MICRO-EPSILON

**Instruction Manual**

# boreCONTROL SDK 1.5

MICRO-EPSILON
MESSTECHNIK
GmbH & Co. KG
Königbacher Strasse 15

94496 Ortenburg / Germany

Tel. +49/8542/168-0
Fax +49/8542/168-90
e-mail info@micro-epsilon.de
www.micro-epsilon.com

**Contents**

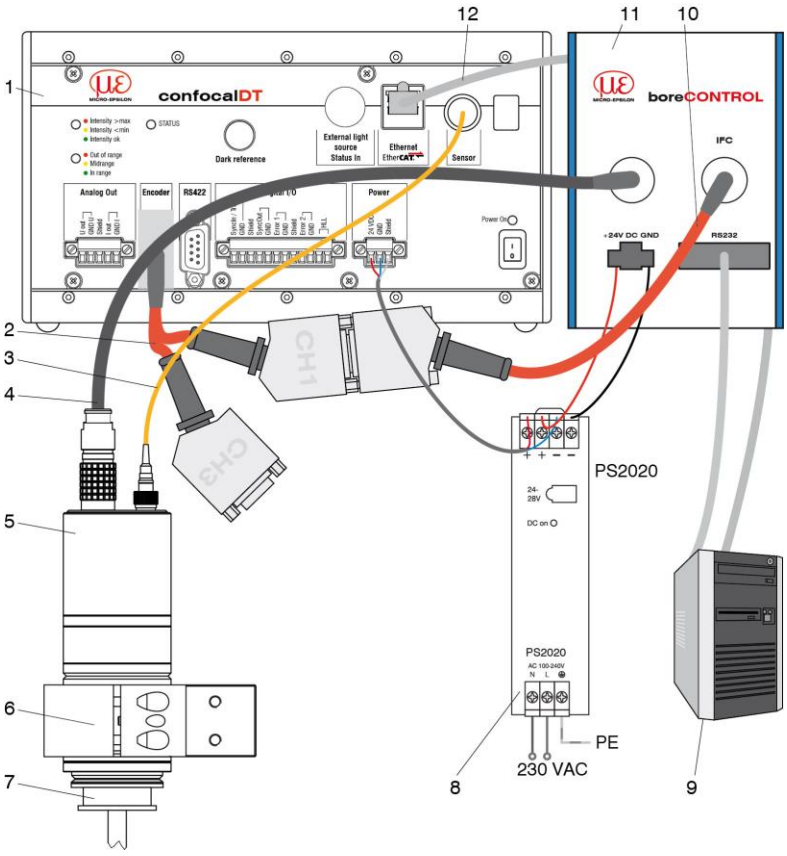
# 1. Introduction

This manual describes the usage of boreCONTROL SDK (software development kit) and the included boreCONTROL.dll (dynamic link library) for integration into customer generated applications.

The boreCONTROL.dll is used to control the boreCONTROL measuring system, which consists out of sensor pen and controller IFC24x1 laid out for applications in industrial environments. By using the supplied dll, it is possible to adjust sensor parameters, scan data can be read and basic measurement values may be calculated.

boreCONTROL is applied for acquisition of 2D scan data, typically generated inside of bore holes. Profile data of the scanned surface is captured by the sensor pen using the confocal measurement principle. The attached rotation unit with controller BCC2410 rotates the sensor pen and therefore allows circular scanning of the target object. 3D scans of the target may be generated by moving the rotation unit in pen direction.

The controller IFC24x1 of boreCONTROL collects position, intensity and distance data on demand and delivers them to a controlling and evaluating PC using an ethernet interface. The boreCONTROL.dll deals with the control of the attached boreCONTROL components and provides data for the application written by the customer. Simple evaluation routines are also included in the range of functions.




1: BCC2451 sensor controller
2: Y adapter cable (optional)
3: C2400/PT-x optical fiber
4: PC2410-x data cable
5: BCM2410 rotation unit
6: Mounting adapter
7: BCS241x-x sensor probe
8: PS2020 power supply
9: PC
10: SC2410-0.5 synchronization cable
11: BCC2410 motor controller
12: LAN cable

*Figure 1: Hardware components of boreCONTROL and connection to PC supplied by the customer*

To be able to realize different measurement ranges and resolutions, different sensor types are available. These sensor pens can be exchanged by the user when needed.

In order to be usable for as many development environments and compilers as possible, the dll delivered with boreCONTROL is based on an interface using C-functions with stdcall calling convention. Therefore the dll may also be used in C++, Delphi, Visual Basic, Python or other programming languages (provided that compatible data types exist).

In this manual, interfacing to the dll is described only for the C programming language. Usage from C++ or other programming languages can be derived from this documentation.

## 1.1 Installation and Preparation for Operation

Information on hardware setup of boreCONTROL can be found in the “boreCONTROL Instruction Manual”. Follow the documented steps for assembly of the hardware components.

Communication to the controller uses an Ethernet connection. The IP address of the controller is pre-configured to 169.254.168.150 and may be reconfigured using the web interface. The address is required to establish the connection to the controller. Installation of additional drivers is not necessary.

To execute the delivered Software, it may be necessary to install the Microsoft redistributable provided on CD (Support\vcredist_x86\vcredist_x86.exe).

After these basic steps the hardware is set up for use with the boreCONTROL.dll.

## 1.2 Basic Definitions

The measurement distances are derived from spectral analysis of the light reflected from the target. The following figure shows a typical spectrum.

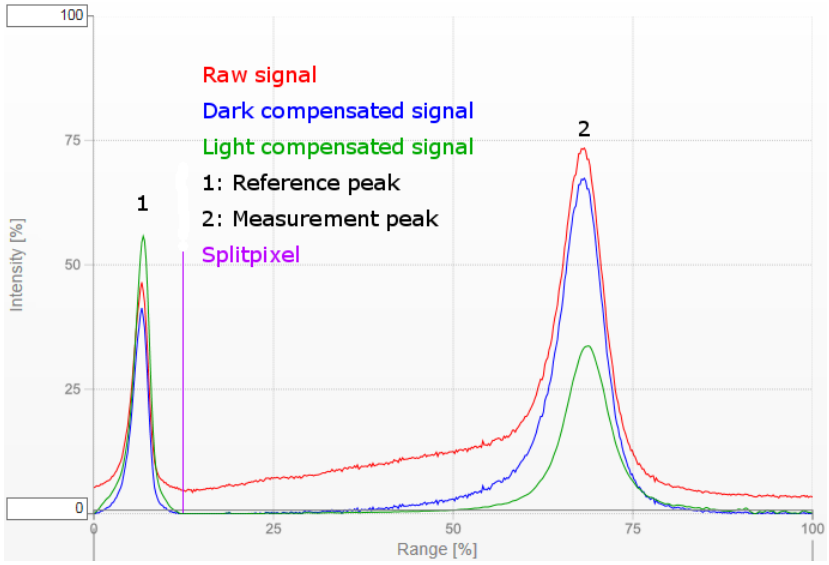



*Figure 2: Video signal acquired by the controller. Description is below.*

The following terms are used throughout the manual:

- **Raw signal:** The spectrum acquired by the controller, including collected effects from the optical components
- **Dark compensated signal:** The spectrum acquired by the controller, effects from the optical components are removed
- **Light compensated signal:** The spectrum acquired by the controller, the spectrum of the integrated light source is compensated.
- **Splitpixel:** A fixed position for each sensor pen which separates the reference peak (left) from the measurement peak (right).
- **Measurement peak:** The reflected light from the target. Brightness and measurement distance in sensor units are generated from the measurement peak. The measurement peak is to the right of the splitpixel.
- **Reference peak:** Sensor signal originating from a reflection coating applied to the sensor pen, mapped into the controller. It’s always present and is used for temperature compensation. The reference peak is evaluated like the measurement peak and both are computed to a measurement distance in $\mu$m using a calibration table. The reference peak is to the left of the splitpixel.
- **Intensity:** Brightness of a measurement point in percent between 0% (dark) and 100% (bright).
- **Data point/measurement point:** A single tuple of data values (angle, intensity, distance, encoder counters, etc.) at a single point of time.
- **Profile:** A profile consists out of several measurement points. A profile normally represents a complete revolution of the borehole sensor.
- **Measurement value:** Using circle fitting, new information is generated for the target object. This includes radius and center of the fitted circle.

