_lock(*Int32 deviceHandle, char* password*)		
JSON method(s) and parameters		
set	method	lock
	params	password
	result	error number

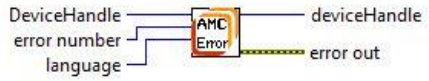
grantAccess

This function requests access to a locked device, so all functions can be called after entering the correct password. Otherwise, each function creates an error.

LabVIEW		
AMC_grantAccess.vi		
		
In	deviceHandle*	device handle
	password	password for accessing the device
	error	error message
Out	deviceHandle*	device handle
	error	error message
C-DLL call		
AMC_grantAccess(*Int32 deviceHandle, char* password*)		
JSON method(s) and parameters		
set	method	grantAccess
	params	password
	result	error number

errorNumberToString

This function “translates” the error code into an error text and adds it to the error out cluster.

LabVIEW		
AMC_errorHandler.vi		
		
In	deviceHandle*	device handle
	error number	error code to translate
	language	Language of error massage
Out	deviceHandle*	device handle
	error	error message
C-DLL call		
AMC_errorNumberToString(*Int32 deviceHandle, int lang, int errcode, char* error*)		
JSON method(s) and parameters		
set	method	com.attocube.system.errorNumberToString
	params	language, error number
	result	error massage

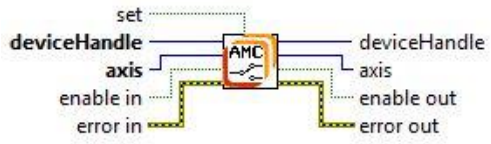
unlock

This function unlocks the device, so it will not be necessary to execute the grantAccess function to run any VI.

LabVIEW		
AMC_unlock.vi		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	error	error message
C-DLL call		
AMC_unlock(*Int32 deviceHandle*)		
JSON method(s) and parameters		
set	method	unlock
	params	
	result	error number

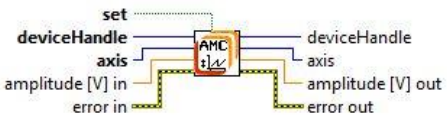
controlOutput

This function sets or gets the status of the output relays of the selected axis.

LabVIEW		
AMC_controlOutput.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	enable	true: enable output false: disable output
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	enable	true: output enabled false: output disabled
	error	error message
C-DLL call		
AMC_controlOutput(*Int32 deviceHandle,Int32 axis,Bln32* enable,Bln32 set*)		
JSON method(s) and parameters		
set	method	com.attocube.amc.control.setControlOutput
	params	axis number,output
	result	error number
get	method	com.attocube.amc.control.getControlOutput
	params	axis number
	result	error number, output

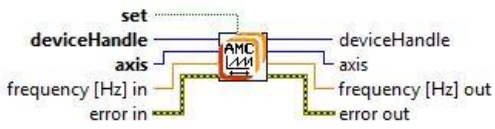
controlAmplitude

This function sets or gets the amplitude of the actuator signal of the selected axis.

LabVIEW		
AMC_controlAmplitude.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	amplitude	amplitude in mV
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error number
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	amplitude	set amplitude in V
	error	error message
C-DLL call		
AMC_controlAmplitude(*Int32 deviceHandle,Int32 axis,Int32* amplitude,BIn32 set*)		
JSON method(s) and parameters		
set	method	com.attocube.amc.control.setControlAmplitude
	params	axis number, amplitude in mV
	result	error number
get	method	com.attocube.amc.control.getControlAmplitude
	params	axis number
	result	error number, amplitude in V

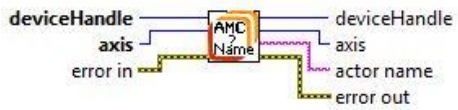
controlFrequency

This function sets or gets the frequency of the actuator signal of the selected axis.

LabVIEW		
AMC_controlFrequency.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	Frequenzy [Hz]	frequency in Hz
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	Frequenzy [Hz]	frequency in Hz
	error	error message
C-DLL call		
AMC_controlFrequency(*Int32 deviceHandle,Int32 axis,Int32* frequency,Bln32 set*)		
JSON method(s) and parameters		
set	method	com.attocube.amc.control.setControlFrequency
	params	axis number, frequency mHz
	result	error number
get	method	com.attocube.amc.control.getControlFrequency
	params	axis number
	result	error number, frequency in mHz

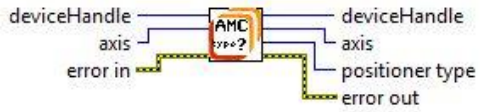
getActorName

This function gets the name of the positioner of the selected axis.

LabVIEW		
AMC_getActorName.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	actor name	name of the positioner
	error	error message
C-DLL call		
AMC_getActorName(*Int32 deviceHandle,Int32 axis,char* name*)		
JSON method(s) and parameters		
get	method	com.attocube.amc.control.getActorName
	params	axis number
	result	error numer, positioner name

getActorType

This function gets the type of the positioner of the selected axis.

