# Functions

## Adjustment

|  |
| --- |
| **getAdjustmentEnabled** This function can be used to see if the adjustment is running |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| enable | true = enabled; false = disabled |

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

|  |
| --- |
| method: com.attocube.ids.adjustment.getAdjustmentEnabled |
| params: [] |
| Result: [errNo, enable] |

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

|  |
| --- |
| int **IDS\_adjustment\_getAdjustmentEnabled**(int deviceHandle, bool\* enable) |

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

|  |
| --- |
| enable = **[dev].adjustment.getAdjustmentEnabled**() |

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

|  |
| --- |
| [enable] = **IDS\_adjustment\_getAdjustmentEnabled**() |

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

|  |
| --- |
| bool value = [Device].**Adjustment\_GetAdjustmentEnabled**() |

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

|  |
| --- |
| getAdjustmentEnabled.vi |

|  |
| --- |
| **getContrastInPermille** This function can be used to monitor the alignment contrast (peak-to-peak of the  basic interference signal amplitude) and the basline (its offset) during alignment  mode. It is used to monitor and optimize the optical alignment such that it allows the  successful initialization of a measurement. To monitor the optical contrast during a  running measurement, please use the getAxisSignalQuality function.  Important: The baseline and contrast needs to be added, otherwise an overload can  occure without being recognized. Example: baseline 15permille and contrast 850permille. Just  from the contrast it looks good. Adding the baseline the overload is visible. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to get the value from [0..2] |
| Out | warningNo | warningNo |
| contast | Contrast of the base band signal in permille |
| baseline | Offset of the contrast measurement in permille |
| mixcontrast | lower contrast measurment when measuring a mix contrast (indicated by error code) |

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

|  |
| --- |
| method: com.attocube.ids.adjustment.getContrastInPermille |
| params: [axis] |
| Result: [warningNo, contast, baseline, mixcontrast] |

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

|  |
| --- |
| int **IDS\_adjustment\_getContrastInPermille**(int deviceHandle, int axis, int\* warningNo, int\* contast, int\* baseline, int\* mixcontrast) |

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

|  |
| --- |
| warningNo, contast, baseline, mixcontrast = **[dev].adjustment.getContrastInPermille**(axis) |

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

|  |
| --- |
| [warningNo, contast, baseline, mixcontrast] = **IDS\_adjustment\_getContrastInPermille**(axis) |

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

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

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

|  |
| --- |
| getContrastInPermille.vi |

## Axis

|  |
| --- |
| **apply** Applies new axis settings. Necessary after JSON set commands. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.axis.apply |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_axis\_apply**(int deviceHandle) |

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

|  |
| --- |
| **[dev].axis.apply**() |

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

|  |
| --- |
| [] = **IDS\_axis\_apply**() |

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

|  |
| --- |
| void value = [Device].**Axis\_Apply**() |

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

|  |
| --- |
| apply.vi |

|  |
| --- |
| **discard** Discards new axis settings. Necessary after JSON set commands instead of apply() in case of failure. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.axis.discard |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_axis\_discard**(int deviceHandle) |

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

|  |
| --- |
| **[dev].axis.discard**() |

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

|  |
| --- |
| [] = **IDS\_axis\_discard**() |

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

|  |
| --- |
| void value = [Device].**Axis\_Discard**() |

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

|  |
| --- |
| discard.vi |

|  |
| --- |
| **getMasterAxis** Returns the master axis (for more information please refer to the IDS User Manual).  ADD MORE INFO |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| masteraxis | Axis which is operating as masteraxis [0..2] |

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

|  |
| --- |
| method: com.attocube.ids.axis.getMasterAxis |
| params: [] |
| Result: [errNo, masteraxis] |

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

|  |
| --- |
| int **IDS\_axis\_getMasterAxis**(int deviceHandle, int\* masteraxis) |

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

|  |
| --- |
| masteraxis = **[dev].axis.getMasterAxis**() |

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

|  |
| --- |
| [masteraxis] = **IDS\_axis\_getMasterAxis**() |

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

|  |
| --- |
| int value = [Device].**Axis\_GetMasterAxis**() |

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

|  |
| --- |
| getMasterAxis.vi |

|  |
| --- |
| **getPassMode** Reads out the current pass mode. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| mode | 0 = single; pass 1 = dual pass |

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

|  |
| --- |
| method: com.attocube.ids.axis.getPassMode |
| params: [] |
| Result: [errNo, mode] |

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

|  |
| --- |
| int **IDS\_axis\_getPassMode**(int deviceHandle, int\* mode) |

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

|  |
| --- |
| mode = **[dev].axis.getPassMode**() |

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

|  |
| --- |
| [mode] = **IDS\_axis\_getPassMode**() |

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

|  |
| --- |
| int value = [Device].**Axis\_GetPassMode**() |

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

|  |
| --- |
| getPassMode.vi |

|  |
| --- |
| **setMasterAxis** Sets the master axis (for more information please refer to the IDS User Manual). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis which is operating as masteraxis [0..2] |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.axis.setMasterAxis |
| params: [axis] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_axis\_setMasterAxis**(int deviceHandle, int axis) |

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

|  |
| --- |
| **[dev].axis.setMasterAxis**(axis) |

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

|  |
| --- |
| [] = **IDS\_axis\_setMasterAxis**(axis) |

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

|  |
| --- |
| void value = [Device].**Axis\_SetMasterAxis**(int axis) |

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

|  |
| --- |
| setMasterAxis.vi |

|  |
| --- |
| **setPassMode** Sets the desired pass mode. Effectively this mode defines, if the correction factor of two (necessary for measurements in optical dual pass configuration) is applied. |

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

|  |  |  |
| --- | --- | --- |
| In | mode | 0 = single pass; 1 = dual pass |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.axis.setPassMode |
| params: [mode] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_axis\_setPassMode**(int deviceHandle, int mode) |

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

|  |
| --- |
| **[dev].axis.setPassMode**(mode) |

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

|  |
| --- |
| [] = **IDS\_axis\_setPassMode**(mode) |

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

|  |
| --- |
| void value = [Device].**Axis\_SetPassMode**(int mode) |

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

|  |
| --- |
| setPassMode.vi |

## Displacement

|  |
| --- |
| **getAbsolutePosition** The absolute position information is estimated at the measurement initialization procedure.  This initial absolute position information is not updated during system  operation, whereas the IDS continuously measures the relative displacement from  that point. This function outputs the sum of the static absolute position register value  and the continuously updated displacement register value for a single axis. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | warningNo | warningNo |
| position | position of the axis in pm |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getAbsolutePosition |
| params: [axis] |
| Result: [warningNo, position] |

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

|  |
| --- |
| int **IDS\_displacement\_getAbsolutePosition**(int deviceHandle, int axis, int\* warningNo, double\* position) |

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

|  |
| --- |
| warningNo, position = **[dev].displacement.getAbsolutePosition**(axis) |

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

|  |
| --- |
| [warningNo, position] = **IDS\_displacement\_getAbsolutePosition**(axis) |

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

|  |
| --- |
| Tuple<int,double> value = [Device].**Displacement\_GetAbsolutePosition**(int axis) |

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

|  |
| --- |
| getAbsolutePosition.vi |

|  |
| --- |
| **getAbsolutePositions** The absolute position information is estimated at the measurement initialization  procedure. This initial absolute position information is not updated during system  operation, whereas the IDS continuously measures the relative displacement from  that point. This function outputs the sum of the static absolute position register value  and the continuously updated displacement register value for all three axes. |

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

|  |  |  |
| --- | --- | --- |
| Out | warningNo | warningNo |
| position0 | position of the axis 0 in pm |
| position1 | position of the axis 1 in pm |
| position2 | position of the axis 2 in pm |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getAbsolutePositions |
| params: [] |
| Result: [warningNo, position0, position1, position2] |

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

|  |
| --- |
| int **IDS\_displacement\_getAbsolutePositions**(int deviceHandle, int\* warningNo, double\* position0, double\* position1, double\* position2) |

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

|  |
| --- |
| warningNo, position0, position1, position2 = **[dev].displacement.getAbsolutePositions**() |

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

|  |
| --- |
| [warningNo, position0, position1, position2] = **IDS\_displacement\_getAbsolutePositions**() |

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

|  |
| --- |
| Tuple<int,double,double,double> value = [Device].**Displacement\_GetAbsolutePositions**() |

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

|  |
| --- |
| getAbsolutePositions.vi |

|  |
| --- |
| **getAverageN** Reads-out the averaging (lowpass) parameter N. The averaging time is calculated by  (2^N)\*40ns, where N is the averaging value. Please refere to the manual for a table with stopband and 3dB cut-off frequency. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| averageN | A value from 0 to 24 |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getAverageN |
| params: [] |
| Result: [errNo, averageN] |

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

|  |
| --- |
| int **IDS\_displacement\_getAverageN**(int deviceHandle, int\* averageN) |

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

|  |
| --- |
| averageN = **[dev].displacement.getAverageN**() |

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

|  |
| --- |
| [averageN] = **IDS\_displacement\_getAverageN**() |

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

|  |
| --- |
| int value = [Device].**Displacement\_GetAverageN**() |

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

|  |
| --- |
| getAverageN.vi |

|  |
| --- |
| **getAxesDisplacement** Reads out the displacement values of all three measurement axes. |

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

|  |  |  |
| --- | --- | --- |
| Out | warningNo | warningNo |
| displacement0 | displacement of the axis 0 in pm |
| displacement1 | displacement of the axis 1 in pm |
| displacement2 | displacement of the axis 2 in pm |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getAxesDisplacement |
| params: [] |
| Result: [warningNo, displacement0, displacement1, displacement2] |

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

|  |
| --- |
| int **IDS\_displacement\_getAxesDisplacement**(int deviceHandle, int\* warningNo, double\* displacement0, double\* displacement1, double\* displacement2) |

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

|  |
| --- |
| warningNo, displacement0, displacement1, displacement2 = **[dev].displacement.getAxesDisplacement**() |

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

|  |
| --- |
| [warningNo, displacement0, displacement1, displacement2] = **IDS\_displacement\_getAxesDisplacement**() |

