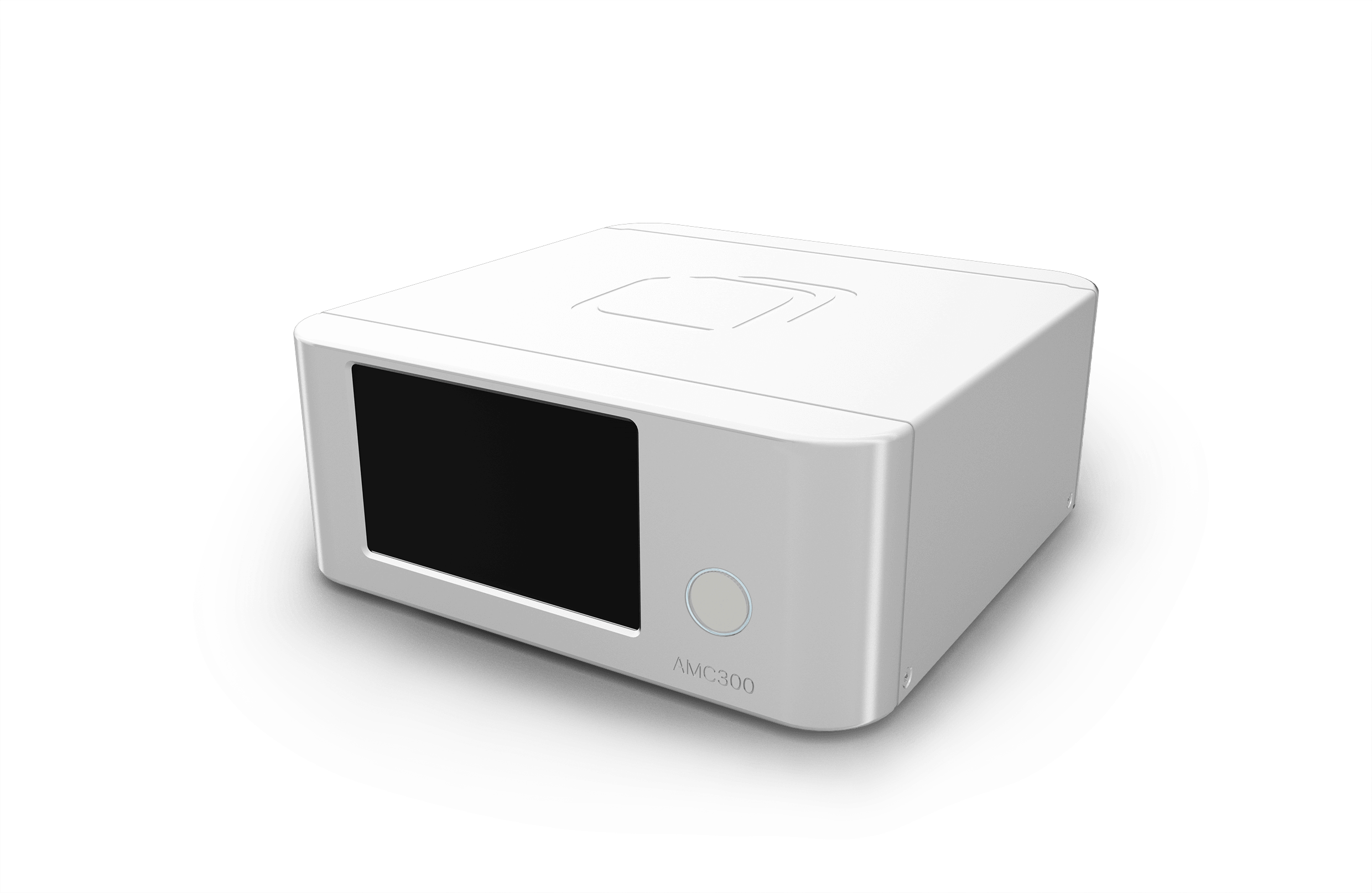


Interface Manual

AMC100 & AMC300





|  |  |
| --- | --- |
|  |  |
| Modified: | 03/2021 |
| Products: | AMC100 | AMC300 | attoDISCOVERY Software | AMC Webserver Application | MOVE |

© 2021 attocube systems AG. Product and company names listed are trademarks or trade names of their respective companies. Any rights not expressly granted herein are reserved. ATTENTION: Specifications and technical data are subject to change without notice.

Table of Contents

[1 System Integration 2](#_Toc67568786)

[1.1 Connecting to Third Party Hardware 2](#_Toc67568787)

[1.1.1 Cabling Restrictions 2](#_Toc67568788)

[1.1.2 Pin Assignments 2](#_Toc67568789)

# System Integration

The AMC100 and the AMC300 can be integrated with external systems or devices by

* combining it with third party hardware
* establishing incoming and outgoing trigger connections (/IO upgrade required)
* controlling it with individual software interfaces

## Connecting to Third Party Hardware

|  |  |
| --- | --- |
|  |  |
| 09 | 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. |
|  |  |
|  |  |
| M00 | 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. |
|  |  |

### 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.

### Pin Assignments

The pins of the device’s positioner control cables can be found in the AMC100 or AMC300 User Manual.

# Software communication & interfaces

You can integrate your attocube Device into complex automated processes via individual software interfaces. Attocube provides APIs for the programming languages C, C#, LabVIEW, Python & Matlab as well as short programming examples to get you started.

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

## Introduction

The Device provides a set of software communication interfaces offering a broad set of functions and options. These can be used to configure the Device as well as to read out data. In particular, these are:

• Web interface (please also refer to the User Manual)

• JSON-RPC

• C DLL

• C# DLL

• LabVIEW VIs

• Matlab library

• Python Library

Most of the different functions are accessible within every interface. This is why we sort by functions not by interfaces. For every function, we show how the implementation in the particular interface is done. Anyway first we give a short explanation of the different interfaces:

|  |  |
| --- | --- |
| M00 | Note  Part of the conventions mentioned below are specific for the handling of attocube devices and are not necessarily applicable in other contexts. |

### Web Interface

The Device runs a built-in webserver. This means that a web interface can be accessed via a common web browser. A how to set up the IP and a first connection is given in the User Manual. The web interface is the most straightforward way to communicate with the Device and almost the full functionality is implemented.

### JSON-RPC

The device allows platform-independent communication using JSON-RPC via TCP/IP and websocket. The JSON commands are the lower level that all other wrappers (e.g. Python or C) use.

### C/C# DLLs

Based on the JSON interface, C/C# libraries are available to implement the functions within C/C#-based coding environments.

### LabVIEW VIs

We offer ready-made VIs to have a fast and easy implementation in National Instrument’s LabVIEW environment.

### Matlab Library

Based on the JSON interface, we offer ready-made Matlab functios to have a fast and easy implementation in Mathwork’s Matlab environment.

## Overview and implementation of the APIs

### JSON-RPC (JRPC2.0)

Your attocube Device allows platform-independent communication using JSON-RPC via TCP/IP. When using JSON-RPC, the following conventions apply.

|  |  |
| --- | --- |
| **Transport protocols** | *TCP*  Uses communication port 9090. |
| **Calling a JSON RPC 2.0 method** | A JSON RPC method is called by sending a message to the device.  { "jsonrpc": "2.0", "method": "<method>", "params": [<param [0]>, <param [1]>, …], "id": <call id>, “api”: <api version>}  *<method>*: String defined in chapter 2.2.  *<param x*>: Parameter for the method call if the PARAM is put between two “, it is a string. Without “ it is a number  *<call id>*: A unique id to find the corresponding answer  *<api version>*: A version identifier for backward compatibility, please set to 2 |
| **Example** | Example:  { "jsonrpc": "2.0", "method": "com.attocube.ids.displacement.getAxisDisplacement", "params": [1], "id": 1, “api”: 2} |
| **Receiving a JSON RPC 2.0 response** | The JSON RPC method answer is then sent back as payload to the OK message:  { "jsonrpc": "2.0", "result": [<return values [0]>, <return values [1]>, …], "id": <call id>}  *<return values [x]*>: The return parameters  *<call id>*: The unique id of the method call |
| **Example** | Example:  { "jsonrpc": "2.0", "results": [0, 4], "id": 1} |
| **Example** | Example:  Communication via PuTTY  Open a Telnet connection with PuTTY.    Sending Json-Rpc commands in the command line interface. |

### C Library

The C API is provided to integrate the Device with all its functionality within your C programs.

|  |  |
| --- | --- |
| **Overview** | The C API contains of following files:  Standard C API:   * attocubeJSON.dll (x64 and x86 versions for a windows environment) * attocubeJSON.lib (x64 and x86 versions for a windows environment) * attocubeJSON.so (x64 and x86 versions for a linux environment) * attocubeJSONCall.h (header file for the general functions) * generatedAPI.h (header file for the device specific functions)   Disovery C API:   * attocube-discovery-dll.dll * attocube discovery-dll.lib * attocube-discovery.h (header file) |
| **Using the .dll’s with different systems** | Note that if you want to use the .dlls within x64 based systems outside the framework of Visual C, you might need to convert the library into a static .a format. |
| **Establishing a connection** | To connect to a device, please use (part of attocubeJSONCall.h):  int ATTOCUBE\_API **Connect**(const char *\*deviceAddress*, int\* *deviceHandle*)  The device handle is the reference to the connection and the device and is input to all other device functions that are following.  To close the connection, please use:  int ATTOCUBE\_API **Disconnect**(int \* *deviceHandle*)  Both functions are included in the API.  For a TCP/IP connection, the port 9090 is used. |
| **Discovering devices within the same network** | The discovery function can be used:  It searches your network for available attocube devices and returns a list of properties. This is done by a SSDP broadcast. If no devices are found, please check the device connection via TCP/IP (e.g. via the websever). The device must be in the same subnet than the requesting PC.  **IMPORTANT NOTE:** These functions are part of an additional discovery .dll – the “attocube discovery dll”, which is also part of the standard delivery content.  Therefore, following functions are available:  int DLL\_EXP AD\_GetDeviceInfos(int index, DeviceInfo\* info)  (Get informations about a discovered device)  void DLL\_EXP AD\_ReleaseInfo();  (Release memory allocated by AD\_Check)  int DLL\_EXP AD\_Check(deviceType)  (Checks discoverable devices on the network and retrieves informations)  **Special data types:**  typedef struct {  char ipAddress[32];  /\*\*< IP address of the device \*/  char modelName[32];  /\*\*< Type of the device \*/  char serialNumber[32];  /\*\*< Serial number of the device \*/  char deviceName[32];  /\*\*< Friendly name assigned to the device \*/  char macAddress[32];  /\*\*< MAC address of the device \*/  bool locked;  /\*\*< Device locked by other program \*/  } DeviceInfo;  typedef enum {  IDS,  MOTION\_CTRLER,  BOTH  } deviceType; |

### C# Library

The C# API is provided to integrate the device with all its functionality within your C# programs

|  |  |
| --- | --- |
| **Overview** | The C# API contains of following files:   * CSharpAPIDLL.dll (compiled as “any” version) * Newtonsoft.Json.dll (compiled as “any” version) * Attocube.chm (helpfile) |
| **Establing a connection** | To connect to a device, please create an device object  public static Attocube<Device> client = new Attocube<Device>()  where <Device> is e.g. AMC or IDS  The connect function is a property of the Attocube<Device> class.  public void **Connect**(string *ipAddress*, int *port*)  The device handle is the reference to the connection to the device and is input to all other device functions that are following.  To close the connection, please use:  public void **Disconnect**()  Both functions are included in the API and part of the device class (so initialize a member of the class first).  For a TCP/IP connection, use the port 9090. |
| **Discovering devices within the same network** | The discovery function can be used:  It searches your network for available devices and returns a list of properties. This is done by an SSDP broadcast. If no devices are found, please check the device connection via TCP/IP (e.g. via the websever) or make sure that the device is in the same subnet than your PC  **IMPORTANT NOTE:** These functions are part of an additional discovery .dll – the “attocube discovery dll”, which is also part of the standard delivery content.  Therefore, following function is available:  public DiscoveryData[] **Check**()  **Special data type:**  Type: DiscoveryData  Class for handling the data of devices discovered using the discovery protocol |

### LabVIEW

The LabVIEW API is provided to integrate the Device with all its functionality within your LabVIEW Vis.

|  |  |
| --- | --- |
| **Overview** | The LabVIEW API contains a LabVIEW project which contains all single function VIs and a master example VI that uses almost all functions available and that mimics the web interfaces UI for easy navigation. |
| **Implementation** | To reduce complexity and external dependencies all TCP/IP calls have been implemented with native LabVIEW TCP/IP elements. For older LabVIEW Versions where there is no native TCP/IP support, DLL based VIs have been created taking care of the TCP/IP communication.  The folders “DLLHandler” or “TCPHandler” contain the respective SubVIs handling the messaging and communication with your attocube device, which are used within all low-level VIs. Those should not be modified or used directly. |
| **High-level Wrapper VIs** | For most functions that do have both a set and a get method a higher level “controlMethod” VI has been created to reduce the number of VIs and also be as backwards compatible as possible to the older motion controller series ECC100 and ANC350. Some additional high level VIs like the deviceInfo VI have been created where multiple low-level VIs are combined into one VI and all In- and Outputs are bundled into clusters.  In case you still want to use those low-level VIs instead, they can be found inside folders that contain the word “SubVIs”. For code cleanliness it is not recommended to use those. However to keep the documentation consistent over all programming languages, only the low-level methods are documented (see chapter 3). |
| **Establing a connection** | To connect to an device, please use the connect VI  The output is the reference to the connection to the device and is needed as an input to all other device functions that are following.  To close the connection, please use the Close VI  Both VIs are included in the API. |
| **Discovering devices within the same network** | The discovery function can be used:  It searches your network for available devices and returns a list of properties. This is done by an SSDP broadcast. If no devices are found, please check the device connection via TCP/IP (e.g. via the websever) or make sure that your device is in the same subnet than your PC.  **IMPORTANT NOTE:** These functions are part of an external DLL – the “attocube discovery dll”, which is also part of the standard delivery content.  Therefore, the “Check.vi” is available. |

### Matlab

The Matlab API is provided to integrate the Device with all its functionality within your Matlab scripts.

|  |  |
| --- | --- |
| **Establing a connection** | To connect to an device, please use:  [success, DeviceHandle] = **connect**(*IPAddress*, *port*)  The device handle is the reference to the connection to the device and is input to all other device functions that are following.  To close the connection, please use:  [success] = **disconnect**(*DeviceHandle*)  Both functions are included in the API.  For a TCP/IP connection, use the port 9090 |