LabVIEW		
`AMC_getActorType.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	positioner type	type of the positioner
	error	error message
C-DLL call		
`AMC_getActorType(`*Int32 deviceHandle, Int32 axis,AMC_actorType* type*`)`		
JSON method(s) and parameters		
get	method	`com.attocube.amc.control.getActorType`
	params	axis number
	result	error number, positioner type

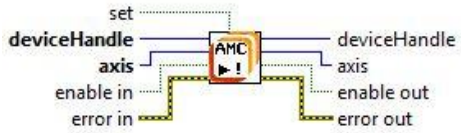
setReset

This function resets the actual position of the selected axis to zero and marks the reference position as invalid.

LabVIEW		
`AMC_setReset.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
C-DLL call		
`AMC_setReset(`*Int32 deviceHandle,Int32 axis*`)`		
JSON method(s) and parameters		
set	method	`com.attocube.amc.control.setReset`
	params	axis number
	result	error number

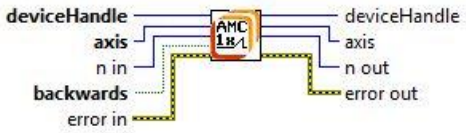
controlMove

This function sets or gets the approach of the selected axis' positioner to the target position.

LabVIEW		
AMC_controlMove.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	enable	true: enable approach false: disable approach
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	enable	true: approach enabled false: approach disabled
	error	error message
C-DLL call		
AMC_controlMove(*Int32 deviceHandle,Int32 axis,Bln32* enable,Bln32 set*)		
JSON method(s) and parameters		
set	method	com.attocube.amc.control.setControlMove
	params	axis number, enable
	result	error number
get	method	com.attocube.amc.control.getControlMove
	params	axis number
	result	error number, enable

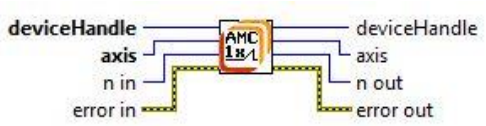
setNSteps

This function triggers n steps on the selected axis in desired direction.

LabVIEW		
AMC_setNSteps.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	n	number of steps
	backwards	true: movement in backward direction false: movement in forward direction
	error	error message
Out	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	n	number of steps
	error	error message
C-DLL call		
AMC_setNSteps(*Int32 deviceHandle,Int32 axis,Bln32 backward,Int32 n*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.move.setNSteps
	params	axis number, steps number, backward
	result	error number

getNSteps

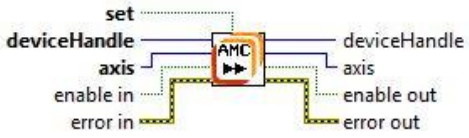
This function triggers a single step in desired direction.

LabVIEW		
`AMC_getNSteps.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	n	number of steps
	error	error message
Out	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	n	number of steps
	error	error message
C-DLL call		
`AMC_getNSteps(`*Int32 deviceHandle,Int32 axis,Int32* n*`)`		
JSON Method(s) and Parameters		
get	method	`com.attocube.amc.move.getNSteps`
	params	axis number
	result	error number, number of steps

controlContinousFwd

This function sets a continuous movement on the selected axis in positive direction or it gets the axis' movement status.

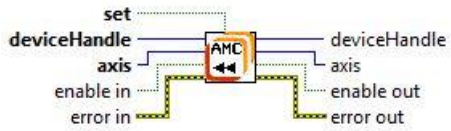
Note
When executing this function, present movements on the axis in negative direction are stopped.

LabVIEW		
AMC_controlContinousFwd.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	enable	true: enable movement in positive direction false: stop any movement
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	enable	true: movement in positive direction enabled false: any movement disabled
	error	error message
C-DLL call		
AMC_controlContinousFwd(*Int32 deviceHandle,Int32 axis,Bln32* enable,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.move.setControlContinousFwd
	params	axis number, enable
	result	error number
get	method	com.attocube.amc.move.getControlContinousFwd
	params	axis number
	result	error number, enable

controlContinousBkwd

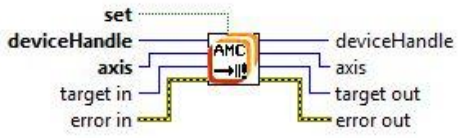
This function sets a continuous movement on the selected axis in backward direction or it gets the axis' movement status.

Note
When executing this function, present movements on the axis in positive direction are stopped.

LabVIEW		
AMC_controlContinousBkwd.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	enable	true: enable movement in negative direction false: stop any movement
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	enable	true: movement in positive direction enabled false: any movement stopped
	error	error message
C-DLL call		
AMC_controlContinousBkwd(*Int32 deviceHandle,Int32 axis,Bln32* enable,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.move.setControlContinousBkwd
	params	axis number, enable
	result	error number
get	method	com.attocube.amc.move.getControlContinousBkwd
	params	axis number
	result	error number, enable

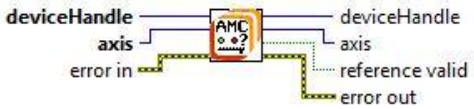
controlTargetPosition

This function sets or gets the target position for the movement on the selected axis.