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

|  |
| --- |
| Tuple<int,double,double,double> value = [Device].**Displacement\_GetAxesDisplacement**() |

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

|  |
| --- |
| getAxesDisplacement.vi |

|  |
| --- |
| **getAxisDisplacement** Reads out the displacement value of a specific measurement axis. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | warningNo | warningNo |
| displacement | Displacement of the axis in pm |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getAxisDisplacement |
| params: [axis] |
| Result: [warningNo, displacement] |

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

|  |
| --- |
| int **IDS\_displacement\_getAxisDisplacement**(int deviceHandle, int axis, int\* warningNo, double\* displacement) |

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

|  |
| --- |
| warningNo, displacement = **[dev].displacement.getAxisDisplacement**(axis) |

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

|  |
| --- |
| [warningNo, displacement] = **IDS\_displacement\_getAxisDisplacement**(axis) |

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

|  |
| --- |
| Tuple<int,double> value = [Device].**Displacement\_GetAxisDisplacement**(int axis) |

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

|  |
| --- |
| getAxisDisplacement.vi |

|  |
| --- |
| **getAxisSignalQuality** This function can be used to monitor the alignment contrast (peak-to-peak of the basic  interference signal amplitude) and the basline (its offset) during a running  measurement. Please note that the getAxisSignalQuality function output is only  updated when a displacement is measured. This means that angular misalignments  without displacement changes on the measurement axes cannot be detected.  Furthermore, we recommend using the IDS high accuracy initialization to obtain  correct values directly after measurement initialization. When using the quick  initialization, the initial value might be falsified at first, but correct after some detected  measurement (because it is updated with the correct value then). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | warningNo | warningNo |
| contrast | Contrast of the base band signal in ‰ |
| baseline | Offset of the contrast measurement in ‰ |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getAxisSignalQuality |
| params: [axis] |
| Result: [warningNo, contrast, baseline] |

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

|  |
| --- |
| int **IDS\_displacement\_getAxisSignalQuality**(int deviceHandle, int axis, int\* warningNo, int\* contrast, int\* baseline) |

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

|  |
| --- |
| warningNo, contrast, baseline = **[dev].displacement.getAxisSignalQuality**(axis) |

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

|  |
| --- |
| [warningNo, contrast, baseline] = **IDS\_displacement\_getAxisSignalQuality**(axis) |

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

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

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

|  |
| --- |
| getAxisSignalQuality.vi |

|  |
| --- |
| **getMeasurementEnabled** This function can be used to see if the measurement is running |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| enable | true = enabled; false = disabled |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getMeasurementEnabled |
| params: [] |
| Result: [errNo, enable] |

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

|  |
| --- |
| int **IDS\_displacement\_getMeasurementEnabled**(int deviceHandle, bool\* enable) |

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

|  |
| --- |
| enable = **[dev].displacement.getMeasurementEnabled**() |

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

|  |
| --- |
| [enable] = **IDS\_displacement\_getMeasurementEnabled**() |

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

|  |
| --- |
| bool value = [Device].**Displacement\_GetMeasurementEnabled**() |

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

|  |
| --- |
| getMeasurementEnabled.vi |

|  |
| --- |
| **getReferencePosition** The reference position information is estimated at the measurement initialization procedure or on reset.  This initial absolute position information is not updated during system  operation, whereas the IDS continuously measures the relative displacement from  that point. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | warningNo | warningNo |
| position | reference position of the axis in pm |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getReferencePosition |
| params: [axis] |
| Result: [warningNo, position] |

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

|  |
| --- |
| int **IDS\_displacement\_getReferencePosition**(int deviceHandle, int axis, int\* warningNo, double\* position) |

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

|  |
| --- |
| warningNo, position = **[dev].displacement.getReferencePosition**(axis) |

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

|  |
| --- |
| [warningNo, position] = **IDS\_displacement\_getReferencePosition**(axis) |

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

|  |
| --- |
| Tuple<int,double> value = [Device].**Displacement\_GetReferencePosition**(int axis) |

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

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

|  |
| --- |
| **getReferencePositions** The reference position information is estimated at the measurement initialization procedure or on reset.  This initial absolute position information is not updated during system  operation, whereas the IDS continuously measures the relative displacement from  that point. |

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

|  |  |  |
| --- | --- | --- |
| Out | warningNo | warningNo |
| position0 | position of the axis 0 in pm |
| position1 | position of the axis 1 in pm |
| position2 | position of the axis 2 in pm |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getReferencePositions |
| params: [] |
| Result: [warningNo, position0, position1, position2] |

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

|  |
| --- |
| int **IDS\_displacement\_getReferencePositions**(int deviceHandle, int\* warningNo, double\* position0, double\* position1, double\* position2) |

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

|  |
| --- |
| warningNo, position0, position1, position2 = **[dev].displacement.getReferencePositions**() |

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

|  |
| --- |
| [warningNo, position0, position1, position2] = **IDS\_displacement\_getReferencePositions**() |

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

|  |
| --- |
| Tuple<int,double,double,double> value = [Device].**Displacement\_GetReferencePositions**() |

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

|  |
| --- |
| getReferencePositions.vi |

|  |
| --- |
| **getSignalQuality** This function can be used to monitor the alignment contrast (peak-to-peak of the basic  interference signal amplitude) and the basline (its offset) during a running  measurement. Please note that the getAxisSignalQuality function output is only  updated when a displacement is measured. This means that angular misalignments  without displacement changes on the measurement axes cannot be detected.  Furthermore, we recommend using the IDS high accuracy initialization to obtain  correct values directly after measurement initialization. When using the quick  initialization, the initial value might be falsified at first, but correct after some detected  measurement (because it is updated with the correct value then). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| Out | warningNo | warningNo |
| contrast | Contrast of the base band signal in ‰ |
| baseline | Offset of the contrast measurement in ‰ |

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

|  |
| --- |
| method: com.attocube.ids.displacement.getSignalQuality |
| params: [axis] |
| Result: [warningNo, contrast, baseline] |

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

|  |
| --- |
| int **IDS\_displacement\_getSignalQuality**(int deviceHandle, int axis, int\* warningNo, int\* contrast, int\* baseline) |

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

|  |
| --- |
| warningNo, contrast, baseline = **[dev].displacement.getSignalQuality**(axis) |

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

|  |
| --- |
| [warningNo, contrast, baseline] = **IDS\_displacement\_getSignalQuality**(axis) |

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

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

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

|  |
| --- |
| getSignalQuality.vi |

|  |
| --- |
| **linProc** Important note: This function is not actively supported anymore. With IDS firmware 1.7.0, a new set of functions corresponding to the linearization procedure was introduced and effectively outdated this function.  Starts linearization procedure (additional IDS feature). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | [0|1|2] |
| fringesnbr | Number of fringes to be acquired |
| samplesperfringe | Number of samples per fringe |
| set | 0 = evaluate current nonlinear amplitude  1 = perform linearization and upload look up table  2 = Clear look up table  3 = Perform only calculation of Phi file |
| Out | errNo | errNo |
| lintable | String, which contains all 512 phase related correction values |
| nonlinearamp | String which contains the residual positive and negative maximal nonlinear amplitude |

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

|  |
| --- |
| method: com.attocube.ids.displacement.linProc |
| params: [axis, fringesnbr, samplesperfringe, set] |
| Result: [errNo, lintable, nonlinearamp] |

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

|  |
| --- |
| int **IDS\_displacement\_linProc**(int deviceHandle, int axis, int fringesnbr, int samplesperfringe, int set, int\* lintable, int\* nonlinearamp) |

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

|  |
| --- |
| lintable, nonlinearamp = **[dev].displacement.linProc**(axis, fringesnbr, samplesperfringe, set) |

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

|  |
| --- |
| [lintable, nonlinearamp] = **IDS\_displacement\_linProc**(axis, fringesnbr, samplesperfringe, set) |

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

|  |
| --- |
| Tuple<int,int> value = [Device].**Displacement\_LinProc**(int axis, int fringesnbr, int samplesperfringe, int set) |

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

|  |
| --- |
| linProc.vi |

|  |
| --- |
| **setAverageN** Sets the averaging (lowpass) parameter N. The averaging time is calculated by  (2^N)\*40ns, where N is the averaging value. Please refere to the manual for a table with stopband and 3dB cut-off frequency. |

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

|  |  |  |
| --- | --- | --- |
| In | value | AverageN value from 0 to 24 |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.displacement.setAverageN |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_displacement\_setAverageN**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].displacement.setAverageN**(value) |

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

|  |
| --- |
| [] = **IDS\_displacement\_setAverageN**(value) |

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

|  |
| --- |
| void value = [Device].**Displacement\_SetAverageN**(int value) |

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

|  |
| --- |
| setAverageN.vi |

## Ecu

|  |
| --- |
| **disable** Disables the ECU interface. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

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

|  |
| --- |
| method: com.attocube.ecu.disable |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_ecu\_disable**(int deviceHandle) |

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

|  |
| --- |
| **[dev].ecu.disable**() |

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

|  |
| --- |
| [] = **IDS\_ecu\_disable**() |

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

|  |
| --- |
| void value = [Device].**Ecu\_Disable**() |

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

|  |
| --- |
| disable.vi |

|  |
| --- |
| **enable** Enables the ECU interface. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

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

|  |
| --- |
| method: com.attocube.ecu.enable |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_ecu\_enable**(int deviceHandle) |

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

|  |
| --- |
| **[dev].ecu.enable**() |

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

|  |
| --- |
| [] = **IDS\_ecu\_enable**() |

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

|  |
| --- |
| void value = [Device].**Ecu\_Enable**() |

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

|  |
| --- |
| enable.vi |

|  |
| --- |
| **getConnected** Reads out whether the ECU interface is physically connected or not. Checking if the ECU is connected can only be done on an enabled ECU interface. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| connected | boolean true if connected, false if not |

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

|  |
| --- |
| method: com.attocube.ecu.getConnected |
| params: [] |
| Result: [errNo, connected] |

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

|  |
| --- |
| int **IDS\_ecu\_getConnected**(int deviceHandle, bool\* connected) |