- **Point rate:** The number of data points acquired per second. From point rate and rotation speed of the borehole sensor the number of data points per profile can be computed. Also referred to as scan rate.
- **Offset:** Added value to adjust small deviations for absolute distances due to sensor pen exchange. The offset must be readjusted by an offset calibration using a calibration target after every exchange of a sensor pen.

## 1.3 Measurement Principle of boreCONTROL

The rotation unit is used for rotating the sensor pen inside the target object with a user defined speed. The sensor collects distance and intensity data and combines them with current angle position and penetration depth, if applicable.

Temperature deviations while measuring are collected using an additional reflection coating on the sensor pen opening and are compensated by the software using provided calibration tables.

After assembly and also on each exchange of a sensor pen an offset calibration has to be done once using a calibration target. The absolute distance of a measurement point is derived from the measured distance plus the calibrated offset.

The data points delivered may be collected until a complete revolution is completed and evaluated using a circle fit.

# 2. Course of Action for selected Tasks

In this chapter the elementary steps for operating the boreCONTROL components are described and the parameters used for scanning are explained.

To assure correct operation of the system, different calibration steps have to be performed. The calibration tables available as file are provided by MICRO-EPSILON and are used to linearize the measured distance values for delivered sensor pens.

For initial operation and after each exchange of a sensor pen, a dark value compensation should be done, the according calibration table has to be selected and an offset calibration must be executed.

The necessary steps are also described in this chapter. The assembly and wiring of the boreCONTROL components must have been done before.

For more detailed information please consult the following chapters containing function overview, data structure description, enumeration of error codes and explanation of the contained demonstration programs. Furthermore, in the scope of supply a documentation viewer named Documentation.exe is available, which contains this document and the automatically generated SDK interface reference, among others.

## 2.1 Connecting with boreCONTROL-Components

To be able to use the functionality provided by the boreCONTROL SDK, it is required to establish a connection with the components. This is done by calling the function “boreConnect”, which initializes the the bore_t data structure.

```
bore_t bore;
bore_result_t result;

result = boreConnect(&bore, NULL);
```

With configurations differing from the delivery status (e.g. modified IP-address of the controller or COM port number of the rotation unit), the second parameter needs to be filled manually. For the delivery status this is done as follows:

```
bore_connect_t param;

memset(&param, 0, sizeof(param));
param.controller = BORE_CONTROLLER_IFC2461;
strcpy(param.tcpip_address, "169.254.168.150");        /* preset, may be omitted */
param.tcpip_port = 23;                                 /* preset, may be omitted */
param.rotation_unit = BORE_ROTATION_BCC2410;
param.rotation_com_port = 1;                           /* preset, may be omitted */
```

The call of “boreConnect” needs to be modified then:

```
result = boreConnect(&bore, &param);
```

If the return value is BORE_OK, the components have been initialized successfully.

To provide an equal temperature distribution in the rotation unit, it is recommended to either run the rotation unit with constant speed all the time, or start and stop it temporarily for measurement only. The latter avoids warm-up of the components and is done by “boreRotationStart” and “boreRotationStop” accordingly.

```
boreRotationSetSpeed(&bore, 2.0);    /* set rotation speed to 2 Hz */
boreRotationStart(&bore);
```

To disconnect all components, call „boreDisconnect”:

```
boreDisconnect(&bore);
```

## 2.2 Settings for boreCONTROL

Use the following parameters to tune the scanning process:

- **Point rate [Hz]:** Supply the scan rate by calling the function “boreSetScanrate”. Note that the number of data points per revolution also depends on the rotation speed set. Setting the point rate determines the maximum exposure time as 1000000 / scan rate $\mu$s. Higher point rates result in

lower maximum exposure times and therefore in lower intensity readings.
For older controllers / firmware releases before 007.119.148.02c the available point rates are restricted to the discrete values 100, 200, 300, 1000, 2500, 5000 and 10000 Hz. Newer IFC2461 controllers allow a free choice of point rate between 100 Hz and 25000 Hz in steps of 100 Hz.

- **Shutter time [$\mu$s]:** By calling the function “boreSetShutterTime”, the brightness of the received signal can be adjusted. Depending on the reflection properties of the target, the value has to be selected so that measurement point intensity values lie between 10% and 90%. Intensity values below 10% indicate a low reflecting target. Eventually reduce the point rate to allow higher shutter time settings. Isolated measurement points with intensity outside of the recommended range are unproblematic.
  For measurement of surfaces with highly varying reflectivity the controller can be operated with an acquisition mode providing automatic shutter time setting.
- **Rotation speed [Hz]:** Using the function “boreRotationSetSpeed”, the rotation speed of the sensor pen can be adjusted in range 0.1 Hz to 10 Hz. Higher rotation speed with same point rate leads to averaging distance and intensity values along a greater distance on the target. To produce optimum accuracy, very slow or very fast rotation speed should be avoided if possible.
- **Intensity thresholds [%]:** By calling the functions “boreEvalSetIntensityThres” and “boreEvalSetIntensityUpperThres”, the intensity thresholds for the evaluation process are set. This is necessary because the distance value may not be accurately determined for very dark or saturated bright measurement points. Data points with intensity below and above the thresholds (recommended values are 10% and 90%) are ignored in circle fit and measurement value computation. Measurement points with intensity less than 5% should never be used.

## 2.3 Sensor Pens with Optical Temperature Compensation and Operation Modes

Up to date sensor pens feature an optical compensation of measurement errors resulting from temperature changes. For that purpose, an additional reference peak is overlaid outside of the measurement range by a coating on the sensor pen. Its position is modified by temperature only. This reference peak is internally used by the software for correction of the measurement peak position.

The dll needs a calibration table to be able to do the linearization and temperature compensation of the measurement value. The table is loaded by “boreRefCalibLoad”.

The temperature compensation is available in four different modes. The mode may be selected by “boreRefCalibSetMode”.

- The mode BORE_REFCALIB_ONREQUEST is used for normal measurement. Before starting the actual measurement, the reference peak needs to be determined by calling “boreRefCalibSingle”. This reference peak is used for temperature compensation of the measurement values acquired afterwards. The acquisition of the reference peak needs to be redone periodically dependent on expected temperature change either within a minute or significantly longer time intervals.
  To achieve an optimum intensity for the reference peak, point rate and shutter time can be adjusted separately from the measurement settings by using “boreRefCalibSetScanrate” and “boreRefCalibSetShutterTime” accordingly. Therefore the shutter time used for measurement does not affect the temperature compensation and either highly reflecting or dark objects can be measured. The recommended values of 400 $\mu$s shutter time at point rate 2500 Hz match the settings at in-house calibration.

- For measurement objects with widely differing reflectance, e.g. bores in composite CFK/metal material, there is no shutter time setting available which would provide acceptable results in both the low and high reflecting part. For this purpose either a dual shutter mode (BORE_REFCALIB_ONREQ2AUTO), automatically altering between two preset shutter times, or an auto shutter mode (BORE_REFCALIB_ONREQAUTO), which adjusts shutter time automatically in the possible range defined by the set point rate. Both automatic modes work analogous to the mode BORE_REFCALIB_ONREQUEST described above, where temperature compensation information is being acquired before the actual measurement.
  While the shutter time does not necessarily need to be set in automatic mode, two sufficiently different shutter times must be set using “boreSetShutterTimes” for dual shutter mode.The shutter time used for acquisition of a measurement point is available as “shutter” entry in bore_meas_t. To get the reflectance of the measurement object it’s advisable to divide the intensity value by the shutter time set by the controller.

