

User Manual

IDS3010



Version: Manual Version 2.2.0
Modified: 23.02.2016
Products: attoIDS3010

User Manual

attoIDS3010

attocube systems AG, Königinstrasse 11a (Rgb), D - 80539 Munich, Germany
Phone: +49 89-2877 8090 Fax: +49 89-2877 80919
E-Mail: info@attocube.com www.attocube.com

For technical queries, contact:
support@attocube.com

attocube systems office Munich:
Phone +49 89 2877 809 0
Fax +49 89 2877 809 19

I. Table of Contents

I.	Table of Contents	4
II.	General handling.....	6
	II.1. Unpacking	6
	II.2. Electrical interface.....	6
	II.2.1. Power supply.....	6
	II.3. Optical interface.....	7
	II.4. Connecting the sensor heads.....	7
	II.5. Working in a standard room environment.....	8
III.	Safety instructions	9
	III.1. Laser	9
	III.2. Infrared Laser	9
	III.3. Visible Laser	9
IV.	Installation	10
	IV.1. Mechanical Installation.....	10
	IV.2. Electrical Installation	11
	IV.3. Optical product cleanliness	11
	IV.4. Waste Electrical and Electronic Equipment (WEEE) Directive	12
V.	Description of the Controller	13
	V.1. General Functionality	14
	V.2. Configuration menu	15
	V.2.1. Networking	15
	V.2.2. Software update	16
	V.3. Interface	18
	V.3.1. HSSL.....	18
	V.3.2. A-quad-B	20
	V.3.3. Sine-Cosine (Analog signal)	22
	V.3.4. Error signal	24
	V.3.5. Pin-Layout	25
	V.3.6. Pin Description.....	26
	V.4. Alignment.....	27
	V.4.1. Axis control.....	27
	V.4.2. Pilot laser	28
	V.4.3. Optics alignment	28
	V.5. Operation	29
	V.5.1. Position.....	29
	V.5.2. Environmental compensation unit (ECU)	30
VI.	Programming.....	31
	VI.1. DLL functions.....	31
	VI.2. JSON-RPC.....	39
	VI.2.1. JSON Methods.....	41

VI.3. Error handling	43
VI.3.1. Implemented Errors	43
VI.3.2. Error handling in C#	45

II. General handling

When using attocube's Industrial Displacement Sensor (IDS), several handling guidelines should be followed.

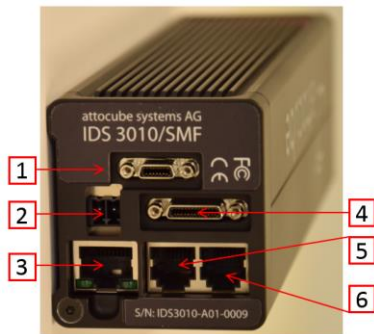
II.1. Unpacking



Setting up. Unpack all the components and retain all packing material and shipping containers for any future shipping needs. Place the unit in a location with proper ventilation. Also make sure that the controller is not placed near any liquids or moisture.

Carefully unpack and visually inspect the controller for any damage.

II.2. Electrical interface



The IDS3010 includes a variety of different interfaces on its back.

- [1] GPIO
- [2] Power supply
- [3] Ethernet
- [4] Realtime output (HSSL/A-quad-B & analogue sine/cosine)
- [5] CAN (Profinet)
- [6] CAN (ECU interface)

For further description of the interfaces refer to the interfaces section.

II.2.1. Power supply

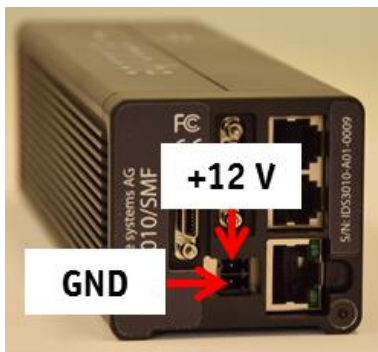
The IDS3010 is supplied with an universal AC input power supply with the following specifications:

Input:

- VOLTAGE RANGE: 90 – 264 VAC
- FREQUENCY RANGE: 47 – 63 Hz
- AC CURRENT: 0.7 A / 100 VAC

Output:

- DC VOLTAGE: 12 V
- CURRENT RANGE: 0 – 2.08 A

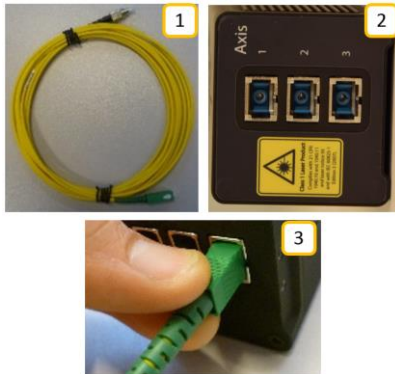


The IDS power supply provides two pins. The polarity of these two pins is shown in the left figure.

IDS shall only be used with the power supply provided by attocube.

The shape of the connector provided by the attocube power supply provides protection against polarity reversal.

II.3. Optical interface

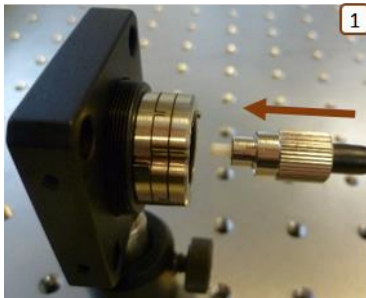


The IDS3010 provides three fiber-optic outputs [2] for simultaneous measurement of the displacement of three axes. The IDS is connected to the sensor heads with SC/APC-FC/PC patch cables [1].

The SC/APC fiber connector (green connector color) is always mounted to the IDS electronic unit [3].



II.4. Connecting the sensor heads

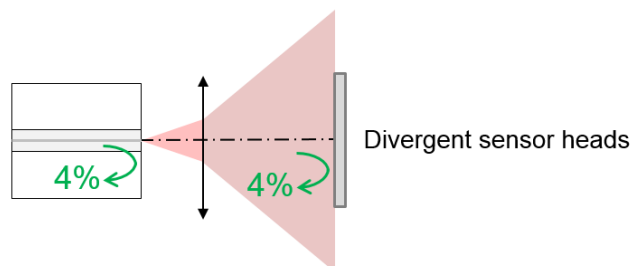


Align the black colored end of the fibers (FC/PC connection with blue or black color) to your optical sensor heads and connect them.

Care must be taken when mounting FC fibers:

- Screw the FC fiber connector to lock the position.
- Make sure the connector key is aligned properly in the mating slot before tightening!
- Do not over tighten the connector. The mechanical/ceramic part of the connector could be damaged.

The standard principle of the IDS works with a divergent sensor head. Here, the target reflects 4 % back into the fiber which has the same intensity compared to the 4 % back reflection of the FC/PC fiber end.



II.5. Working in a standard room environment

To achieve highest precision with the IDS3010, avoid operating your system in dirty environments. For best performance under ambient conditions it is highly recommended to combine the IDS with an environmental compensation unit. Avoid external heating sources in your working area.

III. Safety instructions

attocube's IDS is a Laser Class 1 product. For your own safety it is recommended to use appropriate filters and glasses.

III.1. Laser



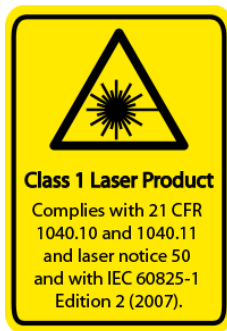
The IDS3010 contains two different types of lasers. There is one infrared laser used for performing the displacement measurements. The second visible laser is used for aligning the setup and is switched off for measurement operation.

The IDS3010 complies with 21 CFR 1040.10 and 1040.11 and laser notice 50 and with IEC 60825-1 Edition 2 (2007).

The light exits the IDS3010 at the front panel.

The laser safety compliance is guaranteed by attocube through output power monitoring.

III.2. Infrared Laser



The DFB laser has a wavelength of 1530 ± 5 nm.

Caution! Invisible laser radiation! Avoid looking directly at the laser beam or at its reflection on mirror-like surfaces.

III.3. Visible Laser



The red alignment laser has a wavelength of 650 ± 10 nm.

Caution! Visible laser radiation! Avoid looking directly at the laser beam or at its reflection on mirror-like surfaces.

IV. Installation

The application of the IDS requires a set of conditions that need to be checked or prepared before using the system.

IV.1. Mechanical Installation

Setting up. Unpack all the components and retain all packing material and shipping containers for any future shipping needs. Place the unit in a location with proper ventilation; do not obstruct the ventilation slots in any way. Also make sure that the controller is not placed near any liquids or moisture.

Carefully unpack and visually inspect the controller for any damage. Place all components on a flat and clean surface.



Caution. When setting up the unit, it should be positioned so that the operation of the rear panel power supply plug and switch is not impeded. Ensure that proper airflow is maintained to the unit.



Warning. Operation outside the following environmental limits may adversely affect operator safety:

Indoor use only

Maximum altitude 2000 m

Temperature range: 5 to 40°C

Maximum humidity: less than 80 % RH (non-condensing) at 31°C

To ensure reliable operation the unit should not be exposed to corrosive agents or excessive moisture, heat or dust. If the unit has been stored at a low temperature or in an environment of high humidity, it must be allowed to reach ambient conditions before being powered up.



Note. In applications requiring the highest level of accuracy and repeatability, it is recommended that the controller unit is powered up approximately 30 minutes before use in order to allow the internal temperature to stabilize.



Caution. Do not connect cabling longer than 3 m. Longer cables may increase the sensitivity of the device to external influences.

IV.2. Electrical Installation



Warnings. The unit must be connected only to a grounded fused supply of 100 to 240 V (Japan, USA and Europe).

Use only power supply cables supplied by attocube systems, as other cables may not be rated to the same current. The unit is shipped with appropriate power cables for use in the UK, Europe, Japan and the USA. When shipped to other territories the appropriate power plug must be fitted by the user.



Prevent electrical shock from electronic. To prevent electrical shock do not remove the cover of the control unit. Unplug power cord and all other electrical connections and consult qualified service personnel when servicing or cleaning. Operate only under dry conditions and at room temperature.

IV.3. Optical product cleanliness



Never attempt to clean the IDS by immersion into any liquid.

The IDS parts are sensitive to any kind of liquid.