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

|  |
| --- |
| connected = **[dev].ecu.getConnected**() |

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

|  |
| --- |
| [connected] = **IDS\_ecu\_getConnected**() |

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

|  |
| --- |
| bool value = [Device].**Ecu\_GetConnected**() |

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

|  |
| --- |
| getConnected.vi |

|  |
| --- |
| **getEnabled** Reads out whether the ECU interface is enabled or not. Enabling the ECU interface is crucial for working with the ECU. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| enabled | boolean true if enabled, false if not |

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

|  |
| --- |
| method: com.attocube.ecu.getEnabled |
| params: [] |
| Result: [errNo, enabled] |

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

|  |
| --- |
| int **IDS\_ecu\_getEnabled**(int deviceHandle, bool\* enabled) |

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

|  |
| --- |
| enabled = **[dev].ecu.getEnabled**() |

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

|  |
| --- |
| [enabled] = **IDS\_ecu\_getEnabled**() |

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

|  |
| --- |
| bool value = [Device].**Ecu\_GetEnabled**() |

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

|  |
| --- |
| getEnabled.vi |

|  |
| --- |
| **getHumidityInPercent** Reads out the ECU measured air humidity in percent. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| humidity | double humidity in % |

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

|  |
| --- |
| method: com.attocube.ecu.getHumidityInPercent |
| params: [] |
| Result: [errNo, humidity] |

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

|  |
| --- |
| int **IDS\_ecu\_getHumidityInPercent**(int deviceHandle, double\* humidity) |

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

|  |
| --- |
| humidity = **[dev].ecu.getHumidityInPercent**() |

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

|  |
| --- |
| [humidity] = **IDS\_ecu\_getHumidityInPercent**() |

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

|  |
| --- |
| double value = [Device].**Ecu\_GetHumidityInPercent**() |

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

|  |
| --- |
| getHumidityInPercent.vi |

|  |
| --- |
| **getPressureInHPa** Reads out the ECU measured air pressure in hPa. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| pressure | double pressure in hPa |

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

|  |
| --- |
| method: com.attocube.ecu.getPressureInHPa |
| params: [] |
| Result: [errNo, pressure] |

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

|  |
| --- |
| int **IDS\_ecu\_getPressureInHPa**(int deviceHandle, double\* pressure) |

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

|  |
| --- |
| pressure = **[dev].ecu.getPressureInHPa**() |

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

|  |
| --- |
| [pressure] = **IDS\_ecu\_getPressureInHPa**() |

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

|  |
| --- |
| double value = [Device].**Ecu\_GetPressureInHPa**() |

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

|  |
| --- |
| getPressureInHPa.vi |

|  |
| --- |
| **getRefractiveIndex** Reads out the ECU estimated refractive index for the current  ECU readings. To get the refractive index for other modes, please see  getRefractiveIndexForCompensation. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| rIndex | double refractive index |

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

|  |
| --- |
| method: com.attocube.ecu.getRefractiveIndex |
| params: [] |
| Result: [errNo, rIndex] |

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

|  |
| --- |
| int **IDS\_ecu\_getRefractiveIndex**(int deviceHandle, double\* rIndex) |

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

|  |
| --- |
| rIndex = **[dev].ecu.getRefractiveIndex**() |

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

|  |
| --- |
| [rIndex] = **IDS\_ecu\_getRefractiveIndex**() |

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

|  |
| --- |
| double value = [Device].**Ecu\_GetRefractiveIndex**() |

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

|  |
| --- |
| getRefractiveIndex.vi |

|  |
| --- |
| **getRefractiveIndexCompensationMode** Reads out the compensation mode (see below) which is currently used for the environmental compensation. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to get the mode for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| mode | int32 mode see defintion in set function |

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

|  |
| --- |
| method: com.attocube.ecu.getRefractiveIndexCompensationMode |
| params: [axis] |
| Result: [errNo, mode] |

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

|  |
| --- |
| int **IDS\_ecu\_getRefractiveIndexCompensationMode**(int deviceHandle, int axis, int\* mode) |

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

|  |
| --- |
| mode = **[dev].ecu.getRefractiveIndexCompensationMode**(axis) |

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

|  |
| --- |
| [mode] = **IDS\_ecu\_getRefractiveIndexCompensationMode**(axis) |

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

|  |
| --- |
| int value = [Device].**Ecu\_GetRefractiveIndexCompensationMode**(int axis) |

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

|  |
| --- |
| getRefractiveIndexCompensationMode.vi |

|  |
| --- |
| **getRefractiveIndexForCompensation** Reads out the refractive index used according to the current environmental compensation mode for this axis. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to get the refractive index for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| rIndex | double refractive index |

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

|  |
| --- |
| method: com.attocube.ecu.getRefractiveIndexForCompensation |
| params: [axis] |
| Result: [errNo, rIndex] |

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

|  |
| --- |
| int **IDS\_ecu\_getRefractiveIndexForCompensation**(int deviceHandle, int axis, double\* rIndex) |

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

|  |
| --- |
| rIndex = **[dev].ecu.getRefractiveIndexForCompensation**(axis) |

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

|  |
| --- |
| [rIndex] = **IDS\_ecu\_getRefractiveIndexForCompensation**(axis) |

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

|  |
| --- |
| double value = [Device].**Ecu\_GetRefractiveIndexForCompensation**(int axis) |

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

|  |
| --- |
| getRefractiveIndexForCompensation.vi |

|  |
| --- |
| **getTemperatureInDegrees** Reads out the ECU measured air temperature in degrees Celsius. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| temperature | double temperature in degrees C |

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

|  |
| --- |
| method: com.attocube.ecu.getTemperatureInDegrees |
| params: [] |
| Result: [errNo, temperature] |

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

|  |
| --- |
| int **IDS\_ecu\_getTemperatureInDegrees**(int deviceHandle, double\* temperature) |

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

|  |
| --- |
| temperature = **[dev].ecu.getTemperatureInDegrees**() |

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

|  |
| --- |
| [temperature] = **IDS\_ecu\_getTemperatureInDegrees**() |

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

|  |
| --- |
| double value = [Device].**Ecu\_GetTemperatureInDegrees**() |

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

|  |
| --- |
| getTemperatureInDegrees.vi |

|  |
| --- |
| **setRefractiveIndexCompensationMode** Sets the refractive index mode. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to set the mode for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| mode | can be 0 for direct ECU mode, 1 to take the manual values and calculate the refractive index from this or 2 to directly take the set refrative index |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

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

|  |
| --- |
| method: com.attocube.ecu.setRefractiveIndexCompensationMode |
| params: [axis, mode] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_ecu\_setRefractiveIndexCompensationMode**(int deviceHandle, int axis, int mode) |

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

|  |
| --- |
| **[dev].ecu.setRefractiveIndexCompensationMode**(axis, mode) |

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

|  |
| --- |
| [] = **IDS\_ecu\_setRefractiveIndexCompensationMode**(axis, mode) |

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

|  |
| --- |
| void value = [Device].**Ecu\_SetRefractiveIndexCompensationMode**(int axis, int mode) |

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

|  |
| --- |
| setRefractiveIndexCompensationMode.vi |

## 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 **IDS\_getLockStatus**(int deviceHandle, bool\* locked, bool\* authorized) |

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

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

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

|  |
| --- |
| [locked, authorized] = **IDS\_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 **IDS\_grantAccess**(int deviceHandle, const char\* password) |

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

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

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

|  |
| --- |
| [] = **IDS\_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 **IDS\_lock**(int deviceHandle, const char\* password) |

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

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

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

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

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

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

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

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

## Manual

|  |
| --- |
| **getHumidityInPercent** Reads out the manually configured humidity (compensation mode 1). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to get the humidity for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| humidity | double humidity in hPa |

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

|  |
| --- |
| method: com.attocube.ecu.manual.getHumidityInPercent |
| params: [axis] |
| Result: [errNo, humidity] |

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

|  |
| --- |
| int **IDS\_manual\_getHumidityInPercent**(int deviceHandle, int axis, double\* humidity) |

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

|  |
| --- |
| humidity = **[dev].manual.getHumidityInPercent**(axis) |

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

|  |
| --- |
| [humidity] = **IDS\_manual\_getHumidityInPercent**(axis) |

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

|  |
| --- |
| double value = [Device].**Manual\_GetHumidityInPercent**(int axis) |

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

|  |
| --- |
| getHumidityInPercent.vi |

|  |
| --- |
| **getPressureInHPa** Reads out the manually configured Pressure (compensation mode 1). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to get the pressure for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| pressure | double pressure in hPa |

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

|  |
| --- |
| method: com.attocube.ecu.manual.getPressureInHPa |
| params: [axis] |
| Result: [errNo, pressure] |

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

|  |
| --- |
| int **IDS\_manual\_getPressureInHPa**(int deviceHandle, int axis, double\* pressure) |

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

|  |
| --- |
| pressure = **[dev].manual.getPressureInHPa**(axis) |

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

|  |
| --- |
| [pressure] = **IDS\_manual\_getPressureInHPa**(axis) |

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

|  |
| --- |
| double value = [Device].**Manual\_GetPressureInHPa**(int axis) |

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

|  |
| --- |
| getPressureInHPa.vi |

|  |
| --- |
| **getRefractiveIndex** Reads out the manually configured value for the refractive index (compensation mode 2). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to get the mode for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| rindex | double refractive index |

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

|  |
| --- |
| method: com.attocube.ecu.manual.getRefractiveIndex |
| params: [axis] |
| Result: [errNo, rindex] |

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

|  |
| --- |
| int **IDS\_manual\_getRefractiveIndex**(int deviceHandle, int axis, double\* rindex) |

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

|  |
| --- |
| rindex = **[dev].manual.getRefractiveIndex**(axis) |

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

|  |
| --- |
| [rindex] = **IDS\_manual\_getRefractiveIndex**(axis) |

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

|  |
| --- |
| double value = [Device].**Manual\_GetRefractiveIndex**(int axis) |

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

|  |
| --- |
| getRefractiveIndex.vi |

|  |
| --- |
| **getTemperatureInDegrees** Reads out the manually configured Temperature (compensation mode 1). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to set the temperature for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |
| temperature | double temperature in degree celsius |