### Python

The Python API is provided to integrate the device with all its functionality within your Python programs.

|  |  |
| --- | --- |
| **Overview** | The Python API contains a folder with domain specific files.  To have access to the python functions, please import the Device within your python script:  import **<Device>**  **where <Device> is your DeviceType, e.g. AMC or IDS** |
| **Establing a connection** | To connect to an device, please use:  device = <Device>.Device(*ipAdress*)  device.**connect()**  The device handle is the reference to the connection to the device and is input to all other device functions that are following.  To close the connection, please use:  device .**close()**  Both functions are included in the API and part of the device class (so initialize a member of the class first).  For a TCP/IP connection, the port 9090 is used per default. |
| **Discovering devices within the same network** | The discovery function can be used:  It searches your network for available devices and returns a list of properties. This is done by an SSDP broadcast. If no devices are found, please check the device connection via TCP/IP (e.g. via the websever) or make sure that the device is in the same subnet than your PC  Therefore, following module function is available:  <Device>.**discover()**  This returns a dictionary containing all found devices combined with their device information. |

## Error handling

### C error handling

|  |  |
| --- | --- |
| **Introduction** | The error handling in C is realized with return values, directly returned by each function. On success, the function yields zero. Negative error numbers indicate an error within the DLL itself and are specified in the header File ( attocubeJSONCall.h). Positive Error numbers indicate an error in the Device, and can be translated to readable strings with system\_errorNumberToString() |
| **Example** | int value;  int ret = Device\_Function(device, &value);  if( ret == ATTOCUBE\_Ok) {  //success  }  else if (res < ATTOCUBE\_Ok) {  //DLL Error, e.g. not connected  }  else if (res > ATTOCUBE\_Ok) {  //Device Error  char errorNameBuf[BUFFER\_SIZE];  system\_errorNumberToString(device, 0, ret, errorNameBuf, BUFFER\_SIZE);  printf("%s", errorNameBuf)  } |

### C# error handling

|  |  |
| --- | --- |
| **Introduction** | The error handling in C# is realized with exceptions not by error numbers. Errors can be caught using a try-catch statement. To include the device specific exceptions, the catch clause will need the AttocubeAPIException as argument. An example code is shown below. |
| **Example** | public class **AttocubeApiException:**  **ApplicationException**  Example: Exception handling  try  {  attoDevice.<Method>();  }  catch (AttocubeApiException e)  {  int err = e.ErrorCode; // passes the errorcode of type int to the variable "err"  string errmsg = attoDevice .ErrorNumberToString(0, err);  // converts "err" into the corresponding errormessage and passes it to "errmsg" of type string  } |

### Python error handling

|  |  |
| --- | --- |
| **Introduction** | The error handling in Python is realized with exceptions not by error numbers. Errors can be caught using a try-except statement. To include the specific exceptions, the catch clause will need the AttoException as argument. An example code is shown below. |
| **Example** | #example for exception handling  from ACS import AttoException  try:  print(dev.<Method> ()) #OK  except AttoException as e:  print(e) |

### LabVIEW error handling

|  |  |
| --- | --- |
| **Introduction** | The error handling in LabVIEW is realized by using an error message variable which should be looped through all VIs. Therefore, every VI provides an error in and error out connector. Note that we divide the error variable in error messages and “real” errors, which are treated differently. Error messages have a positive error number value combined with the Boolean error status set on inactive (Boolean value on false – visualized by a green hook icon), whereas “real” errors have a negative error number values with an active (Boolean value on true – visualized by a red cross icon) error status. Error messages do not influence the execution of the following VIs, they are used to inform the user. “Real” errors hinder the execution of following VIs, they are meant to stop the program. Examples are shown below: |
| **Example** | Example for an error message  An indicator is used to visualize the error message in the front panel ( see picture below ). Facing a positive error code value, an error message is indicated. The status is still set on green meaning no real error available. Error messages are intended to inform the user or other functions about certain cases. If an error message is inputted in a following VI, this VI still is executed. |
|  | Example for a “real” error  Facing a negative error code value, a “real” error is indicated (see picture below) Therefore, also the status is set on red, which means that the error is active. When an active error is inputted in a following VI, the VI will not execute its function. |

# Functions

## Access

|  |
| --- |
| **getLockStatus** This function returns if the device is locked and if the current client is authorized to use the device. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| locked | Is the device locked? |
| authorized | Is the client authorized? |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: getLockStatus |
| params: [] |
| Result: [errNo, locked, authorized] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_getLockStatus**(int deviceHandle, bool\* locked, bool\* authorized) |

|  |
| --- |
| **Python** |

|  |
| --- |
| locked, authorized = **[dev].access.getLockStatus**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [locked, authorized] = **AMC\_getLockStatus**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| Tuple<bool,bool> value = [Device].**GetLockStatus**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getLockStatus.vi |

|  |
| --- |
| **grantAccess** Grants access to a locked device for the requesting IP by checking against the password |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | password | string the current password |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: grantAccess |
| params: [password] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_grantAccess**(int deviceHandle, const char\* password) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].access.grantAccess**(password) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **AMC\_grantAccess**(password) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**GrantAccess**(string password) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| grantAccess.vi |

|  |
| --- |
| **lock** This function locks the device with a password, so the calling of functions is only possible with this password. The locking IP is automatically added to the devices which can access functions |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | password | string the password to be set |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: lock |
| params: [password] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_lock**(int deviceHandle, const char\* password) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].access.lock**(password) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **AMC\_lock**(password) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Lock**(string password) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| lock.vi |

|  |
| --- |
| **unlock** This function unlocks the device, so it will not be necessary to execute the grantAccess function to run any function |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | password | string the current password |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: unlock |
| params: [password] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_unlock**(int deviceHandle, const char\* password) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].access.unlock**(password) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **AMC\_unlock**(password) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Unlock**(string password) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| unlock.vi |

## Amcids

|  |
| --- |
| **getLowerSoftLimit** Gets the lower boundary of the soft limit protection.  This protection is needed if the IDS working range is smaller than the positioners travel range.  It is no hard limit, so, it is possible to overshoot it! |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | Axis of the AMC to get the soft limit status from |
| Out | errNo | int32 Error number if one occured, 0 in case of no error |
| limit | double Lower boundary in pm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.amcids.getLowerSoftLimit |
| params: [axis] |
| Result: [errNo, limit] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_amcids\_getLowerSoftLimit**(int deviceHandle, int axis, double\* limit) |

|  |
| --- |
| **Python** |

|  |
| --- |
| limit = **[dev].amcids.getLowerSoftLimit**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [limit] = **amcids\_getLowerSoftLimit**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| double value = [Device].**Amcids\_GetLowerSoftLimit**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getLowerSoftLimit.vi |

|  |
| --- |
| **getSoftLimitEnabled** Gets whether the soft limit protection is enabled.  This protection is needed if the IDS working range is smaller than the positioners travel range.  It is no hard limit, so, it is possible to overshoot it! |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | Axis of the AMC to get the soft limit status from |
| Out | errNo | int32 Error number if one occured, 0 in case of no error |
| enabled | boolean True, if the soft limit should be enabled on this axis |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.amcids.getSoftLimitEnabled |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_amcids\_getSoftLimitEnabled**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].amcids.getSoftLimitEnabled**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **amcids\_getSoftLimitEnabled**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Amcids\_GetSoftLimitEnabled**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getSoftLimitEnabled.vi |

|  |
| --- |
| **getSoftLimitReached** Gets whether the current position is out of the soft limit boundaries.  This protection is needed if the IDS working range is smaller than the positioners travel range.  It is no hard limit, so, it is possible to overshoot it! |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | Axis of the AMC to get the soft limit status from |
| Out | errNo | int32 Error number if one occured, 0 in case of no error |
| enabled | boolean True, if the position is not within the boundaries |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.amcids.getSoftLimitReached |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_amcids\_getSoftLimitReached**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].amcids.getSoftLimitReached**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **amcids\_getSoftLimitReached**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Amcids\_GetSoftLimitReached**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getSoftLimitReached.vi |

|  |
| --- |
| **getUpperSoftLimit** Gets the upper lower boundary of the soft limit protection.  This protection is needed if the IDS working range is smaller than the positioners travel range.  It is no hard limit, so, it is possible to overshoot it! |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | Axis of the AMC to get the soft limit status from |
| Out | errNo | int32 Error number if one occured, 0 in case of no error |
| limit | double Upper boundary in pm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.amcids.getUpperSoftLimit |
| params: [axis] |
| Result: [errNo, limit] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_amcids\_getUpperSoftLimit**(int deviceHandle, int axis, double\* limit) |

|  |
| --- |
| **Python** |

|  |
| --- |
| limit = **[dev].amcids.getUpperSoftLimit**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [limit] = **amcids\_getUpperSoftLimit**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| double value = [Device].**Amcids\_GetUpperSoftLimit**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getUpperSoftLimit.vi |

|  |
| --- |
| **resetIdsAxis** Resets the position value to zero of a specific measurement axis.  Use this for positioners with an IDS as sensor.  This method does not work for NUM and RES sensors. Use com.attocube.amc.control.resetAxis instead. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | Axis of the IDS to reset the position |
| Out | errNo | int32 Error number if one occured, 0 in case of no error |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.amcids.resetIdsAxis |
| params: [axis] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_amcids\_resetIdsAxis**(int deviceHandle, int axis) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].amcids.resetIdsAxis**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **amcids\_resetIdsAxis**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Amcids\_ResetIdsAxis**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| resetIdsAxis.vi |

|  |
| --- |
| **setLowerSoftLimit** Sets the lower boundary of the soft limit protection in pm.  This protection is needed if the IDS working range is smaller than the positioners travel range.  It is no hard limit, so, it is possible to overshoot it! |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | Axis of the AMC where the soft limit should be changed |
| limit | Lower boundary in pm |
| Out | errNo | int32 Error number if one occured, 0 in case of no error |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.amcids.setLowerSoftLimit |
| params: [axis, limit] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_amcids\_setLowerSoftLimit**(int deviceHandle, int axis, double limit) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].amcids.setLowerSoftLimit**(axis, limit) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **amcids\_setLowerSoftLimit**(axis, limit) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Amcids\_SetLowerSoftLimit**(int axis, double limit) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setLowerSoftLimit.vi |

|  |
| --- |
| **setSoftLimitEnabled** Enables/disables the soft limit protection.  This protection is needed if the IDS working range is smaller than the positioners travel range.  It is no hard limit, so, it is possible to overshoot it! |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | Axis of the AMC where the soft limit should be changed |
| enabled | True, if the soft limit should be enabled on this axis |
| Out | errNo | int32 Error number if one occured, 0 in case of no error |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.amcids.setSoftLimitEnabled |
| params: [axis, enabled] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_amcids\_setSoftLimitEnabled**(int deviceHandle, int axis, bool enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].amcids.setSoftLimitEnabled**(axis, enabled) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **amcids\_setSoftLimitEnabled**(axis, enabled) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Amcids\_SetSoftLimitEnabled**(int axis, bool enabled) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setSoftLimitEnabled.vi |

|  |
| --- |
| **setUpperSoftLimit** Sets the upper boundary of the soft limit protection in pm.  This protection is needed if the IDS working range is smaller than the positioners travel range.  It is no hard limit, so, it is possible to overshoot it! |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | Axis of the AMC where the soft limit should be changed |
| limit | Upper boundary in pm |
| Out | errNo | int32 Error number if one occured, 0 in case of no error |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.amcids.setUpperSoftLimit |
| params: [axis, limit] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_amcids\_setUpperSoftLimit**(int deviceHandle, int axis, double limit) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].amcids.setUpperSoftLimit**(axis, limit) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **amcids\_setUpperSoftLimit**(axis, limit) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Amcids\_SetUpperSoftLimit**(int axis, double limit) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setUpperSoftLimit.vi |

## Control

|  |
| --- |
| **MultiAxisPositioning** Simultaneously set 3 axes positions  and get positions to minimize network latency |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | set1 | axis1 otherwise pos1 target is ignored |
| set2 | axis2 otherwise pos2 target is ignored |
| set3 | axis3 otherwise pos3 target is ignored |
| target1 | target position of axis 1 |
| target2 | target position of axis 2 |
| target3 | target position of axis 3 |
| Out | errNo | errNo |
| ref1 | Status of axis 1 |
| ref2 | Status of axis 2 |
| ref3 | Status of axis 3 |
| refpos1 | reference Position of axis 1 |
| refpos2 | reference Position of axis 2 |
| refpos3 | reference Position of axis 3 |
| pos1 | position of axis 1 |
| pos2 | position of axis 2 |
| pos3 | position of axis 3 |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.MultiAxisPositioning |
| params: [set1, set2, set3, target1, target2, target3] |
| Result: [errNo, ref1, ref2, ref3, refpos1, refpos2, refpos3, pos1, pos2, pos3] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_MultiAxisPositioning**(int deviceHandle, bool set1, bool set2, bool set3, int target1, int target2, int target3, bool\* ref1, bool\* ref2, bool\* ref3, int\* refpos1, int\* refpos2, int\* refpos3, int\* pos1, int\* pos2, int\* pos3) |

|  |
| --- |
| **Python** |

|  |
| --- |
| ref1, ref2, ref3, refpos1, refpos2, refpos3, pos1, pos2, pos3 = **[dev].control.MultiAxisPositioning**(set1, set2, set3, target1, target2, target3) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [ref1, ref2, ref3, refpos1, refpos2, refpos3, pos1, pos2, pos3] = **control\_MultiAxisPositioning**(set1, set2, set3, target1, target2, target3) |

|  |
| --- |
| **C#** |

|  |
| --- |
| Tuple<bool,bool,bool,int,int,int, Tuple<int,int,int>> value = [Device].**Control\_MultiAxisPositioning**(bool set1, bool set2, bool set3, int target1, int target2, int target3) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| MultiAxisPositioning.vi |

|  |
| --- |
| **getActorName** This function gets the name of the positioner of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| actor\_name | actor\_name |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getActorName |
| params: [axis] |
| Result: [errNo, actor\_name] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getActorName**(int deviceHandle, int axis, char\* actor\_name, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| actor\_name = **[dev].control.getActorName**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [actor\_name] = **control\_getActorName**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Control\_GetActorName**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getActorName.vi |

|  |
| --- |
| **getActorParametersActorName** Control the actors parameter: actor name |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| actorname | actorname |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getActorParametersActorName |
| params: [axis] |
| Result: [errNo, actorname] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getActorParametersActorName**(int deviceHandle, int axis, char\* actorname, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| actorname = **[dev].control.getActorParametersActorName**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [actorname] = **control\_getActorParametersActorName**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Control\_GetActorParametersActorName**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getActorParametersActorName.vi |