Please note that all parts of IDS are cleaned in our production facility. If any surface is contaminated, please clean all surfaces only as follows:

- If there are dust particles on the surface of the IDS collimator head or fiber, please use dust and oil free air to clean them off.
- If there is dirt on the fiber or collimator head surface, please use a dust-free tissue slightly tintured with isopropanol and/or acetone to clean the surface. Please make sure that no droplets of solvent get into contact with the glued parts of the IDS.

Because of the inherent fragility of the collimator head and ferrule sleeves, the manufacturer takes no responsibility for the breakdowns of these parts. In case of breakage, please contact the manufacturer for details of the repair service.

IV.4. Waste Electrical and Electronic Equipment (WEEE) Directive



Compliance

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

This offer is valid for attocube systems electrical and electronic equipment:

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

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

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

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

Waste treatment on your own responsibility

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

Ecological background

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

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

V. Description of the Controller

attoIDS Control

- 1530 nm laser modulated light with 340 μ W output power
- Stabilization of the laser wavelength to $5 \cdot 10^{-8}$ ($\Delta\lambda/\lambda$)
- Digital measurement of the direct attoIDS Fabry-Perot interference pattern
- Digital measurement of the demodulated attoIDS Fabry-Perot interference pattern
- Processing of the relative target position displacement
- 48 bits position output resolution
- Single and double path attoIDS signal processing.

Laser Input/Output

Fiber connector:	SC/APC single mode fiber
Laser light wavelength:	1530 ± 5 nm
laser light output power:	340 μ W
Modulation rate:	12.5 MHz
Measurement bandwidth:	10 MHz

V.1. General Functionality

attoIDS is a patented plug and play fiber based low finesse Fabry-Perot interferometer.

Coherent light is emitted from a telecom laser diode comprised in the IDS electronics. This source is stabilized in both current and temperature. The laser light is routed towards each axis of the IDS through integrated, fiber based components. Each axis of the standard IDS is composed of a fiber with a FC/PC terminal that is attached to a lens and the laser light is divergent. The free beam is directed towards a reflecting target that back-reflects the light into the axis fiber. A low finesse Fabry-Perot resonator cavity is hence defined between the FC/PC fiber end and the target.

Unlike usual homodyne techniques, the **IDS laser wavelength is modulated** with a 12.5 MHz frequency (K. Karrai, EP2045572 (A1), 2009). The detected sinusoid-like interference pattern is small signal modulated. Two components are extracted from this overall signal: a low pass filtered component ($f < 10\text{MHz}$), which is the standard interferogram, and a demodulated component (see *Figure*). They are in phase quadrature. This allows overcoming the usual Fabry-Perot restrictions (existence of low sensitivity sensing spots and impossibility of defining displacement direction). This configuration also permits both measurement and processing electronics to be deported from the actual interferometer fiber optic based sensing head.

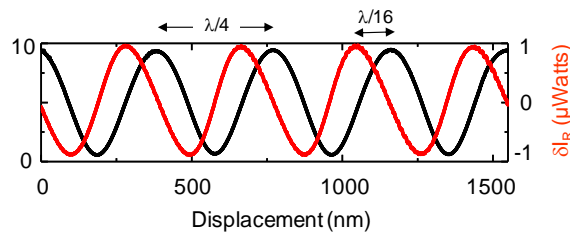


Figure: Low pass filtered and demodulated phase quadrature low finesse measured interferogram.

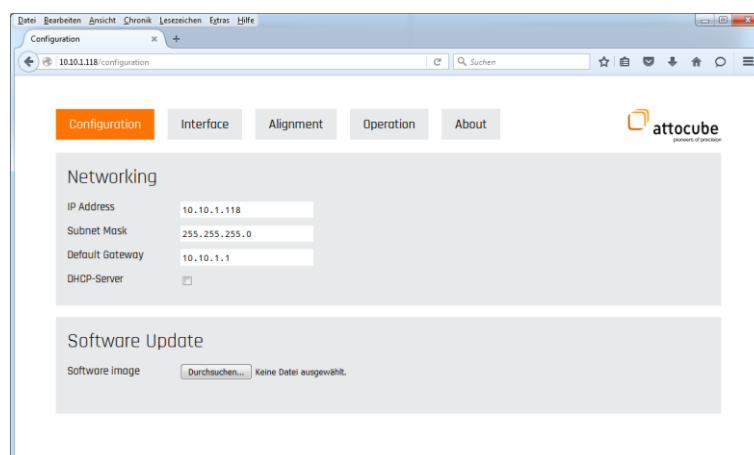
V.2. Configuration menu

V.2.1. Networking

The configuration menu allows setting the network settings of the IDS Ethernet interface. The IDS supports IPv4 networking. By default, the IDS runs a DHCP server on the Ethernet interface. The DHCP server provides an IP address (lease) to a client (PC, Notebook) connected to the IDS.

In order to connect the IDS to an existing network, it is possible to disable the DHCP server and configure a static IP address, subnet mask and default gateway.

When connecting the first time with the IDS, connect the IDS directly with your PC or Notebook and open your web browser. The IDS supports Internet Explorer, Google Chrome and Firefox. You will reach the IDS Web Interface under the default IP address "192.168.1.1".



The IDS Web Interface allows the following configurations of the Ethernet interface:

IP address

Here you can configure a static IP address in order to connect the IDS to an existing network. Make sure the IP address does not conflict with an existing IP address in your network. The default IP address is "192.168.1.1".

Please write down the configured IP address and make it visible on the device. The IP address cannot be simply recalled or reset. In case of loss please contact attocube systems.

Subnet mask

The default value is "255.255.255.0".

Default Gateway

The default value is "192.168.1.1".

DHCP server

Here you can enable and disable the IDS integrated DHCP server. The DHCP server is activated by default upon delivery.



Configuration Interface Alignment Operation

attocube
partners of precision

Networking

DHCP-Server ☒

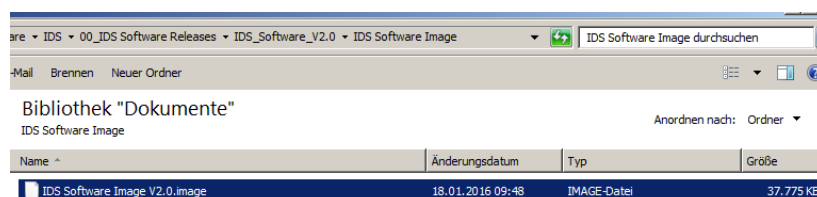
Software Update

Software image Keine Datei ausgewählt

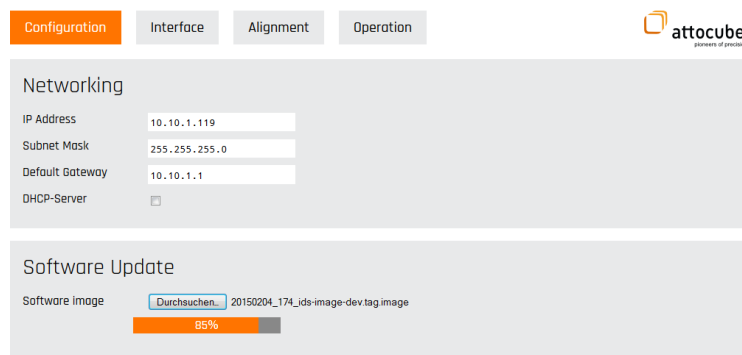
If you have problems opening the specific IDS website with the correct IP address, please erase the cookies of the web browser.

V.2.2. Software update

attocube occasionally provides software updates for the IDS. The updates are delivered in a “.zip” file. Please save this file to your local disk and unpack it. In the web interface, press the “Durchsuchen...” button and select the “.image” file from the unpacked folder. Press “Open” to load the file into the IDS.



A status bar on the web interface shows the upload progress.



Configuration Interface Alignment Operation

attocube
partners of precision

Networking

IP Address 10.10.1.119

Subnet Mask 255.255.255.0

Default Gateway 10.10.1.1

DHCP-Server ☐

Software Update

Software image 20150204_174_ids-image-dev.tag.image

85%


After uploading the image file, press “Install & Reboot”.

Configuration

Interface

Alignment

Operation



Networking

IP Address	<input type="text" value="10.10.1.119"/>
Subnet Mask	<input type="text" value="255.255.255.0"/>
Default Gateway	<input type="text" value="10.10.1.1"/>
DHCP-Server	<input type="checkbox"/>

Software Update

Software image	<input type="button" value="Durchsuchen..."/> 20150214_190_ids-image-dev.tag.image
----------------	--

The installation takes a while. Do not disconnect the IDS during this time.

After the installation, a red bar at the top of the web interface indicates the reboot process.


Connection to server lost. Waiting for server to reconnect...

Configuration

Interface

Alignment

Operation



Networking

IP Address	<input type="text" value="10.10.1.119"/>
Subnet Mask	<input type="text" value="255.255.255.0"/>
Default Gateway	<input type="text" value="10.10.1.1"/>
DHCP-Server	<input type="checkbox"/>

Software Update

Software image	<input type="button" value="Durchsuchen..."/> 20150214_190_ids-image-dev.tag.image
----------------	--

Software update successful, rebooting system...

The IDS will restart after the reboot is complete. This is indicated by a green bar.


Connection to server restored. You can continue working now

Configuration

Interface

Alignment

Operation



Networking

IP Address	<input type="text" value="10.10.1.119"/>
Subnet Mask	<input type="text" value="255.255.255.0"/>
Default Gateway	<input type="text" value="10.10.1.1"/>
DHCP-Server	<input type="checkbox"/>

Software Update

Software image	<input type="button" value="Durchsuchen..."/> 20150214_190_ids-image-dev.tag.image
----------------	--

Software update successful, rebooting system...

V.3. Interface

The IDS provides real time position signals for each axis. There are three different output modes and different output signal levels which can be selected by the user. Only one mode and signal level is available at a time. The setting applies to all outputs simultaneously.

- HSSL (LVTTTL or LVDS)
- A-quadr-B (LVTTTL or LVDS)
- Sine-Cosine (Analog signal)

The settings are configured in the Interface tab.