- For standard reflecting measurement objects the mode BORE_REFCALIB_ALWAYS exists, which acquires the reference peak with every single measurement point. Using this mode, a separate

acquisition before measurement is not necessary. However, one has to make sure that the reference peak intensity is in range 10% to 90% with the shutter time used for measurement. Because of this restriction, this mode is rarely used.

## 2.4 Dark Level Compensation for boreCONTROL

For initial operation and after exchanging a sensor pen, a dark level compensation needs to be performed, so the signal level without any measurement target is retrieved.

For optically compensated sensor pens the reference peak is always visible. The area around the reference peak therefore has to be ignored when doing a dark value compensation. The information about the location of the reference peak is a sensor property and is supplied by the according calibration file. After having loaded the calibration file once, the controller button for dark value compensation may be used as well. The following figure shows a typical dark table, which consists out of the spectral effects of the optical components which are always present, and the removed reference peak to the left of the splitpixel.

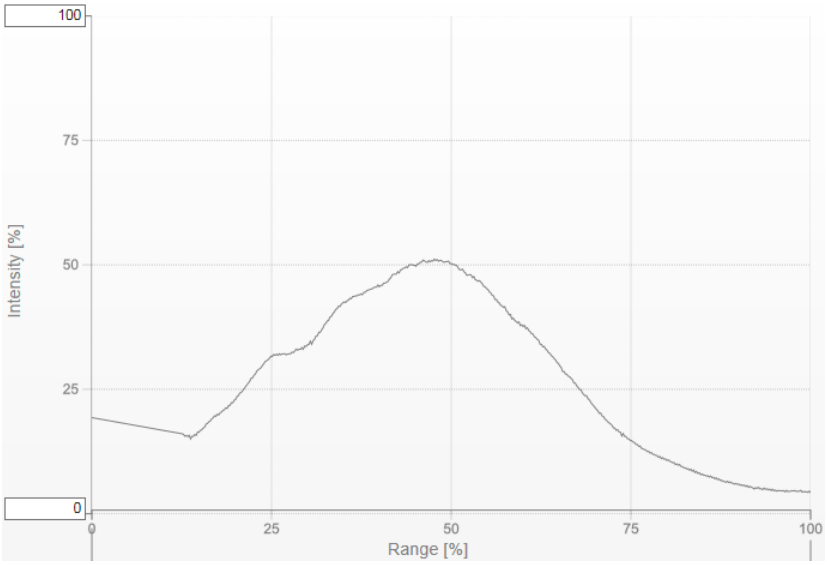



*Figure 3: Typical dark table, with linear part left of the splitpixel*

The always present basic signal level is subtracted from the acquired signal values by the controller even before the peak detection in sensor pixel coordinates is done. Therefore, the raw signal level is hidden from the user.

To perform a dark level compensation, there must not be a target inside or close to the measurement range. The point rate is set to 300 Hz by calling “boreSetScanrate” and the shutter time is set to 3000 $\mu$s by “boreSetShutterTime”. Then the function “boreCalibDark” has to be called. It receives the position of the splitpixel between reference peak and measurement peak (it separates the part of the spectrum used for temperature compensation from the part used for measurement distance determination and can be picked up from the calibration file, look for SplitPixel entry) and the number of averaging operations to be performed. Assumed enough time for the task is available, it’s recommended to do an averaging over an integer multiple of measurement points per rotation while the sensor pen is rotating.

```
boreSetScanrate(&bore, 300);
boreSetShutterTime(&bore, 3000);
boreCalibDark(&bore, 65, 300);
```

Hint: Older sensor pens supplied with the IFC2451 controller use 110 instead of 65 as splitpixel-position, and shutter time 10000 $\mu$s at 100 Hz point rate.

## 2.5 Acquisition and Evaluation of Measurement Data

To fill a data buffer with measurement data, the buffer needs to be allocated with the required size. If not done yet, the rotation needs to be started now by “boreRotationStart”. Then data acquisition is started, the buffer is filled and evaluated and data acquisition is stopped afterwards:

```
bore_data_t data;
bore_eval_t eval;
```

```
boreDataAlloc(&data, 10000);
boreDataStart(&bore);
boreDataGetRevolution(&bore, &data);
boreEvaluate(&bore, &data, &eval);
boreDataStop(&bore);
boreDataFree(&data);
```

Evaluation results like circle fit center and diameter, and some more, may be further processed using the filled data structure bore_eval_t.

To center the sensor pen inside of the measurement target, data acquisition and circle fit evaluation may be repeated arbitrarily while centering as required.

## 2.6 Angle Calibration of boreCONTROL

After power-on, the angle positions of the rotation controller and confocal controller are not necessarily corresponding. The correct position in the rotation controller is initialized by using “boreRotationHome”. The rotation unit will move to its zero-position slowly. After a single revolution the confocal controller also knows this zero-position. The angle ticks counted by the confocal controller will be reset to zero on each completed revolution and therefore are synchronized with the rotation unit.

The sensor pen and the rotation unit both have a mark, which have to be matched when inserting the sensor pen. The direction of the light output of different sensor pens will be the same relative to the rotation unit, but the angle zero position of the rotation unit is not recognizable from outside.

To center the sensor pen inside of the measurement object it may be helpful to orient the coordinate system in a defined way, so that evaluated center coordinates and movement for centering match.

To define the orientation of the coordinate system, an angle offset from the sensor coordinate system to the world coordinate system has to be determined once for each sensor pen. This may be accomplished by using a target with a scanable marker (e.g. a dark lasered line) defining the direction of the coordinate system.

After acquiring a complete revolution of data and evaluating the marker position, the tick counter “tick_counter_of_mark” for that position is known. The desired direction for the zero angle can be adjusted by calling

```
boreRotationAddTickOffset(&bore, - tick_counter_of_mark);
```

or, alternatively by

```
boreRotationSetAngleZero(&bore, angle_position);
```

This step has to be repeated after each restart of the controller, with the already known value for the offset.

The angle adjustment alternatively can be done manually by altering the mounting orientation. The rotation has to be stopped (“boreRotationStop”), the sensor pen rotated into 0° angle position (“boreRotationMoveTo”) and the rotation unit needs to be oriented as needed. For subsequent data acquisition, the rotation should be started again afterwards using “boreRotationStart”.

## 2.7 Offset-Calibration of boreCONTROL

By offset calibration small deviations of the position of the sensor pen from the rotation axis are corrected. This compensates small deviations occurring when inserting the sensor pen.

To perform an offset calibration, the sensor pen has to be placed into the precision gauge ring delivered with boreCONTROL or a similar precise target with known diameter. The ring has to be centered relative to the sensor pen and the shutter time must be adjusted accordingly.

The following steps have to be done for an offset calibration:

```
bore_data_t data;
bore_eval_t eval;

boreRefCalibLoad(&bore, " calibration_file.txt ");
boreRefCalibSetMode(&bore, BORE_REFCALIB_ONREQUEST );
boreRotationSetSpeed(&bore, 2.0);
```

```
boreRotationStart(&bore);
boreSetScanrate(&bore, 1000);
boreSetShutterTime(&bore, 200);
boreRefCalibSingle(&bore);

boreDataAlloc(&data, 10000);
boreDataStart(&bore);
do
{
   boreDataGetRawRevolution(&bore, &data);
   boreEvaluate(&bore, &data, &eval);
}
while(sqrt(eval.center_x * eval_center_x + eval.center_y * eval.center_y) > 100.0);
boreDataStop(&bore);

boreCalibrateOffset(&bore, 4000.0 – eval.radius);
boreDataFree(&data);
```

In detail, this means:

- Load the calibration table intended for use with the current sensor pen by calling the function “boreRefCalibLoad” and activate the use of the calibration table by calling “boreRefCalibSetMode” with parameter BORE_REFCALIB_ONREQUEST.
- Set the rotation speed (“boreRotationSetSpeed”) and start the rotation (“boreRotationStart”).
- Adjust the point rate to an appropriate value by “boreSetScanrate”, for ceramics gauge rings typically 1000 Hz, and adjust the shutter time using “boreSetShutterTime” so that the produced data point intensity lies between 10% and 90%. Setting the shutter time may be omitted if the mode BORE_REFCALIB_ONREQAUTOSH is used instead of BORE_REFCALIB_ONREQUEST.
- Initialize the temperature compensation using “boreRefCalibSingle”. Without this step the acquired measurement values cannot be converted to micron units in onrequest-modes.
- Allocate a data buffer with sufficient size using “boreDataAlloc”.
- Start the data acquisition (“boreDataStart”).
- Adjust the center position. Acquire a revolution of data using “boreDataGetRawRevolution” (identical to boreDataGetRevolution if Offset is 0.0) and evaluate data by “boreEvaluate”. Repeat the process until evaluation of the circle fit center produces absolute values of below 50 $\mu$m, preferably below 10 $\mu$m.
- Stop data acquisition by calling “boreDataStop”.
- The difference between ring radius and latest evaluated circle fit radius is set as new offset by calling “boreCalibrateOffset”.
- In case the data buffer isn’t used any longer, release it by calling “boreDataFree”.