|  |
| --- |
| **getActorSensitivity** Get the setting for the actor parameter sensitivity |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| sensitivity | sensitivity |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getActorSensitivity |
| params: [axis] |
| Result: [errNo, sensitivity] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getActorSensitivity**(int deviceHandle, int axis, int\* sensitivity) |

|  |
| --- |
| **Python** |

|  |
| --- |
| sensitivity = **[dev].control.getActorSensitivity**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [sensitivity] = **control\_getActorSensitivity**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetActorSensitivity**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getActorSensitivity.vi |

|  |
| --- |
| **getActorType** This function gets the type of the positioner of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| actor\_type | 0: linear, 1: rotator, 2: goniometer |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getActorType |
| params: [axis] |
| Result: [errNo, actor\_type] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getActorType**(int deviceHandle, int axis, int\* actor\_type) |

|  |
| --- |
| **Python** |

|  |
| --- |
| actor\_type = **[dev].control.getActorType**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [actor\_type] = **control\_getActorType**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetActorType**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getActorType.vi |

|  |
| --- |
| **getAutoMeasure** This function returns if the automeasurement on axis enable is enabled |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enable | true: enable automeasurement, false: disable automeasurement |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getAutoMeasure |
| params: [axis] |
| Result: [errNo, enable] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getAutoMeasure**(int deviceHandle, int axis, bool\* enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enable = **[dev].control.getAutoMeasure**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enable] = **control\_getAutoMeasure**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Control\_GetAutoMeasure**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getAutoMeasure.vi |

|  |
| --- |
| **getControlAmplitude** This function gets the amplitude of the actuator signal of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| amplitude | in mV |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getControlAmplitude |
| params: [axis] |
| Result: [errNo, amplitude] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getControlAmplitude**(int deviceHandle, int axis, int\* amplitude) |

|  |
| --- |
| **Python** |

|  |
| --- |
| amplitude = **[dev].control.getControlAmplitude**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [amplitude] = **control\_getControlAmplitude**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetControlAmplitude**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlAmplitude.vi |

|  |
| --- |
| **getControlAutoReset** This function resets the position every time the reference position is detected. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enabled | boolean |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getControlAutoReset |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getControlAutoReset**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].control.getControlAutoReset**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **control\_getControlAutoReset**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Control\_GetControlAutoReset**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlAutoReset.vi |

|  |
| --- |
| **getControlFixOutputVoltage** This function gets the DC level output of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| amplitude\_mv | in mV |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getControlFixOutputVoltage |
| params: [axis] |
| Result: [errNo, amplitude\_mv] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getControlFixOutputVoltage**(int deviceHandle, int axis, int\* amplitude\_mv) |

|  |
| --- |
| **Python** |

|  |
| --- |
| amplitude\_mv = **[dev].control.getControlFixOutputVoltage**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [amplitude\_mv] = **control\_getControlFixOutputVoltage**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetControlFixOutputVoltage**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlFixOutputVoltage.vi |

|  |
| --- |
| **getControlFrequency** This function gets the frequency of the actuator signal of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| frequency | in mHz |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getControlFrequency |
| params: [axis] |
| Result: [errNo, frequency] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getControlFrequency**(int deviceHandle, int axis, int\* frequency) |

|  |
| --- |
| **Python** |

|  |
| --- |
| frequency = **[dev].control.getControlFrequency**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [frequency] = **control\_getControlFrequency**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetControlFrequency**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlFrequency.vi |

|  |
| --- |
| **getControlMove** This function gets the approach of the selected axis’ positioner to the target position. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enable | boolean true: closed loop control enabled, false: closed loop control disabled |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getControlMove |
| params: [axis] |
| Result: [errNo, enable] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getControlMove**(int deviceHandle, int axis, bool\* enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enable = **[dev].control.getControlMove**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enable] = **control\_getControlMove**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Control\_GetControlMove**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlMove.vi |

|  |
| --- |
| **getControlOutput** This function gets the status of the output relays of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enabled | power status (true = enabled,false = disabled) |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getControlOutput |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getControlOutput**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].control.getControlOutput**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **control\_getControlOutput**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Control\_GetControlOutput**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlOutput.vi |

|  |
| --- |
| **getControlReferenceAutoUpdate** This function gets the status of whether the reference position is updated when the reference mark is hit.  When this function is disabled, the reference marking will be considered only the first time and after then ignored. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enabled | boolen |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getControlReferenceAutoUpdate |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getControlReferenceAutoUpdate**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].control.getControlReferenceAutoUpdate**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **control\_getControlReferenceAutoUpdate**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Control\_GetControlReferenceAutoUpdate**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlReferenceAutoUpdate.vi |

|  |
| --- |
| **getControlTargetRange** This function gets the range around the target position in which the flag "In Target Range" becomes active. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| targetrange | in nm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getControlTargetRange |
| params: [axis] |
| Result: [errNo, targetrange] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getControlTargetRange**(int deviceHandle, int axis, int\* targetrange) |

|  |
| --- |
| **Python** |

|  |
| --- |
| targetrange = **[dev].control.getControlTargetRange**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [targetrange] = **control\_getControlTargetRange**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetControlTargetRange**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlTargetRange.vi |

|  |
| --- |
| **getCrosstalkThreshold** This function gets the threshold range and slip phase time which is used while moving another axis |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| range | in pm |
| time | after slip phase which is waited until the controller is acting again in microseconds |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getCrosstalkThreshold |
| params: [axis] |
| Result: [errNo, range, time] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getCrosstalkThreshold**(int deviceHandle, int axis, int\* range, int\* time) |

|  |
| --- |
| **Python** |

|  |
| --- |
| range, time = **[dev].control.getCrosstalkThreshold**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [range, time] = **control\_getCrosstalkThreshold**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| Tuple<int,int> value = [Device].**Control\_GetCrosstalkThreshold**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getCrosstalkThreshold.vi |

|  |
| --- |
| **getCurrentOutputVoltage** This function gets the current Voltage which is applied to the Piezo |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| amplitude | in mV |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getCurrentOutputVoltage |
| params: [axis] |
| Result: [errNo, amplitude] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getCurrentOutputVoltage**(int deviceHandle, int axis, int\* amplitude) |

|  |
| --- |
| **Python** |

|  |
| --- |
| amplitude = **[dev].control.getCurrentOutputVoltage**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [amplitude] = **control\_getCurrentOutputVoltage**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetCurrentOutputVoltage**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getCurrentOutputVoltage.vi |

|  |
| --- |
| **getExternalSensor** This function gets whether the sensor source of closed loop is IDS  It is only available when the feature AMC/IDS closed loop has been activated |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enabled | enabled |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getExternalSensor |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getExternalSensor**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].control.getExternalSensor**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **control\_getExternalSensor**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Control\_GetExternalSensor**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getExternalSensor.vi |

|  |
| --- |
| **getFinePositioningRange** This function gets the fine positioning DC-range |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| range | in nm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getFinePositioningRange |
| params: [axis] |
| Result: [errNo, range] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getFinePositioningRange**(int deviceHandle, int axis, int\* range) |

|  |
| --- |
| **Python** |

|  |
| --- |
| range = **[dev].control.getFinePositioningRange**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [range] = **control\_getFinePositioningRange**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetFinePositioningRange**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getFinePositioningRange.vi |

|  |
| --- |
| **getFinePositioningSlewRate** This function gets the fine positioning slew rate |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| slewrate | [0|1|2|3] |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getFinePositioningSlewRate |
| params: [axis] |
| Result: [errNo, slewrate] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getFinePositioningSlewRate**(int deviceHandle, int axis, int\* slewrate) |

|  |
| --- |
| **Python** |

|  |
| --- |
| slewrate = **[dev].control.getFinePositioningSlewRate**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [slewrate] = **control\_getFinePositioningSlewRate**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetFinePositioningSlewRate**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getFinePositioningSlewRate.vi |

|  |
| --- |
| **getMotionControlThreshold** This function gets the threshold range within the closed-loop controlled movement stops to regulate. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| threshold | in pm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getMotionControlThreshold |
| params: [axis] |
| Result: [errNo, threshold] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getMotionControlThreshold**(int deviceHandle, int axis, int\* threshold) |

|  |
| --- |
| **Python** |

|  |
| --- |
| threshold = **[dev].control.getMotionControlThreshold**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [threshold] = **control\_getMotionControlThreshold**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetMotionControlThreshold**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getMotionControlThreshold.vi |

|  |
| --- |
| **getPositionsAndVoltages** Simultaneously get 3 axes positions as well as the DC offset  to maximize sampling rate over network |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| pos1 | position of axis 1 |
| pos2 | position of axis 2 |
| pos3 | position of axis 3 |
| val1 | dc voltage of of axis 1 in mV |
| val2 | dc voltage of of axis 2 in mV |
| val3 | dc voltage of of axis 3 in mV |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getPositionsAndVoltages |
| params: [] |
| Result: [errNo, pos1, pos2, pos3, val1, val2, val3] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getPositionsAndVoltages**(int deviceHandle, int\* pos1, int\* pos2, int\* pos3, int\* val1, int\* val2, int\* val3) |

|  |
| --- |
| **Python** |

|  |
| --- |
| pos1, pos2, pos3, val1, val2, val3 = **[dev].control.getPositionsAndVoltages**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [pos1, pos2, pos3, val1, val2, val3] = **control\_getPositionsAndVoltages**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| Tuple<int,int,int,int,int,int> value = [Device].**Control\_GetPositionsAndVoltages**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getPositionsAndVoltages.vi |

|  |
| --- |
| **getReferencePosition** This function gets the reference position of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| position | position: For linear type actors the position is defined in nm for goniometer an rotator type actors it is µ°. |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getReferencePosition |
| params: [axis] |
| Result: [errNo, position] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getReferencePosition**(int deviceHandle, int axis, int\* position) |

|  |
| --- |
| **Python** |

|  |
| --- |
| position = **[dev].control.getReferencePosition**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [position] = **control\_getReferencePosition**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_GetReferencePosition**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getReferencePosition.vi |

|  |
| --- |
| **getSensorDirection** This function gets whether the IDS sensor source of closed loop is inverted  It is only available when the feature AMC/IDS closed loop has been activated |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| inverted | boolen |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getSensorDirection |
| params: [axis] |
| Result: [errNo, inverted] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getSensorDirection**(int deviceHandle, int axis, bool\* inverted) |

|  |
| --- |
| **Python** |

|  |
| --- |
| inverted = **[dev].control.getSensorDirection**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [inverted] = **control\_getSensorDirection**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Control\_GetSensorDirection**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getSensorDirection.vi |

|  |
| --- |
| **getSensorEnabled** Get sensot power supply status |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| value | true if enabled, false otherwise |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getSensorEnabled |
| params: [axis] |
| Result: [errNo, value] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getSensorEnabled**(int deviceHandle, int axis, bool\* value) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value = **[dev].control.getSensorEnabled**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value] = **control\_getSensorEnabled**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Control\_GetSensorEnabled**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getSensorEnabled.vi |

|  |
| --- |
| **getStatusMovingAllAxes** Get Status of all axes, see getStatusMoving for coding of the values |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| moving1 | status of axis 1 |
| moving2 | status of axis 2 |
| moving3 | status of axis 3 |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.getStatusMovingAllAxes |
| params: [] |
| Result: [errNo, moving1, moving2, moving3] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_getStatusMovingAllAxes**(int deviceHandle, int\* moving1, int\* moving2, int\* moving3) |

|  |
| --- |
| **Python** |

|  |
| --- |
| moving1, moving2, moving3 = **[dev].control.getStatusMovingAllAxes**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [moving1, moving2, moving3] = **control\_getStatusMovingAllAxes**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| Tuple<int,int,int> value = [Device].**Control\_GetStatusMovingAllAxes**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getStatusMovingAllAxes.vi |

|  |
| --- |
| **searchReferencePosition** This function searches for the reference position of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.searchReferencePosition |
| params: [axis] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_searchReferencePosition**(int deviceHandle, int axis) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.searchReferencePosition**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_searchReferencePosition**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SearchReferencePosition**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| searchReferencePosition.vi |

|  |
| --- |
| **setActorParametersByName** This function sets the name for the positioner on the selected axis. The possible names can be retrieved by executing getPositionersList |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| actorname | name of the actor |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setActorParametersByName |
| params: [axis, actorname] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setActorParametersByName**(int deviceHandle, int axis, const char\* actorname) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setActorParametersByName**(axis, actorname) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setActorParametersByName**(axis, actorname) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetActorParametersByName**(int axis, string actorname) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setActorParametersByName.vi |

|  |
| --- |
| **setActorParametersJson** Select and override a positioner out of the Current default list only override given parameters set others default |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| json\_dict | dict with override params |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setActorParametersJson |
| params: [axis, json\_dict] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setActorParametersJson**(int deviceHandle, int axis, const char\* json\_dict) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setActorParametersJson**(axis, json\_dict) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setActorParametersJson**(axis, json\_dict) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetActorParametersJson**(int axis, string json\_dict) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setActorParametersJson.vi |

|  |
| --- |
| **setActorSensitivity** Control the actor parameter closed loop sensitivity |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| sensitivity |  |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setActorSensitivity |
| params: [axis, sensitivity] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setActorSensitivity**(int deviceHandle, int axis, int sensitivity) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setActorSensitivity**(axis, sensitivity) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setActorSensitivity**(axis, sensitivity) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetActorSensitivity**(int axis, int sensitivity) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setActorSensitivity.vi |

|  |
| --- |
| **setAutoMeasure** This function enables/disables the automatic C/R measurement on axis enable |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | true: enable automeasurement, false: disable automeasurement |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setAutoMeasure |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setAutoMeasure**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setAutoMeasure**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setAutoMeasure**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetAutoMeasure**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setAutoMeasure.vi |

|  |
| --- |
| **setControlAmplitude** This function sets the amplitude of the actuator signal of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| amplitude | in mV |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setControlAmplitude |
| params: [axis, amplitude] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setControlAmplitude**(int deviceHandle, int axis, int amplitude) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setControlAmplitude**(axis, amplitude) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setControlAmplitude**(axis, amplitude) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetControlAmplitude**(int axis, int amplitude) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlAmplitude.vi |

|  |
| --- |
| **setControlAutoReset** This function resets the position every time the reference position is detected. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | boolean |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setControlAutoReset |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setControlAutoReset**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setControlAutoReset**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setControlAutoReset**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetControlAutoReset**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlAutoReset.vi |