The screenshot shows the 'Interface' tab of the attocube web interface. Under the 'Real Time Interface' section, the 'Output Mode' dropdown is open, showing several options. 'HSSL (LVDS)' is currently selected. Below the dropdown, there are input fields for 'Resolution HSSL Low' (2^n pm), 'Resolution HSSL High' (2^n pm), 'Period HSSL clock' (400 ns), and 'Period HSSL gap' (1 clock). At the bottom, the 'Periphery' section includes a checkbox for 'Environmental Compensation Unit'.

Output Mode

The Real Time Interface section allows setting the output format and the corresponding signal level. You can choose between

- HSSL (LVTTTL or LVDS)
- A-quadr-B (LVTTTL or LVDS)
- Sine-Cosine (LVTTTL or LVDS error signal)

V.3.1. HSSL

In HSSL (High Speed Serial Link) mode, the current displacement value is periodically transferred using a binary serial format based on 1 pm resolution. The HSSL protocol is defined by its resolution, clock time and gap. Clock time represents the bit output rate, and resolution defines the binary bit configuration, which provides the position information. The low resolution tab indicates the starting bit and the high resolution tab sets the final bit. An example is shown below. The signal itself starts with the low resolution bit.

The HSSL word is encoded using the two's complement system.

The required gap is a time value in terms of clock time, which separates the different position signals from each other. The clock time is limited by 25 MHz, equal to 40 ns. For real time outputs it is recommended to use a DIO or DAQ card, which is compatible to a multiple of 40 ns. The clock period, gap and number of bits can be user adjusted by the web interface. Between two sets of position information, synchronization of reader/sender can be gained through a continuous stream of position information. The data are synchronized with the rising edge of the CLK signal.

The signals are available with two different signaling standards (only one at a time):

- Single-ended with LVTTTL levels
- Differential with LVDS levels

A third signal (POS_ERROR) signals an error condition when the position is lost due to laser beam interruption or too high of a displacement speed.

The HSSL signal starts with a zero displacement value as soon as the calibration process is done and the absolute position is shown on the Operation Tab.

The screenshot shows the 'Interface' tab of the attocube software. Under 'Real Time Interface', the 'Output Mode' is set to 'HSSL (TTL)'. Below this, there are input fields for 'Resolution HSSL Low' (2), 'Resolution HSSL High' (45), 'Period HSSL clock' (400), and 'Period HSSL gap' (1). Each field has a unit indicator: '2^n pm' for resolution and '[ns], multiple of 40ns' for period. At the bottom of the configuration section are 'Apply' and 'Discard' buttons. Below the configuration section is a 'Periphery' section with a checkbox for 'Environmental Compensation Unit'.

The digital resolution of the HSSL interface is 1 pm. However, for high speed measurements it might be necessary to limit the number of transmitted bits.

Resolution HSSL Low (Resolution)

“Resolution HSSL Low” defines the position resolution of the HSSL interface. It allows specifying the lowest bit of the 48-bit distance value in units of bits. The position resolution is therefore given by

$$\text{Resolution} = 2^{(\text{Resolution HSSL Low})} * 1 \text{ pm}$$

The lowest value is 0.

Resolution HSSL High

“Resolution HSSL High” specifies the highest bit of the 48-bit distance value. The highest value is 47. The HSSL word size is therefore given by

$$\text{Word Size} = \text{Resolution HSSL High} - \text{Resolution HSSL Low} + 1$$

Period HSSL Clock

The clock defines the time period at which the serial word bits are outputted (inverse of the bit rate). The clock is programmable in integer, nanosecond multiples of 40 ns.

Period HSSL Gap

The gap argument is given in HSSL clock periods that are omitted and specifies the gap between the end of a HSSL word and the beginning of the subsequent HSSL word.

Total length of signal and maximum frequency

The total HSSL length of the signal is defined by the formula:

$$(\text{Word Size} + \text{Number of gap bits}) * \text{clock time} = \text{total HSSL length of signal}$$

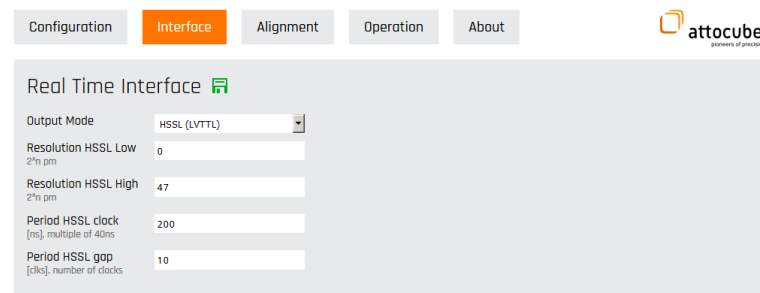
The maximum total HSSL length of signal is 2.00 ms (500 Hz).

Example trace of the HSSL signal

The following figure shows an example trace of the HSSL signal with full 48 bit word length and a gap of 10 bits. Here, the first and second line represents the time trace of the data channel and the clock channel, respectively. Each word is separated by 10 omitted clock bits.



The successful saving of the user defined settings is indicated by a green floppy disk.

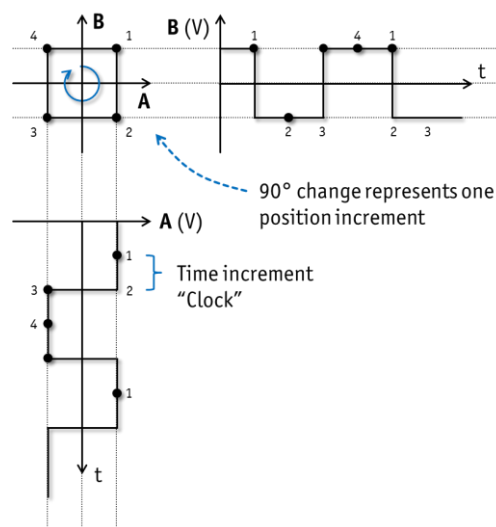


V.3.2. A-quad-B

The A-quad-B process is a digital interface that allows high resolution over large sizes of data. Resolution and Clock define the quadrature process. The signal is encoded over two different 2-level-channels, A and B. The levels' amplitude is either LVTTTL or LVDS (one at a time):

- Single-ended with LVTTTL levels
- Differential with LVDS levels

Each channel increment/decrement represents a position change of a value Resolution. A position change of size Resolution results in an increment or decrement of the signals A and B, depending on the change direction. In this way, the A-quad-B differential interface allows transmitting both the displacement and direction of the target movement. A third signal POS_ERROR uses the same definition as the HSSL format.



Plotting both Channel A and B against each other creates a square. In this picture, a signal change of 90° represents, depending of the direction, a displacement of \pm Resolution.

In this mode, change of position can only be displayed by stepping through the four possible states of the A/B quadrature signals. The scaling and the update rate of the A/B outputs are configurable.

The fastest possible update interval is 40 ns, equal to the update interval of the absolute distance accumulator. The range of possible resolution settings is 1 pm to 64.93 nm.

When the target displacement is above the maximum velocity that can be handled by the A-quad-B output, then the data need more time to be transmitted, which can be seen as a low-pass filtering of the data.

Resolution

The resolution defines the position resolution of the real time interface. The resolution of the A-quad-B interface is user adjustable and ranges from 1 pm to 64.93 nm (programmable in 2^n steps, where n is an integer number). We recommend to start to use a high resolution for the first test, afterwards you can decrease the resolution and adjust it to the application.

Period Sin-Cos clock

The clock defines the minimal period at which the quadrature signal increment can be outputted. The clock of the A-quad-B interface is limited to a minimal period of 40 ns (i.e. 25 MHz) and a maximal period of 1.298 ms (i.e. 770 Hz). The clock is programmable in integer multiples of 40 ns.



Note: When using the AquadB interface, the signal must be recorded constantly; otherwise the position information is lost.



Note: The combination of *Resolution* and *Clock* parameters defines a type of low pass filter.

The combination of *Resolution* and *Clock* parameters defines a speed limit of

$$\text{maximum speed} = \text{Resolution} / \text{Clock}.$$

If the target velocity increases that limit, the AquadB interface cannot follow and an error signal is raised. To clear the error, the position has to be reset or a calibration has to be started.



Note: If the signal to noise ratio is not high enough during the target displacement or if the speed limit is exceeded, the AquadB interface error signal will activate.

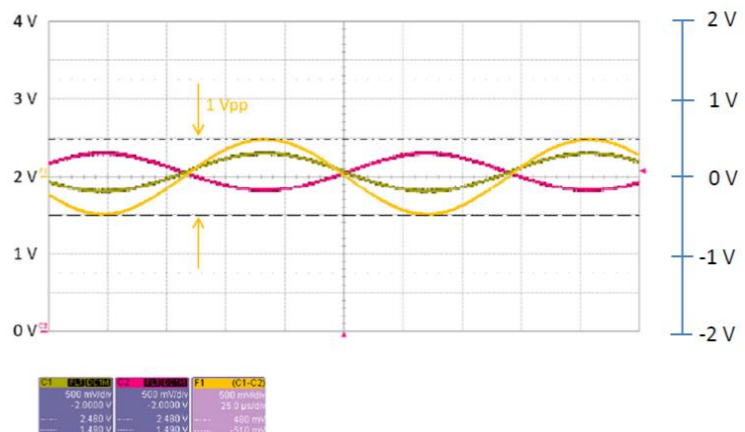
V.3.3. Sine-Cosine (Analog signal)

Sinus/Cosine uses two analog signals in quadrature. Levels are chosen according the following definition:

- Differential signal 1 Vpp +/- 10%
- Common Mode voltage: 2 V +/- 5%
- $R_{SOURCE} = 30\ \Omega$
- $7\ \text{mA} \leq I_{LOAD} \leq 20\ \text{mA}$
- Max. output bandwidth 25 MHz

Each channel increment/decrement represents a signal change of a value Resolution. A position change of size Resolution results in an increment or decrement of the signals A and B, depending on the change direction. This allows the determination of the signal direction and the value with a step unit of Resolution. The clock parameter defines the maximum frequency at which the incremental change of the signal can be outputted. A third signal POS_ERROR uses the same definition as the HSSL format. In this mode, only a change in distance can be displayed by means of an analog quadrature signal pair. The scaling and the update rate of the sin/cos output is configurable. The fastest possible update interval is 40 ns. The resolution specifies the distance corresponding to a 90 degree rotation at the sin/cos output.