By averaging evaluations from different positions inside of the calibration ring, the offset value can be determined even more accurate.

After this procedure the system is calibrated for absolute measurement. Offset values greater than 30 $\mu$m are unlikely to occur.

Alternatively, you can use multiple calibration rings for an adjustment of the calibration table. For that, acquire scan data with at least 50 revolutions per ring, either in equidistant planes or as spiral scan, and collect the data into a sufficiently large data buffer. Alternatively you can combine partial scans of each ring into one data buffer. For pre-assembled calibration stacks a file describing the rings is supplied. The file format is as follows:

```
# BCCSI 1 4
3996    3000
6000    7000
7999    10000
9998    10000
```

The first line defines the file format (BCCSI, boreCONTROL Calibration Stack Info), the file format version (1) and the number of contained rings (4). Subsequent lines hold the nominal diameter and ring height in $\mu$m.

Using this information, a modified calibration file can be created by the previously acquired data in bore_data_t as follows:

```
bore_recalibrate_eval_t refcal_eval;
bore_calstack_info_t calstack;
bore_refcalib_t refcal;

boreCalStackReadInfo(&calstack, stackinfo_name);
boreCalStackProcess(&data, &calstack);
boreRefCalibRead(&refcal, calfile_name);
boreRefCalibRecalibrateStack(&refcal, &caleval, calstack.count, &refcal_eval);
boreRefCalibWrite(&refcal, outfile_name);
boreRefCalibFree(&refcal);
```

The check of status values returned from function calls has been omitted for clearness.

## 2.8 Quality Control by Video Data Acquisition

For normal measurement operation, reference and measurement peak position are extracted as barycenters of a spectrum being mapped onto a video sensor line, and provided by the confocal controller. To detect contamination in the optical way the position information is not sufficient, but data from the sensor line is required. For that purpose an interface for retrieving the spectrum information is available. The procedure is as follows:

- To inspect the optical way, it is mandatory to place the sensor pen outside of a measurement object and protect it from significant environmental light.
- Just as for dark compensation, the shutter time is set high while the scan rate is low, typically 3000 $\mu$s at 300 Hz.
- After the normal measurement data acquisition is stopped, the transfer of video data is activated by calling “boreVideoStreamStart“. The additionally provided parameter is a bitwise combination of bit masks defined in bore_videotype_t and selects one or more channels. For checking the optical way it is sufficient to use BORE_VIDEOTYPE_RAW.
- After having started the video data transfer, the controller delivers video data sets with high speed, which have to be fetched continuously by the application to avoid buffer overflows. This is done by transferring data to an internal buffer by “boreVideoStreamGet“, followed by copying the needed channels to the application buffer using “boreVideoStreamCopy“. The required size for the application-side buffer can be retrieved by “boreVideoStreamGetTableSize“, which results in 512 for the currently used controllers IFC24x1.
- After having acquired the needed number of video data sets, the transfer is stopped by calling “boreVideoStreamStop“.
- For the simple case of quality control with a single channel holding the raw information, the steps above with starting acquisition, fetching data, stopping the acquisition and copying the data are combined in a single function “boreVideoStreamSingle“ for convenience:

```
double table[512];

boreSetScanrate(&bore, 300.0);
boreSetShutterTime(&bore, 3000.0);
boreVideoStreamSingle(&bore, BORE_VIDEOTYPE_RAW, table, 512);
```

When using long shutter times, the reference peak will be too bright and therefore this part of the spectrum can be ignored. The maximum intensity in the area of the measurement peak without presence of a measurement target consists out of accumulated optical effects and is expected to be lower than 90%, better 70%. Otherwise some cleaning is required. However, high values also might indicate a damaged component.

To identify the component which needs cleaning, the test above is repeated with successively removed components (sensor pen, rotation unit, optical fiber plug at the controller), keeping in mind

that the accumulated effect reduces with each component. If the maximum intensity massively drops when unplugging a component, the polluted coupling is identified and can be cleaned.

## 2.9 Data Filtering

### 2.9.1 Sector Filtering

If the sensor pen is de-centered in the borehole, the ray in places hits the surface in a suboptimal angle, and the measurement value might be inexact, whereas close to the positions with smallest and largest distance the angle is nearly perpendicular.

To optimize the computation of the fit circle it is possible to exclude measurement points which lie offside the positions with minimum and maximum distance. To achieve this, the bit BORE_FLAG_VALID in element “flag“ of the data structure bore_meas_t is unset for points which shouldn’t be used.

This is done by the function

boreFilterInvalidateTwoAngleSectors(bore_data_t *data, double selang, double selrng)

which invalidates points with angle distance larger than „selrng“ from given angle „selang“ and it’s opposing counterpart (all in degree).

To determine the angle with smallest distance, the function

boreFilterGetMinDistAngle(double offset_x, double offset_y)

which converts the circle fit center coordinates into the according angle position, may be used.

Dependent on de-centering of the sensor pen it makes sense to vary the sector size for selected points used for circle fit. For small de-centering it’s preferred to use all points, whereas with increasing de-centering the sector sizes should get smaller, but never go below a certain minimum size to have enough points available for circle fit. The computation of the half sector size to be used in „boreFilterInvalidateTwoAngleSectors“ is done by the function

boreFilterGetAngleRangeFromRamp(
double offset_x, double offset_y, double rampstart, double rampend, double endangle)

which gets the circle fit center (offset_x, offset_y) and the parameters of a ramp function as arguments. For de-centering less than rampstart, 90° is returned, values above rampend return the last argument endangle. Between rampstart and rampend the return value is computed by affine-linear interpolation between these two points.

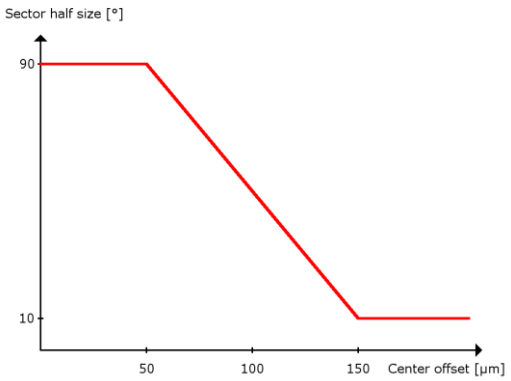



*Figure 4: Example of a ramp function*

Reasonable values are rampstart=50 [$\mu$m] and rampend=150 [$\mu$m] with endangle=10 [°]. Particularly the choice for the angle range depends on used point rate and rotation speed, so the values should be optimized experimentally.