|  |
| --- |
| **setControlFixOutputVoltage** This function sets the DC level output of the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| amplitude\_mv | in mV |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setControlFixOutputVoltage |
| params: [axis, amplitude\_mv] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setControlFixOutputVoltage**(int deviceHandle, int axis, int amplitude\_mv) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setControlFixOutputVoltage**(axis, amplitude\_mv) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setControlFixOutputVoltage**(axis, amplitude\_mv) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetControlFixOutputVoltage**(int axis, int amplitude\_mv) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlFixOutputVoltage.vi |

|  |
| --- |
| **setControlFrequency** This function sets the frequency of the actuator signal of the selected axis.  Note: Approximate the slewrate of the motion controller according to Input Frequency |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| frequency | in mHz |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setControlFrequency |
| params: [axis, frequency] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setControlFrequency**(int deviceHandle, int axis, int frequency) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setControlFrequency**(axis, frequency) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setControlFrequency**(axis, frequency) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetControlFrequency**(int axis, int frequency) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlFrequency.vi |

|  |
| --- |
| **setControlMove** This function sets the approach of the selected axis’ positioner to the target position. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | boolean true: eanble the approach , false: disable the approach |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setControlMove |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setControlMove**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setControlMove**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setControlMove**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetControlMove**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlMove.vi |

|  |
| --- |
| **setControlOutput** This function sets the status of the output relays of the selected axis.  Enable only if cable is connected and FlyBack is enabled  use a PWM startup of 1sec |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | true: enable drives, false: disable drives |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setControlOutput |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setControlOutput**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setControlOutput**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setControlOutput**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetControlOutput**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlOutput.vi |

|  |
| --- |
| **setControlReferenceAutoUpdate** This function sets the status of whether the reference position is updated when the reference mark is hit.  When this function is disabled, the reference marking will be considered only the first time and after then ignored. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | boolean |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setControlReferenceAutoUpdate |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setControlReferenceAutoUpdate**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setControlReferenceAutoUpdate**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setControlReferenceAutoUpdate**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetControlReferenceAutoUpdate**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlReferenceAutoUpdate.vi |

|  |
| --- |
| **setControlTargetRange** This function sets the range around the target position in which the flag "In Target Range" (see VIII.7.a) becomes active. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| range | in nm |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setControlTargetRange |
| params: [axis, range] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setControlTargetRange**(int deviceHandle, int axis, int range) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setControlTargetRange**(axis, range) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setControlTargetRange**(axis, range) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetControlTargetRange**(int axis, int range) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlTargetRange.vi |

|  |
| --- |
| **setCrosstalkThreshold** This function sets the threshold range and slip phase time which is used while moving another axis |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| threshold | in pm |
| slipphasetime | time after slip phase which is waited until the controller is acting again in microseconds |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setCrosstalkThreshold |
| params: [axis, threshold, slipphasetime] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setCrosstalkThreshold**(int deviceHandle, int axis, int threshold, int slipphasetime) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setCrosstalkThreshold**(axis, threshold, slipphasetime) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setCrosstalkThreshold**(axis, threshold, slipphasetime) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetCrosstalkThreshold**(int axis, int threshold, int slipphasetime) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setCrosstalkThreshold.vi |

|  |
| --- |
| **setExternalSensor** This function sets the sensor source of closed loop to the IDS when enabled. Otherwise the normal AMC Sensor depending on the configuration (e.g. NUM or RES) is used  It is only available when the feature AMC/IDS closed loop has been activated |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enabled |  |
| Out | warningNo | Warning code, can be converted into a string using the errorNumberToString function |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setExternalSensor |
| params: [axis, enabled] |
| Result: [warningNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setExternalSensor**(int deviceHandle, int axis, bool enabled, int\* warningNo) |

|  |
| --- |
| **Python** |

|  |
| --- |
| warningNo = **[dev].control.setExternalSensor**(axis, enabled) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [warningNo] = **control\_setExternalSensor**(axis, enabled) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Control\_SetExternalSensor**(int axis, bool enabled) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setExternalSensor.vi |

|  |
| --- |
| **setFinePositioningRange** This function sets the fine positioning DC-range |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| range | in nm |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setFinePositioningRange |
| params: [axis, range] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setFinePositioningRange**(int deviceHandle, int axis, int range) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setFinePositioningRange**(axis, range) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setFinePositioningRange**(axis, range) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetFinePositioningRange**(int axis, int range) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setFinePositioningRange.vi |

|  |
| --- |
| **setFinePositioningSlewRate** This function sets the fine positioning slew rate |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| slewrate | [0|1|2|3] |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setFinePositioningSlewRate |
| params: [axis, slewrate] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setFinePositioningSlewRate**(int deviceHandle, int axis, int slewrate) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setFinePositioningSlewRate**(axis, slewrate) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setFinePositioningSlewRate**(axis, slewrate) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetFinePositioningSlewRate**(int axis, int slewrate) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setFinePositioningSlewRate.vi |

|  |
| --- |
| **setMotionControlThreshold** This function sets the threshold range within the closed-loop controlled movement stops to regulate. Default depends on connected sensor type |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| threshold | in pm |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setMotionControlThreshold |
| params: [axis, threshold] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setMotionControlThreshold**(int deviceHandle, int axis, int threshold) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setMotionControlThreshold**(axis, threshold) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setMotionControlThreshold**(axis, threshold) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetMotionControlThreshold**(int axis, int threshold) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setMotionControlThreshold.vi |

|  |
| --- |
| **setReset** This function resets the actual position of the selected axis given by the NUM sensor to zero and marks the reference position as invalid.  It does not work for RES positioners and positions read by IDS.  For IDS, use com.attocube.ids.displacement.resetAxis() or com.attocube.amc.amcids.resetIdsAxis() instead. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setReset |
| params: [axis] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setReset**(int deviceHandle, int axis) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setReset**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setReset**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetReset**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setReset.vi |

|  |
| --- |
| **setSensorDirection** This function sets the IDS sensor source of closed loop to inverted when true.  It is only available when the feature AMC/IDS closed loop has been activated |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| inverted |  |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setSensorDirection |
| params: [axis, inverted] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setSensorDirection**(int deviceHandle, int axis, bool inverted) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setSensorDirection**(axis, inverted) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setSensorDirection**(axis, inverted) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetSensorDirection**(int axis, bool inverted) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setSensorDirection.vi |

|  |
| --- |
| **setSensorEnabled** Set sensor power supply status, can be switched off to save heat generated by sensor [NUM or RES]  Positions retrieved will be invalid when activating this, so closed-loop control should be switched off beforehand |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| value | true if enabled, false otherwise |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.control.setSensorEnabled |
| params: [axis, value] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_control\_setSensorEnabled**(int deviceHandle, int axis, bool value) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].control.setSensorEnabled**(axis, value) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **control\_setSensorEnabled**(axis, value) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Control\_SetSensorEnabled**(int axis, bool value) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setSensorEnabled.vi |

## Description

|  |
| --- |
| **checkChassisNbr** Get Chassis and Slot Number, only works when AMC is within a Rack |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| slotNbr | slotNbr |
| chassisNbr | chassisNbr |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.description.checkChassisNbr |
| params: [] |
| Result: [errNo, slotNbr, chassisNbr] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_description\_checkChassisNbr**(int deviceHandle, int\* slotNbr, int\* chassisNbr) |

|  |
| --- |
| **Python** |

|  |
| --- |
| slotNbr, chassisNbr = **[dev].description.checkChassisNbr**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [slotNbr, chassisNbr] = **description\_checkChassisNbr**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| Tuple<int,int> value = [Device].**Description\_CheckChassisNbr**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| checkChassisNbr.vi |

|  |
| --- |
| **getDeviceType** This function gets the device type based on its EEPROM configuration. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| devicetype | Device name (AMC100, AMC150, AMC300) with attached feature ( AMC100\\NUM, AMC100\\NUM\\PRO) |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.description.getDeviceType |
| params: [] |
| Result: [errNo, devicetype] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_description\_getDeviceType**(int deviceHandle, char\* devicetype, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| devicetype = **[dev].description.getDeviceType**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [devicetype] = **description\_getDeviceType**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Description\_GetDeviceType**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getDeviceType.vi |

|  |
| --- |
| **getFeaturesActivated** Get the activated features and return as a string |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| features | activated on device concatenated by comma e.g. Closed loop Operation, Pro, Wireless Controller, IO |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.description.getFeaturesActivated |
| params: [] |
| Result: [errNo, features] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_description\_getFeaturesActivated**(int deviceHandle, char\* features, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| features = **[dev].description.getFeaturesActivated**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [features] = **description\_getFeaturesActivated**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Description\_GetFeaturesActivated**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getFeaturesActivated.vi |

|  |
| --- |
| **getPositionersList** This function reads the actor names that can be connected to the device. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| PositionersList | PositionersList |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.description.getPositionersList |
| params: [] |
| Result: [errNo, PositionersList] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_description\_getPositionersList**(int deviceHandle, char\* PositionersList, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| PositionersList = **[dev].description.getPositionersList**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [PositionersList] = **description\_getPositionersList**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Description\_GetPositionersList**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getPositionersList.vi |

## Diagnostic

|  |
| --- |
| **getDiagnosticPower** Returns the current power consumption |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| power | power |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.diagnostic.getDiagnosticPower |
| params: [axis] |
| Result: [errNo, power] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_diagnostic\_getDiagnosticPower**(int deviceHandle, int axis, int\* power) |

|  |
| --- |
| **Python** |

|  |
| --- |
| power = **[dev].diagnostic.getDiagnosticPower**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [power] = **diagnostic\_getDiagnosticPower**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Diagnostic\_GetDiagnosticPower**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getDiagnosticPower.vi |

|  |
| --- |
| **getDiagnosticResults** Returns the results of the last diagnostic run and an error, if there was no run, it is currently running or the run failed |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| capacity | in nF |
| resistance | in Ohm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.diagnostic.getDiagnosticResults |
| params: [axis] |
| Result: [errNo, capacity, resistance] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_diagnostic\_getDiagnosticResults**(int deviceHandle, int axis, int\* capacity, int\* resistance) |

|  |
| --- |
| **Python** |

|  |
| --- |
| capacity, resistance = **[dev].diagnostic.getDiagnosticResults**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [capacity, resistance] = **diagnostic\_getDiagnosticResults**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| Tuple<int,int> value = [Device].**Diagnostic\_GetDiagnosticResults**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getDiagnosticResults.vi |

|  |
| --- |
| **getDiagnosticStepSize** Performs 10 steps in forward and backward and calculates the average step size in both directions on a specific axis |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| stepsize\_fwd | stepsize\_fwd |
| stepsize\_bwd | stepsize\_bwd |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.diagnostic.getDiagnosticStepSize |
| params: [axis] |
| Result: [errNo, stepsize\_fwd, stepsize\_bwd] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_diagnostic\_getDiagnosticStepSize**(int deviceHandle, int axis, int\* stepsize\_fwd, int\* stepsize\_bwd) |

|  |
| --- |
| **Python** |

|  |
| --- |
| stepsize\_fwd, stepsize\_bwd = **[dev].diagnostic.getDiagnosticStepSize**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [stepsize\_fwd, stepsize\_bwd] = **diagnostic\_getDiagnosticStepSize**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| Tuple<int,int> value = [Device].**Diagnostic\_GetDiagnosticStepSize**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getDiagnosticStepSize.vi |

|  |
| --- |
| **getDiagnosticTemperature** Returns the current axis temperature |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| temperature | temperature |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.diagnostic.getDiagnosticTemperature |
| params: [axis] |
| Result: [errNo, temperature] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_diagnostic\_getDiagnosticTemperature**(int deviceHandle, int axis, int\* temperature) |

|  |
| --- |
| **Python** |

|  |
| --- |
| temperature = **[dev].diagnostic.getDiagnosticTemperature**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [temperature] = **diagnostic\_getDiagnosticTemperature**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Diagnostic\_GetDiagnosticTemperature**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getDiagnosticTemperature.vi |

|  |
| --- |
| **startDiagnostic** Start the diagnosis procedure for the given axis |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.diagnostic.startDiagnostic |
| params: [axis] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_diagnostic\_startDiagnostic**(int deviceHandle, int axis) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].diagnostic.startDiagnostic**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **diagnostic\_startDiagnostic**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Diagnostic\_StartDiagnostic**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| startDiagnostic.vi |

## Move

|  |
| --- |
| **getControlContinuousBkwd** This function gets the axis’ movement status in backward direction. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enabled | true if movement backward is active , false otherwise |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.getControlContinuousBkwd |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_getControlContinuousBkwd**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].move.getControlContinuousBkwd**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **move\_getControlContinuousBkwd**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Move\_GetControlContinuousBkwd**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlContinuousBkwd.vi |

|  |
| --- |
| **getControlContinuousFwd** This function gets the axis’ movement status in positive direction. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enabled | true if movement Fwd is active, false otherwise |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.getControlContinuousFwd |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_getControlContinuousFwd**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].move.getControlContinuousFwd**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **move\_getControlContinuousFwd**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Move\_GetControlContinuousFwd**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlContinuousFwd.vi |

|  |
| --- |
| **getControlEotOutputDeactive** This function gets the output applied to the selected axis on the end of travel. /PRO feature. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enabled | If true, the output of the axis will be deactivated on positive EOT detection. |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.getControlEotOutputDeactive |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_getControlEotOutputDeactive**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].move.getControlEotOutputDeactive**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **move\_getControlEotOutputDeactive**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Move\_GetControlEotOutputDeactive**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlEotOutputDeactive.vi |

|  |
| --- |
| **getControlTargetPosition** This function gets the target position for the movement on the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| position | defined in nm for goniometer an rotator type actors it is µ°. |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.getControlTargetPosition |
| params: [axis] |
| Result: [errNo, position] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_getControlTargetPosition**(int deviceHandle, int axis, double\* position) |

|  |
| --- |
| **Python** |

|  |
| --- |
| position = **[dev].move.getControlTargetPosition**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [position] = **move\_getControlTargetPosition**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| double value = [Device].**Move\_GetControlTargetPosition**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlTargetPosition.vi |

|  |
| --- |
| **getGroundAxis** Checks if the axis piezo drive is actively grounded  only in AMC300 |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | montion controler axis [0|1|2] |
| Out | errNo | 0 or error |
| grounded | true or false |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.getGroundAxis |
| params: [axis] |
| Result: [errNo, grounded] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_getGroundAxis**(int deviceHandle, int axis, bool\* grounded) |

|  |
| --- |
| **Python** |

|  |
| --- |
| grounded = **[dev].move.getGroundAxis**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [grounded] = **move\_getGroundAxis**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Move\_GetGroundAxis**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getGroundAxis.vi |