Oscilloscope trace



C1 (green) = SIN+ (A+)

C2 (pink) = SIN- (A-)

F1 (yellow) = Math-function C1 - C2

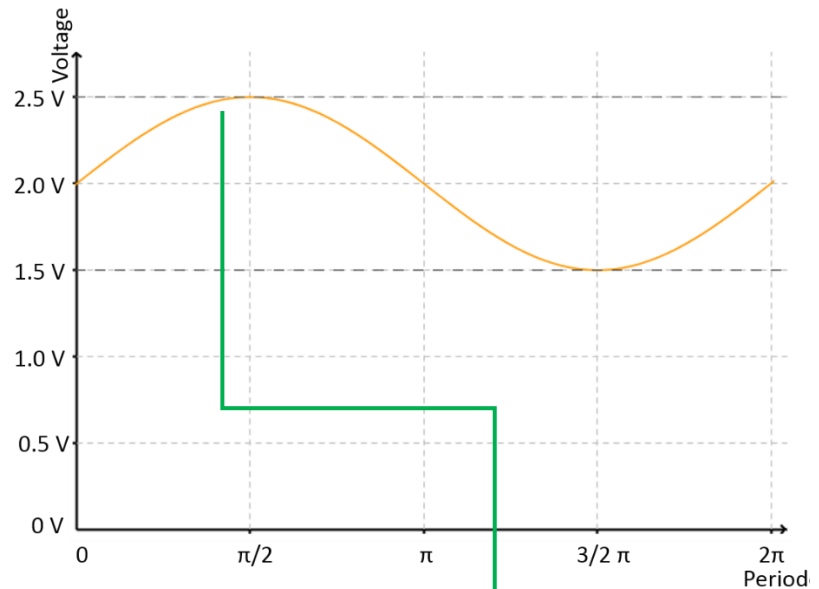
Signal F1 shows 1 Vpp (right scale bar)

Signals C1 and C2 show the common mode voltage (CMV) of 2.05 V (left scale bar).

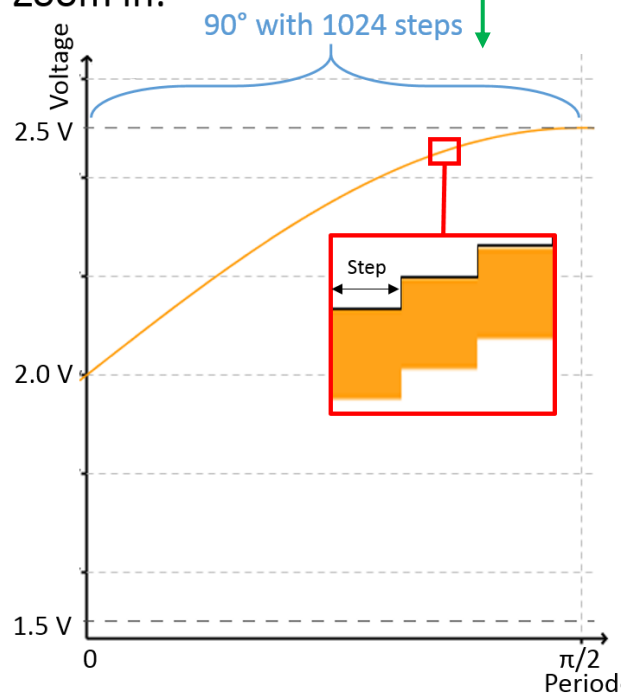
The SIN/COS outputs are controlled by D/A converters with a word length of 12 bit, providing $2^{12} = 4096$ incremental voltage steps.

Resolution Sin-Cos

The largest resolution is $16.8 \mu\text{m}/90^\circ$ (corresponding to a resolution range of 24 bits given in pm units), the smallest resolution is 1 pm.



Zoom in:

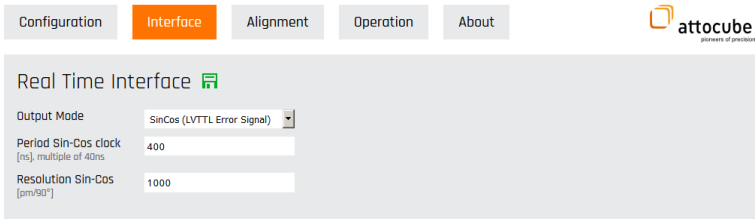


The Sine/Cosine output is controlled by 12 bit D/A converter. Therefore the resolution for a whole sine/cosine period (360°) contains $2^{12} = 4096$ steps.

For example setting the position resolution to 5000 nm for a 90° turn of the sine/cosine signal leads to 1024 steps with a resolution per step of 4.88 nm.

Period Sin-Cos clock

The clock period can be adjusted in multiples of 40 ns. The longest possible period is 10.2 μ s, the shortest is 40 ns.

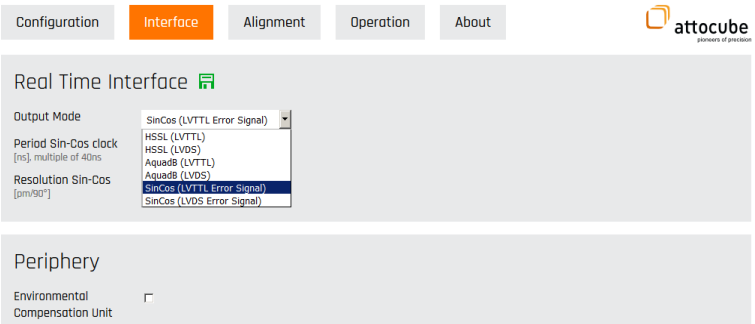


V.3.4. Error signal

The real-time interface provides an error signal for each individual interferometric axis. The voltage level of the error signal can be either differential LVDS or single ended LVTTTL.

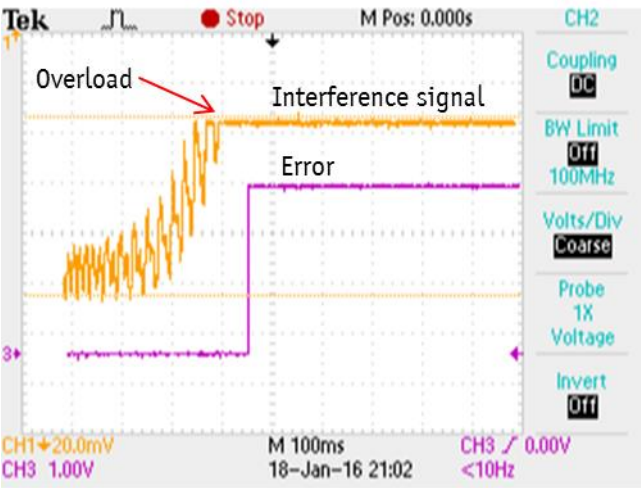
Voltage level

The voltage level of the error signal corresponds to that of the chosen interface. For the Sin/Cos signal, which is only provided as differential signal, the interface tab provides two possible signal levels for the error signal, either LVDS or LVTTTL.



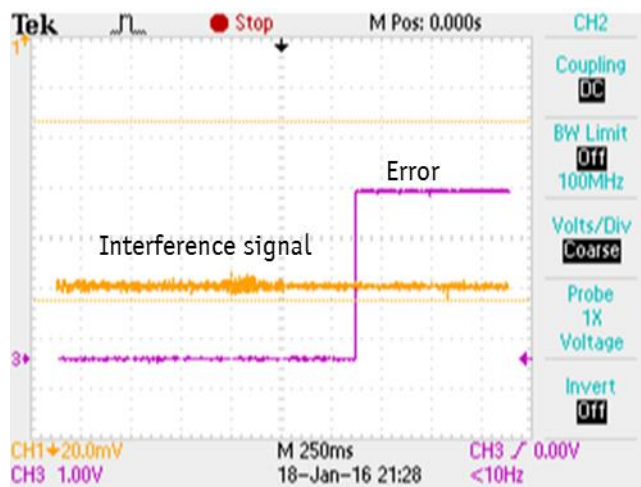
Overload

The error signal turns on in case that the interference signal saturates the detector.



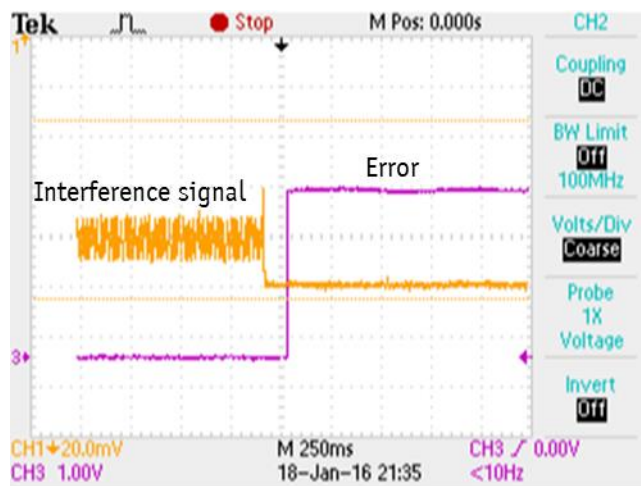
Signal too low

The error signal turns on in case that the interference signal becomes too low.



Beam interruption

The error signal turns on in case that the beam is interrupted.

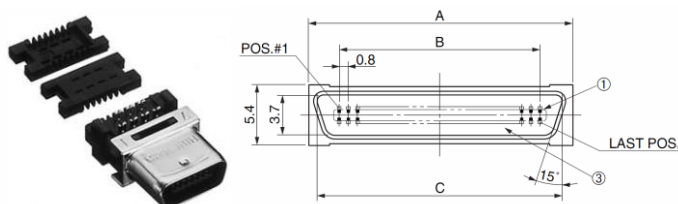


V.3.5. Pin-Layout

A 26 pin HDR connector transports the realtime output signals. (front view of the female HDR connector):



The IDS is delivered with a 26 pin IDC type male connector (HDR-E26 MAG1+):



Furthermore, the IDS Prototypes are delivered with a cable with open ends (three quad cables (1,2, and 3) and six twisted pairs):



V.3.6. Pin Description

In order to reduce the noise level of the real-time outputs, the signal output can be terminated with a proper termination impedance.

The SIN/COS output is optimized for a differential termination impedance of 120 Ω . This termination results in a voltage level of 1 V.