### 2.9.2 Median Filtering

To eliminate individual distance outliers, e.g. when scanning rough surfaces, a median filter function

```
boreFilterMedian(
  const bore_data_t *data, int halfsize, bore_filter_flags_t flags, bore_data_t *outdata)
```

respectively for parts of data buffers (startidx to endidx, excluded, destination index outidx)

```
boreFilterMedianRange(
  const bore_data_t *data, int startidx, int endidx,  int halfsize, bore_filter_flags_t flags,
  bore_data_t *outdata, int outidx);
```

is provided, and takes input data, half of the filter kernel size halfsize and filter parameters as input. It fills an output buffer, which has been initialized with suitable size before. The filter options flags include:

- BORE_FILTER_NO_WRAPAROUND:
  Filtering is not continued at the other end of the buffer. This is reasonable for spiral scans, as the data at start and end of the data buffer originates from different depth inside of the target.
- BORE_FILTER_OMIT_INVALID:
  Previously invalid measurement points are not replaced by filtered values, but stay invalid in the output buffer. Gaps in data are preserved.
- BORE_FILTER_BARY:
  Instead of the distance the entry for the measurement barycenter is filtered.

### 2.9.3 Average Filtering

To smooth data values, an average filter analogous to median filter is provided by boreFilterAverage and boreFilterAverageRange, accordingly.

# 3. Short Description of Functions

The functions provided by boreCONTROL.dll are arranged in several functional groups.

| | |
|---|---|
| Connection management | Connect and disconnect boreCONTROL |
| Rotation unit | Functions to control the rotation unit |
| Z-axis control | Functions for z-axis interpretation |
| Scan parameters | Functions to set and get scan parameters |
| Video acquisition | Functions for video signal acquisition |
| Calibration | Functions for calibration control |
| Offset-calibration | Functions for calibration of sensor offset |
| Data acquisition | Functions for data acquisition |
| Data IO | Functions to read and write data |
| Data types support | Functions to support special datatypes |
| Filters | Functions for filtering data |
| Evaluation | Functions for data evaluation |
| Data exchange | Functions for data exchange |
| Reference calibration | Functions for optical temperature compensation |
| Reference calibration processing | Functions to process reference calibrations |
| Calibration stack handling | Functions to evaluate calibration stacks |

A detailed description of the function calls can be found in the enclosed HTML-SDK-documentation.

## 3.1 Functions for Connection Management

| | |
|---|---|
| boreConnect | Connects to boreCONTROL system components |
| boreDisconnect | Disconnects previously connected boreCONTROL components |
| boreGetControllerVersion | Get version information from sensor controller |
| boreGetDllVersion | Get version information of dll |
| boreGetErrorMessage | Retrieve description text for error codes |
| boreGetControllerTemperature | Retrieve internal controller temperature |

## 3.2 Functions for Rotation Unit Control

| | |
|---|---|
| boreRotationSetSpeed | Set rotation unit speed |
| boreRotationGetSpeed | Get rotation unit speed |
| boreRotationStart | Start rotation unit |
| boreRotationStop | Stop rotation unit |
| boreRotationGetTicks | Get ticks for complete rotation |
| boreRotationSetTickOffset | Set tick offset for angle ticks |
| boreRotationGetTickOffset | Get tick offset for angle ticks |
| boreRotationAddTickOffset | Add tick offset for angle ticks |
| boreRotationGetAngle | Get angle position of rotation unit relative to zero position |

| | |
|---|---|
| boreRotationGetAngleRaw | Get hardware angle position of rotation unit |
| boreRotationSetAngleZero | Set zero position of rotation unit |
| boreRotationGetAngleZero | Get zero position of rotation unit |
| boreRotationAddAngleZero | Modify zero position relative to current value |
| boreRotationHome | Rotate to zero-position |
| boreRotationMoveTo | Rotate to angle tick position |
| boreRotationMoveAbsAngle | Rotate to absolute position |
| boreRotationMoveRelAngle | Rotate relative to current position |

## 3.3 Functions for Z-Axis Interpretation

| | |
|---|---|
| boreZAxisSetScale | Set z-axis scaling factor ticks to $\mu$m |
| boreZAxisGetScale | Get z-axis scaling factor ticks to $\mu$m |
| boreZAxisSetOffset | Set z-axis offset value in $\mu$m |
| boreZAxisGetOffset | Get z-axis offset value in $\mu$m |
| boreZAxisZero | Set z-axis ticks to zero |
| boreZAxisSetPosition | Set z-axis position in $\mu$m |

## 3.4 Functions to manage Scan Parameters

| | |
|---|---|
| boreSetIntensity | Set controller LED intensity (IFC2401 only) |
| boreGetIntensity | Get controller LED intensity (IFC2401 only) |
| boreSetShutterTime | Set shutter time |
| boreGetShutterTime | Get shutter time |
| boreSetShutterTimes | Set shutter times in dual shutter mode |
| boreGetShutterTimes | Get shutter times in dual shutter mode |
| boreSetScanrate | Set controller scan rate |
| boreGetScanrate | Get controller scan rate |

## 3.5 Functions for Video Data Acquisition

| | |
|---|---|
| boreVideoStreamStart | Start of video data acquisition |
| boreVideoStreamStop | Stop of video data acquisition |
| boreVideoStreamGet | Retrieve video data from controller |
| boreVideoStreamCopy | Copy retrieved video data |
| boreVideoStreamSingle | Retrieve and copy selected video data |
| boreVideoStreamGetTableSize | Get size of video data tables |

## 3.6 Functions for Calibration

| | |
|---|---|
| boreCalibSetTable | Set calibration table |

| | |
|---|---|
| boreCalibGetTable | Get calibration table |
| boreCalibGetRange | Get measurement range of currently used sensor table |
| boreCalibDarkSignal | Do dark signal calibration (IFC2401 only) |
| boreCalibDark | Do dark compensation for optically compensated sensor pens |
| boreCalibAngleReference | Adjustment of angle reference (IFC2401 only) |
| boreCalibGetRanges | Get measurement ranges of all calibration tables |

## 3.7 Functions for Offset-Calibration

| | |
|---|---|
| boreSetOffset | Set new offset value for offset calibration |
| boreGetOffset | Get currently set offset value for offset calibration |
| boreDataAddOffset | Do offset calibration on data buffer |
| boreDataApplyFactorOffsetRange | Apply factor and offset on part of data buffer |
| boreDataApplyFactorOffset | Apply factor and offset on data buffer |
| boreSetFactor | Set new slope correction factor for offset calibration |
| boreGetFactor | Get currently set slope correction factor for offset calibration |
| boreCalibrateOffset | Fill offset calibration structure for supplied diameter in $\mu$m |
| boreCalibrateOffsetMulti | Offset calibration for multiple supplied calibration ring data |

## 3.8 Functions for Data Acquisition

| | |
|---|---|
| boreDataAlloc | Allocate a measurement data buffer |
| boreDataFree | Free previously allocated buffer |
| boreDataReset | Reset data buffer to empty state |
| boreDataGetRevCount | Get number of revolutions contained in data buffer |
| boreDataGetRevIndices | Get start indices of revolutions in data buffer |
| boreDataRevalidate | Recompute valid mark in data buffer |
| boreDataStart | Start acquisition and data transfer |
| boreDataStop | Stop acquisition and data transfer |
| boreDataRestart | Empty internal data buffers while acquisition is active |
| boreDataAvailable | Get number of currently available data points |
| boreDataGetRawRevolutionPartial | Collect available data without offset added, non-blocking |
| boreDataGetRevolutionPartial | Collect available data and add offset, non-blocking |
| boreDataGetRawRevolution | Get single circle of data without offset added |
| boreDataGetRevolution | Get single circle of data and add offset value |
| boreDataGetRaw | Read measurement data without adding offset value |
| boreDataGet | Read measurement data and add offset value |
| boreDataAppendRange | Append part of a data buffer to another one |

| | |
|---|---|
| boreDataAppend | Append data buffer to another data buffer |

## 3.9 Functions for Data IO

| | |
|---|---|
| boreDataIOGetSize | Get number of measurement values from file |
| boreDataIOReadAppend | Append file measurement values to data buffer |
| boreDataIORead | Read data buffer from file |
| boreDataIOReadAlloc | Read data buffer from file, buffer allocated accordingly |
| boreDataIOWrite | Write data buffer to file |
| boreDataIOGetFilesetSize | Get file and measurement count for files matching pattern |
| boreDataIOReadFilesetAppend | Append files matching pattern to data buffer |
| boreDataIOReadFileset | Read files matching pattern to data buffer |
| boreDataIOReadFilesetAlloc | Read files matching pattern to data buffer, allocate accordingly |