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

|  |
| --- |
| method: com.attocube.ecu.manual.getTemperatureInDegrees |
| params: [axis] |
| Result: [errNo, temperature] |

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

|  |
| --- |
| int **IDS\_manual\_getTemperatureInDegrees**(int deviceHandle, int axis, double\* temperature) |

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

|  |
| --- |
| temperature = **[dev].manual.getTemperatureInDegrees**(axis) |

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

|  |
| --- |
| [temperature] = **IDS\_manual\_getTemperatureInDegrees**(axis) |

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

|  |
| --- |
| double value = [Device].**Manual\_GetTemperatureInDegrees**(int axis) |

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

|  |
| --- |
| getTemperatureInDegrees.vi |

|  |
| --- |
| **setHumidityInPercent** Sets the manually configured Humidity (compensation mode 1). The input range is defined to 0 to 100 % (valid range for the Ciddor Equation). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to set the humidity for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| humidity | in Percent |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

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

|  |
| --- |
| method: com.attocube.ecu.manual.setHumidityInPercent |
| params: [axis, humidity] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_manual\_setHumidityInPercent**(int deviceHandle, int axis, double humidity) |

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

|  |
| --- |
| **[dev].manual.setHumidityInPercent**(axis, humidity) |

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

|  |
| --- |
| [] = **IDS\_manual\_setHumidityInPercent**(axis, humidity) |

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

|  |
| --- |
| void value = [Device].**Manual\_SetHumidityInPercent**(int axis, double humidity) |

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

|  |
| --- |
| setHumidityInPercent.vi |

|  |
| --- |
| **setPressureInHPa** Sets the manually configured Pressure (compensation mode 1). The input range is defined to 800 to 1200 hPa (valid range for the Ciddor Equation). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to set the pressure for- Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| pressure | in hPa |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

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

|  |
| --- |
| method: com.attocube.ecu.manual.setPressureInHPa |
| params: [axis, pressure] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_manual\_setPressureInHPa**(int deviceHandle, int axis, double pressure) |

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

|  |
| --- |
| **[dev].manual.setPressureInHPa**(axis, pressure) |

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

|  |
| --- |
| [] = **IDS\_manual\_setPressureInHPa**(axis, pressure) |

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

|  |
| --- |
| void value = [Device].**Manual\_SetPressureInHPa**(int axis, double pressure) |

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

|  |
| --- |
| setPressureInHPa.vi |

|  |
| --- |
| **setRefractiveIndex** Sets the refractive index for the direct mode (compensation mode 2). The input range is defined to be greater than 1. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to set the mode for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| rindex | refractive index |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

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

|  |
| --- |
| method: com.attocube.ecu.manual.setRefractiveIndex |
| params: [axis, rindex] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_manual\_setRefractiveIndex**(int deviceHandle, int axis, double rindex) |

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

|  |
| --- |
| **[dev].manual.setRefractiveIndex**(axis, rindex) |

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

|  |
| --- |
| [] = **IDS\_manual\_setRefractiveIndex**(axis, rindex) |

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

|  |
| --- |
| void value = [Device].**Manual\_SetRefractiveIndex**(int axis, double rindex) |

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

|  |
| --- |
| setRefractiveIndex.vi |

|  |
| --- |
| **setTemperatureInDegrees** Sets the manually configured Temperature (compensation mode 1). The input range is defined to -40 to +100 °C (valid range for the Ciddor Equation). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis to set the temperature for. Parameter has to be -1 for the moment, individual axes will be supported in the next firmware release |
| temperature | in degree celcius |
| Out | errNo | int32 Error code, if there was an error, otherwise 0 for ok |

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

|  |
| --- |
| method: com.attocube.ecu.manual.setTemperatureInDegrees |
| params: [axis, temperature] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_manual\_setTemperatureInDegrees**(int deviceHandle, int axis, double temperature) |

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

|  |
| --- |
| **[dev].manual.setTemperatureInDegrees**(axis, temperature) |

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

|  |
| --- |
| [] = **IDS\_manual\_setTemperatureInDegrees**(axis, temperature) |

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

|  |
| --- |
| void value = [Device].**Manual\_SetTemperatureInDegrees**(int axis, double temperature) |

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

|  |
| --- |
| setTemperatureInDegrees.vi |

## Nlc

|  |
| --- |
| **clearLut** Deactivates the LUT and removes it from the device |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS |
| Out | errNo | int32 Error number if one occured during deletion of the LUT |

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

|  |
| --- |
| method: com.attocube.ids.nlc.clearLut |
| params: [axis] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_nlc\_clearLut**(int deviceHandle, int axis) |

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

|  |
| --- |
| **[dev].nlc.clearLut**(axis) |

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

|  |
| --- |
| [] = **IDS\_nlc\_clearLut**(axis) |

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

|  |
| --- |
| void value = [Device].**Nlc\_ClearLut**(int axis) |

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

|  |
| --- |
| clearLut.vi |

|  |
| --- |
| **createLut** Creates a new LUT for the given axis. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS where the LUT should be generated |
| Out | errNo | int32 Error number if one occured while starting the creation. Errors during creation can be pulled by getLutStatus |

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

|  |
| --- |
| method: com.attocube.ids.nlc.createLut |
| params: [axis] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_nlc\_createLut**(int deviceHandle, int axis) |

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

|  |
| --- |
| **[dev].nlc.createLut**(axis) |

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

|  |
| --- |
| [] = **IDS\_nlc\_createLut**(axis) |

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

|  |
| --- |
| void value = [Device].**Nlc\_CreateLut**(int axis) |

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

|  |
| --- |
| createLut.vi |

|  |
| --- |
| **estimateNonlinearities** Estimates the nonlinearity error for the current device settings without changing or updating any settings. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS of which the nonlinearity error is to be estimated |
| Out | errNo | int32 Error number if one occured while estimating the nonlinearity error |

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

|  |
| --- |
| method: com.attocube.ids.nlc.estimateNonlinearities |
| params: [axis] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_nlc\_estimateNonlinearities**(int deviceHandle, int axis) |

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

|  |
| --- |
| **[dev].nlc.estimateNonlinearities**(axis) |

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

|  |
| --- |
| [] = **IDS\_nlc\_estimateNonlinearities**(axis) |

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

|  |
| --- |
| void value = [Device].**Nlc\_EstimateNonlinearities**(int axis) |

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

|  |
| --- |
| estimateNonlinearities.vi |

|  |
| --- |
| **getDynamicNormalization** Returns the normalization mode of a specific axis. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS of which the normalization mode is queried |
| Out | errNo | int32 Error number if an error occured while getting normalization mode |
| mode | int32 Normalization Mode 0 Dynamic normalization 1 Normalization frozen 2 Normalization mode determined by target velocity |

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

|  |
| --- |
| method: com.attocube.ids.nlc.getDynamicNormalization |
| params: [axis] |
| Result: [errNo, mode] |

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

|  |
| --- |
| int **IDS\_nlc\_getDynamicNormalization**(int deviceHandle, int axis, int\* mode) |

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

|  |
| --- |
| mode = **[dev].nlc.getDynamicNormalization**(axis) |

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

|  |
| --- |
| [mode] = **IDS\_nlc\_getDynamicNormalization**(axis) |

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

|  |
| --- |
| int value = [Device].**Nlc\_GetDynamicNormalization**(int axis) |

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

|  |
| --- |
| getDynamicNormalization.vi |

|  |
| --- |
| **getHistogram** Returns a histogram of the measured nonlinearity errors obtained from the last call of createLut or estimateNonlinearites. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS |
| Out | errNo | int32 Error number if one occured during retrieving the histogram |
| histogram | string Json dumped histogram array |

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

|  |
| --- |
| method: com.attocube.ids.nlc.getHistogram |
| params: [axis] |
| Result: [errNo, histogram] |

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

|  |
| --- |
| int **IDS\_nlc\_getHistogram**(int deviceHandle, int axis, char\* histogram, int size0) |

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

|  |
| --- |
| histogram = **[dev].nlc.getHistogram**(axis) |

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

|  |
| --- |
| [histogram] = **IDS\_nlc\_getHistogram**(axis) |

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

|  |
| --- |
| string value = [Device].**Nlc\_GetHistogram**(int axis) |

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

|  |
| --- |
| getHistogram.vi |

|  |
| --- |
| **getLut** Returns the LUT determined by createLut (which can be applied by setLutApplied). |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS |
| Out | errNo | int32 Error number if one occured during retrieving the LUT |
| lut | string Json dumped LUT array with 512 integers |

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

|  |
| --- |
| method: com.attocube.ids.nlc.getLut |
| params: [axis] |
| Result: [errNo, lut] |

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

|  |
| --- |
| int **IDS\_nlc\_getLut**(int deviceHandle, int axis, char\* lut, int size0) |

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

|  |
| --- |
| lut = **[dev].nlc.getLut**(axis) |

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

|  |
| --- |
| [lut] = **IDS\_nlc\_getLut**(axis) |

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

|  |
| --- |
| string value = [Device].**Nlc\_GetLut**(int axis) |

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

|  |
| --- |
| getLut.vi |

|  |
| --- |
| **getLutApplied** Returns whether a LUT is applied or not for a given axis. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS of which the LUT apply rule is queried |
| Out | errNo | int32 Error number if an error occured while quering the LUT apply rule |
| apply | boolean True, if LUT is applied on this axis else False |

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

|  |
| --- |
| method: com.attocube.ids.nlc.getLutApplied |
| params: [axis] |
| Result: [errNo, apply] |

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

|  |
| --- |
| int **IDS\_nlc\_getLutApplied**(int deviceHandle, int axis, bool\* apply) |

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

|  |
| --- |
| apply = **[dev].nlc.getLutApplied**(axis) |

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

|  |
| --- |
| [apply] = **IDS\_nlc\_getLutApplied**(axis) |

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

|  |
| --- |
| bool value = [Device].**Nlc\_GetLutApplied**(int axis) |

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

|  |
| --- |
| getLutApplied.vi |

|  |
| --- |
| **getLutStatus** Returns if a LUT is available and if warnings or errors occurred during creation. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS of which the status of the LUT should be returned |
| Out | creationWarning | int32 Error or warning number if one occured while creating the LUT, 0 in case of no error |
| status | boolean True, if a LUT exists else False |