Pin IDS top view	Pin 3M SDR Cable	Signal	HSSL LV TTL	HSSL LV DS	A-quad-B LV TTL	A-quad-B LV DS	Sin/Cos (error: LV TTL)	Sin/Cos (error: LV DS)
Axis 1								
21	blue (white)	POSITION-1A(+)	CL 1	CLK1(+)	1A	1A(+)	1A(+)	1A(+)
8	white (blue)	POSITION-1A(-)	-	CLK1(-)	-	1A(-)	1A(-)	1A(-)
22	green (white)	POSITION-1B(+)	DATA1	DATA1(+)	1B	1B(+)	1B(+)	1B(+)
9	white (green)	POSITION-1B(-)	-	DATA1(-)	-	1B(-)	1B(-)	1B(-)
23	yellow (white)	POSITION-1E(+)	-	-	1E	1E(+)	1E	1E(+)
10	white (yellow)	POSITION-1E(-)	-	-	-	1E(-)	-	1E(-)
Axis 2								
18	1 blue	POSITION-2A(+)	CLK2	CLK2(+)	2A	2A(+)	2A(+)	2A(+)
5	1 green	POSITION-2A(-)	-	CLK2(-)	-	2A(-)	2A(-)	2A(-)
19	3 brown	POSITION-2B(+)	DATA2	DATA2(+)	2B	2B(+)	2B(+)	2B(+)
6	3 black	POSITION-2B(-)	-	DATA2(-)	-	2B(-)	2B(-)	2B(-)
20	3 yellow	POSITION-2E(+)	-	-	2E	2E(+)	2E	2E(+)
7	3 green	POSITION-2E(-)	-	-	-	2E(-)	-	2E(-)
Axis 3								
15	2 brown	POSITION-3A(+)	CLK3	CLK3(+)	3A	3A(+)	3A(+)	3A(+)
2	2 black	POSITION-3A(-)	-	CLK3(-)	-	3A(-)	3A(-)	3A(-)
16	2 red	POSITION-3B(+)	DATA3	DATA3(+)	3B	3B(+)	3B(+)	3B(+)
3	2 green	POSITION-3B(-)	-	DATA3(-)	-	3B(-)	3B(-)	3B(-)
17	1 brown	POSITION-3E(+)	-	-	3E	3E(+)	3E	3E(+)
4	1 black	POSITION-3E(-)	-	-	-	3E(-)	-	3E(-)
GND								
12	black (red)	GND						
26	red (white)	GND						
13 + 14	inner shield	GND						
24	gold (white)	GND						
11	white (gold)	GND						
1	white (red)	GND						
25	red (black)	GPIO-RT						

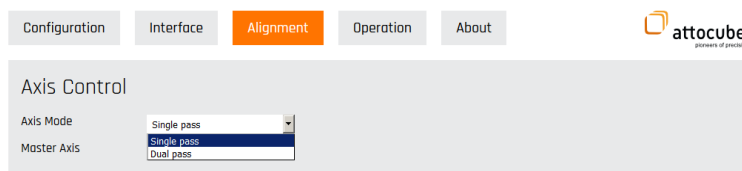
General purpose IO, always LV TTL levels (same circuit as on GPIO)

V.4. Alignment

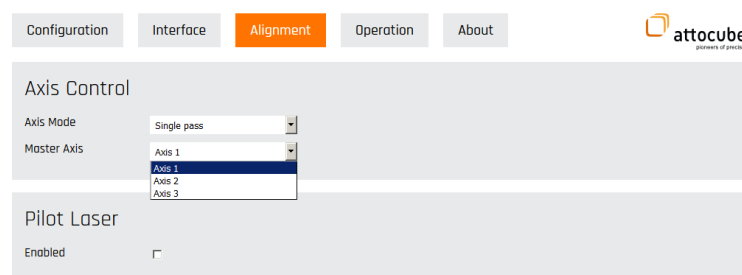
V.4.1. Axis control

The alignment tab provides some tools for the mechanical alignment of the setup.

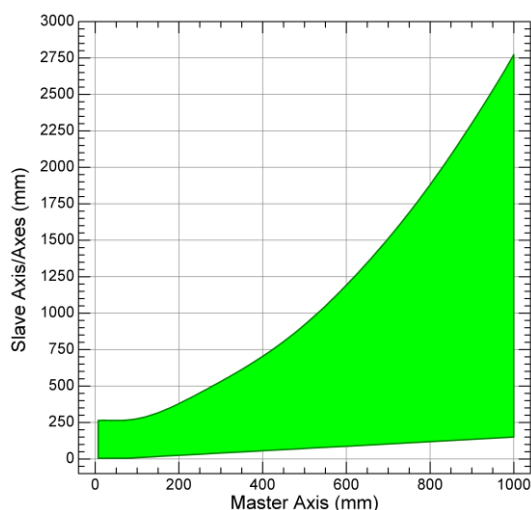
The axis control section allows setting the specification of the master axis and the folding order of the Fabry-Pérot cavity of each single axis.



The master axis is used for open loop control of the modulation amplitude.



The master axis should be set to the longest absolute distance axis number, in order to maximize the displacement region of the two slave axes. The following figure depicts the possible displacement range (see the green area) of the slave axis/axes as a function of the master axis. Here, the measurement was taken with an alignment contrast signal of 700 ‰ for each IDS axis. If the alignment signal has a lower (higher) value in comparison to the 700 ‰, the displacement range of the slave axis/axes is smaller (larger).



Example 1: The master axis is on the absolute position 200 mm. This means that the slave axis/axes can be moved between 25 and 380 mm.

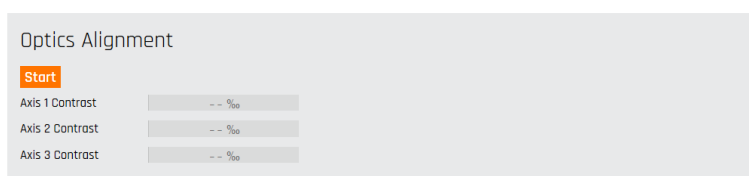
Example 2: The master axis is on the absolute position 800 mm. This means that the slave axis/axes can be moved between 120 and 1900 mm.

V.4.2. Pilot laser

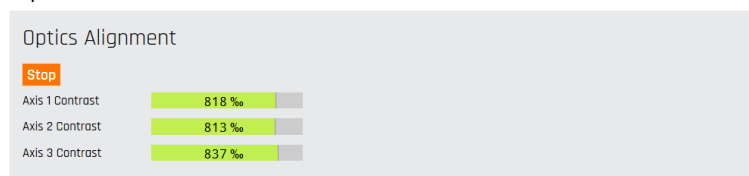
The pilot laser checkbox allows you to enable a red alignment laser. The red laser helps to roughly align the setup.

V.4.3. Optics alignment

The optics alignment section provides a contrast bar for each axis. It aims at maximizing the signal contrast of each axis. The contrast bar is enabled by pressing the “Start” button.



It needs to be stopped by pressing the “Stop” button in order to enable the displacement measurement.

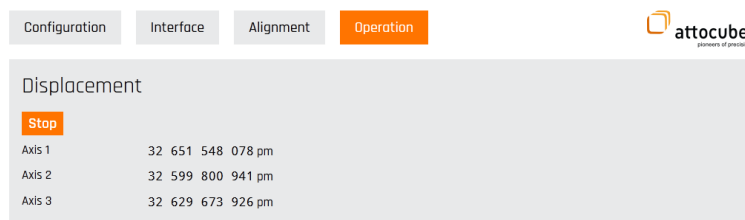


We recommend, that the interference contrast (in per mil ‰) should be between 700 and 1000 ‰. Moreover, the target as well as the sensor head should be adjusted so that the alignment signal stays relative constant over the whole measurement range.

V.5. Operation

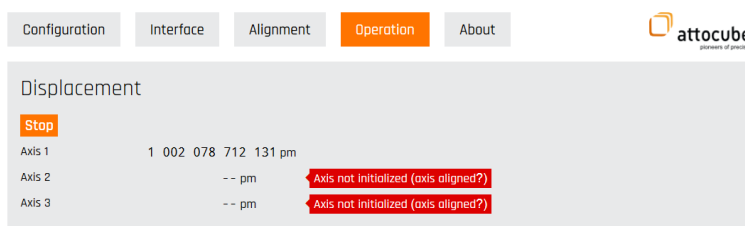
V.5.1. Position

The Operation tab displays the current displacement value of all three axes in picometers. This calibration process to obtain absolute positions for all three axes requires less than two minutes.

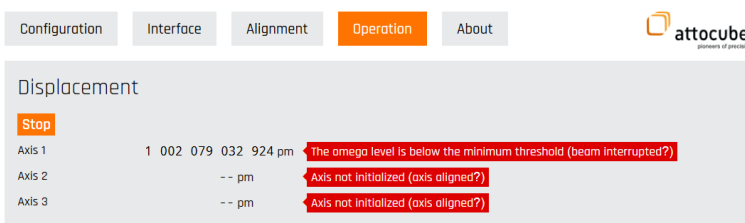


Error display on the IDS Web Interface

The IDS web interface shows error events happened during the displacement, as shown in the following example:



Axis 1 shows an error message after beam interruption:



The error can be reset using the DLL method `IDS_ResetError()` or be recalibration the IDS system.

V.5.2. Environmental compensation unit (ECU)

Status

Indicates if the ECU is connected and activated in the interface tab.

Temperature

Indicates the current temperature.

Relative Humidity

Indicates the current relative humidity in air.

Pressure

Indicates the current pressure.

Refractive index

Indicates the current refractive index, which was deduced from temperature, relative humidity, and pressure.

Environmental Compensation Unit	
Status	connected
Temperature	21.45634 °C
Relative Humidity	29.89 %
Pressure	958.25 hPa
Refraction Index	1.000252466160699

VI. Programming

The IDS3010 provides a .NET DLL for reading and writing settings on the IDS.
The DLL uses communication port **9090**.

VI.1. DLL functions

This chapter provides an overview over the available DLL functions.