## 3.10 Functions to Support Special Data Types

| | |
|---|---|
| boreMatrixAlloc | Allocate matrix |
| boreMatrixFree | Free matrix |
| boreMatrixSet | Set matrix to constant value |

## 3.11 Functions for Data Filtering

| | |
|---|---|
| boreFilterInvalidateAngleSectors | Invalidate data outside of provided sectors |
| boreFilterInvalidateTwoAngleSectors | Invalidate data outside of opposing sectors |
| boreFilterGetAngleRangeFromRamp | Computation of sector size using ramp function |
| boreFilterGetMinDistAngle | Computation of angle with minimum distance |
| boreFilterMedian | Median filter |
| boreFilterMedianRange | Mendian filter on part of a data buffer |
| boreFilterAverage | Average filter |
| boreFilterAverageRange | Average filter on part of a data buffer |
| boreFilterResample | Resample data buffer on equidistant grid |
| boreFilterResampleRange | Resample part of data buffer on equidistant grid |

## 3.12 Functions for Evaluation

| | |
|---|---|
| boreEvalSetIntensityThres | Set evaluation intensity lower threshold in percent |
| boreEvalGetIntensityThres | Get evaluation intensity lower threshold in percent |
| boreEvalSetIntensityUpperThres | Set evaluation intensity upper threshold in percent |
| boreEvalGetIntensityUpperThres | Get evaluation intensity upper threshold in percent |
| boreEvaluate | Evaluates measurement data using circle fit |
| boreEvaluateCal | Evaluates raw measurement data for calibration purposes only |

| | |
|---|---|
| boreEvaluateData | Evaluates measurement data, only data buffer required |
| boreEvaluateCalData | Evaluates raw measurement data, only data buffer required |
| boreEvaluateDataRange | Evaluates measurement data using index range |
| boreEvaluateCalDataRange | Evaluates raw measurement values using index range |
| boreEvaluateDataRevs | Evaluates multiple revolutions using index vector |
| boreEvaluateCalDataRevs | Evaluates multiple raw revolutions using index vector |

## 3.13 Functions for Data Exchange

| | |
|---|---|
| boreMeasGetFlag | Get flag element of measurement data into array |
| boreMeasGetDistance | Get distance element of measurement data into array |
| boreMeasGetIntensity | Get intensity element of measurement data into array |
| boreMeasGetCounter | Get data counter element of measurement data into array |
| boreMeasGetBarycenter | Get barycenter element of measurement data into array |
| boreMeasGetAngleRad | Get angle element of measurement data into array |
| boreMeasGetXPos | Get x-position element of measurement data into array |
| boreMeasGetYPos | Get y-position element of measurement data into array |
| boreMeasGetZPos | Get z-position element of measurement data into array |
| boreMeasGetTempCelsius | Get temperature element of measurement data into array |
| boreMeasGetAngleTicks | Get angle tick element of measurement data into array |
| boreMeasGetRevTicks | Get revolution count element of measurement data into array |
| boreMeasGetZTicks | Get z-axis tick count element of measurement data into array |
| boreMeasGetRefIntensity | Get reference peak intensity element of measurement data into array |
| boreMeasGetRefBarycenter | Get reference peak barycenter element of measurement data into array |
| boreMeasGetShutter | Get shutter element of measurement data into array |

## 3.14 Functions for Reference Calibration

| | |
|---|---|
| boreRefCalibLoad | Load a calibration table for optical temperature compensation |
| boreRefCalibSingleData | Reference peak determination from data buffer |
| boreRefCalibSingle | Do a single measurement for determining reference peak position |
| boreRefCalibSetMode | Set temperature compensation mode |
| boreRefCalibGetMode | Get temperature compensation mode |
| boreRefCalibSetSingleMode | Set reference calibration mode for single reference peak acquisition |
| boreRefCalibGetSingleMode | Get reference calibration mode for single reference peak acquisition |
| boreRefCalibSetShutterTime | Set controller shutter time for reference peak acquisition |
| boreRefCalibGetShutterTime | Get controller shutter time for reference peak acquisition |
| boreRefCalibSetScanrate | Set controller scan rate for reference peak acquisition |

| | |
|---|---|
| boreRefCalibGetScanrate | Get controller scan rate for reference peak acquisition |

## 3.15 Functions for Processing Reference Calibrations

| | |
|---|---|
| boreRefCalibFree | Release an initialized reference calibration structure |
| boreRefCalibRead | Read a reference calibration from file |
| boreRefCalibWrite | Write a reference calibration to file |
| boreRefCalibSet | Use an initialized reference calibration |
| boreRefCalibRecalibrate | Recalibration of a reference calibration according to new data |
| boreRefCalibRecalibrateStack | Recalibration of a reference calibration using cal stack evaluation |
| boreRefCalibGetComment | Get comment of reference calibration, user description |
| boreRefCalibSetComment | Set comment of reference calibration, user description |
| boreRefCalibGetPeakRanges | Retrieve reference peak and measurement peak range |
| boreRefCalibEvaluate | Conversion of peak positions to $\mu$m |

## 3.16 Functions for Processing Calibration Stacks

| | |
|---|---|
| boreCalStackReadInfo | Read calibration stack information from file |
| boreCalStackWriteInfo | Write calibration stack info to file |
| boreCalStackProcessRevs | Process calibration stack data |
| boreCalStackProcess | Completely automated processing of calibration stack data |

# 4. Description of Data Structures

## 4.1 Connection – bore_connect_t and Enumeration Types

The data type bore_connect_t is used only when connecting to boreCONTROL components using the function “boreConnect”.

```
/// boreCONTROL hardware controller
typedef enum {
  BORE_CONTROLLER_NONE    = 0,  ///< no controller selected
  BORE_CONTROLLER_IFC2401 = 1,  ///< boreCONTROL IFC2401 controller
  BORE_CONTROLLER_IFC2431 = 2,  ///< boreCONTROL IFC2431 controller
  BORE_CONTROLLER_IFC2451 = 3,  ///< boreCONTROL IFC2451 controller
  BORE_CONTROLLER_IFC2471 = 4,  ///< boreCONTROL IFC2471 controller
  BORE_CONTROLLER_IFC2461 = 5   ///< boreCONTROL IFC2461 controller
} bore_controller_t;

/// boreCONTROL hardware rotation unit
typedef enum {
  BORE_ROTATION_NONE    = 0,    ///< no controller selected
  BORE_ROTATION_BCC2410 = 1     ///< boreCONTROL BCC2410 controller
} bore_rotation_t;

/// boreCONTROL hardware temperature
typedef enum {
  BORE_TEMPERATURE_NONE = 0,    ///< no temperature sensor selected
  BORE_TEMPERATURE_NN   = 1     ///< boreCONTROL temperature sensor
} bore_temperature_t;

/// parameters for boreCONTROL connection
typedef struct bore_connect_ {
  bore_controller_t controller; ///< boreCONTROL controller selection
  int controller_table;         ///< boreCONTROL sensor table
  int usb_instance_number;      ///< usb based controllers, 0 is first instance
  char tcpip_address[32];       ///< eth based controllers, server address
  int tcpip_port;               ///< eth based controllers, command port

  bore_rotation_t rotation_unit;///< rotation unit hardware
  int rotation_com_port;        ///< serial port of rotation unit, COM1 == 1

  bore_temperature_t temp_unit; ///< boreCONTROL temperature sensor
} bore_connect_t;
```

The data structure needs to be filled as follows:

- **controller:** Type of controller to use, normally BORE_CONTROLLER_IFC2461, but also BORE_CONTROLLER_IFC2451 or BORE_CONTROLLER_IFC2471. BORE_CONTROLLER_IFC2401 and BORE_CONTROLLER_IFC2431 are not used any longer.
- **controller_table:** Always 0, as the calibration table for IFC2451, IFC2461 and IFC2471 is provided by software.
- **usb_instance_number:** 0, not used for IFC2451, IFC2461 and IFC2471
- **tcpip_address:** The IP-address of IFC2451, IFC2461 or IFC2471, normally „169.254.168.150“, as long as the address isn’t modified by customer
- **tcpip_port:** 0 (or 23), is set to 23 for values <= 0 automatically
- **rotation_unit:** BORE_ROTATION_BCC2410
- **rotation_com_port:** Serial port for BCC2410, 1 corresponds to COM 1 etc.
- **temp_unit:** Not available, therefore BORE_TEMPERATURE_NONE

In case only default parameters should be used, a NULL-pointer can be passed to the function boreConnect instead of the bore_connect_t parameter.