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

|  |
| --- |
| method: com.attocube.ids.nlc.getLutStatus |
| params: [axis] |
| Result: [creationWarning, status] |

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

|  |
| --- |
| int **IDS\_nlc\_getLutStatus**(int deviceHandle, int axis, int\* creationWarning, bool\* status) |

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

|  |
| --- |
| creationWarning, status = **[dev].nlc.getLutStatus**(axis) |

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

|  |
| --- |
| [creationWarning, status] = **IDS\_nlc\_getLutStatus**(axis) |

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

|  |
| --- |
| Tuple<int,bool> value = [Device].**Nlc\_GetLutStatus**(int axis) |

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

|  |
| --- |
| getLutStatus.vi |

|  |
| --- |
| **getNonlinearityEstimation** Returns the LUT created by estimateNonlinearities (read-only mode) to compensate the nonlinearity error of the device for the current device settings. If no estimation was created an array of zeros is returned. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error number if one occured loading the LUT |
| lut | string Json dumped LUT array with 512 integers |

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

|  |
| --- |
| method: com.attocube.ids.nlc.getNonlinearityEstimation |
| params: [] |
| Result: [errNo, lut] |

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

|  |
| --- |
| int **IDS\_nlc\_getNonlinearityEstimation**(int deviceHandle, char\* lut, int size0) |

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

|  |
| --- |
| lut = **[dev].nlc.getNonlinearityEstimation**() |

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

|  |
| --- |
| [lut] = **IDS\_nlc\_getNonlinearityEstimation**() |

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

|  |
| --- |
| string value = [Device].**Nlc\_GetNonlinearityEstimation**() |

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

|  |
| --- |
| getNonlinearityEstimation.vi |

|  |
| --- |
| **getVelocityThresholds** Returns the threshold velocity (in µm/s) for mode 2 of setDynamicNormalization. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | int32 Error number if an error occured |
| velocityOn | int32 Velocity of the target in µm/s when to switch the normalization on (default: 10 µm/s) |
| velocityOff | int32 Velocity of the target in µm/s when to switch the normalization off (default: 5 µm/s) |

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

|  |
| --- |
| method: com.attocube.ids.nlc.getVelocityThresholds |
| params: [] |
| Result: [errNo, velocityOn, velocityOff] |

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

|  |
| --- |
| int **IDS\_nlc\_getVelocityThresholds**(int deviceHandle, int\* velocityOn, int\* velocityOff) |

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

|  |
| --- |
| velocityOn, velocityOff = **[dev].nlc.getVelocityThresholds**() |

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

|  |
| --- |
| [velocityOn, velocityOff] = **IDS\_nlc\_getVelocityThresholds**() |

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

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

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

|  |
| --- |
| getVelocityThresholds.vi |

|  |
| --- |
| **setDynamicNormalization** Sets the normalization mode of a specific axis. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS of which the normalization mode should be set |
| mode | Normalization Mode  0 Dynamic normalization (default)  1 Normalization frozen (for slow target drifts)  2 Automatic alternation between mode 0 and 1 depending on the target velocity |
| Out | errNo | int32 Error number if an error occured during switching the normalization mode |

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

|  |
| --- |
| method: com.attocube.ids.nlc.setDynamicNormalization |
| params: [axis, mode] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_nlc\_setDynamicNormalization**(int deviceHandle, int axis, int mode) |

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

|  |
| --- |
| **[dev].nlc.setDynamicNormalization**(axis, mode) |

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

|  |
| --- |
| [] = **IDS\_nlc\_setDynamicNormalization**(axis, mode) |

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

|  |
| --- |
| void value = [Device].**Nlc\_SetDynamicNormalization**(int axis, int mode) |

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

|  |
| --- |
| setDynamicNormalization.vi |

|  |
| --- |
| **setLut** Uploads a LUT for a specific axis (which can be applied by setLutApplied) |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS |
| lut | Json dumped LUT array with 512 integers |
| Out | errNo | int32 Error number if one occured during uploading the LUT |

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

|  |
| --- |
| method: com.attocube.ids.nlc.setLut |
| params: [axis, lut] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_nlc\_setLut**(int deviceHandle, int axis, const char\* lut) |

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

|  |
| --- |
| **[dev].nlc.setLut**(axis, lut) |

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

|  |
| --- |
| [] = **IDS\_nlc\_setLut**(axis, lut) |

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

|  |
| --- |
| void value = [Device].**Nlc\_SetLut**(int axis, string lut) |

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

|  |
| --- |
| setLut.vi |

|  |
| --- |
| **setLutApplied** Sets the apply rule for the given axis |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Axis [0|1|2] of the IDS of which the apply rule should be set |
| apply | True for applying a LUT, False for disabling a LUT |
| Out | errNo | int32 Error number if an error occured while enabling or disabling a LUT |

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

|  |
| --- |
| method: com.attocube.ids.nlc.setLutApplied |
| params: [axis, apply] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_nlc\_setLutApplied**(int deviceHandle, int axis, bool apply) |

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

|  |
| --- |
| **[dev].nlc.setLutApplied**(axis, apply) |

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

|  |
| --- |
| [] = **IDS\_nlc\_setLutApplied**(axis, apply) |

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

|  |
| --- |
| void value = [Device].**Nlc\_SetLutApplied**(int axis, bool apply) |

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

|  |
| --- |
| setLutApplied.vi |

|  |
| --- |
| **setVelocityThresholds** Sets the threshold velocity (in µm/s) for mode 2 of setDynamicNormalization. By default, the normalization in mode 2 is frozen for velocities below 5 µm/s and switched to dynamic mode for velocities above 10 µm/s. |

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

|  |  |  |
| --- | --- | --- |
| In | velocityOn | Velocity of the target in µm/s when to switch the normalization on (default: 10 µm/s) |
| velocityOff | Velocity of the target in µm/s when to switch the normalization off (default: 5 µm/s) |
| Out | errNo | int32 Error number if an error occured |

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

|  |
| --- |
| method: com.attocube.ids.nlc.setVelocityThresholds |
| params: [velocityOn, velocityOff] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_nlc\_setVelocityThresholds**(int deviceHandle, int velocityOn, int velocityOff) |

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

|  |
| --- |
| **[dev].nlc.setVelocityThresholds**(velocityOn, velocityOff) |

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

|  |
| --- |
| [] = **IDS\_nlc\_setVelocityThresholds**(velocityOn, velocityOff) |

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

|  |
| --- |
| void value = [Device].**Nlc\_SetVelocityThresholds**(int velocityOn, int velocityOff) |

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

|  |
| --- |
| setVelocityThresholds.vi |

## Pilotlaser

|  |
| --- |
| **disable** Disables the pilot laser. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.pilotlaser.disable |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_pilotlaser\_disable**(int deviceHandle) |

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

|  |
| --- |
| **[dev].pilotlaser.disable**() |

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

|  |
| --- |
| [] = **IDS\_pilotlaser\_disable**() |

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

|  |
| --- |
| void value = [Device].**Pilotlaser\_Disable**() |

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

|  |
| --- |
| disable.vi |

|  |
| --- |
| **enable** Enables the pilot laser. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.pilotlaser.enable |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_pilotlaser\_enable**(int deviceHandle) |

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

|  |
| --- |
| **[dev].pilotlaser.enable**() |

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

|  |
| --- |
| [] = **IDS\_pilotlaser\_enable**() |

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

|  |
| --- |
| void value = [Device].**Pilotlaser\_Enable**() |

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

|  |
| --- |
| enable.vi |

|  |
| --- |
| **getEnabled** Reads out whether the pilot laser is enabled or not. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| enable | true = enabled; false = disabled |

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

|  |
| --- |
| method: com.attocube.ids.pilotlaser.getEnabled |
| params: [] |
| Result: [errNo, enable] |

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

|  |
| --- |
| int **IDS\_pilotlaser\_getEnabled**(int deviceHandle, bool\* enable) |

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

|  |
| --- |
| enable = **[dev].pilotlaser.getEnabled**() |

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

|  |
| --- |
| [enable] = **IDS\_pilotlaser\_getEnabled**() |

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

|  |
| --- |
| bool value = [Device].**Pilotlaser\_GetEnabled**() |

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

|  |
| --- |
| getEnabled.vi |

## Realtime

|  |
| --- |
| **AafIsEnabled** Checks if the anti-aliasing filter is enabled. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| enabled | false: Anti-Aliasing Filter is disabled  true: Anti-Aliasing Filter is enabled |

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

|  |
| --- |
| method: com.attocube.ids.realtime.AafIsEnabled |
| params: [] |
| Result: [errNo, enabled] |

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

|  |
| --- |
| int **IDS\_realtime\_AafIsEnabled**(int deviceHandle, int\* enabled) |

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

|  |
| --- |
| enabled = **[dev].realtime.AafIsEnabled**() |

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

|  |
| --- |
| [enabled] = **IDS\_realtime\_AafIsEnabled**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_AafIsEnabled**() |

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

|  |
| --- |
| AafIsEnabled.vi |

|  |
| --- |
| **apply** Applies new real time settings. Necessary after JSON realtime set commands. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.apply |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_apply**(int deviceHandle) |

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

|  |
| --- |
| **[dev].realtime.apply**() |

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

|  |
| --- |
| [] = **IDS\_realtime\_apply**() |

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

|  |
| --- |
| void value = [Device].**Realtime\_Apply**() |

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

|  |
| --- |
| apply.vi |

|  |
| --- |
| **disableTestChannel** Disables the test channel. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.disableTestChannel |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_disableTestChannel**(int deviceHandle) |

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

|  |
| --- |
| **[dev].realtime.disableTestChannel**() |

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

|  |
| --- |
| [] = **IDS\_realtime\_disableTestChannel**() |

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

|  |
| --- |
| void value = [Device].**Realtime\_DisableTestChannel**() |

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

|  |
| --- |
| disableTestChannel.vi |

|  |
| --- |
| **discard** Discards new real time settings. Necessary after JSON set commands in case of failure. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.discard |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_discard**(int deviceHandle) |