	System
<pre>public void IDS_Connect(string ipAddress, int port)</pre>	<p>Connect to the target with a hostname or IP address and port provided. If a valid port number (> 0) is supplied, the target URL is constructed like this:</p> <p><code>http://\$(ipAddress):\$(port)/api/json</code></p> <p>If an invalid port number (0 or negative value) is supplied, then the string in "ipAddress" is taken as the literal string for the base service URL (used for e.g. testing).</p> <p>Parameters</p> <p><i>ipAddress</i></p> <p>Type: System.String IP Address for target system</p> <p><i>port</i></p> <p>Type: System.Int32 Port</p>
<pre>public void IDS_Disconnect()</pre>	Disconnect from the target system.
<pre>public void IDS_FactoryReset()</pre>	Do a factory reset on next reboot.
<pre>public void IDS_RebootSystem()</pre>	Reboot the IDS system
<pre>public void IDS_SoftwareUpdate(string filePath)</pre>	<p>Loads the specific binary update file and converts it to base64.</p> <p>Parameters</p> <p><i>filePath</i></p> <p>Type: System.String Path to the binary update image file</p>
<pre>public void IDS_StartMeasurement()</pre>	Start the measurement, this can take a while because the system needs to heat up the laser.
<pre>public void IDS_StartOpticsAlignment()</pre>	Start the optics alignment (Contrast measurement).
<pre>public void IDS_StopMeasurement()</pre>	Stop the measurement.
<pre>public void IDS_StopOpticsAlignment()</pre>	Stop the optics alignment (Contrast measurement).

	Alignment
<code>public void IDS_EnablePilotLaser()</code>	Enable the pilot laser.
<code>public void IDS_DisablePilotLaser()</code>	Disable the pilot laser.
<code>public bool IDS_GetPilotLaserEnabled()</code>	<p>Returns if the red pilot laser is enabled.</p> <p>Return Value</p> <p>Type: Boolean true if enabled, false if disabled</p>
<code>ContrastData IDS_GetContrastInPermille(int axis)</code>	<p>Returns the contrast of the signal (ratio signal maximum to signal minimum) in per mill.</p> <p>Parameters</p> <p><i>axis</i></p> <p>Type: System.Int32 Axis to get the value from {0-2}</p> <p>Return Value</p> <p>Type: ContrastData The contrast in per mill</p> <pre>public ContrastData(int contrast, int warningCode)</pre> <p>Parameters</p> <p><i>contrast</i></p> <p>Type: System.Int32 Result value - contrast in promille</p> <p><i>warningCode</i></p> <p>Type: System.Int32 Warning code</p>
<u>Example:</u>	<p>Getting the contrast of axis 1</p> <pre>int ContrastAxis1 = client.IDS_GetContrastInPermille(0).ContrastInPromille;</pre>
<code>public int IDS_GetBaseBandAmplitude()</code>	<p>Get the amplitude of the base band signal, this will allow the customer to readjust the measurement setup, if it is too low. The amplitude is the difference between the interference maximum and the minimum.</p> <p>Return Value</p> <p>Type: Int32</p> <p>The amplitude of the base band signal, the return values range is between 0 and 17740. If the signal is <900, the IDS assumes that no interferometer axis is connected.</p>

Position	
<pre>public AxesDisplacements IDS_GetAxesDisplacement()</pre>	<p>Return Value</p> <p>Type: AxesDisplacements A class with the three displacement values requests</p> <pre>public AxesDisplacements(int warning, long displacement0, long displacement1, long displacement2)</pre> <p>Parameters</p> <p><i>warning</i></p> <p>Type: System.Int32 Possible warning value from the embedded system</p> <p><i>displacement0</i></p> <p>Type: System.Int64 Value for displacement of axis no. 0</p> <p><i>displacement1</i></p> <p>Type: System.Int64 Value for displacement of axis no. 1</p> <p><i>displacement2</i></p> <p>Type: System.Int64 Value for displacement of axis no. 2</p>
<pre>public long IDS_GetAxisDisplacement(int axis)</pre>	<p>Get the displacement of axis n</p> <p>Parameters</p> <p><i>axis</i></p> <p>Type: System.Int32 Axis to get the relative displacement from {0-2}</p> <p>Return Value</p> <p>Type: Int64 Displacement of the axis in pm</p>
<pre>public void IDS_ResetAxes()</pre>	<p>Reset the displacement of all axes to 0 to start a new measurement on the fly.</p>
<pre>public void IDS_ResetAxis(int axis)</pre>	<p>Reset the displacement of axis to 0 to start a new measurement on the fly.</p> <p>Parameters</p> <p><i>axis</i></p> <p>Type: System.Int32 Axis to reset {0-2}</p>
<pre>public long IDS_GetAbsolutePosition(int axis)</pre>	<p>Get the absolute position of an axis taken upon measurement start. The absolute position is not updated during system operation. A dynamic absolute position can be</p>

)	obtained by adding the displacement value to the absolute position.
	Parameters
	<i>axis</i>
	Type: System.Int32
	Number of the axis (valid values: 0 - 2)
	Return Value
	Type: Int64
	Length/absolute position of the axis in pm
public AxesPositions IDS_GetAbsolutePositions()	Get the absolute position of all axis
	Return Value
	Type: AxesPositions
	Instance of a class with the three absolute positions for the three axes
	public AxesPositions(<div data-bbox="734 840 861 873"><i>int warning,</i></div> <div data-bbox="734 884 893 918"><i>long position0,</i></div> <div data-bbox="734 929 893 963"><i>long position1,</i></div> <div data-bbox="734 974 893 1008"><i>long position2</i></div>
)
	Parameters
	<i>warning</i>
	Type: System.Int32
	Possible warning value from the embedded system
	<i>position0</i>
	Type: System.Int64
	Value for position no. 0
	<i>position1</i>
	Type: System.Int64
	Value for position no. 1
	<i>position2</i>
	Type: System.Int64
	Value for position no. 2
public void IDS_SetAverageN (<div data-bbox="303 1691 478 1724"><i>int averageN</i></div>	Sets the averaging value. The averaging time is calculated by $(2^n) \cdot 40\text{ns}$, where n is the averaging value.
	Parameters
	<i>averageN</i>
	Type: System.Int32
	A value from 0 to 24

<code>public int IDS_GetAverageN()</code>	<p>Returns the current averaging value. The averaging time is calculated by $(2^n) \cdot 40\text{ns}$, where n is the averaging value.</p> <p>Return Value</p> <p>Type: Int32 A value from 0 to 24</p>
Error handling	
<code>public int IDS_GetSystemError()</code>	<p>Returns the system error if any occurred. ERR_OK=0 if no error occurred since the last time this method was called. The method automatically resets the system error to ERR_OK. The function returns an integer number which represents the error. The number can be converted into a string using the function IDS_ErrorNumberToString, which is described below.</p> <p>Return Value</p> <p>Type: Int32 System Error</p>
<code>public void IDS_ResetError()</code>	Clear displacement errors (e.g. beam interruption), while measurement is running.
<code>public class AttocubeIdsException : ApplicationException</code>	Base exception for all custom exceptions in the Attocube IDS library
<u>Example:</u>	<p>Getting the axis dependent error</p> <pre>try { DisplacementAxis[i,0] = client.IDS_GetAxisDisplacement(0); } catch (AttocubeIdsException exc) { labelErrorMessage.Text = client.IDS_ErrorNumberToString(0, exc.ErrorCode); }</pre>
<code>public string IDS_ErrorNumberToString(int language, int errorNumber)</code>	<p>Return a string that represents the error number</p> <p>Parameters</p> <p><i>language</i></p> <p>Type: System.Int32 Language of the error description { 0 = System, 1 = English }</p> <p><i>errorNumber</i></p> <p>Type: System.Int32 Error number to translate</p>
<u>Example:</u>	<p>Getting the warning message during measurement</p> <pre>int Error = client.IDS_GetAxesDisplacement().Warning; string ErrorString = client.IDS_ErrorNumberToString(0, Error);</pre>
<code>public string IDS_GetCurrentMode()</code>	<p>Return a string that describes the current mode in which the ids currently is</p> <p>Return Value</p> <p>Type: String A short description of the current mode: "system idle", "measurement starting",</p>

"measurement running", "optics alignment starting", "optics alignment running",
"pilot laser enabled"

Info	
<code>public string IDS_GetDllInfo()</code>	<p>Return internal info about the DLL - whether or not it's connected, and if it is connected, what address it's connected to</p> <p>Return Value</p> <p>Type: String String indicating some info on connection status of client DLL</p>
<code>public string IDS_GetDllVersion()</code>	<p>Returns the current version of the Attocube IDS DLL</p> <p>Return Value</p> <p>Type: String Version in the form X.Y.Z</p>
<code>public string IDS_GetFpgaVersion()</code>	<p>Get the version of the current FPGA in use</p> <p>Return Value</p> <p>Type: String Version in the form X.Y.Z</p>
<code>public string IDS_GetSoftwareVersion()</code>	<p>Get the version of the ids software.</p> <p>Return Value</p> <p>Type: String Version in the form X.Y.Z</p>
<code>public string IDS_GetGateway()</code>	<p>Get the gateway of the target</p> <p>Return Value</p> <p>Type: String Gateway in the form XXX.XXX.XXX.XXX</p>
<code>public string IDS_GetIpAddress()</code>	<p>Get the IP address of the target</p> <p>Return Value</p> <p>Type: String IP address in the form XXX.XXX.XXX.XXX</p>
<code>public int IDS_GetPassMode()</code>	<p>Returns the pass mode, also called the sensor head information</p> <p>Return Value</p> <p>Type: Int32 Mode { 0->single pass, 1->dual pass, 2->xs sensor single pass, 3 -> customized sensor head }</p>
<code>public string IDS_GetSubnetMask()</code>	<p>Get the subnet mask of the target.</p> <p>Return Value</p> <p>Type: String Subnet mask in the form XXX.XXX.XXX.XXX</p>
<code>public int IDS_GetSwUpdateProgress()</code>	<p>Get the progress software update progress. The values are 0-99 during update and 100 for done. The update first must be startet.</p> <p>Return Value</p> <p>Type: Int32</p>

	progress
public void IDS_SetPassMode (int mode)	<p>Returns the pass mode, also called the sensor head information.</p> <p>Parameters</p> <p>mode</p> <p>Type: System.Int32 Mode { 0->single pass, 1->dual pass, 2->xs sensor single pass, 3 -> customized sensor head }</p>
ECU	
public void IDS_EnableEcu ()	Enable the Environment Compensation Unit.
public void IDS_DisableEcu ()	Disable the Environment Compensation Unit.
public bool IDS_GetEcuConnected ()	<p>Returns if the ECU is connected. This means it is enabled and recognized by the IDS.</p> <p>Return Value</p> <p>Type: Boolean True if connected, false if not</p>
public bool IDS_GetEcuEnabled ()	<p>Returns if the ECU is enabled or disabled</p> <p>Return Value</p> <p>Type: Boolean True if enabled, false if not</p>
public double IDS_GetRefractiveIndex ()	<p>Get the refractive index from the ECU (as index).</p> <p>Return Value</p> <p>Type: Double Refractive index</p>
public double IDS_GetTemperatureInDegrees ()	<p>Get the temperature in Degrees from the ECU.</p> <p>Return Value</p> <p>Type: Double Temperature in degrees</p>
public double IDS_GetPressureInHPa ()	<p>Get the pressure from the ECU in hPa.</p> <p>Return Value</p> <p>Type: Double Pressure in hPa</p>
public double IDS_GetHumidityInPercent ()	<p>Get the humidity from the ECU in Percent.</p> <p>Return Value</p> <p>Type: Double Humidity in percent</p>
Realtime Output	
public void IDS_SetRtOutMode (int rtOutMode)	<p>Set the mode of the realtime output.</p> <p>Parameters</p> <p>rtOutMode</p> <p>Type: System.Int32</p>