## 4.2 Acquisition Modes – bore_refcalib_mode_t

By supplying bore_refcalib_mode_t to “boreRefCalibSetMode”, the acquisition mode is selected:

```
/// when to do reference calibration
typedef enum {
  BORE_REFCALIB_OFF          = 0,  ///< reference calibration not used
  BORE_REFCALIB_ONREQUEST    = 1,  ///< reference calibration on request
  BORE_REFCALIB_ALWAYS       = 2,  ///< reference calibration done always
  BORE_REFCALIB_ONREQ2AUTO   = 3,  ///< on request, autoselect from 2 shutters
  BORE_REFCALIB_ONREQAUTOSH  = 4,  ///< on request, auto shutter
  BORE_REFCALIB_REFAUTOSH    = 5   ///< reference peak only, auto shutter
} bore_refcalib_mode_t;
```

## 4.3 Sensor Data - bore_data_t and bore_meas_t

bore_data_t provides a vector of measurement points. The vector is allocated und initialized by “boreDataAlloc” and freed by “boreDataFree”, which releases the previously allocated memory resources.

To reset the data structure, use “boreDataReset”. This will set the number of data points and the offset to 0, but doesn’t release the allocated memory.

```
/// buffer with measurement points
typedef struct bore_data_ {
  bore_meas_t *data;            ///< data array
  int maxsize;                  ///< allocated size of data array
  int currsize;                 ///< used size of data array
  double offset;                ///< offset added to data
} bore_data_t;
```

The data structure consists out of the following entries:

- **data:** Pointer to measurement point data, see bore_meas_t
- **maxsize:** The number of allocated measurement points in “data”
- **currsize:** The number of currently used data points
- **offset:** The offset from the offset calibration in micron units, which was added to the distances of the measurement points. For data acquisition, two variants are provided. For acquisition using the “Raw” variant (boreDataGetRawRevolution/boreDataGetRaw) the offset is set to 0. To add the offset from the offset calibration use “boreDataAddOffset” function. For variants not containing the “Raw” name part (boreDataGetRevolution/boreDataGet) the offset from the offset calibration is already added to distance data.

Data points are realized by bore_meas_t:

```
/// single measurement point
typedef struct bore_meas_ {
  bore_flag_t flag;             ///< valid flag, first of circle, last of circle
  double distance;              ///< measurement peak distance in µm
  double intensity;             ///< measurement peak intensity, range [0%-100%]
  int counter;                  ///< measurement point counter
  double barycenter;            ///< measurement raw distance in sensor units
  double angle_rad;             ///< angle converted to radian 0..2PI from enc1
  double x_pos;                 ///< angle and distance converted to x position
  double y_pos;                 ///< angle and distance converted to y position
  double z_pos;                 ///< z_ticks converted to µm
  double temp_celsius;          ///< if temperature sensor connected, else 20°C
  int angle_ticks;              ///< angle tick counter
  int rev_ticks;                ///< revolution tick counter
  int z_ticks;                  ///< z-axis tick counter
  double ref_intensity;         ///< reference peak intensity
  double ref_barycenter;        ///< reference peak distance in sensor units
  double shutter;               ///< shutter time in microseconds
} bore_meas_t;
```

The elements hold the following information:

- **flag:** Markers for measurement points, defined in enumeration data type bore_flag_t. Valid measurement points (BORE_FLAG_VALID) are those with intensity value above the provided threshold. When acquiring complete revolutions, the first (BORE_FLAG_FIRST) and last (BORE_FLAG_LAST) measurement point are marked also.
- **distance:** The calibrated distance to the target in µm. The conversion from sensor units is provided by selection the according calibration table in the controller.
- **intensity:** The brightness information of the measurement point in percent.
- **counter:** A continuously incremented counter for acquired measurement points.
- **barycenter:** Measurement peak coordinate of the sensor in pixel units for calibration purposes.
- **angle_rad:** Angle in radian, computed from angle tick counter
- **x_pos:** x-coordinate in µm, computed from distance und angle_rad
- **y_pos:** y-coordinate in µm, computed from distance und angle_rad
- **z_pos:** z-coordinate in µm, computed from z_ticks
- **temp_celsius:** Temperature in °C, may be filled with internal controller temperature if needed.
- **angle_ticks:** Angle tick counter from rotation unit encoder
- **rev_ticks:** Revolution count from the rotation unit encoder, connected only for IFC2401, else free for different use.
- **z_ticks:** z tick counter from optional z axis encoder
- **ref_intensity:** The brightness information of the reference peak, in percent
- **ref_barycenter:** Reference peak coordinate of the sensor in pixel units, for optical calibration.
- **shutter:** Used shutter time in µs

## 4.4 Evaluation Information – bore_eval_t

bore_eval_t provides selected information after a computed circle fit and is filled by either “boreEvaluate”, “boreEvaluateCal”, “boreEvaluateData”, ”boreEvaluateCalData” or as part of bore_offset_cal_t by “boreCalibrateOffset”.

```
/// evaluation of data
typedef struct bore_eval_ {
  int total_points;              ///< total number of points supplied
  int valid_points;              ///< number of valid data points

  double radius;                 ///< fit circle radius in µm
  double diameter;               ///< fit circle diameter in µm

  double center_x;               ///< signed center x deviation
  double center_y;               ///< signed center y deviation

  double avg_intensity;          ///< average intensity in %
  double min_intensity;          ///< minimum intensity in %
  double max_intensity;          ///< maximum intensity in %

  double deviation_outside;      ///< max deviation away from center, positive
  double deviation_inside;       ///< max deviation towards center, positive

  // extended information, mainly for calibration purposes

  bore_eval_flag_t flags;        ///< evaluation warnings
  double ring_diameter;          ///< filled before recalib: ring diameter in µm

  double avg_refbary;            ///< reference peak bary center, if available
  double avg_refintensity;       ///< reference peak intensity, if available
  double min_refintensity;       ///< minimum reference peakintensity in %
  double max_refintensity;       ///< maximum reference peak intensity in %

  double avg_measbary;           ///< measurement peak bary center, pixel
  double measbary_center_x;      ///< measurement peak center x position, pixel
```

```
  double measbary_center_y;     ///< measurement peak center y position, pixel

  double measdev_outside;       ///< meas peak max deviation away from center
  double measdev_inside;        ///< meas peak max deviation towards center

  double avg_shutter;           ///< average shutter time
  double min_shutter;           ///< minimum shutter time
  double max_shutter;           ///< maximum shutter time
} bore_eval_t;
```

The data structure elements contain the following information:

- **total_points:** Number of data points in the evaluated data buffer
- **valid_points:** Number of valid data points for one revolution, which have an intensity value inside of the supplied intensity range. (see “boreEvalSetIntensityThres” and “boreEvalSetIntensityUpperThres”) and therefore are used for evaluation and circle fit (“boreEvaluate”).
- **radius:** Radius of the fitted circle in $\mu$m
- **diameter:** Diameter of the fitted circle in $\mu$m
- **center_x:** x-coordinate of the center position in $\mu$m determined by a circle fit. The orientation of the coordinate system is configured by defining the position of the 0° angle, which provides the direction of the positive x-axis (see “boreRotationSetTickOffset”).
- **center_y:** y-coordinate of the center position in $\mu$m determined by a circle fit. The positive y-axis is the direction of the x-axis rotated mathematically positive by 90° as seen from above.
- **avg_intensity:** Average intensity of all valid data points in %
- **min_intensity:** Minimum intensity of all data points in %. This minimum intensity cannot be smaller than the configured intensity threshold (see “boreEvalSetIntensityThres”).
- **max_intensity:** Maximum intensity of all valid data points in %. This maximum intensity cannot be larger than the configured upper intensity threshold (see “boreEvalSetIntensityUpperThres”).
- **deviation_outside:** Maximum deviation of a data point outside of the fitted circle relative to the fitted circle with positive sign in $\mu$m
- **deviation_inside:** Maximum deviation of a data point inside of the fitted circle relative to the fitted circle with positive sign, in $\mu$m

The measurement values “min_intensity” and “max_intensity” also provide the saturation value. It is affected not only by the material and the reflection properties of the target, but also influenced by the point rate and shutter time parameters of the scan process. The saturation of a measurement should be in range 10 % to 90 %.