|  |
| --- |
| **getGroundAxisAutoOnTarget** Pull axis piezo drive to GND if positioner is in ground target range  only in AMC300 |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | montion controler axis [0|1|2] |
| Out | errNo | 0 or error |
| value | true or false |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.getGroundAxisAutoOnTarget |
| params: [axis] |
| Result: [errNo, value] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_getGroundAxisAutoOnTarget**(int deviceHandle, int axis, bool\* value) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value = **[dev].move.getGroundAxisAutoOnTarget**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value] = **move\_getGroundAxisAutoOnTarget**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Move\_GetGroundAxisAutoOnTarget**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getGroundAxisAutoOnTarget.vi |

|  |
| --- |
| **getGroundTargetRange** Retrieves the range around the target position in which the auto grounding becomes active.  only in AMC300 |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| targetrange | in nm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.getGroundTargetRange |
| params: [axis] |
| Result: [errNo, targetrange] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_getGroundTargetRange**(int deviceHandle, int axis, int\* targetrange) |

|  |
| --- |
| **Python** |

|  |
| --- |
| targetrange = **[dev].move.getGroundTargetRange**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [targetrange] = **move\_getGroundTargetRange**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Move\_GetGroundTargetRange**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getGroundTargetRange.vi |

|  |
| --- |
| **getNSteps** This function gets the number of Steps in desired direction. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| nbrstep | nbrstep |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.getNSteps |
| params: [axis] |
| Result: [errNo, nbrstep] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_getNSteps**(int deviceHandle, int axis, int\* nbrstep) |

|  |
| --- |
| **Python** |

|  |
| --- |
| nbrstep = **[dev].move.getNSteps**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [nbrstep] = **move\_getNSteps**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Move\_GetNSteps**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getNSteps.vi |

|  |
| --- |
| **getPosition** This function gets the current position of the positioner on the selected axis.  The axis on the web application are indexed from 1 to 3 |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| position | defined in nm for goniometer an rotator type actors it is µ°. |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.getPosition |
| params: [axis] |
| Result: [errNo, position] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_getPosition**(int deviceHandle, int axis, double\* position) |

|  |
| --- |
| **Python** |

|  |
| --- |
| position = **[dev].move.getPosition**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [position] = **move\_getPosition**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| double value = [Device].**Move\_GetPosition**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getPosition.vi |

|  |
| --- |
| **moveReference** This function starts an approach to the reference position. A running motion command is aborted; closed loop moving is switched on. Requires a valid reference position. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.moveReference |
| params: [axis] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_moveReference**(int deviceHandle, int axis) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.moveReference**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_moveReference**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_MoveReference**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| moveReference.vi |

|  |
| --- |
| **performNSteps** Perform the OL command for N steps |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| backward | Selects the desired direction. False triggers a forward step, true a backward step |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.performNSteps |
| params: [axis, backward] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_performNSteps**(int deviceHandle, int axis, bool backward) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.performNSteps**(axis, backward) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_performNSteps**(axis, backward) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_PerformNSteps**(int axis, bool backward) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| performNSteps.vi |

|  |
| --- |
| **setControlContinuousBkwd** This function sets a continuous movement on the selected axis in backward direction. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | If enabled a present movement in the opposite direction is stopped. The parameter "false" stops all movement of the axis regardless its direction |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.setControlContinuousBkwd |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_setControlContinuousBkwd**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.setControlContinuousBkwd**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_setControlContinuousBkwd**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_SetControlContinuousBkwd**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlContinuousBkwd.vi |

|  |
| --- |
| **setControlContinuousFwd** This function sets a continuous movement on the selected axis in positive direction. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | If enabled a present movement in the opposite direction is stopped. The parameter "false" stops all movement of the axis regardless its direction. |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.setControlContinuousFwd |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_setControlContinuousFwd**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.setControlContinuousFwd**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_setControlContinuousFwd**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_SetControlContinuousFwd**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlContinuousFwd.vi |

|  |
| --- |
| **setControlEotOutputDeactive** This function sets the output applied to the selected axis on the end of travel. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | if enabled, the output of the axis will be deactivated on positive EOT detection. |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.setControlEotOutputDeactive |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_setControlEotOutputDeactive**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.setControlEotOutputDeactive**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_setControlEotOutputDeactive**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_SetControlEotOutputDeactive**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlEotOutputDeactive.vi |

|  |
| --- |
| **setControlTargetPosition** This function sets the target position for the movement on the selected axis.  careful: the maximum positon in nm is 2\*\*47/1000 |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| target | absolute position : For linear type actors the position is defined in nm for goniometer an rotator type actors it is µ°. |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.setControlTargetPosition |
| params: [axis, target] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_setControlTargetPosition**(int deviceHandle, int axis, double target) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.setControlTargetPosition**(axis, target) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_setControlTargetPosition**(axis, target) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_SetControlTargetPosition**(int axis, double target) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setControlTargetPosition.vi |

|  |
| --- |
| **setGroundAxis** Pull axis piezo drive to GND actively  only in AMC300  this is used in MIC-Mode |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | motion controler axis [0|1|2] |
| enabled | true or false |
| Out | errNo | 0 or error |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.setGroundAxis |
| params: [axis, enabled] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_setGroundAxis**(int deviceHandle, int axis, bool enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.setGroundAxis**(axis, enabled) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_setGroundAxis**(axis, enabled) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_SetGroundAxis**(int axis, bool enabled) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setGroundAxis.vi |

|  |
| --- |
| **setGroundAxisAutoOnTarget** Pull axis piezo drive to GND actively if positioner is in ground target range  only in AMC300  this is used in MIC-Mode |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | montion controler axis [0|1|2] |
| enabled | true or false |
| Out | errNo | 0 or error |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.setGroundAxisAutoOnTarget |
| params: [axis, enabled] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_setGroundAxisAutoOnTarget**(int deviceHandle, int axis, bool enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.setGroundAxisAutoOnTarget**(axis, enabled) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_setGroundAxisAutoOnTarget**(axis, enabled) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_SetGroundAxisAutoOnTarget**(int axis, bool enabled) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setGroundAxisAutoOnTarget.vi |

|  |
| --- |
| **setGroundTargetRange** Set the range around the target position in which the auto grounding becomes active.  only in AMC300 |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| range | in nm |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.setGroundTargetRange |
| params: [axis, range] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_setGroundTargetRange**(int deviceHandle, int axis, int range) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.setGroundTargetRange**(axis, range) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_setGroundTargetRange**(axis, range) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_SetGroundTargetRange**(int axis, int range) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setGroundTargetRange.vi |

|  |
| --- |
| **setNSteps** This function triggers n steps on the selected axis in desired direction. /PRO feature. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| backward | Selects the desired direction. False triggers a forward step, true a backward step |
| step | number of step |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.setNSteps |
| params: [axis, backward, step] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_setNSteps**(int deviceHandle, int axis, bool backward, int step) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.setNSteps**(axis, backward, step) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_setNSteps**(axis, backward, step) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_SetNSteps**(int axis, bool backward, int step) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setNSteps.vi |

|  |
| --- |
| **setSingleStep** This function triggers one step on the selected axis in desired direction. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| backward | Selects the desired direction. False triggers a forward step, true a backward step |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.setSingleStep |
| params: [axis, backward] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_setSingleStep**(int deviceHandle, int axis, bool backward) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.setSingleStep**(axis, backward) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_setSingleStep**(axis, backward) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_SetSingleStep**(int axis, bool backward) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setSingleStep.vi |

|  |
| --- |
| **writeNSteps** Sets the number of steps to perform on stepwise movement. /PRO feature. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| step | number of step |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.move.writeNSteps |
| params: [axis, step] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_move\_writeNSteps**(int deviceHandle, int axis, int step) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].move.writeNSteps**(axis, step) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **move\_writeNSteps**(axis, step) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Move\_WriteNSteps**(int axis, int step) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| writeNSteps.vi |

## Res

|  |
| --- |
| **getChainGain** Get chain gain, see setChainGain for parameter description |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | number of axis |
| Out | errNo | errNo |
| gaincoeff | gaincoeff |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.res.getChainGain |
| params: [axis] |
| Result: [errNo, gaincoeff] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_res\_getChainGain**(int deviceHandle, int axis, int\* gaincoeff) |

|  |
| --- |
| **Python** |

|  |
| --- |
| gaincoeff = **[dev].res.getChainGain**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [gaincoeff] = **res\_getChainGain**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Res\_GetChainGain**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getChainGain.vi |

|  |
| --- |
| **getLinearization** Gets wether linearization is enabled or not |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enabled | true when enabled |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.res.getLinearization |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_res\_getLinearization**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].res.getLinearization**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **res\_getLinearization**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Res\_GetLinearization**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getLinearization.vi |

|  |
| --- |
| **getLutSn** get the identifier of the loaded lookuptable (will be empty if disabled) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| value\_string1 | string : identifier |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.res.getLutSn |
| params: [axis] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_res\_getLutSn**(int deviceHandle, int axis, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].res.getLutSn**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **res\_getLutSn**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Res\_GetLutSn**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getLutSn.vi |

|  |
| --- |
| **getMode** Get mode of RES application, see setMode for the description of possible parameters |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| mode | mode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.res.getMode |
| params: [] |
| Result: [errNo, mode] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_res\_getMode**(int deviceHandle, int\* mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| mode = **[dev].res.getMode**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [mode] = **res\_getMode**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Res\_GetMode**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getMode.vi |

|  |
| --- |
| **getSensorStatus** Gets wether a valid RES position signal is present (always true for a disabled sensor and for rotators) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| present | true when present |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.res.getSensorStatus |
| params: [axis] |
| Result: [errNo, present] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_res\_getSensorStatus**(int deviceHandle, int axis, bool\* present) |

|  |
| --- |
| **Python** |

|  |
| --- |
| present = **[dev].res.getSensorStatus**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [present] = **res\_getSensorStatus**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Res\_GetSensorStatus**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getSensorStatus.vi |

|  |
| --- |
| **setChainGain** Set signal chain gain to control overall power |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | number of axis |
| gainconfig | 0: 0dB ( power 600mVpkpk^2/R), 1 : -10 dB , 2 : -15 dB , 3 : -20 dB |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.res.setChainGain |
| params: [axis, gainconfig] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_res\_setChainGain**(int deviceHandle, int axis, int gainconfig) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].res.setChainGain**(axis, gainconfig) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **res\_setChainGain**(axis, gainconfig) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Res\_SetChainGain**(int axis, int gainconfig) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setChainGain.vi |

|  |
| --- |
| **setConfigurationFile** Load configuration file which either contains a JSON dict with parameters for the positioner on the axis or the LUT file itself (as legacy support for ANC350 .aps files) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| content | JSON Dictionary or .aps File.  The JSON Dictonary can/must contain the following keys:  'type': mandatory This field has to be one of the positioner list (see getPositionersList)  'lut': optional, contains an array of 1024 LUT values that are a mapping between ratio of the RES element travelled (0 to 1) and the corresponding absolute value at this ratio given in [nm].  Note: when generating these tables with position data in absolute units, the scaling of the travel ratio with the current sensor range has to be reversed.  'lut\_sn': optional, a string to uniquely identify the loaded LUT |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.res.setConfigurationFile |
| params: [axis, content] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_res\_setConfigurationFile**(int deviceHandle, int axis, const char\* content) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].res.setConfigurationFile**(axis, content) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **res\_setConfigurationFile**(axis, content) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Res\_SetConfigurationFile**(int axis, string content) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setConfigurationFile.vi |

|  |
| --- |
| **setLinearization** Control if linearization is enabled or not |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | boolean ( true: enable linearization) |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.res.setLinearization |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_res\_setLinearization**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].res.setLinearization**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **res\_setLinearization**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Res\_SetLinearization**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setLinearization.vi |

|  |
| --- |
| **setMode** Sets the mode of the RES position measurement  This selects which frequency/ies are used for the lock-in measurement of the RES position, currently there are two possibilities:  1: Individual per axis: each axis is measured on a different frequency; this mode reduces noise coupling between axes, while requiring more wiring  2: Shared line/MIC-Mode: each axis is measured on the same frequency, which reduces the number of required wires while more coupling noise is excpected |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | mode | 1: Individual per axis 2: Shared line mode |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.res.setMode |
| params: [mode] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_res\_setMode**(int deviceHandle, int mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].res.setMode**(mode) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **res\_setMode**(mode) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Res\_SetMode**(int mode) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setMode.vi |

## Rotcomp

|  |
| --- |
| **getControlTargetRanges** Checks if all three axis are in target range. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| in\_target\_range | boolean true all three axes are in target range, false at least one axis is not in target range |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rotcomp.getControlTargetRanges |
| params: [] |
| Result: [errNo, in\_target\_range] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rotcomp\_getControlTargetRanges**(int deviceHandle, bool\* in\_target\_range) |

|  |
| --- |
| **Python** |

|  |
| --- |
| in\_target\_range = **[dev].rotcomp.getControlTargetRanges**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [in\_target\_range] = **rotcomp\_getControlTargetRanges**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Rotcomp\_GetControlTargetRanges**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getControlTargetRanges.vi |

|  |
| --- |
| **getEnabled** Gets the enabled status of the rotation compensation |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| enabled | boolean true Rotation compensation is enabled, false Rotation compensation is disabled |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rotcomp.getEnabled |
| params: [] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rotcomp\_getEnabled**(int deviceHandle, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].rotcomp.getEnabled**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **rotcomp\_getEnabled**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Rotcomp\_GetEnabled**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getEnabled.vi |

|  |
| --- |
| **getLUT** Gets the LUT file as JSON string |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| lut | string JSON string of the LUT file for the rotation compensation |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rotcomp.getLUT |
| params: [] |
| Result: [errNo, lut] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rotcomp\_getLUT**(int deviceHandle, char\* lut, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| lut = **[dev].rotcomp.getLUT**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [lut] = **rotcomp\_getLUT**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Rotcomp\_GetLUT**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getLUT.vi |

|  |
| --- |
| **setEnabled** Enables and disables the rotation compensation |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | enabled | true Rotation compensation is enabled, false Rotation compensation is disabled |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rotcomp.setEnabled |
| params: [enabled] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rotcomp\_setEnabled**(int deviceHandle, bool enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rotcomp.setEnabled**(enabled) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rotcomp\_setEnabled**(enabled) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rotcomp\_SetEnabled**(bool enabled) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setEnabled.vi |

|  |
| --- |
| **setLUT** Sets the LUT file from a JSON string |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | lut\_string | JSON string of the LUT file for the rotation compensation |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rotcomp.setLUT |
| params: [lut\_string] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rotcomp\_setLUT**(int deviceHandle, const char\* lut\_string) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rotcomp.setLUT**(lut\_string) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rotcomp\_setLUT**(lut\_string) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rotcomp\_SetLUT**(string lut\_string) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setLUT.vi |

|  |
| --- |
| **updateOffsets** Updates the start offsets of the axes |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | offset\_axis0 | Offset of axis 1 in [nm] |
| offset\_axis1 | Offset of axis 2 in [nm] |
| offset\_axis2 | Offset of axis 3 in [nm] |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rotcomp.updateOffsets |
| params: [offset\_axis0, offset\_axis1, offset\_axis2] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rotcomp\_updateOffsets**(int deviceHandle, int offset\_axis0, int offset\_axis1, int offset\_axis2) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rotcomp.updateOffsets**(offset\_axis0, offset\_axis1, offset\_axis2) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rotcomp\_updateOffsets**(offset\_axis0, offset\_axis1, offset\_axis2) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rotcomp\_UpdateOffsets**(int offset\_axis0, int offset\_axis1, int offset\_axis2) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| updateOffsets.vi |