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

|  |
| --- |
| **[dev].realtime.discard**() |

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

|  |
| --- |
| [] = **IDS\_realtime\_discard**() |

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

|  |
| --- |
| void value = [Device].**Realtime\_Discard**() |

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

|  |
| --- |
| discard.vi |

|  |
| --- |
| **enableTestChannel** Enables the Test Channel, which can be used for estimating the maximum signal range. |

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

|  |  |  |
| --- | --- | --- |
| In | axis | Test Channel Master Axis |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.enableTestChannel |
| params: [axis] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_enableTestChannel**(int deviceHandle, int axis) |

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

|  |
| --- |
| **[dev].realtime.enableTestChannel**(axis) |

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

|  |
| --- |
| [] = **IDS\_realtime\_enableTestChannel**(axis) |

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

|  |
| --- |
| void value = [Device].**Realtime\_EnableTestChannel**(int axis) |

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

|  |
| --- |
| enableTestChannel.vi |

|  |
| --- |
| **getAafAttenuation** Returns the current attenuation at f\_nyquist of the anti-aliasing filter. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| attenuation | [3-30] dB m f\_nyquist |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getAafAttenuation |
| params: [] |
| Result: [errNo, attenuation] |

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

|  |
| --- |
| int **IDS\_realtime\_getAafAttenuation**(int deviceHandle, int\* attenuation) |

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

|  |
| --- |
| attenuation = **[dev].realtime.getAafAttenuation**() |

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

|  |
| --- |
| [attenuation] = **IDS\_realtime\_getAafAttenuation**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetAafAttenuation**() |

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

|  |
| --- |
| getAafAttenuation.vi |

|  |
| --- |
| **getAafEnabled** Checks if the anti-aliasing filter is enabled. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| enabled | 0 - the Anti-Aliasing Filter is disabled   1 - the Anti-Aliasing Filter is enabled |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getAafEnabled |
| params: [] |
| Result: [errNo, enabled] |

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

|  |
| --- |
| int **IDS\_realtime\_getAafEnabled**(int deviceHandle, int\* enabled) |

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

|  |
| --- |
| enabled = **[dev].realtime.getAafEnabled**() |

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

|  |
| --- |
| [enabled] = **IDS\_realtime\_getAafEnabled**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetAafEnabled**() |

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

|  |
| --- |
| getAafEnabled.vi |

|  |
| --- |
| **getAafWindow** Returns the current filter window of the anti-aliasing filter. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| window | 0 = Rectangular,  1 = Cosine,  2 = Cosine^2,  3 = Hamming,  4 = Raised Cosine,  5 = Automatic |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getAafWindow |
| params: [] |
| Result: [errNo, window] |

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

|  |
| --- |
| int **IDS\_realtime\_getAafWindow**(int deviceHandle, int\* window) |

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

|  |
| --- |
| window = **[dev].realtime.getAafWindow**() |

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

|  |
| --- |
| [window] = **IDS\_realtime\_getAafWindow**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetAafWindow**() |

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

|  |
| --- |
| getAafWindow.vi |

|  |
| --- |
| **getHighPassCutOffFreq** Reads out the high pass filter number of Linear/Analog output mode. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| value | N, Linear Analog High Pass Cut-Off freqency is 1600/2^N kHz, with N \\in [1,24] |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getHighPassCutOffFreq |
| params: [] |
| Result: [errNo, value] |

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

|  |
| --- |
| int **IDS\_realtime\_getHighPassCutOffFreq**(int deviceHandle, int\* value) |

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

|  |
| --- |
| value = **[dev].realtime.getHighPassCutOffFreq**() |

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

|  |
| --- |
| [value] = **IDS\_realtime\_getHighPassCutOffFreq**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetHighPassCutOffFreq**() |

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

|  |
| --- |
| getHighPassCutOffFreq.vi |

|  |
| --- |
| **getLinearRange** Reads out the range number of Linear/Analog output mode. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| rangenumber | N, Linear Analog Range is +-2^(N+11) pm, with N \\in [0, 34] |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getLinearRange |
| params: [] |
| Result: [errNo, rangenumber] |

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

|  |
| --- |
| int **IDS\_realtime\_getLinearRange**(int deviceHandle, int\* rangenumber) |

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

|  |
| --- |
| rangenumber = **[dev].realtime.getLinearRange**() |

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

|  |
| --- |
| [rangenumber] = **IDS\_realtime\_getLinearRange**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetLinearRange**() |

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

|  |
| --- |
| getLinearRange.vi |

|  |
| --- |
| **getPeriodHsslClk** Reads out the HSSL period clock. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| period | Period in the Range of [40ns..10200ns] |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getPeriodHsslClk |
| params: [] |
| Result: [errNo, period] |

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

|  |
| --- |
| int **IDS\_realtime\_getPeriodHsslClk**(int deviceHandle, int\* period) |

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

|  |
| --- |
| period = **[dev].realtime.getPeriodHsslClk**() |

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

|  |
| --- |
| [period] = **IDS\_realtime\_getPeriodHsslClk**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetPeriodHsslClk**() |

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

|  |
| --- |
| getPeriodHsslClk.vi |

|  |
| --- |
| **getPeriodHsslGap** Reads out the HSSL period gap. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| gap | Number of clocks |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getPeriodHsslGap |
| params: [] |
| Result: [errNo, gap] |

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

|  |
| --- |
| int **IDS\_realtime\_getPeriodHsslGap**(int deviceHandle, int\* gap) |

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

|  |
| --- |
| gap = **[dev].realtime.getPeriodHsslGap**() |

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

|  |
| --- |
| [gap] = **IDS\_realtime\_getPeriodHsslGap**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetPeriodHsslGap**() |

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

|  |
| --- |
| getPeriodHsslGap.vi |

|  |
| --- |
| **getPeriodSinCosClk** Reads out the Sine-Cosine and AquadB period clock. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| period | 40ns to 10200ns |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getPeriodSinCosClk |
| params: [] |
| Result: [errNo, period] |

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

|  |
| --- |
| int **IDS\_realtime\_getPeriodSinCosClk**(int deviceHandle, int\* period) |

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

|  |
| --- |
| period = **[dev].realtime.getPeriodSinCosClk**() |

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

|  |
| --- |
| [period] = **IDS\_realtime\_getPeriodSinCosClk**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetPeriodSinCosClk**() |

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

|  |
| --- |
| getPeriodSinCosClk.vi |

|  |
| --- |
| **getResolutionBissC** Reads out the BissC resolution. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| resolution | 1pm to 65535pm |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getResolutionBissC |
| params: [] |
| Result: [errNo, resolution] |

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

|  |
| --- |
| int **IDS\_realtime\_getResolutionBissC**(int deviceHandle, int\* resolution) |

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

|  |
| --- |
| resolution = **[dev].realtime.getResolutionBissC**() |

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

|  |
| --- |
| [resolution] = **IDS\_realtime\_getResolutionBissC**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetResolutionBissC**() |

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

|  |
| --- |
| getResolutionBissC.vi |

|  |
| --- |
| **getResolutionHsslHigh** Reads out the HSSL resolution high bit. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| resolution | Resolution in the Range of [0..46] |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getResolutionHsslHigh |
| params: [] |
| Result: [errNo, resolution] |

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

|  |
| --- |
| int **IDS\_realtime\_getResolutionHsslHigh**(int deviceHandle, int\* resolution) |

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

|  |
| --- |
| resolution = **[dev].realtime.getResolutionHsslHigh**() |

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

|  |
| --- |
| [resolution] = **IDS\_realtime\_getResolutionHsslHigh**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetResolutionHsslHigh**() |

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

|  |
| --- |
| getResolutionHsslHigh.vi |

|  |
| --- |
| **getResolutionHsslLow** Reads out the HSSL resolution low bit.# |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| resolution | Resolution in the range of [0..46] |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getResolutionHsslLow |
| params: [] |
| Result: [errNo, resolution] |

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

|  |
| --- |
| int **IDS\_realtime\_getResolutionHsslLow**(int deviceHandle, int\* resolution) |

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

|  |
| --- |
| resolution = **[dev].realtime.getResolutionHsslLow**() |

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

|  |
| --- |
| [resolution] = **IDS\_realtime\_getResolutionHsslLow**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetResolutionHsslLow**() |

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

|  |
| --- |
| getResolutionHsslLow.vi |

|  |
| --- |
| **getResolutionSinCos** Reads out the Sine-Cosine and AquadB resolution. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| resolution | 1pm to 65535pm |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getResolutionSinCos |
| params: [] |
| Result: [errNo, resolution] |

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

|  |
| --- |
| int **IDS\_realtime\_getResolutionSinCos**(int deviceHandle, int\* resolution) |

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

|  |
| --- |
| resolution = **[dev].realtime.getResolutionSinCos**() |

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

|  |
| --- |
| [resolution] = **IDS\_realtime\_getResolutionSinCos**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetResolutionSinCos**() |

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

|  |
| --- |
| getResolutionSinCos.vi |

|  |
| --- |
| **getRtDistanceMode** Reads out the distance mode. Depending on the realtime output mode, the mode can  be Displacement (returns 1), Absolute Distance (returns 2) or Vibrometry (returns 3). |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| linearmode | 1 = Displacement (Available in HSSL mode and Linear Mode)  2 = Absolute Distance (Available in HSSL mode only)  3 = Vibrometry (Available in Linear mode) |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getRtDistanceMode |
| params: [] |
| Result: [errNo, linearmode] |

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

|  |
| --- |
| int **IDS\_realtime\_getRtDistanceMode**(int deviceHandle, int\* linearmode) |

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

|  |
| --- |
| linearmode = **[dev].realtime.getRtDistanceMode**() |

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

|  |
| --- |
| [linearmode] = **IDS\_realtime\_getRtDistanceMode**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetRtDistanceMode**() |