LabVIEW		
AMC_controlTargetPosition.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	target	target position in nm
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	target	target position in nm
	error	error message
C-DLL call		
AMC_controlTargetPosition(*Int32 deviceHandle,Int32 axis,Int32* target,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.move.setControlTargetPosition
	params	axis number, target position
	result	error number
get	method	com.attocube.amc.move.getControlTargetPosition
	params	axis number
	result	error number, target position

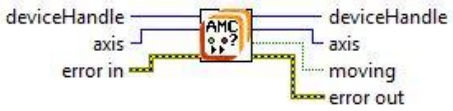
getStatusReference

This function gets information about the status of the reference position.

LabVIEW		
`AMC_getStatusReference.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	reference valid	true: reference position valid false: reference position invalid
	error	error message
C-DLL call		
`AMC_getStatusReference` (*Int32 deviceHandle,Int32 axis,Bln32* valid*)		
JSON Method(s) and Parameters		
get	method	`com.attocube.amc.status.getStatusReference`
	params	axis number
	result	error number, status reference

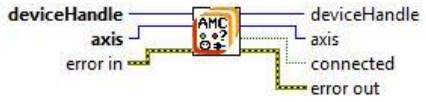
getStatusMoving

This function gets information about the status of the stage output.

LabVIEW		
`AMC_getStatusMoving.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	moving	0: idle (no movement commands for positioner pending) 1: moving (positioner actively driven to target position) 2: pending (positioner in target range and not actively driven)
	error	error message
C-DLL call		
`AMC_getStatusMoving(`*Int32 deviceHandle,Int32 axis,Int32* moving*`)`		
JSON Method(s) and Parameters		
get	method	`com.attocube.amc.status.getStatusMoving`
	params	axis number
	result	error number, status moving

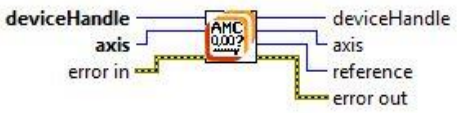
getStatusConnected

This function gets information about the connection status of the selected axis' positioner.

LabVIEW		
`AMC_getStatusConnected.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	connected	true: positioner electrically connected to controller false: positioner not electrically connected to controller
	error	error message
C-DLL call		
`AMC_getStatusConnected`(*Int32 deviceHandle,Int32 axis,Bln32* connected*)		
JSON Method(s) and Parameters		
get	method	`com.attocube.amc.status.getStatusConnected`
	params	axis number
	result	error number, status connected

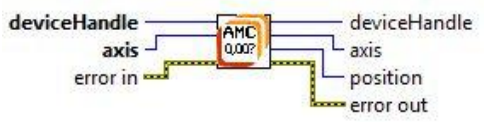
getReferencePosition

This function gets the reference position of the selected axis.

LabVIEW		
`AMC_getReferencePosition.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	reference	reference position in nm
	error	error message
C-DLL call		
`AMC_getReferencePosition(`*Int32 deviceHandle,Int32 axis,Int32* reference*`)`		
JSON Method(s) and Parameters		
get	method	`com.attocube.amc.control.getReferencePosition`
	params	axis number
	result	error number, reference position

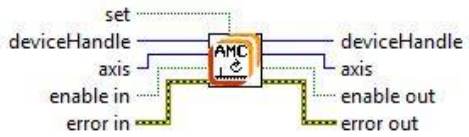
getPosition

This function gets the current position of the positioner on the selected axis.

LabVIEW		
AMC_getPosition.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	position	positioner's position in nm
	error	error message
C-DLL call		
AMC_getPosition(*Int32 deviceHandle,Int32 axis,Int32* position*)		
JSON Method(s) and Parameters		
get	method	com.attocube.amc.move.getPosition
	params	axis number
	result	error number, position

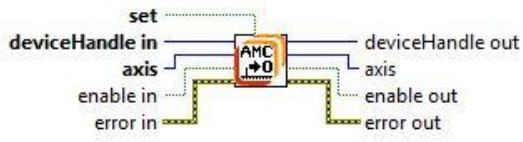
controlReferenceAutoUpdate

This function sets and gets the status of whether the reference position is updated when the reference mark is hit.

LabVIEW		
AMC_controlReferenceAutoUpdate.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	enable	true: update reference position every time the reference mark is hit false: update reference position just once when the reference mark is hit for the first time, ignore further hits
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	enable	true: reference position is updated every time the reference mark is hit false: reference position is updated just once when the reference mark is hit for the first time, further hits are ignored
	error	error message
C-DLL call		
AMC_controlReferenceAutoUpdate (*Int32 deviceHandle,Int32 axis,Bln32* enable,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.control.setControlReferenceAutoUpdate
	params	axis number, enable
	result	error number
get	method	com.attocube.amc.control.getControlReferenceAutoUpdate
	params	axis number
	result	error number, enable

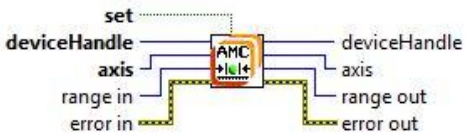
controlAutoReset

This function resets the position every time the reference position is detected.