## Rtin

|  |
| --- |
| **apply** Apply all realtime input function |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.apply |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_apply**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.apply**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_apply**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_Apply**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_apply.vi |

|  |
| --- |
| **discard** Discard all values beting set and not yet applieds |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.discard |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_discard**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.discard**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_discard**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_Discard**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_discard.vi |

|  |
| --- |
| **getControlAQuadBInResolution** This function gets the AQuadB input resolution for setpoint parameter. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| resolution | ion nm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.getControlAQuadBInResolution |
| params: [axis] |
| Result: [errNo, resolution] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_getControlAQuadBInResolution**(int deviceHandle, int axis, int\* resolution) |

|  |
| --- |
| **Python** |

|  |
| --- |
| resolution = **[dev].rtin.getControlAQuadBInResolution**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [resolution] = **rtin\_getControlAQuadBInResolution**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtin\_GetControlAQuadBInResolution**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_getControlAQuadBInResolution.vi |

|  |
| --- |
| **getControlMoveGPIO** This function gets the status for real time input on the selected axis in closed-loop mode. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enable | boolean true: approach enabled , false: approach disabled |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.getControlMoveGPIO |
| params: [axis] |
| Result: [errNo, enable] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_getControlMoveGPIO**(int deviceHandle, int axis, bool\* enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enable = **[dev].rtin.getControlMoveGPIO**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enable] = **rtin\_getControlMoveGPIO**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Rtin\_GetControlMoveGPIO**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_getControlMoveGPIO.vi |

|  |
| --- |
| **getGpioMode** get the GPIO mode for Mic Mode feature |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| gpio\_mode | gpio\_mode: 0: Standard GPIO 1: NSL-/Mic-Mode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.getGpioMode |
| params: [] |
| Result: [errNo, gpio\_mode] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_getGpioMode**(int deviceHandle, int\* gpio\_mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| gpio\_mode = **[dev].rtin.getGpioMode**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [gpio\_mode] = **rtin\_getGpioMode**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtin\_GetGpioMode**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_getGpioMode.vi |

|  |
| --- |
| **getNslMux** get the axis the NSL multiplexer is set to |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | mux\_mode | [0|1|2|3]  0: Off  1: Axis 1  2: Axis 2  3: Axis 3 |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.getNslMux |
| params: [mux\_mode] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_getNslMux**(int deviceHandle, int mux\_mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.getNslMux**(mux\_mode) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_getNslMux**(mux\_mode) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_GetNslMux**(int mux\_mode) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_getNslMux.vi |

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

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| resolution | to be added in current pos in nm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.getRealTimeInChangePerPulse |
| params: [axis] |
| Result: [errNo, resolution] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_getRealTimeInChangePerPulse**(int deviceHandle, int axis, int\* resolution) |

|  |
| --- |
| **Python** |

|  |
| --- |
| resolution = **[dev].rtin.getRealTimeInChangePerPulse**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [resolution] = **rtin\_getRealTimeInChangePerPulse**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtin\_GetRealTimeInChangePerPulse**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_getRealTimeInChangePerPulse.vi |

|  |
| --- |
| **getRealTimeInFeedbackLoopMode** Get if the realtime function must operate in close loop operation or open loop operation |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| mode | 0: open loop, 1 : close-loop |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.getRealTimeInFeedbackLoopMode |
| params: [axis] |
| Result: [errNo, mode] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_getRealTimeInFeedbackLoopMode**(int deviceHandle, int axis, int\* mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| mode = **[dev].rtin.getRealTimeInFeedbackLoopMode**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [mode] = **rtin\_getRealTimeInFeedbackLoopMode**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtin\_GetRealTimeInFeedbackLoopMode**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_getRealTimeInFeedbackLoopMode.vi |

|  |
| --- |
| **getRealTimeInMode** This function sets or gets the real time input mode for the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| mode | see `RT\_IN\_MODES` |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.getRealTimeInMode |
| params: [axis] |
| Result: [errNo, mode] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_getRealTimeInMode**(int deviceHandle, int axis, int\* mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| mode = **[dev].rtin.getRealTimeInMode**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [mode] = **rtin\_getRealTimeInMode**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtin\_GetRealTimeInMode**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_getRealTimeInMode.vi |

|  |
| --- |
| **getRealTimeInStepsPerPulse** Get the change in step per pulse of the realtime input when trigger and stepper mode is used |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| steps | number of steps to applied |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.getRealTimeInStepsPerPulse |
| params: [axis] |
| Result: [errNo, steps] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_getRealTimeInStepsPerPulse**(int deviceHandle, int axis, int\* steps) |

|  |
| --- |
| **Python** |

|  |
| --- |
| steps = **[dev].rtin.getRealTimeInStepsPerPulse**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [steps] = **rtin\_getRealTimeInStepsPerPulse**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtin\_GetRealTimeInStepsPerPulse**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_getRealTimeInStepsPerPulse.vi |

|  |
| --- |
| **setControlAQuadBInResolution** This function sets the AQuadB input resolution for setpoint parameter. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| resolution | ion nm |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.setControlAQuadBInResolution |
| params: [axis, resolution] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_setControlAQuadBInResolution**(int deviceHandle, int axis, int resolution) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.setControlAQuadBInResolution**(axis, resolution) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_setControlAQuadBInResolution**(axis, resolution) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_SetControlAQuadBInResolution**(int axis, int resolution) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_setControlAQuadBInResolution.vi |

|  |
| --- |
| **setControlMoveGPIO** This function sets the status for real time input on the selected axis in closed-loop mode. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| enable | boolean true: eanble the approach , false: disable the approach |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.setControlMoveGPIO |
| params: [axis, enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_setControlMoveGPIO**(int deviceHandle, int axis, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.setControlMoveGPIO**(axis, enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_setControlMoveGPIO**(axis, enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_SetControlMoveGPIO**(int axis, bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_setControlMoveGPIO.vi |

|  |
| --- |
| **setGpioMode** set the GPIO mode for Mic Mode feature |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | gpio\_mode | [0|1]  0: Standard GPIO  1: NSL-/Mic-Mode |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.setGpioMode |
| params: [gpio\_mode] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_setGpioMode**(int deviceHandle, int gpio\_mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.setGpioMode**(gpio\_mode) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_setGpioMode**(gpio\_mode) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_SetGpioMode**(int gpio\_mode) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_setGpioMode.vi |

|  |
| --- |
| **setNslMux** set the axis the NSL multiplexer is set to |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | mux\_mode | [0|1|2|3]  0: Off  1: Axis 1  2: Axis 2  3: Axis 3 |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.setNslMux |
| params: [mux\_mode] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_setNslMux**(int deviceHandle, int mux\_mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.setNslMux**(mux\_mode) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_setNslMux**(mux\_mode) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_SetNslMux**(int mux\_mode) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_setNslMux.vi |

|  |
| --- |
| **setRealTimeInChangePerPulse** This function sets the change per pulse for the selected axis under real time input in the closed-loop mode.  only used in closed loop operation |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| delta | to be added to current position in nm |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.setRealTimeInChangePerPulse |
| params: [axis, delta] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_setRealTimeInChangePerPulse**(int deviceHandle, int axis, int delta) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.setRealTimeInChangePerPulse**(axis, delta) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_setRealTimeInChangePerPulse**(axis, delta) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_SetRealTimeInChangePerPulse**(int axis, int delta) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_setRealTimeInChangePerPulse.vi |

|  |
| --- |
| **setRealTimeInFeedbackLoopMode** Set if the realtime function must operate in close loop operation or open loop operation |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| mode | 0: open loop, 1 : close-loop |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.setRealTimeInFeedbackLoopMode |
| params: [axis, mode] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_setRealTimeInFeedbackLoopMode**(int deviceHandle, int axis, int mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.setRealTimeInFeedbackLoopMode**(axis, mode) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_setRealTimeInFeedbackLoopMode**(axis, mode) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_SetRealTimeInFeedbackLoopMode**(int axis, int mode) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_setRealTimeInFeedbackLoopMode.vi |

|  |
| --- |
| **setRealTimeInMode** This function sets the real time input mode for the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| mode | see `RT\_IN\_MODES` @see realtime |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.setRealTimeInMode |
| params: [axis, mode] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_setRealTimeInMode**(int deviceHandle, int axis, int mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.setRealTimeInMode**(axis, mode) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_setRealTimeInMode**(axis, mode) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_SetRealTimeInMode**(int axis, int mode) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_setRealTimeInMode.vi |

|  |
| --- |
| **setRealTimeInStepsPerPulse** Set the change in step per pulse of the realtime input when trigger and stepper mode is used  only used in open loop operation |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| steps | number of steps to applied |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtin.setRealTimeInStepsPerPulse |
| params: [axis, steps] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtin\_setRealTimeInStepsPerPulse**(int deviceHandle, int axis, int steps) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtin.setRealTimeInStepsPerPulse**(axis, steps) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtin\_setRealTimeInStepsPerPulse**(axis, steps) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtin\_SetRealTimeInStepsPerPulse**(int axis, int steps) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtin\_setRealTimeInStepsPerPulse.vi |

## Rtout

|  |
| --- |
| **apply** Apply for all rtout function |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.apply |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_apply**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.apply**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_apply**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_Apply**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_apply.vi |

|  |
| --- |
| **applyAxis** Apply for rtout function of specific axis |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.applyAxis |
| params: [axis] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_applyAxis**(int deviceHandle, int axis) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.applyAxis**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_applyAxis**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_ApplyAxis**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_applyAxis.vi |

|  |
| --- |
| **discard** Discard all rtout value set by the set function(not applied yet) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.discard |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_discard**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.discard**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_discard**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_Discard**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_discard.vi |

|  |
| --- |
| **discardAxis** Discard rtout value of specific axis set by the set function(not applied yet) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.discardAxis |
| params: [axis] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_discardAxis**(int deviceHandle, int axis) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.discardAxis**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_discardAxis**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_DiscardAxis**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_discardAxis.vi |

|  |
| --- |
| **discardSignalMode** Discard value set by setSignalMode |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.discardSignalMode |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_discardSignalMode**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.discardSignalMode**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_discardSignalMode**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_DiscardSignalMode**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_discardSignalMode.vi |

|  |
| --- |
| **getControlAQuadBOut** This function gets if of AQuadB output for position indication is enabled |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| enabled | boolean |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.getControlAQuadBOut |
| params: [axis] |
| Result: [errNo, enabled] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_getControlAQuadBOut**(int deviceHandle, int axis, bool\* enabled) |

|  |
| --- |
| **Python** |

|  |
| --- |
| enabled = **[dev].rtout.getControlAQuadBOut**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [enabled] = **rtout\_getControlAQuadBOut**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Rtout\_GetControlAQuadBOut**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_getControlAQuadBOut.vi |

|  |
| --- |
| **getControlAQuadBOutClock** This function gets the clock for AQuadB output. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| clock\_in\_ns | Clock in multiples of 20ns. Minimum 2 (40ns), maximum 65535 (1,310700ms) |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.getControlAQuadBOutClock |
| params: [axis] |
| Result: [errNo, clock\_in\_ns] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_getControlAQuadBOutClock**(int deviceHandle, int axis, int\* clock\_in\_ns) |

|  |
| --- |
| **Python** |

|  |
| --- |
| clock\_in\_ns = **[dev].rtout.getControlAQuadBOutClock**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [clock\_in\_ns] = **rtout\_getControlAQuadBOutClock**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtout\_GetControlAQuadBOutClock**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_getControlAQuadBOutClock.vi |

|  |
| --- |
| **getControlAQuadBOutResolution** This function gets the AQuadB output resolution for position indication. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| resolution | in nm |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.getControlAQuadBOutResolution |
| params: [axis] |
| Result: [errNo, resolution] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_getControlAQuadBOutResolution**(int deviceHandle, int axis, int\* resolution) |

|  |
| --- |
| **Python** |

|  |
| --- |
| resolution = **[dev].rtout.getControlAQuadBOutResolution**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [resolution] = **rtout\_getControlAQuadBOutResolution**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtout\_GetControlAQuadBOutResolution**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_getControlAQuadBOutResolution.vi |

|  |
| --- |
| **getMode** Get Mode |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| mode | 0: Off, 1: AquadB, 2: Trigger |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.getMode |
| params: [axis] |
| Result: [errNo, mode] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_getMode**(int deviceHandle, int axis, int\* mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| mode = **[dev].rtout.getMode**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [mode] = **rtout\_getMode**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtout\_GetMode**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_getMode.vi |

|  |
| --- |
| **getSignalMode** This function gets the real time output mode for the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| mode | 0: TTL, 1: LVDS |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.getSignalMode |
| params: [] |
| Result: [errNo, mode] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_getSignalMode**(int deviceHandle, int\* mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| mode = **[dev].rtout.getSignalMode**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [mode] = **rtout\_getSignalMode**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Rtout\_GetSignalMode**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_getSignalMode.vi |

|  |
| --- |
| **getTriggerConfig** Get the real time output trigger config |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| higher | upper limit in nm / µdeg |
| lower | lower limit in nm / µdeg |
| epsilon | hysteresis in nm / µdeg |
| polarity | 0: active high, 1: active low |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.getTriggerConfig |
| params: [axis] |
| Result: [errNo, higher, lower, epsilon, polarity] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_getTriggerConfig**(int deviceHandle, int axis, int\* higher, int\* lower, int\* epsilon, int\* polarity) |

|  |
| --- |
| **Python** |

|  |
| --- |
| higher, lower, epsilon, polarity = **[dev].rtout.getTriggerConfig**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [higher, lower, epsilon, polarity] = **rtout\_getTriggerConfig**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| Tuple<int,int,int,int> value = [Device].**Rtout\_GetTriggerConfig**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_getTriggerConfig.vi |