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

|  |
| --- |
| getRtDistanceMode.vi |

|  |
| --- |
| **getRtOutMode** Reads out the current realtime output mode. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| rtOutMode | 0 = HSSL (TTL), 1 = HSSL (LVDS), 2 = AquadB (TTL),  3 = AquadB (LVDS), 4 = SinCos (TTL Error Signal),  5 = SinCos (LVDS Error Signal), 6 = Linear (TTL), 7 = Linear (LVDS),  8 = BiSS-C, 9 = Deactivated |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getRtOutMode |
| params: [] |
| Result: [errNo, rtOutMode] |

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

|  |
| --- |
| int **IDS\_realtime\_getRtOutMode**(int deviceHandle, int\* rtOutMode) |

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

|  |
| --- |
| rtOutMode = **[dev].realtime.getRtOutMode**() |

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

|  |
| --- |
| [rtOutMode] = **IDS\_realtime\_getRtOutMode**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetRtOutMode**() |

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

|  |
| --- |
| getRtOutMode.vi |

|  |
| --- |
| **getTestChannelEnabled** Checks if the test channel is enabled |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| enabled | true = enabled, false = disabled |

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

|  |
| --- |
| method: com.attocube.ids.realtime.getTestChannelEnabled |
| params: [] |
| Result: [errNo, enabled] |

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

|  |
| --- |
| int **IDS\_realtime\_getTestChannelEnabled**(int deviceHandle, int\* enabled) |

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

|  |
| --- |
| enabled = **[dev].realtime.getTestChannelEnabled**() |

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

|  |
| --- |
| [enabled] = **IDS\_realtime\_getTestChannelEnabled**() |

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

|  |
| --- |
| int value = [Device].**Realtime\_GetTestChannelEnabled**() |

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

|  |
| --- |
| getTestChannelEnabled.vi |

|  |
| --- |
| **setAaf** Sets the anti-aliasing filter with assigned filter window. |

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

|  |  |  |
| --- | --- | --- |
| In | enabled | 0 - disables the Anti-Aliasing Filter   1 - enables the Anti-Aliasing Filter |
| attenuation | [3-30] dB m f\_nyquist |
| window | 0 = Rectangular,  1 = Cosine,  2 = Cosine^2,  3 = Hamming,  4 = Raised Cosine,  5 = Automatic |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setAaf |
| params: [enabled, attenuation, window] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setAaf**(int deviceHandle, int enabled, int attenuation, int window) |

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

|  |
| --- |
| **[dev].realtime.setAaf**(enabled, attenuation, window) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setAaf**(enabled, attenuation, window) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetAaf**(int enabled, int attenuation, int window) |

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

|  |
| --- |
| setAaf.vi |

|  |
| --- |
| **setHighPassCutOffFreq** Sets the high pass filter number of Linear/Analog output mode. |

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

|  |  |  |
| --- | --- | --- |
| In | value | N, Linear Analog High Pass Cut-Off freqency is 1600/2^N kHz, with N \\in [1,24] |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setHighPassCutOffFreq |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setHighPassCutOffFreq**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].realtime.setHighPassCutOffFreq**(value) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setHighPassCutOffFreq**(value) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetHighPassCutOffFreq**(int value) |

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

|  |
| --- |
| setHighPassCutOffFreq.vi |

|  |
| --- |
| **setLinearRange** Sets the range number of Linear/Analog output mode. |

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

|  |  |  |
| --- | --- | --- |
| In | rangenumber | N, Linear Analog Range is +-2^(N+11) pm, with N \\in [0, 34] |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setLinearRange |
| params: [rangenumber] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setLinearRange**(int deviceHandle, int rangenumber) |

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

|  |
| --- |
| **[dev].realtime.setLinearRange**(rangenumber) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setLinearRange**(rangenumber) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetLinearRange**(int rangenumber) |

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

|  |
| --- |
| setLinearRange.vi |

|  |
| --- |
| **setPeriodHsslClk** Set the HSSL period clock. The value has to be a multiple of 40ns. If not, the value automatically is rounded. |

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

|  |  |  |
| --- | --- | --- |
| In | period | Period in the Range of [40ns..10200ns] |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setPeriodHsslClk |
| params: [period] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setPeriodHsslClk**(int deviceHandle, int period) |

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

|  |
| --- |
| **[dev].realtime.setPeriodHsslClk**(period) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setPeriodHsslClk**(period) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetPeriodHsslClk**(int period) |

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

|  |
| --- |
| setPeriodHsslClk.vi |

|  |
| --- |
| **setPeriodHsslGap** Set the HSSL gap. |

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

|  |  |  |
| --- | --- | --- |
| In | value | Number of clocks |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setPeriodHsslGap |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setPeriodHsslGap**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].realtime.setPeriodHsslGap**(value) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setPeriodHsslGap**(value) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetPeriodHsslGap**(int value) |

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

|  |
| --- |
| setPeriodHsslGap.vi |

|  |
| --- |
| **setPeriodSinCosClk** Sets the Sine-Cosine and AquadB period clock. The value has to be a multiple of 40ns. If not, the value automatically is rounded. |

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

|  |  |  |
| --- | --- | --- |
| In | value | period 40ns to 10200ns |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setPeriodSinCosClk |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setPeriodSinCosClk**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].realtime.setPeriodSinCosClk**(value) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setPeriodSinCosClk**(value) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetPeriodSinCosClk**(int value) |

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

|  |
| --- |
| setPeriodSinCosClk.vi |

|  |
| --- |
| **setResolutionBissC** Sets the BissC resolution. |

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

|  |  |  |
| --- | --- | --- |
| In | value | resolution 1pm to 65535pm |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setResolutionBissC |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setResolutionBissC**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].realtime.setResolutionBissC**(value) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setResolutionBissC**(value) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetResolutionBissC**(int value) |

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

|  |
| --- |
| setResolutionBissC.vi |

|  |
| --- |
| **setResolutionHsslHigh** Sets the HSSL resolution high bit. |

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

|  |  |  |
| --- | --- | --- |
| In | value | Resolution in the Range of [0..46] |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setResolutionHsslHigh |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setResolutionHsslHigh**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].realtime.setResolutionHsslHigh**(value) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setResolutionHsslHigh**(value) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetResolutionHsslHigh**(int value) |

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

|  |
| --- |
| setResolutionHsslHigh.vi |

|  |
| --- |
| **setResolutionHsslLow** Sets the HSSL resolution low bit. |

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

|  |  |  |
| --- | --- | --- |
| In | value | Resolution in the Range of [0..46] |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setResolutionHsslLow |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setResolutionHsslLow**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].realtime.setResolutionHsslLow**(value) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setResolutionHsslLow**(value) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetResolutionHsslLow**(int value) |

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

|  |
| --- |
| setResolutionHsslLow.vi |

|  |
| --- |
| **setResolutionSinCos** Sets the Sine-Cosine and AquadB resolution. |

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

|  |  |  |
| --- | --- | --- |
| In | value | resolution 1pm to 65535pm |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setResolutionSinCos |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setResolutionSinCos**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].realtime.setResolutionSinCos**(value) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setResolutionSinCos**(value) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetResolutionSinCos**(int value) |

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

|  |
| --- |
| setResolutionSinCos.vi |

|  |
| --- |
| **setRtDistanceMode** Sets the distance mode. Depending on the configuration of the IDS the mode can be  Displacement (returns 1), Absolute Distance (returns 2) or Vibrometry (returns 3). |

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

|  |  |  |
| --- | --- | --- |
| In | value | 1 = Displacement (HSSL mode and Linear Mode)  2 = Absolute Distance (HSSL mode only)  3 = Vibrometry (Linear mode) |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setRtDistanceMode |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setRtDistanceMode**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].realtime.setRtDistanceMode**(value) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setRtDistanceMode**(value) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetRtDistanceMode**(int value) |

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

|  |
| --- |
| setRtDistanceMode.vi |

|  |
| --- |
| **setRtOutMode** Sets the real time output mode. |

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

|  |  |  |
| --- | --- | --- |
| In | value | rtOutMode 0 = HSSL (TTL), 1 = HSSL (LVDS), 2 = AquadB (TTL),  3 = AquadB (LVDS), 4 = SinCos (TTL Error Signal),  5 = SinCos (LVDS Error Signal), 6 = Linear (TTL), 7 = Linear (LVDS),  8 = BiSS-C, 9 = Deactivated |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.realtime.setRtOutMode |
| params: [value] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_realtime\_setRtOutMode**(int deviceHandle, int value) |

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

|  |
| --- |
| **[dev].realtime.setRtOutMode**(value) |

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

|  |
| --- |
| [] = **IDS\_realtime\_setRtOutMode**(value) |

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

|  |
| --- |
| void value = [Device].**Realtime\_SetRtOutMode**(int value) |

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

|  |
| --- |
| setRtOutMode.vi |

## System

|  |
| --- |
| **getCurrentMode** Reads out the current IDS system state. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| mode | Values: "system idle", "measurement starting", "measurement running", "optics alignment starting", "optics alignment running", "test channels enabled" |

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

|  |
| --- |
| method: com.attocube.ids.system.getCurrentMode |
| params: [] |
| Result: [errNo, mode] |

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

|  |
| --- |
| int **IDS\_system\_getCurrentMode**(int deviceHandle, char\* mode, int size0) |

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

|  |
| --- |
| mode = **[dev].system.getCurrentMode**() |

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

|  |
| --- |
| [mode] = **IDS\_system\_getCurrentMode**() |

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

|  |
| --- |
| string value = [Device].**System\_GetCurrentMode**() |

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

|  |
| --- |
| system\_getCurrentMode.vi |

|  |
| --- |
| **getDeviceType** Reads out the IDS device type. For differences between the device types please refer to the IDS User Manual. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| type | Type of IDS (e.g. "IDS3010") |

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

|  |
| --- |
| method: com.attocube.ids.system.getDeviceType |
| params: [] |
| Result: [errNo, type] |

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

|  |
| --- |
| int **IDS\_system\_getDeviceType**(int deviceHandle, char\* type, int size0) |

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

|  |
| --- |
| type = **[dev].system.getDeviceType**() |

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

|  |
| --- |
| [type] = **IDS\_system\_getDeviceType**() |

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

|  |
| --- |
| string value = [Device].**System\_GetDeviceType**() |

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

|  |
| --- |
| system\_getDeviceType.vi |

|  |
| --- |
| **getFeaturesName** Converts the IDS feature number to its corresponding name. |