LabVIEW		
AMC_controlAutoReset.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	enable	true: reset position every time the reference position is detected false: do not reset the position every time the reference position is detected
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	enable	enables/ disable functionality
	error	error message
C-DLL call		
AMC_controlAutoReset(*Int32 deviceHandle,Int32 axis,Bln32* enable,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.control.setControlAutoReset
	params	axis number, enable
	result	error number
get	method	com.attocube.amc.control.getControlAutoReset
	params	axis number
	result	error number, enable

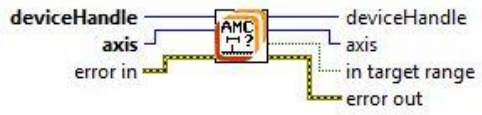
controlTargetRange

This function sets and gets the range around the target position in which the flag "In Target Range" (see VII.7.a) becomes active.

LabVIEW		
AMC_controlTargetRange.vi		
		
Input	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	range	target range in nm
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Output	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	range	target range in nm
	error	error message
C-DLL call		
AMC_controlTargetRange (*Int32 deviceHandle,Int32 axis,Int32* range,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.control.setControlTargetRange
	params	axis number, target range
	result	error number
get	method	com.attocube.amc.control.getControlTargetRange
	params	axis number
	result	error number, target range

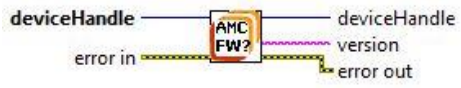
getStatusTargetRange

This function gets information about whether the selected axis' positioner is in target range or not.

<table>
<tr><td colspan="3">LabVIEW</td></tr>
<tr><td colspan="3">AMC_getStatusTargetRange.vi</td></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><td rowspan="3">In</td><td>deviceHandle*</td><td>device handle</td></tr>
<tr><td>axis*</td><td>number of the axis to be configured (0 – 2)</td></tr>
<tr><td>error</td><td>error message</td></tr>
<tr><td rowspan="4">Out</td><td>deviceHandle</td><td>device handle</td></tr>
<tr><td>axis</td><td>number of the configured axis (0 – 2)</td></tr>
<tr><td>in target range</td><td>true: positioner is within target range
false: positioner is not within target range</td></tr>
<tr><td>error</td><td>error message</td></tr>
<tr><td colspan="3">C-DLL call</td></tr>
<tr><td colspan="3">AMC_getStatusTargetRange(Int32 deviceHandle,Int32 axis,Bln32* target)</td></tr>
<tr><td colspan="3">JSON Method(s) and Parameters</td></tr>
<tr><td rowspan="3">get</td><td>method</td><td>com.attocube.amc.status.getStatusTargetRange</td></tr>
<tr><td>params</td><td>axis number</td></tr>
<tr><td>result</td><td>error number, status target range</td></tr>
</table>

getFirmwareVersion

This function gets the version number of the controller's firmware.

LabVIEW		
AMC_getFirmwareVersion.vi		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	version	firmware version number
	error	error message
C-DLL call		
AMC_getFirmwareVersion(*Int32 deviceHandle,char* version*)		
JSON Method(s) and Parameters		
get	method	com.attocube.system.getFirmwareVersion
	params	
	result	firmware version number

getFpgaVersion

This function gets the version number of the controller's FPGA.

LabVIEW		
AMC_getFpgaVersion.vi		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	version	FPGA version number
	error	error message
C-DLL call		
AMC_getFpgaVersion(*Int32 deviceHandle,char* version*)		
JSON Method(s) and Parameters		
get	method	com.attocube.amc.description.getFpgaVersion
	params	
	result	FPGA version number

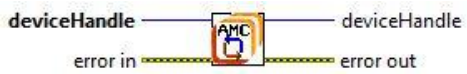
rebootSystem

This function reboots the device.

LabVIEW		
AMC_rebootSystem.vi		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	error	error message
C-DLL call		
AMC_rebootSystem(*Int32 deviceHandle*)		
JSON Method(s) and Parameters		
set	method	rebootSystem
	params	
	result	error number

factoryReset

This function resets the device to the factory settings when it's booted the next time.

LabVIEW		
`AMC_factoryReset.vi`		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	error	error message
C-DLL call		
`AMC_factoryReset` (*Int32 deviceHandle*)		
JSON Method(s) and Parameters		
set	method	`factoryReset`
	params	
	result	error number

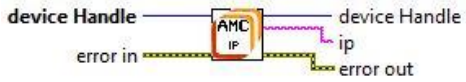
getMacAddress

This function gets the MAC address of the device.