	Realtime output mode {0=HSSL (LVTTTL), 1=HSSL (LVDS), 2=AquadB (LVTTTL), 3=AquadB (LVDS), 4=SinCos (LVTTTL Error Signal), 5=SinCos (LVDS Error Signal) }
public int IDS_GetRtOutMode()	Get configured mode of the realtime output Return Value Type: Int32 The output mode {0=HSSL (LVTTTL), 1=HSSL (LVDS), 2=AquadB (LVTTTL), 3=AquadB (LVDS), 4=SinCos (LVTTTL Error Signal), 5=SinCos (LVDS Error Signal) }
Realtime Output: HSSL	
public void IDS_SetPeriodHsslClk(int period)	The HSSL clock period in ns. Parameters <i>period</i> Type: System.Int32 Period {40ns - 10200ns}
public int IDS_GetPeriodHsslClk()	The period of the HSSL clock. Return Value Type: Int32 Period {40ns - 10200ns}
public void IDS_SetResolutionHsslHigh(int resolution)	The high parameter of the HSSL resolution. The hssl output will be a value between hssl low resolution and HSSL high resolution. Parameters <i>resolution</i> Type: System.Int32 Resolution {1-48}
public int IDS_GetResolutionHsslHigh()	The high parameter of the HSSL resolution. The hssl output will be a value between HSSL low resolution and HSSL high resolution. Return Value Type: Int32 Resolution {1-48}
public void IDS_SetResolutionHsslLow(int resolution)	The low parameter of the HSSL resolution. The hssl output will be a value between hssl low resolution and HSSL high resolution. Parameters <i>resolution</i> Type: System.Int32 Resolution {1 - 48}
public int IDS_GetResolutionHsslLow()	The low parameter of the HSSL resolution. The hssl output will be a value between hssl low resolution and HSSL high resolution. Return Value Type: Int32 Resolution {0 - 47}
public void IDS_SetPeriodHsslGap(The gap between two values in ns. Parameters

<code>int gap</code>	<i>gap</i>
)	Type: System.Int32 Gap {40pm - 10200pm}
<code>public int IDS_GetPeriodHsslGap()</code>	The gap between two values in ns. Return Value Type: Int32 Gap {40pm - 10200pm}
Realtime Output: Sin/Cos and A-quad-B	
<code>public void IDS_SetPeriodSinCosClk(int period)</code>	The period of sin/cos and AquadB update clock in ns. Parameters <i>period</i> Type: System.Int32 Period {40ns - 10200ns}
<code>public int IDS_GetPeriodSinCosClk()</code>	The period of sin/cos and A-quad-B update clock in ns. Return Value Type: Int32 Period {40pm - 10200pm}
<code>public void IDS_SetResolutionSinCos(int resolution)</code>	The resolution of sin/cos and A-quad-B in 1pm/90°. Parameters <i>resolution</i> Type: System.Int32 Resolution {1pm - 65535pm}
<code>public int IDS_GetResolutionSinCos()</code>	The resolution of sin/cos and AquadB in 1pm/90° Return Value Type: Int32 Resolution { 1pm to 65535pm }

VI.2. JSON-RPC

The IDS3010 allows platform-independent communication using JSON-RPC. It supports the transport protocols TCP, HTTP and WebSocket. Attocube provides only limited support for communication via JSON-RPC.

Transport protocols

TCP

Uses communication port 9090.

HTTP

Uses communication port 8080.

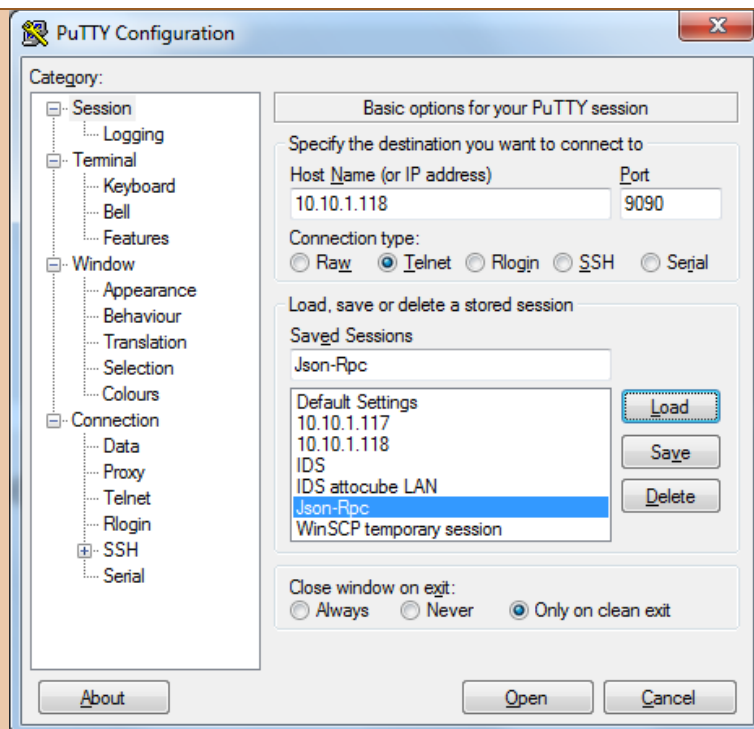
Target URL is constructed like this: `http://$(ipAddress):$(port)/api/json`

WebSocket

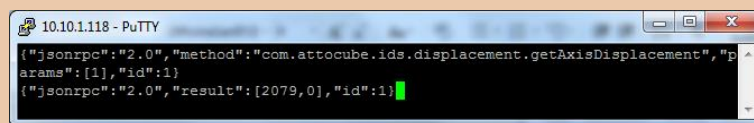
Uses communication port 8081.

Supported from software version 22 or software image 1.1.11 (versions with "About" page in WebServer).

Calling a JSON RPC 2.0 method	<p>A JSON RPC method is called by sending a POST Request over http to the specified URL with the following JSON Data:</p> <pre>{ "jsonrpc": "2.0", "method": "<method>", "params": [<param [0]>, <param [1]>, ...], "id": <call id>}</pre> <p><i><method></i>: String defined in chapter 2.2.</p> <p><i><param x></i>: Parameter for the method call if the PARAM is put between two ", it is a string. Without " it is a number</p> <p><i><call id></i>: A unique id to find the corresponding answer</p>
<u>Example:</u>	<pre>{ "jsonrpc": "2.0", "method": "com.attocube.ids.displacement.getAxisDisplacement", "params": [1], "id": 1}</pre>
Receiving a JSON RPC 2.0 response	<p>The JSON RPC method answer is then sent back as payload to the OK message:</p> <pre>{ "jsonrpc": "2.0", "result": [<return value [0]>, <return value [1]>, ...], "id": <call id>}</pre> <p><i><return value [x]></i>: The return parameters</p> <p><i><call id></i>: The unique id of the method call</p>
<u>Example:</u>	<pre>{ "jsonrpc": "2.0", "results": [0, 4], "id": 1}</pre>
Naming convention	<p>JSON RPC:</p> <pre><return value[0]> <METHOD> (<param[0]>, <param[1]>, <&return value[1]>, <&return value[2]>)</pre> <p>All return values beside "return value [0]" are marked with a "&" sign. If the JSON RPC method returns an error code it is marked with a *. In this case the first return value is the error code, "return value[0]" is then the second return value.</p> <p>A JSON RPC result that can return an error would then look as following:</p> <pre>{ "jsonrpc": "2.0", "result": [<error number>, <return value [0]>, ...], "id": <call id>}</pre>
<u>Example:</u>	Open a Telnet connection with PuTTY.
Communication via PuTTY	



Sending Json-Rpc commands in the command line interface.



WebSocket

Usage of Socket.IO on server side

When Socket.IO is used on the server side, the WebSocket connection must be established in a special way. In this case, the URL is „ws://<IP>:8081/socket.io/?EIO=2&transport=websocket“.

Example:

Implementation in Python:

```
import websocket

ws = websocket.WebSocket()

ws.connect("ws://172.17.1.116:8081/socket.io/?EIO=2&transport=websocket")

ws.send("test") # From here on communication via JSON-RPC
```

VI.2.1.JSON Methods

All DLL functions are also available by means of Json-Rpc. The following list provides an overview over the Json-Rpc commands. A detailed description is provided in the DLL chapter.