|  |
| --- |
| **setControlAQuadBOutClock** This function sets the clock for AQuadB output. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| clock | Clock in multiples of 20ns. Minimum 2 (40ns), maximum 65535 (1,310700ms) |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.setControlAQuadBOutClock |
| params: [axis, clock] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_setControlAQuadBOutClock**(int deviceHandle, int axis, int clock) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.setControlAQuadBOutClock**(axis, clock) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_setControlAQuadBOutClock**(axis, clock) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_SetControlAQuadBOutClock**(int axis, int clock) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_setControlAQuadBOutClock.vi |

|  |
| --- |
| **setControlAQuadBOutResolution** This function sets the AQuadB output resolution for position indication. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| resolution | in nm |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.setControlAQuadBOutResolution |
| params: [axis, resolution] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_setControlAQuadBOutResolution**(int deviceHandle, int axis, int resolution) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.setControlAQuadBOutResolution**(axis, resolution) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_setControlAQuadBOutResolution**(axis, resolution) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_SetControlAQuadBOutResolution**(int axis, int resolution) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_setControlAQuadBOutResolution.vi |

|  |
| --- |
| **setMode** Set the real time output signal mode |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| mode | 0: Off, 1: AquadB, 2: Trigger |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.setMode |
| params: [axis, mode] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_setMode**(int deviceHandle, int axis, int mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.setMode**(axis, mode) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_setMode**(axis, mode) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_SetMode**(int axis, int mode) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_setMode.vi |

|  |
| --- |
| **setSignalMode** This function sets the real time output mode for the selected axis. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | mode | 0: TTL, 1: LVDS |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.setSignalMode |
| params: [mode] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_setSignalMode**(int deviceHandle, int mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.setSignalMode**(mode) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_setSignalMode**(mode) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_SetSignalMode**(int mode) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_setSignalMode.vi |

|  |
| --- |
| **setTriggerConfig** Control the real time output trigger config |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| higher | upper limit in nm / µdeg |
| lower | lower limit in nm / µdeg |
| epsilon | hysteresis in nm / µdeg |
| polarity | 0: active high, 1: active low |
| Out | errNo | errNo |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.rtout.setTriggerConfig |
| params: [axis, higher, lower, epsilon, polarity] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_rtout\_setTriggerConfig**(int deviceHandle, int axis, int higher, int lower, int epsilon, int polarity) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].rtout.setTriggerConfig**(axis, higher, lower, epsilon, polarity) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **rtout\_setTriggerConfig**(axis, higher, lower, epsilon, polarity) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Rtout\_SetTriggerConfig**(int axis, int higher, int lower, int epsilon, int polarity) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rtout\_setTriggerConfig.vi |

## Status

|  |
| --- |
| **getFullCombinedStatus** Get the full combined status of a positioner axis and return the status as a string (to be used in the Webapplication) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| value\_string1 | string can be "moving","in target range", "backward limit reached", "forward limit reached", "positioner not connected", "grounded" (only AMC300), "output not enabled" |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.status.getFullCombinedStatus |
| params: [axis] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_status\_getFullCombinedStatus**(int deviceHandle, int axis, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].status.getFullCombinedStatus**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **status\_getFullCombinedStatus**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Status\_GetFullCombinedStatus**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getFullCombinedStatus.vi |

|  |
| --- |
| **getOlStatus** Get the Feedback status of the positioner |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| sensorstatus | as integer 0: NUM Positioner connected 1: OL positioner connected 2: No positioner connected , 3: RES positione connected |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.status.getOlStatus |
| params: [axis] |
| Result: [errNo, sensorstatus] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_status\_getOlStatus**(int deviceHandle, int axis, int\* sensorstatus) |

|  |
| --- |
| **Python** |

|  |
| --- |
| sensorstatus = **[dev].status.getOlStatus**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [sensorstatus] = **status\_getOlStatus**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Status\_GetOlStatus**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getOlStatus.vi |

|  |
| --- |
| **getStatusConnected** This function gets information about the connection status of the selected axis’ positioner. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| connected | If true, the actor is connected |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.status.getStatusConnected |
| params: [axis] |
| Result: [errNo, connected] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_status\_getStatusConnected**(int deviceHandle, int axis, bool\* connected) |

|  |
| --- |
| **Python** |

|  |
| --- |
| connected = **[dev].status.getStatusConnected**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [connected] = **status\_getStatusConnected**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Status\_GetStatusConnected**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getStatusConnected.vi |

|  |
| --- |
| **getStatusEot** Retrieves the status of the end of travel (EOT) detection in backward direction or in forward direction. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| detected | true when EoT in either direction was detected |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.status.getStatusEot |
| params: [axis] |
| Result: [errNo, detected] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_status\_getStatusEot**(int deviceHandle, int axis, bool\* detected) |

|  |
| --- |
| **Python** |

|  |
| --- |
| detected = **[dev].status.getStatusEot**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [detected] = **status\_getStatusEot**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Status\_GetStatusEot**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getStatusEot.vi |

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

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| detected | true when EoT was detected |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.status.getStatusEotBkwd |
| params: [axis] |
| Result: [errNo, detected] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_status\_getStatusEotBkwd**(int deviceHandle, int axis, bool\* detected) |

|  |
| --- |
| **Python** |

|  |
| --- |
| detected = **[dev].status.getStatusEotBkwd**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [detected] = **status\_getStatusEotBkwd**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Status\_GetStatusEotBkwd**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getStatusEotBkwd.vi |

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

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| detected | true when EoT was detected |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.status.getStatusEotFwd |
| params: [axis] |
| Result: [errNo, detected] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_status\_getStatusEotFwd**(int deviceHandle, int axis, bool\* detected) |

|  |
| --- |
| **Python** |

|  |
| --- |
| detected = **[dev].status.getStatusEotFwd**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [detected] = **status\_getStatusEotFwd**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Status\_GetStatusEotFwd**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getStatusEotFwd.vi |

|  |
| --- |
| **getStatusMoving** This function gets information about the status of the stage output. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| status | 0: Idle, i.e. within the noise range of the sensor, 1: Moving, i.e the actor is actively driven by the output stage either for closed-loop approach or continous/single stepping and the output is active.  2 : Pending means the output stage is driving but the output is deactivated |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.status.getStatusMoving |
| params: [axis] |
| Result: [errNo, status] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_status\_getStatusMoving**(int deviceHandle, int axis, int\* status) |

|  |
| --- |
| **Python** |

|  |
| --- |
| status = **[dev].status.getStatusMoving**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [status] = **status\_getStatusMoving**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Status\_GetStatusMoving**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getStatusMoving.vi |

|  |
| --- |
| **getStatusReference** This function gets information about the status of the reference position. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| valid | true = valid, false = not valid |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.status.getStatusReference |
| params: [axis] |
| Result: [errNo, valid] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_status\_getStatusReference**(int deviceHandle, int axis, bool\* valid) |

|  |
| --- |
| **Python** |

|  |
| --- |
| valid = **[dev].status.getStatusReference**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [valid] = **status\_getStatusReference**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Status\_GetStatusReference**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getStatusReference.vi |

|  |
| --- |
| **getStatusTargetRange** This function gets information about whether the selected axis’ positioner is in target range or not.  The detection only indicates whether the position is within the defined range. This status is updated periodically but currently not in real-time.  If a fast detection is desired, please check the position in a loop |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | errNo | errNo |
| in\_range | true within the target range, false not within the target range |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.amc.status.getStatusTargetRange |
| params: [axis] |
| Result: [errNo, in\_range] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **AMC\_status\_getStatusTargetRange**(int deviceHandle, int axis, bool\* in\_range) |

|  |
| --- |
| **Python** |

|  |
| --- |
| in\_range = **[dev].status.getStatusTargetRange**(axis) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [in\_range] = **status\_getStatusTargetRange**(axis) |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Status\_GetStatusTargetRange**(int axis) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getStatusTargetRange.vi |

## About

|  |
| --- |
| **getInstalledPackages** Get list of packages installed on the device |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_string1 | string: Comma separated list of packages |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.about.getInstalledPackages |
| params: [] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_about\_getInstalledPackages**(int deviceHandle, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].about.getInstalledPackages**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **system\_about\_getInstalledPackages**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**About\_GetInstalledPackages**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getInstalledPackages.vi |

|  |
| --- |
| **getPackageLicense** Get the license for a specific package |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | pckg | string: Package name |
| Out | errNo | errorCode |
| value\_string1 | string: License for this package |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.about.getPackageLicense |
| params: [pckg] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_about\_getPackageLicense**(int deviceHandle, const char\* pckg, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].about.getPackageLicense**(pckg) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **system\_about\_getPackageLicense**(pckg) |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**About\_GetPackageLicense**(string pckg) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getPackageLicense.vi |

## System\_service

|  |
| --- |
| **apply** Apply temporary system configuration |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.apply |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_apply**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].system\_service.apply**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_apply**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Apply**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| apply.vi |

|  |
| --- |
| **errorNumberToRecommendation** Get a recommendation for the error code |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | language | integer: Language code |
| errNbr | interger: Error code to translate |
| Out | errNo | errorCode |
| value\_string1 | string: Error recommendation (currently returning an int = 0 until we have recommendations) |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.errorNumberToRecommendation |
| params: [language, errNbr] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_errorNumberToRecommendation**(int deviceHandle, int language, int errNbr, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].system\_service.errorNumberToRecommendation**(language, errNbr) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **system\_errorNumberToRecommendation**(language, errNbr) |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**ErrorNumberToRecommendation**(int language, int errNbr) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| errorNumberToRecommendation.vi |

|  |
| --- |
| **errorNumberToString** Get a description of an error code |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | language | integer: Language code 0 for the error name, 1 for a more user friendly error message |
| errNbr | interger: Error code to translate |
| Out | errNo | errorCode |
| value\_string1 | string: Error description |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.errorNumberToString |
| params: [language, errNbr] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_errorNumberToString**(int deviceHandle, int language, int errNbr, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].system\_service.errorNumberToString**(language, errNbr) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **system\_errorNumberToString**(language, errNbr) |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**ErrorNumberToString**(int language, int errNbr) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| errorNumberToString.vi |

|  |
| --- |
| **factoryReset** Turns on the factory reset flag. To perform the factory reset, a reboot is necessary afterwards. All settings will be set to default and the IDS will be configured as DHCP server. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.factoryReset |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_factoryReset**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].system\_service.factoryReset**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_factoryReset**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**FactoryReset**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| factoryReset.vi |

## Functions

|  |
| --- |
| **checkAMCinRack** If AMC is on Rack position 0, use it as DHCP server, else use it as DHCP client |

|  |
| --- |
| **Function specific parameters** |



|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.functions.checkAMCinRack |
| params: [] |
| Result: [] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_functions\_checkAMCinRack**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].functions.checkAMCinRack**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_functions\_checkAMCinRack**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Functions\_CheckAMCinRack**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| checkAMCinRack.vi |

## System\_service

|  |
| --- |
| **getDeviceName** Get the actual device name |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_string1 | string: actual device name |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.getDeviceName |
| params: [] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_getDeviceName**(int deviceHandle, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].system\_service.getDeviceName**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **system\_getDeviceName**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**GetDeviceName**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getDeviceName.vi |

|  |
| --- |
| **getFirmwareVersion** Get the firmware version of the system |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_string1 | string: The firmware version |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.getFirmwareVersion |
| params: [] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_getFirmwareVersion**(int deviceHandle, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].system\_service.getFirmwareVersion**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **system\_getFirmwareVersion**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**GetFirmwareVersion**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getFirmwareVersion.vi |

|  |
| --- |
| **getFluxCode** Get the flux code of the system |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_string1 | string: flux code |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.getFluxCode |
| params: [] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_getFluxCode**(int deviceHandle, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].system\_service.getFluxCode**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **system\_getFluxCode**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**GetFluxCode**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getFluxCode.vi |

|  |
| --- |
| **getHostname** Return device hostname |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| available | available |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.getHostname |
| params: [] |
| Result: [errNo, available] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_getHostname**(int deviceHandle, char\* available, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| available = **[dev].system\_service.getHostname**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [available] = **system\_getHostname**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**GetHostname**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getHostname.vi |

|  |
| --- |
| **getMacAddress** Get the mac address of the system |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_string1 | string: Mac address of the system |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.getMacAddress |
| params: [] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_getMacAddress**(int deviceHandle, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].system\_service.getMacAddress**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **system\_getMacAddress**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**GetMacAddress**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getMacAddress.vi |

|  |
| --- |
| **getSerialNumber** Get the serial number of the system |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_string1 | string: Serial number |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.getSerialNumber |
| params: [] |
| Result: [errNo, value\_string1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_getSerialNumber**(int deviceHandle, char\* value\_string1, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_string1 = **[dev].system\_service.getSerialNumber**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_string1] = **system\_getSerialNumber**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**GetSerialNumber**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getSerialNumber.vi |