LabVIEW		
`AMC_getMacAddress.vi`		
		
Input	deviceHandle*	device handle
	error	error message
Output	deviceHandle*	device handle
	mac	MAC address
	error	error message
C-DLL call		
`AMC_getMacAddress` (*Int32 deviceHandle, char* mac*)		
JSON Method(s) and Parameters		
get	method	`com.attocube.system.getMacAddress`
	params	
	result	MAC address

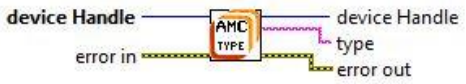
getIpAddress

This function gets the device's IP address.

LabVIEW		
AMC_getIpAddress.vi		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	ip	IP address of device
	error	error message
C-DLL call		
AMC_getIpAddress(*Int32 deviceHandle, char* ip*)		
JSON Method(s) and Parameters		
get	method	com.attocube.system.network.getIpAddress
	params	
	result	IP adress

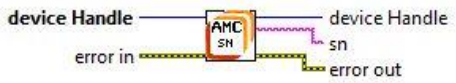
getDeviceType

This function gets the device type based on its EEPROM configuration.

LabVIEW		
AMC_getDeviceType.vi		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	type	type of the device
	error	error message
C-DLL call		
AMC_getDeviceType(*Int32 deviceHandle, char* type*)		
JSON Method(s) and Parameters		
get	method	com.attocube.amc.description.getDeviceType
	params	
	result	device type

getSerialNumber

This function gets the device's serial number.

LabVIEW		
AMC_getSerialNumber.vi		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	sn	Serial number
	error	error message
C-DLL call		
AMC_getSerialNumber(*Int32 deviceHandle, char* sn*)		
JSON Method(s) and Parameters		
get	method	com.attocube.system.getSerialNumber
	params	
	result	serial number

getDeviceName

This function gets the device's name.

LabVIEW		
AMC_getDeviceName.vi		
		
In	deviceHandle*	device handle
	error	error message
Out	deviceHandle*	device handle
	name	get device Name
	error	error message
C-DLL call		
AMC_getDeviceName(*Int32 deviceHandle, char* name*)		
JSON Method(s) and Parameters		
get	method	com.attocube.system.getDeviceName
	params	
	result	device name

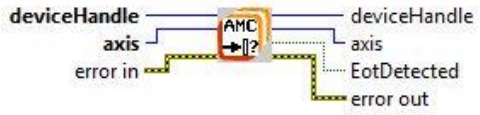
setDeviceName

This function sets the device's name.

LabVIEW		
AMC_setDeviceName.vi		
		
In	deviceHandle*	device handle
	name	set device name
	error	error message
Out	deviceHandle*	device handle
	error	error message
C-DLL call		
AMC_setDeviceName(*Int32 deviceHandle, const char* name*)		
JSON Method(s) and Parameters		
set	method	com.attocube.system.setDeviceName
	params	device name
	result	error number

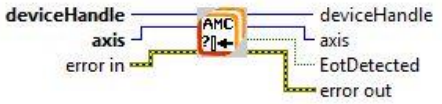
getStatusEotFwd

This function gets the status of the end of travel detection on the selected axis in forward direction.

LabVIEW		
AMC_getStatusEotFwd.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	EotDetected	true: end of travel in forward direction detected false: end of travel in forward direction not detected
	error	error message
C-DLL call		
AMC_getStatusEotFwd(*Int32 deviceHandle,Int32 axis,Bln32* EotDetected*)		
JSON Method(s) and Parameters		
get	method	com.attocube.amc.status.getStatusEotFwd
	params	axis number
	result	error number, end of travel detected

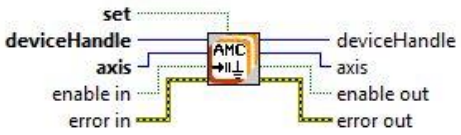
getStatusEotBkwd

This function gets the status of the end of travel detection on the selected axis in backward direction.

LabVIEW		
AMC_getStatusEotBkwd.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	EotDetected	true: end of travel in backward direction detected false: end of travel in backward direction not detected
	error	error message
C-DLL call		
AMC_getStatusEotBkwd(*Int32 deviceHandle,Int32 axis,Bln32* EotDetected*)		
JSON Method(s) and Parameters		
get	method	com.attocube.amc.status.getStatusEotBkwd
	params	axis number
	result	error number, end of travel detected

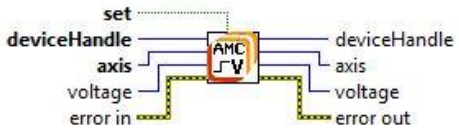
controlEotOutputDeactive

This function sets or gets the output applied to the selected axis on the end of travel.

LabVIEW		
AMC_controlEotOutputDeactive.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	enable	true: deactivate output on end of travel false: keep output active on end of travel
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	enable	true: output is deactivated on end of travel false: output stays active on end of travel
	error	error message
C-DLL call		
AMC_controlEotOutputDeactive(*Int32 deviceHandle,Int32 axis,Bln32* enable,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.move.setControlEotOutputDeactive
	params	axis number, enable
	result	error number
get	method	com.attocube.amc.move.getControlEotOutputDeactive
	params	axis number
	result	error number, enable

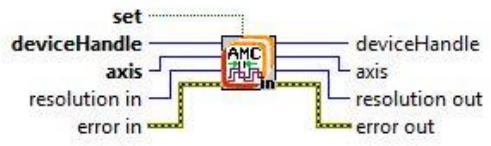
controlFixOutputVoltage

This function sets or gets the DC level output of the selected axis.