	System (Update/reset)
Network	string com.attocube.system.network.getIpAddress()
	string com.attocube.system.network.getSubnetMask()
	string com.attocube.system.network.getGateway()
	void com.attocube.system.uploadSoftwareImageBas64 (int32 filePos, string base64Block)*

	<code>void com.attocube.system.softwareUpdateBase64()*</code>
	<code>int32 com.attocube.system.getSwUpdateProgress()*</code>
	<code>void com.attocube.system.factoryReset()*</code>
	<code>void com.attocube.system.rebootSystem()*</code>
	<code>string com.attocube.system.getSoftwareVersion()</code>
	<code>string com.attocube.system.getFpgaVersion()</code>
	<code>string com.attocube.system.errorNumberToString (int32 language, int32 errorNumber)</code>

	Realtime Output
Output mode	<code>int32 com.attocube.ids.realtime.getRtOutMode()</code> <code>void com.attocube.ids.realtime.setRtOutMode(int32 rtOutMode)*</code>
HSSL	<code>int32 com.attocube.ids.realtime.getResolutionHsslLow()</code> <code>void com.attocube.ids.realtime.setResolutionHsslLow(int32 resolution)*</code> <code>int32 com.attocube.ids.realtime.getResolutionHsslHigh()</code> <code>void com.attocube.ids.realtime.setResolutionHsslHigh(int32 resolution)*</code> <code>int32 com.attocube.ids.realtime.getPeriodHsslClk()</code> <code>void com.attocube.ids.realtime.setPeriodHsslClk(int32 period)*</code> <code>int32 com.attocube.ids.realtime.getPeriodHsslGap()</code> <code>void com.attocube.ids.realtime.setPeriodHsslGap(int32 gap)*</code>
A-quad-B	<code>int32 com.attocube.ids.realtime.getPeriodSinCosClk()</code>
Sin/Cos	<code>void com.attocube.ids.realtime.setPeriodSinCosClk(int period)*</code> <code>int32 com.attocube.ids.realtime.getResolutionSinCos()</code> <code>void com.attocube.ids.realtime.setResolutionSinCos(int resolution)*</code>

	Alignment
Pilot laser	<code>bool com.attocube.ids.pilotlaser.isEnabled()</code> <code>void com.attocube.ids.pilotlaser.enable()*</code> <code>void com.attocube.ids.pilotlaser.disable()*</code>
Optics alignment	<code>void com.attocube.ids.system.startOpticsAlignment()*</code> <code>void com.attocube.ids.system.stopOpticsAlignment()*</code>
Contrast	<code>int32 com.attocube.ids.adjustment.getContrastInPermill(int axisNumber)*</code>
Test signal stability	<code>int32 com.attocube.ids.test.getBaseBandAmplitude()</code>

	Position
Averaging	<code>int32 com.attocube.ids.displacement.getAverageN()</code> <code>void com.attocube.ids.displacement.setAverageN(int32 averageN)*</code>

Absolute position	<code>void com.attocube.ids.displacement.getAbsolutePosition (int axis, int64 & position)*</code>
	<code>void com.attocube.ids.displacement.getAbsolutePositions(int64 &position0, int64 &position1, int64 &position2)*</code>
Displacement	<code>int64 com.attocube.ids.displacement.getAxisDisplacement(int axis)*</code>
	<code>Void com.attocube.ids. displacement.getAxesDisplacement(int64 &displacement0, int64 &displacement1, int64 &displacement2)*</code>
Reset axes to zero	<code>void com.attocube.ids.system.resetAxis(int32 axis)*</code>
	<code>void com.attocube.ids.system.resetAxes()*</code>
IDS System	
	<code>string com.attocube.ids.system.getCurrentMode()</code>
	<code>void com.attocube.ids.system.startMeasurement()*</code>
	<code>void com.attocube.ids.system.stopMeasurement()*</code>
	<code>void com.attocube.ids.system.getSystemError()*</code>
Single/Double pass	<code>int32 com.attocube.ids.axis.getPassMode()</code>
	<code>void com.attocube.ids.axis.setPassMode(int32 mode)*</code>
ECU	
Status	<code>bool com.attocube.ecu.isEnabled()</code>
	<code>void com.attocube.ids.ecu.enable()</code>
	<code>void com.attocube.ids.ecu.disable()</code>
Get sensor data	<code>double com.attocube.ecu.getTemperatureInDegrees()</code>
	<code>double com.attocube.ecu.getPressureInHPa()</code>
	<code>double com.attocube.ecu.getHumidityInPercent()</code>
	<code>double com.attocube.ecu.getRefractiveIndex()</code>

VI.3. Error handling

The error status of the IDS can be controlled using the warning and exception function.

VI.3.1.Implemented Errors

No error		Error Description
0	ERR_OK=0x00,	No error
Warnings (return value):		
1	ERR_MEASUREMENT_AXIS_0,	Value of axis 0 is invalid (is axis aligned?)
2	ERR_MEASUREMENT_AXIS_1,	Value of axis 1 is invalid (is axis aligned?)

3	ERR_MEASUREMENT_AXIS_01,	Values of axes 0 and 1 are invalid (are axis aligned?)
4	ERR_MEASUREMENT_AXIS_2,	Value of axis 2 is invalid (is axis aligned?)
5	ERR_MEASUREMENT_AXIS_02,	Values of axes 0 and 2 are invalid (are axis aligned?)
6	ERR_MEASUREMENT_AXIS_12,	Values of axes 1 and 2 are invalid (are axis aligned?)
7	ERR_MEASUREMENT_AXIS_123,	Values of axes 0, 1 and 2 are invalid (are axis aligned?)
8	ERR_SIGNAL_OVERDRIVEN,	Signal too strong, overdrives the input

Errors (error number when exception was thrown):

2048	ERR_INT_RANGE,	Integer range check fails
2049	ERR_DOUB_RANGE,	Double range check fails
2050	ERR_INV_IP_ADDR,	IP format is invalid
2051	ERR_IP_CONF,	IP configuration is invalid
2052	ERR_HSSL_RESOLUTION,	HSSL resolution is not valid
2053	ERR_SYSTEM_IS_BUSY,	System is busy
2054	ERR_STATE_MACHINE_TIMEOUT,	State machine is not responding
2055	ERR_SWUPDATE_FAILED,	Software update fails with unknown error
2056	ERR_SWUPDATE_FILE_NOT_FOUND,	Software update can not find file
2057	ERR_EXTERNAL_PROGRAM_FAILED,	An external program call failed by this action
2058	ERR_ACTION_DENIED,	This action is currently not permitted (measurement/alignment running?)
2059	ERR_NOT_AVAILABLE_IN_THIS_VERSION,	This feature is not supported in the current Firmware Version
2060	ERR_DISPLACEMENT_NOT_ENABLED,	Displacement is currently not enabled
2061	ERR_DISPLACEMENT_ERROR,	Displacement error (e.g. beam interrupted?)
2062	ERR_ADJUST_LASER_TEMPERATURE,	Error during laser temperature adjustment
2063	ERR_ADJUST_MODULATION_AMPLITUDE,	Error during modulation amplitude adjustment (master axis correct?)
2064	ERR_ADJUST_MODULATION_PHASE,	Error during modulation phase adjustment
2065	ERR_START_OPTICS_ALIGNMENT,	Can not start optics alignment
2066	ERR_SET_MEASUREMENT_ZERO,	Can not set measurement to zero
2067	ERR_MEASURE_DISPLACEMENT,	Error during displacement measurement
2068	ERR_STOP_SYSTEM,	Can not stop system
2069	ERR_ENABLE_PILOT_LASER,	Can not enable pilot laser
2070	ERR_DISABLE_PILOT_LASER,	Can not disable pilot laser
2071	ERR_INITIALIZE_FPGA,	Can not initialize FPGA
2072	ERR_STOP_OPTICS_ALIGNMENT,	Can not stop optics alignment
2073	ERR_ADJUST_PHOTO_OFFSET,	Can not adjust photo offset

2074	ERR_ENABLE_WAVELENGTH_STABILIZATION,	Can not enable wavelength stabilization
2075	ERR_ENABLE_LASER,	Can not enable the laser
2076	ERR_DISABLE_LASER,	Can not disable the laser
2077	ERR_GET_ABSOLUTE_DISTANCE,	Can not get the absolute distance
2078	ERR_SET_DISTANCE_SCALING,	Can not set the distance scaling factor
2079	ERR_AXIS_NOT_INITIALIZED,	Axis not initialized (axis aligned?)
2080	ERR_SET_ABS_DISTANCE_SCALING,	Can not set absolute distance scaling
2081	ERR_SET_DISTANCE_CORRECTION,	Can not set the distance correction calculated by ECU
2082	ERR_ENABLE_LUA_SYSLOG,	Can not enable syslog for LUA
2083	ERR_OMEGA_LEVEL_BELOW_MINIMUM,	The omega level is below the minimum threshold (beam interrupted?)
2084	ERR_OMEGA_AMPLITUDE_BELOW_MINIMUM,	The omega amplitude is below the minimum threshold (beam interrupted?)
2085	ERR_BASEBAND_AMPLITUDE_BELOW_MINIMUM,	The baseband amplitude is below the minimum threshold (beam interrupted?)
2086	ERR_WAVELENGTH_NOT_STABILIZED,	Not able to stabilize the wavelength (problem with gas cell?)
2087	ERR_UNKNOWN_MEASUREMENT_PROBLEM,	Unknown error during measurement (beam interrupted?)
2088	ERR_REFRACTIVE_INDEX_NOT_AVAILABLE,	Refractive index is not readable from ECU (is ECU connected correctly?)
2089	ERR_PACKAGE_INEXISTENT,	Specified license package does not exist
2090	ERR_LICENSE_MISSING,	License text is missing

VI.3.2.Error handling in C#

The error handling in C# is realized using a try-catch statement. An example code is shown below.

Example code:	<code>try</code>
Exception handling	<code>{</code>
	<code> attocubeIds.IDS_StartMeasurement();</code>
	<code>}</code>
	<code>catch (AttocubeIdsException exc)</code>
	<code>{</code>
	<code> labelErrorMessage.Text =</code>
	<code> client.IDS_ErrorNumberToString(0, exc.ErrorCode);</code>
	<code>}</code>

During operation, the condition of the IDS can be controlled using the warning signal implemented in the DLL.

Example code:	<code>int Error = client.IDS_GetAxesDisplacement().Warning;</code>
Warning handling	<code>string ErrorString = client.IDS_ErrorNumberToString(0, Error);</code>

attocube systems
Königinstrasse 11a
D - 80539 München, Germany
Phone: +49 89-2877 809-0
Fax: +49 89-2877 809-19
E-Mail: info@attocube.com
www.attocube.com

For technical queries, contact:
support@attocube.com

North America Support Hotlines:
+1 212 962 6930 (East Coast Office)
+1 510 649 9245 (West Coast Office)

South America Support Hotline:
+1 510 649 9245