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

|  |  |  |
| --- | --- | --- |
| In | featurenumber | Number of feature |
| Out | errNo | errNo |
| names | The name of the corresponding feature |

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

|  |
| --- |
| method: com.attocube.ids.system.getFeaturesName |
| params: [featurenumber] |
| Result: [errNo, names] |

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

|  |
| --- |
| int **IDS\_system\_getFeaturesName**(int deviceHandle, int featurenumber, char\* names, int size0) |

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

|  |
| --- |
| names = **[dev].system.getFeaturesName**(featurenumber) |

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

|  |
| --- |
| [names] = **IDS\_system\_getFeaturesName**(featurenumber) |

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

|  |
| --- |
| string value = [Device].**System\_GetFeaturesName**(int featurenumber) |

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

|  |
| --- |
| system\_getFeaturesName.vi |

|  |
| --- |
| **getFpgaVersion** Reads out the IDS FPGA version. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| version | Version in the form X.Y.Z |

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

|  |
| --- |
| method: com.attocube.ids.system.getFpgaVersion |
| params: [] |
| Result: [errNo, version] |

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

|  |
| --- |
| int **IDS\_system\_getFpgaVersion**(int deviceHandle, char\* version, int size0) |

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

|  |
| --- |
| version = **[dev].system.getFpgaVersion**() |

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

|  |
| --- |
| [version] = **IDS\_system\_getFpgaVersion**() |

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

|  |
| --- |
| string value = [Device].**System\_GetFpgaVersion**() |

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

|  |
| --- |
| system\_getFpgaVersion.vi |

|  |
| --- |
| **getInitMode** Returns the Initialization mode. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| mode | 0 = High Accuracy Initialization; 1 = Quick Initialization |

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

|  |
| --- |
| method: com.attocube.ids.system.getInitMode |
| params: [] |
| Result: [errNo, mode] |

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

|  |
| --- |
| int **IDS\_system\_getInitMode**(int deviceHandle, int\* mode) |

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

|  |
| --- |
| mode = **[dev].system.getInitMode**() |

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

|  |
| --- |
| [mode] = **IDS\_system\_getInitMode**() |

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

|  |
| --- |
| int value = [Device].**System\_GetInitMode**() |

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

|  |
| --- |
| system\_getInitMode.vi |

|  |
| --- |
| **getNbrFeaturesActivated** Reads out the amount of activated features activated on the IDS. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |
| nbr | Gives the number of activated features. |

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

|  |
| --- |
| method: com.attocube.ids.system.getNbrFeaturesActivated |
| params: [] |
| Result: [errNo, nbr] |

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

|  |
| --- |
| int **IDS\_system\_getNbrFeaturesActivated**(int deviceHandle, int\* nbr) |

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

|  |
| --- |
| nbr = **[dev].system.getNbrFeaturesActivated**() |

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

|  |
| --- |
| [nbr] = **IDS\_system\_getNbrFeaturesActivated**() |

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

|  |
| --- |
| int value = [Device].**System\_GetNbrFeaturesActivated**() |

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

|  |
| --- |
| system\_getNbrFeaturesActivated.vi |

|  |
| --- |
| **getSystemError** Reads out the system error. The function returns an integer number which represents  the error. The number can be converted into a string using the errorNumberToString  function. Use this function to query errors occured while starting and running measurements and alignments. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.system.getSystemError |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_system\_getSystemError**(int deviceHandle) |

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

|  |
| --- |
| **[dev].system.getSystemError**() |

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

|  |
| --- |
| [] = **IDS\_system\_getSystemError**() |

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

|  |
| --- |
| void value = [Device].**System\_GetSystemError**() |

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

|  |
| --- |
| system\_getSystemError.vi |

|  |
| --- |
| **resetAxes** Resets the position value of all measurement axes to zero. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.system.resetAxes |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_system\_resetAxes**(int deviceHandle) |

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

|  |
| --- |
| **[dev].system.resetAxes**() |

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

|  |
| --- |
| [] = **IDS\_system\_resetAxes**() |

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

|  |
| --- |
| void value = [Device].**System\_ResetAxes**() |

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

|  |
| --- |
| system\_resetAxes.vi |

|  |
| --- |
| **resetAxis** Resets the position value of a specific measurement axis to zero. |

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

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

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

|  |
| --- |
| method: com.attocube.ids.system.resetAxis |
| params: [axis] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_system\_resetAxis**(int deviceHandle, int axis) |

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

|  |
| --- |
| **[dev].system.resetAxis**(axis) |

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

|  |
| --- |
| [] = **IDS\_system\_resetAxis**(axis) |

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

|  |
| --- |
| void value = [Device].**System\_ResetAxis**(int axis) |

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

|  |
| --- |
| system\_resetAxis.vi |

|  |
| --- |
| **resetError** Resets a measurement error that can have occurred with the aim to continue the interrupted measurement. It is configurable if an additional renormalization process (please refer to the IDS User Manual) should be performed or not.  This function can be used for two concerns: 1. FALSE: This function only clears displacement errors (e.g. after a beam interruption) of all three axes, while the displacement measurement is running. 2. TRUE: This function can be utilized to normalize the Lissajous-Figure of all three optical axes during the running displacement measurement by sweeping the laser temperature and to clear all displacement errors. This normalization process takes around 14-20 seconds. This function can be used, for example, in two main applications: A. The alignment contrast decreases due to the angular change of the target and/or sensor head without any displacements (see Figure 45). Using this function, the Lissajous-Figure of each measurement axis gets normalized and high-resolution measurements are guaranteed. B. After changing the optical components as, for example, the retro reflector this function can be used to normalize the Lissajous-Figure after completely losing the signal (see Figure 46). Attention: Depending on the Boolean input parameter performRenormalisation, it can be decided, if the renormalization process should be executed or skipped. If it is executed, the IDS system needs around 14-20 seconds to get back to the measurement mode. It is also important to comment that the recommended action after an error is to stop and to restart the displacement measurement. Moreover, losing displacement values due to an occurred error the internal absolute position can be different to the real absolute position and this can could result in dynamic movement errors or a wrong ECU compensation during long-term displacement measurements. |

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

|  |  |  |
| --- | --- | --- |
| In | perform | renormalization |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.system.resetError |
| params: [perform] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_system\_resetError**(int deviceHandle, bool perform) |

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

|  |
| --- |
| **[dev].system.resetError**(perform) |

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

|  |
| --- |
| [] = **IDS\_system\_resetError**(perform) |

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

|  |
| --- |
| void value = [Device].**System\_ResetError**(bool perform) |

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

|  |
| --- |
| system\_resetError.vi |

|  |
| --- |
| **setInitMode** Sets the mode for the initialization procedure that is performed when starting a measurement. |

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

|  |  |  |
| --- | --- | --- |
| In | mode | 0 = High Accuracy Initialization; 1 = Quick Initialization |
| Out | errNo | errNo |

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

|  |
| --- |
| method: com.attocube.ids.system.setInitMode |
| params: [mode] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_system\_setInitMode**(int deviceHandle, int mode) |

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

|  |
| --- |
| **[dev].system.setInitMode**(mode) |

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

|  |
| --- |
| [] = **IDS\_system\_setInitMode**(mode) |

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

|  |
| --- |
| void value = [Device].**System\_SetInitMode**(int mode) |

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

|  |
| --- |
| system\_setInitMode.vi |

|  |
| --- |
| **startMeasurement** Starts the displacement measurement system state.  Please query errors during starting and running measurements by system.getSystemError |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | Error on initiating the system mode change |

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

|  |
| --- |
| method: com.attocube.ids.system.startMeasurement |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_system\_startMeasurement**(int deviceHandle) |

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

|  |
| --- |
| **[dev].system.startMeasurement**() |

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

|  |
| --- |
| [] = **IDS\_system\_startMeasurement**() |

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

|  |
| --- |
| void value = [Device].**System\_StartMeasurement**() |

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

|  |
| --- |
| system\_startMeasurement.vi |

|  |
| --- |
| **startOpticsAlignment** Starts the optical alignment system state.  Please query errors during starting and running alignments by system.getSystemError |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | Error on initiating the system mode change |

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

|  |
| --- |
| method: com.attocube.ids.system.startOpticsAlignment |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_system\_startOpticsAlignment**(int deviceHandle) |

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

|  |
| --- |
| **[dev].system.startOpticsAlignment**() |

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

|  |
| --- |
| [] = **IDS\_system\_startOpticsAlignment**() |

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

|  |
| --- |
| void value = [Device].**System\_StartOpticsAlignment**() |

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

|  |
| --- |
| system\_startOpticsAlignment.vi |

|  |
| --- |
| **stopMeasurement** Stops the displacement measurement system state. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | Error on initiating the system mode change |

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

|  |
| --- |
| method: com.attocube.ids.system.stopMeasurement |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_system\_stopMeasurement**(int deviceHandle) |

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

|  |
| --- |
| **[dev].system.stopMeasurement**() |

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

|  |
| --- |
| [] = **IDS\_system\_stopMeasurement**() |

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

|  |
| --- |
| void value = [Device].**System\_StopMeasurement**() |

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

|  |
| --- |
| system\_stopMeasurement.vi |

|  |
| --- |
| **stopOpticsAlignment** Stops the optical alignment system state. |

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

|  |  |  |
| --- | --- | --- |
| Out | errNo | Error on initiating the system mode change |

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

|  |
| --- |
| method: com.attocube.ids.system.stopOpticsAlignment |
| params: [] |
| Result: [errNo] |

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

|  |
| --- |
| int **IDS\_system\_stopOpticsAlignment**(int deviceHandle) |

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

|  |
| --- |
| **[dev].system.stopOpticsAlignment**() |

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

|  |
| --- |
| [] = **IDS\_system\_stopOpticsAlignment**() |

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

|  |
| --- |
| void value = [Device].**System\_StopOpticsAlignment**() |

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

|  |
| --- |
| system\_stopOpticsAlignment.vi |

## Access

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

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

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

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

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

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

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

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

|  |
| --- |
| unlock.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 |

|  |
| --- |
| **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 |

## 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 |