LabVIEW		
AMC_controlFixOutputVoltage.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	voltage	DC output in mV
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	voltage	DC output in mV
	error	error message
C-DLL call		
AMC_controlFixOutputVoltage(*Int32 deviceHandle,Int32 axis,Int32* voltage,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.control.setControlFixOutputVoltage
	params	axis number, voltage
	result	error number
get	method	com.attocube.amc.control.getControlFixOutputVoltage
	params	axis number
	result	error number, voltage

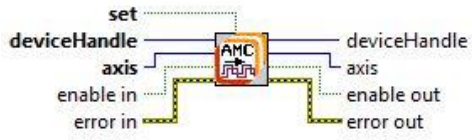
controlAQuadBInResolution

This function sets or gets the AQuadB input resolution for setpoint parameter.

LabVIEW		
AMC_controlAQuadBInResolution.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	resolution	resolution in nm
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	resolution	resolution in nm
	error	error message
C-DLL call		
AMC_controlAQuadBInResolution(*Int32 deviceHandle,Int32 axis,Int32* resolution,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.rtin.setControlAQuadBInResolution
	params	axis number, resolution
	result	error number
get	method	com.attocube.amc.rtin.getControlAQuadBInResolution
	params	axis number
	result	error number, resolution

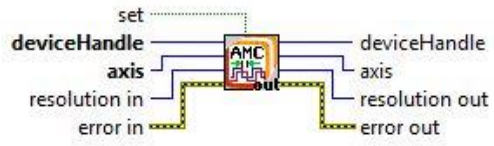
controlAQuadBOut

This function sets or gets status of AQuadB output for position indication.

LabVIEW		
AMC_controlAQuadBOut.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	enable	true: enable AQuadB output false: disable AQuadB output
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	enable	true: AQuadB output enabled false: AQuadB output disabled
	error	error message
C-DLL call		
AMC_controlAQuadBOut(*Int32 deviceHandle,Int32 axis,Bln32* enable,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.rtout.setControlAQuadBOut
	params	axis number, enable
	result	error number
get	method	com.attocube.amc.rtout.getControlAQuadBOut
	params	axis number
	result	error number, enable

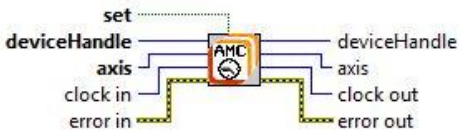
controlAQuadBOutResolution

This function sets or gets the AQuadB output resolution for position indication.

LabVIEW		
AMC_controlAQuadBOutResolution.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	resolution	resolution in nm
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	resolution	resolution in nm
	error	error message
C-DLL call		
AMC_controlAQuadBOutResolution(*Int32 deviceHandle,Int32 axis,Int32* resolution,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.rtout.setControlAQuadBOutResolution
	params	axis number, resolution
	result	error number
get	method	com.attocube.amc.rtout.getControlAQuadBOutResolution
	params	axis number
	result	error number, resolution

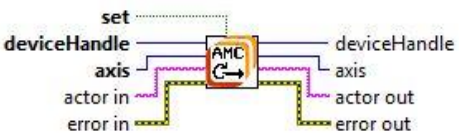
controlAQuadBOutClock

This function sets or gets the clock for AQuadB output.

LabVIEW		
AMC_controlAQuadBOutClock.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	clock	signal interval in multiples of 20 ns – minimum 2 (40 ns), maximum 65535 (1,310,700 ns)
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	clock	signal interval in multiples of 20 ns – minimum 2 (40 ns), maximum 65535 (1,310,700 ns)
	error	error message
C-DLL call		
AMC_controlAQuadBOutClock(*Int32 deviceHandle,Int32 axis,Int32* clock,BIn32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.rtout.setControlAQuadBOutClock
	params	axis number, clock
	result	error number
get	method	com.attocube.amc.rtout.getControlAQuadBOutClock
	params	axis number
	result	error number, clock

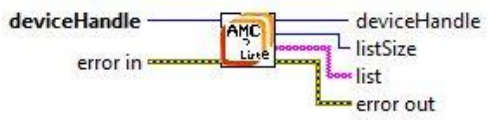
setActorParametersByName

This function sets the name for the positioner on the selected axis.

LabVIEW		
`AMC_setActorParametersByName.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	actor	name of the positioner
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	actor	name of the positioner
	error	error message
C-DLL call		
`AMC_setActorParametersByName` (*Int32 deviceHandle,Int32 axis,const char* actorname*)		
JSON Method(s) and Parameters		
set	method	`com.attocube.amc.control.setActorParametersByName`
	params	axis number, positioner
	result	error number

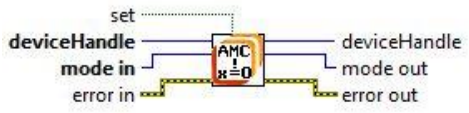
getPositionersList

This function gets a list of the positioners connected to the device.