## Network

|  |
| --- |
| **apply** Apply temporary IP configuration and load it |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.apply |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_apply**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.apply**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_apply**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_Apply**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| apply.vi |

|  |
| --- |
| **configureWifi** Change the wifi configuration and applies it |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | mode | 0: Access point, 1: Wifi client |
| ssid |  |
| psk | Pre-shared key |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.configureWifi |
| params: [mode, ssid, psk] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_configureWifi**(int deviceHandle, int mode, const char\* ssid, const char\* psk) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.configureWifi**(mode, ssid, psk) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_configureWifi**(mode, ssid, psk) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_ConfigureWifi**(int mode, string ssid, string psk) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| configureWifi.vi |

|  |
| --- |
| **discard** Discard temporary IP configuration |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.discard |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_discard**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.discard**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_discard**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_Discard**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| discard.vi |

|  |
| --- |
| **getDefaultGateway** Get the default gateway of the device |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| Default | gateway |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getDefaultGateway |
| params: [] |
| Result: [errNo, Default] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getDefaultGateway**(int deviceHandle, char\* Default, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| Default = **[dev].network.getDefaultGateway**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [Default] = **system\_network\_getDefaultGateway**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Network\_GetDefaultGateway**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getDefaultGateway.vi |

|  |
| --- |
| **getDnsResolver** Get the DNS resolver |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | priority | of DNS resolver (Usually: 0 = Default, 1 = Backup) |
| Out | errNo | errorCode |
| IP | address of DNS resolver |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getDnsResolver |
| params: [priority] |
| Result: [errNo, IP] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getDnsResolver**(int deviceHandle, int priority, char\* IP, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| IP = **[dev].network.getDnsResolver**(priority) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [IP] = **system\_network\_getDnsResolver**(priority) |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Network\_GetDnsResolver**(int priority) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getDnsResolver.vi |

|  |
| --- |
| **getEnableDhcpClient** Get the state of DHCP client |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_boolean1 | boolean: true = DHCP client enable, false = DHCP client disable |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getEnableDhcpClient |
| params: [] |
| Result: [errNo, value\_boolean1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getEnableDhcpClient**(int deviceHandle, bool\* value\_boolean1) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_boolean1 = **[dev].network.getEnableDhcpClient**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_boolean1] = **system\_network\_getEnableDhcpClient**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Network\_GetEnableDhcpClient**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getEnableDhcpClient.vi |

|  |
| --- |
| **getEnableDhcpServer** Get the state of DHCP server |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_boolean1 | boolean: true = DHCP server enable, false = DHCP server disable |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getEnableDhcpServer |
| params: [] |
| Result: [errNo, value\_boolean1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getEnableDhcpServer**(int deviceHandle, bool\* value\_boolean1) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_boolean1 = **[dev].network.getEnableDhcpServer**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_boolean1] = **system\_network\_getEnableDhcpServer**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Network\_GetEnableDhcpServer**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getEnableDhcpServer.vi |

|  |
| --- |
| **getIpAddress** Get the IP address of the device |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| IP | address as string |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getIpAddress |
| params: [] |
| Result: [errNo, IP] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getIpAddress**(int deviceHandle, char\* IP, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| IP = **[dev].network.getIpAddress**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [IP] = **system\_network\_getIpAddress**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Network\_GetIpAddress**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getIpAddress.vi |

|  |
| --- |
| **getProxyServer** Get the proxy settings of the devide |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| Proxy | Server String, empty for no proxy |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getProxyServer |
| params: [] |
| Result: [errNo, Proxy] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getProxyServer**(int deviceHandle, char\* Proxy, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| Proxy = **[dev].network.getProxyServer**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [Proxy] = **system\_network\_getProxyServer**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Network\_GetProxyServer**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getProxyServer.vi |

|  |
| --- |
| **getRealIpAddress** Get the real IP address of the device set to the network interface (br0, eth1 or eth0) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| IP | address as string |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getRealIpAddress |
| params: [] |
| Result: [errNo, IP] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getRealIpAddress**(int deviceHandle, char\* IP, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| IP = **[dev].network.getRealIpAddress**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [IP] = **system\_network\_getRealIpAddress**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Network\_GetRealIpAddress**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getRealIpAddress.vi |

|  |
| --- |
| **getSubnetMask** Get the subnet mask of the device |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| Subnet | mask as string |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getSubnetMask |
| params: [] |
| Result: [errNo, Subnet] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getSubnetMask**(int deviceHandle, char\* Subnet, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| Subnet = **[dev].network.getSubnetMask**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [Subnet] = **system\_network\_getSubnetMask**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Network\_GetSubnetMask**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getSubnetMask.vi |

|  |
| --- |
| **getWifiMode** Get the operation mode of the wifi adapter |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| mode | 0: Access point, 1: Wifi client |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getWifiMode |
| params: [] |
| Result: [errNo, mode] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getWifiMode**(int deviceHandle, int\* mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| mode = **[dev].network.getWifiMode**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [mode] = **system\_network\_getWifiMode**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Network\_GetWifiMode**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getWifiMode.vi |

|  |
| --- |
| **getWifiPassphrase** Get the the passphrase of the network hosted (mode: Access point) or connected to (mode: client) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| psk | Pre-shared key |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getWifiPassphrase |
| params: [] |
| Result: [errNo, psk] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getWifiPassphrase**(int deviceHandle, char\* psk, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| psk = **[dev].network.getWifiPassphrase**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [psk] = **system\_network\_getWifiPassphrase**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Network\_GetWifiPassphrase**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getWifiPassphrase.vi |

|  |
| --- |
| **getWifiPresent** Returns is a Wifi interface is present |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| True | True, if interface is present |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getWifiPresent |
| params: [] |
| Result: [errNo, True] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getWifiPresent**(int deviceHandle, bool\* True) |

|  |
| --- |
| **Python** |

|  |
| --- |
| True = **[dev].network.getWifiPresent**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [True] = **system\_network\_getWifiPresent**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| bool value = [Device].**Network\_GetWifiPresent**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getWifiPresent.vi |

|  |
| --- |
| **getWifiSSID** Get the the SSID of the network hosted (mode: Access point) or connected to (mode: client) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| SSID | SSID |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.getWifiSSID |
| params: [] |
| Result: [errNo, SSID] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_getWifiSSID**(int deviceHandle, char\* SSID, int size0) |

|  |
| --- |
| **Python** |

|  |
| --- |
| SSID = **[dev].network.getWifiSSID**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [SSID] = **system\_network\_getWifiSSID**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| string value = [Device].**Network\_GetWifiSSID**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getWifiSSID.vi |

|  |
| --- |
| **setDefaultGateway** Set the default gateway of the device |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | gateway | Default gateway as string |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setDefaultGateway |
| params: [gateway] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setDefaultGateway**(int deviceHandle, const char\* gateway) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setDefaultGateway**(gateway) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setDefaultGateway**(gateway) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetDefaultGateway**(string gateway) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setDefaultGateway.vi |

|  |
| --- |
| **setDnsResolver** Set the DNS resolver |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | priority | of DNS resolver (Usually: 0 = Default, 1 = Backup) |
| resolver | The resolver's IP address as string |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setDnsResolver |
| params: [priority, resolver] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setDnsResolver**(int deviceHandle, int priority, const char\* resolver) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setDnsResolver**(priority, resolver) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setDnsResolver**(priority, resolver) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetDnsResolver**(int priority, string resolver) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setDnsResolver.vi |

|  |
| --- |
| **setEnableDhcpClient** Enable or disable DHCP client |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | enable | boolean: true = enable DHCP client, false = disable DHCP client |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setEnableDhcpClient |
| params: [enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setEnableDhcpClient**(int deviceHandle, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setEnableDhcpClient**(enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setEnableDhcpClient**(enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetEnableDhcpClient**(bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setEnableDhcpClient.vi |

|  |
| --- |
| **setEnableDhcpServer** Enable or disable DHCP server |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | enable | boolean: true = enable DHCP server, false = disable DHCP server |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setEnableDhcpServer |
| params: [enable] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setEnableDhcpServer**(int deviceHandle, bool enable) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setEnableDhcpServer**(enable) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setEnableDhcpServer**(enable) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetEnableDhcpServer**(bool enable) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setEnableDhcpServer.vi |

|  |
| --- |
| **setIpAddress** Set the IP address of the device |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | address | IP address as string |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setIpAddress |
| params: [address] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setIpAddress**(int deviceHandle, const char\* address) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setIpAddress**(address) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setIpAddress**(address) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetIpAddress**(string address) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setIpAddress.vi |

|  |
| --- |
| **setProxyServer** Set the proxy server of the device |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | proxyServer | Proxy Server Setting as string |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setProxyServer |
| params: [proxyServer] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setProxyServer**(int deviceHandle, const char\* proxyServer) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setProxyServer**(proxyServer) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setProxyServer**(proxyServer) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetProxyServer**(string proxyServer) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setProxyServer.vi |

|  |
| --- |
| **setSubnetMask** Set the subnet mask of the device |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | netmask | Subnet mask as string |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setSubnetMask |
| params: [netmask] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setSubnetMask**(int deviceHandle, const char\* netmask) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setSubnetMask**(netmask) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setSubnetMask**(netmask) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetSubnetMask**(string netmask) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setSubnetMask.vi |

|  |
| --- |
| **setWifiMode** Change the operation mode of the wifi adapter |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | mode | 0: Access point, 1: Wifi client |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setWifiMode |
| params: [mode] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setWifiMode**(int deviceHandle, int mode) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setWifiMode**(mode) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setWifiMode**(mode) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetWifiMode**(int mode) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setWifiMode.vi |

|  |
| --- |
| **setWifiPassphrase** Change the passphrase of the network hosted (mode: Access point) or connected to (mode: client) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | psk | Pre-shared key |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setWifiPassphrase |
| params: [psk] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setWifiPassphrase**(int deviceHandle, const char\* psk) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setWifiPassphrase**(psk) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setWifiPassphrase**(psk) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetWifiPassphrase**(string psk) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setWifiPassphrase.vi |

|  |
| --- |
| **setWifiSSID** Change the SSID of the network hosted (mode: Access point) or connected to (mode: client) |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | ssid |  |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.setWifiSSID |
| params: [ssid] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_setWifiSSID**(int deviceHandle, const char\* ssid) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.setWifiSSID**(ssid) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_setWifiSSID**(ssid) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_SetWifiSSID**(string ssid) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setWifiSSID.vi |

|  |
| --- |
| **verify** Verify that temporary IP configuration is correct |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.network.verify |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_network\_verify**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].network.verify**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_network\_verify**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Network\_Verify**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| verify.vi |

## System\_service

|  |
| --- |
| **rebootSystem** Reboot the system |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.rebootSystem |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_rebootSystem**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].system\_service.rebootSystem**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_rebootSystem**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**RebootSystem**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| rebootSystem.vi |

|  |
| --- |
| **setDeviceName** Set custom name for the device |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | name | string: device name |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.setDeviceName |
| params: [name] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_setDeviceName**(int deviceHandle, const char\* name) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].system\_service.setDeviceName**(name) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_setDeviceName**(name) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**SetDeviceName**(string name) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setDeviceName.vi |

|  |
| --- |
| **setTime** Set system time manually |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | day | integer: Day (1-31) |
| month | integer: Day (1-12) |
| year | integer: Day (eg. 2021) |
| hour | integer: Day (0-23) |
| minute | integer: Day (0-59) |
| second | integer: Day (0-59) |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.setTime |
| params: [day, month, year, hour, minute, second] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_setTime**(int deviceHandle, int day, int month, int year, int hour, int minute, int second) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].system\_service.setTime**(day, month, year, hour, minute, second) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_setTime**(day, month, year, hour, minute, second) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**SetTime**(int day, int month, int year, int hour, int minute, int second) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| setTime.vi |

|  |
| --- |
| **softReset** Performs a soft reset (Reset without deleting the network settings). Please reboot the device directly afterwards. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.softReset |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_softReset**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].system\_service.softReset**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_softReset**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**SoftReset**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| softReset.vi |

|  |
| --- |
| **updateTimeFromInternet** Update system time by querying attocube.com |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.updateTimeFromInternet |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_updateTimeFromInternet**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].system\_service.updateTimeFromInternet**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_updateTimeFromInternet**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**UpdateTimeFromInternet**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| updateTimeFromInternet.vi |

## Update

|  |
| --- |
| **getLicenseUpdateProgress** Get the progress of running license update |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_int1 | int: progress in percent |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.update.getLicenseUpdateProgress |
| params: [] |
| Result: [errNo, value\_int1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_update\_getLicenseUpdateProgress**(int deviceHandle, int\* value\_int1) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_int1 = **[dev].update.getLicenseUpdateProgress**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_int1] = **system\_update\_getLicenseUpdateProgress**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Update\_GetLicenseUpdateProgress**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getLicenseUpdateProgress.vi |

|  |
| --- |
| **getSwUpdateProgress** Get the progress of running update |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |
| value\_int1 | int: progress in percent |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.update.getSwUpdateProgress |
| params: [] |
| Result: [errNo, value\_int1] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_update\_getSwUpdateProgress**(int deviceHandle, int\* value\_int1) |

|  |
| --- |
| **Python** |

|  |
| --- |
| value\_int1 = **[dev].update.getSwUpdateProgress**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [value\_int1] = **system\_update\_getSwUpdateProgress**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| int value = [Device].**Update\_GetSwUpdateProgress**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| getSwUpdateProgress.vi |

|  |
| --- |
| **licenseUpdateBase64** Execute the license update with base64 file uploaded. After execution, a manual reboot is nevessary. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.update.licenseUpdateBase64 |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_update\_licenseUpdateBase64**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].update.licenseUpdateBase64**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_update\_licenseUpdateBase64**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Update\_LicenseUpdateBase64**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| licenseUpdateBase64.vi |

|  |
| --- |
| **softwareUpdateBase64** Execute the update with base64 file uploaded. After completion, a manual reboot is necessary. |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.update.softwareUpdateBase64 |
| params: [] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_update\_softwareUpdateBase64**(int deviceHandle) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].update.softwareUpdateBase64**() |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_update\_softwareUpdateBase64**() |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Update\_SoftwareUpdateBase64**() |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| softwareUpdateBase64.vi |

|  |
| --- |
| **uploadLicenseBase64** Upload new license file in format base 64 |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | offset | int: offset of the data |
| b64Data | string: base64 data |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.update.uploadLicenseBase64 |
| params: [offset, b64Data] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_update\_uploadLicenseBase64**(int deviceHandle, int offset, const char\* b64Data) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].update.uploadLicenseBase64**(offset, b64Data) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_update\_uploadLicenseBase64**(offset, b64Data) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Update\_UploadLicenseBase64**(int offset, string b64Data) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| uploadLicenseBase64.vi |

|  |
| --- |
| **uploadSoftwareImageBase64** Upload new firmware image in format base 64 |

|  |
| --- |
| **Function specific parameters** |

|  |  |  |
| --- | --- | --- |
| In | offset | int: offset of the data |
| b64Data | string: base64 data |
| Out | errNo | errorCode |

|  |
| --- |
| **JSON Method** |

|  |
| --- |
| method: com.attocube.system.update.uploadSoftwareImageBase64 |
| params: [offset, b64Data] |
| Result: [errNo] |

|  |
| --- |
| **C-DLL call** |

|  |
| --- |
| int **system\_update\_uploadSoftwareImageBase64**(int deviceHandle, int offset, const char\* b64Data) |

|  |
| --- |
| **Python** |

|  |
| --- |
| **[dev].update.uploadSoftwareImageBase64**(offset, b64Data) |

|  |
| --- |
| **Matlab** |

|  |
| --- |
| [] = **system\_update\_uploadSoftwareImageBase64**(offset, b64Data) |

|  |
| --- |
| **C#** |

|  |
| --- |
| void value = [Device].**Update\_UploadSoftwareImageBase64**(int offset, string b64Data) |

|  |
| --- |
| **LabVIEW** |

|  |
| --- |
| uploadSoftwareImageBase64.vi |

|  |
| --- |
| attocube systems AG Eglfinger Weg 2 D - 85540 Haar, Germany  Phone: +49 89 - 4207 97 0  Fax: +49 89 - 4207 97 20 190  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)  **South America Support Hotline:**  +1 510 649 9245 |