LabVIEW		
AMC_getPositionersList.vi		
		
Input	deviceHandle*	device handle
	error	error message
Output	deviceHandle*	device handle
	listSize	size of the buffer
	list	retrieve positioners list from device
	error	error message
C-DLL call		
AMC_getPositionersList(*Int32 deviceHandle,char* list,Int32 listSize*)		
JSON Method(s) and Parameters		
get	method	com.attocube.amc.description.getPositionersList
	params	
	result	list

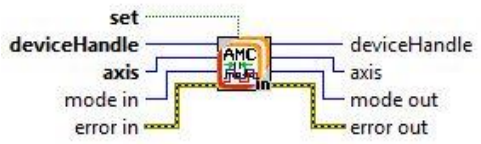
controlRtOutSignalMode

This function sets or gets the real time output mode for the selected axis.

LabVIEW		
AMC_controlRtOutSignalMode.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	mode	0: AquadB (LVTTL) 1: AquadB (LVDS)
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	mode	0: AquadB (LVTTL) 1: AquadB (LVDS)
	error	error message
C-DLL call		
AMC_controlRtOutSignalMode(*Int32 deviceHandle,Int32 *mode,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.rtout.setRtOutSignalMode
	params	Axis number, mode
	result	error number
get	method	com.attocube.amc.rtout.getRtOutSignalMode
	params	axis number
	result	error number, mode

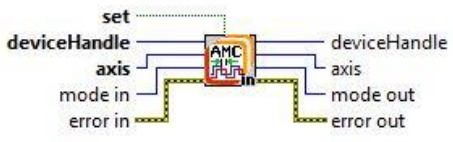
controlRealtimeInputMode

This function sets or gets the real time input mode for the selected axis.

LabVIEW		
AMC_controlRealtimeInputMode.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	mode	0: Aquadb (LVTTL) 1: AquadB (LVDS) 8: Stepper (LVTTL) 9: Stepper(LVDS) 0: Trigger (LVTTL 11: Trigger (LVDS) 15: disable
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	mode	0: Aquadb (LVTTL) 1: AquadB (LVDS) 8: Stepper (LVTTL) 9: Stepper(LVDS) 0: Trigger (LVTTL 11: Trigger (LVDS) 15: disabled
	error	error message
C-DLL call		
AMC_controlRealtimeInputMode (*Int32 deviceHandle,Int32 axis,Int32* mode,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.rtin.setRealTimeInMode
	params	axis number, mode
	result	error number
get	method	com.attocube.amc.rtin.getRealTimeInMode
	params	axis number
	result	error number, mode

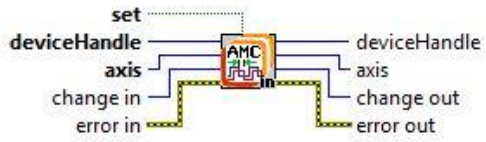
controlRealtimeInputLoopMode

This function sets or gets the real time input loop mode for the selected axis.

LabVIEW		
AMC_controlRealtimeInputLoopMode.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	mode	0: open-loop 1: closed-loop
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	mode	0: open-loop 1: closed-loop
	error	error message
C-DLL call		
AMC_controlRealtimeInputLoopMode(*Int32 deviceHandle,Int32 axis,Int32* mode,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.rtin.setRealTimeInFeedbackLoopMode
	params	axis number mode
	result	error number
get	method	com.attocube.amc.rtin.getRealTimeInFeedbackLoopMode
	params	axis number
	result	error number, mode

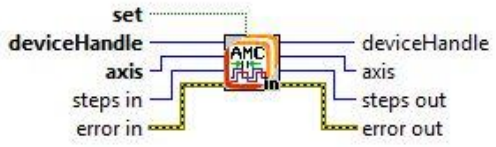
controlRealtimeInputChangePerPulse

This function sets or gets the change per pulse for the selected axis under real time input in the closed-loop mode.

LabVIEW		
AMC_controlRealtimeInputChangePerPulse.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	change	change per pulse in nm – maximum 1,000,000 nm
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	change	change per pulse in nm
	error	error message
C-DLL call		
AMC_controlRealtimeInputChangePerPulse(*Int32 deviceHandle,Int32 axis,Int32* change,Bln32 set*)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.rtin.setRealTimeInChangePerPulse
	params	axis number change per pulse
	result	error number
get	method	com.attocube.amc.rtin.getRealTimeInChangePerPulse
	params	axis number
	result	error number, change per pulse

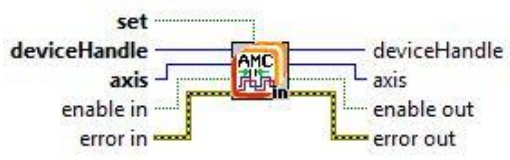
controlRealtimeInputStepsPerPulse

This function sets or gets the steps per pulse for the selected axis under real time input in closed-loop mode.

LabVIEW		
`AMC_controlRealtimeInputStepsPerPulse.vi`		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	steps	number of steps per pulse – maximum 10,000 steps
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	steps	number of steps per pulse
	error	error message
C-DLL call		
`AMC_controlRealtimeInputStepsPerPulse(`*Int32 deviceHandle,Int32 axis,Int32* steps,Bln32 set*`)`		
JSON Method(s) and Parameters		
set	method	`com.attocube.amc.rtin.setRealTimeInStepsPerPulse`
	params	axis number steps per pulse
	result	error number
get	method	`com.attocube.amc.rtin.getRealTimeInStepsPerPulse`
	params	axis number
	result	error number, steps per pulse

controlRealtimeInputMove

This function sets or gets the status for real time input on the selected axis in closed-loop mode.

LabVIEW		
AMC_controlRealtimeInputMove.vi		
		
In	deviceHandle*	device handle
	axis*	number of the axis to be configured (0 – 2)
	enable	true: enable movements false: disable movements
	set*	true: send the supplied values to controller false: ignore input, only retrieve results
	error	error message
Out	deviceHandle	device handle
	axis	number of the configured axis (0 – 2)
	enable	true: movements enabled false: movements disabled
	error	error message
C-DLL call		
AMC_controlRealtimeInputMove(Int32 deviceHandle,Int32 axis,Bln32* enable,Bln32 set)		
JSON Method(s) and Parameters		
set	method	com.attocube.amc.rtin.setControlMoveGPIO
	params	axis number enable
	result	error number
get	method	com.attocube.amc.rtin.getControlMoveGPIO
	params	axis number
	result	error number, enable

XII.3.e. Finding Devices via Discovery

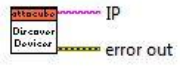
attocube provides a discovery DLL that allows to search for attocube devices in all connected networks using the SSDP protocol.

Tip
The deviceType parameter in the AMC_check_SubVI.vi, as well as the AD_Check function, can be used to also search for attocube's interferometer IDS3010 (using parameter value "0" to only search for IDS3010, or using "2" to search for both device types).

Note
MOVE uses the same SSDP mechanism on its "Find Devices" screen.

Finding Devices in LabVIEW

The provided discovery DLL can also be used in LabVIEW, an example on how to use it can be found in the Discovery.vi (and its SubVIs). It can be found in the Discovery folder on the provided USB flash drive.

Discovery.vi		
		
	IP	IP address of the selected device
	error	Error message

Depending on the Number of found devices the VI behaves differently:

- If no device is found, the following dialog appears:

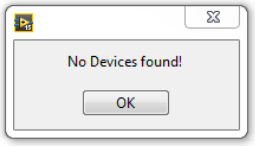


Figure 31 "No devices found" dialog

- If only one device is found, the device and therefore IP is chosen automatically.
- If more than one device is found, the following dialog appears, urging you to select one of the found devices. You can go through all found devices by clicking on the "please select IP" drop-down menu.

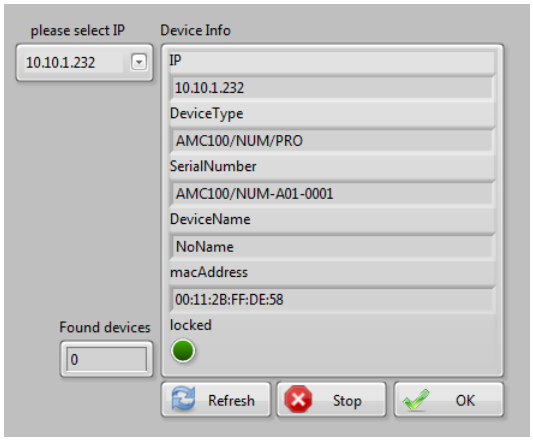


Figure 32 "Select device" dialog

Find Devices via C-DLL

To gather device information about found devices using the discovery DLL, use the following functions inside the DLL:

C-DLL call
`int32_t AD_Check(`*deviceType*`)`
Sends a Broadcast to look for all available devices specified under deviceType in all connected networks and returns the number of found devices.

C-DLL call
`int32_t AD_GetDeviceInfos(`*int32_t DeviceIndex, void *DeviceInfo*`)`
Can be used to gather more specific Information about found devices. The maximum number of times you can iterate over this function (using the DeviceIndex) is defined by the number of found devices (as returned from the AD_Check function).

C-DLL call
`void AD_ReleaseInfo(void)`
Releases the memory used for the storage of the gathered device information, allocated by the AD_Check and AD_GetDeviceInfos functions.

attocube systems AG
Königinstrasse 11a
D - 80539 München, Germany
Phone: +49 89-2877 809-0
Fax: +49 89-2877 809-19

E-Mail: info@attocube.com
www.attocube.com

For technical queries, contact:
support@attocube.com

North America Support Hotlines:

+1 212 962 6930 (East Coast Office)
+1 510 649 9245 (West Coast Office)