The rest of the data structure contains additional information mainly used for recalibration purposes:

- **flags:** Additional markers for error conditions while evaluation, see below
- **ring_diameter:** Diameter of the object to measure in $\mu$m specified on calling the evaluation by “boreEvaluateCal” or “boreEvaluateCalData”, typically a gauge ring.
- **avg_refbary:** Average position of the reference peak in pixel
- **avg_refintensity:** Average intensity of reference peak in %
- **min_refintensity:** Minimum intensity of reference peaks in %
- **max_refintensity:** Maximum intensity of reference peaks in %
- **avg_measbary:** Average measurement peak using circle fit, in pixel
- **measbary_center_x:** x-coordinate of the circle center for the measurement peak in pixel
- **measbary_center_y:** y-coordinate of the circle center for the measurement peak in pixel
- **measdev_outside:** Maximum deviation of a measurement peak outside of the fitted circle to the fitted circle with positive sign, in pixel

- **measdev_inside:** Maximum deviation of a measurement peak inside of the fitted circle towards the circle center with positive sign, in pixel
- **avg_shutter:** Average value for shutter time in µs
- **min_shutter:** Minimum value for shutter time in µs
- **max_shutter:** Maximum value for shutter time in µs

In bore_eval_flag_t markers for suspicious data on evaluation are collected. They are intended to be interpreted as warnings. The thresholds used are suitable for gauge rings only, not for real bores.

```
/// extended evaluation bits, primarily for use with gauge rings
typedef enum {
  /// general evaluation flags, set by evaluation
  BORE_EVAL_NONE                 = 0x0000,    ///< unmarked
  BORE_EVAL_LOW_MIN_INTENSITY    = 0x0001,    ///< minimum intensity too low
  BORE_EVAL_HIGH_MAX_INTENSITY   = 0x0002,    ///< maximum intensity too high
  BORE_EVAL_LOW_REF_INTENSITY    = 0x0004,    ///< min ref intensity too low
  BORE_EVAL_HIGH_REF_INTENSITY   = 0x0008,    ///< max ref intensity too high
  BORE_EVAL_LOW_INTENSITY        = 0x0010,    ///< overall intensity too low
  BORE_EVAL_HIGH_INTENSITY       = 0x0020,    ///< overall intensity too high
  BORE_EVAL_WARN_OFF_CENTER      = 0x0040,    ///< center deviation too large
  BORE_EVAL_WARN_HIGH_DEVIATION = 0x0080,    ///< suspicious deviation to fit
  BORE_EVAL_WARN_REF_POSITION    = 0x0100,    ///< suspicious ref peak position
  BORE_EVAL_WARN_MEAS_POSITION   = 0x0200,    ///< suspicious meas peak pos
  BORE_EVAL_WARN_FEW_VALID       = 0x0400,    ///< low percentage of valid points
  BORE_EVAL_WARN_REF_INTENSITY   = 0x0800,    ///< high ref peak intensity deviation

  /// set by recalibration
  BORE_EVAL_HIGH_TABLE_OFFSET    = 0x010000,  ///< high offset to original table
  BORE_EVAL_RING_IGNORED_AVG     = 0x020000,  ///< ring ignored for averaging

  /// user defined content
  BORE_EVAL_USER_MASK            = 0xFF000000 ///< bitmask for user defined bits
} bore_eval_flag_t;
```

## 4.5 Offset Calibration Information – bore_offsetcal_t

bore_offsetcal_t consists out of the results of an offset calibration and the evaluation information from the circle fit done for offset calibration. The data structure is filled by the function “boreCalibrateOffset”.

```
/// offset calibration result info
typedef struct bore_offsetcal_ {
  int successful;                ///< bool: overall offset calibration status
  double offset;                 ///< generated calibration offset
  double factor;                 ///< generated calibration slope factor
  double expected_diameter;      ///< expected diameter in µm, calibration target

  int intensity_ok;              ///< bool: whether intensity was ok
  int center_ok;                 ///< bool: whether centering was ok
  int deviation_ok;              ///< bool: whether deviation from fit circle was ok

  bore_eval_t eval;              ///< evaluation results for detailed analysis
} bore_offsetcal_t;
```

The entries of the data structure have got the following meaning:

- **successful:** Is set on successful completion of an offset calibration. For this, all of intensity_ok, center_ok and deviation_ok must have been set to true.
- **offset:** The computed offset value from the offset calibration
- **factor:** The computed correction factor for slope correction
- **expected_diameter:** The calibration ring diameter of the target used for offset calibration
- **intensity_ok:** Whether the intensity was inside of a reasonable range. For a closer analysis have a look to min_intensity and max_intensity, section 4.4.
- **center_ok:** Whether the center of the fitted circle was inside of a preconfigured interval
- **deviation_ok:** Whether the point deviations from fitted circle are inside of a reasonable range

- **eval:** Detailed information concerning circle fit, see section 4.4.

## 4.6 Calibration Stack Information – bore_calstack_info_t and bore_calstack_eval_t

The evaluation information for calibration stacks is provided by the following set of data structures:

```
/** calibration stack evaluation for single calibration ring */
typedef struct bore_calstack_eval_ {
  /* segmentation result */
  int ring_processed;   /**< whether processed successfully */
  int start_rev;        /**< start revolution index, included */
  int end_rev;          /**< end revoultion index, included */
  int inner_start_rev;  /**< start revolution index inner_height, included */
  int inner_end_rev;    /**< end revolution index inner_height, included */
  double height;        /**< detected height in um */
  double inner_height;  /**< used inner height of ring in um (def 5mm) */

  /* averaging result */
  double avg_diam;       /**< average diameter (inner part only)*/
  double avg_refbary;    /**< average refbary position */
  double avg_measbary;   /**< average measbary position */
  double avg_center_x;   /**< average center x position */
  double avg_center_y;   /**< average center y position */
  double start_center_x; /**< center x position at start of ring */
  double start_center_y; /**< center y position at start of ring */
  double end_center_x;   /**< center x position at end of ring*/
  double end_center_y;   /**< center y position at end of ring*/
} bore_calstack_eval_t;


/** maximum number of rings in calibration stack */
#define BORE_CALSTACK_INFO_MAX 25

/** calibration ring stack information */
typedef struct bore_calstack_info_ {
  /* required input information */
  int count;                                  /**< number of rings present */
  double diameter[BORE_CALSTACK_INFO_MAX];    /**< ring diameter in micron units */
  double height[BORE_CALSTACK_INFO_MAX];      /**< ring height in micron units */

  /* computation result */
  bore_calstack_eval_t eval[BORE_CALSTACK_INFO_MAX];  /**< evaluation result */
} bore_calstack_info_t;
```

The entries of the bore_calstack_info_t data structure are normally filled by loading stack information files using “boreCalStackReadInfo”, but may be filled manually also.

- **count:** Number of contained calibration rings, maximum BORE_CALSTACK_INFO_MAX
- **diameter:** Array with ring diameters of each calibration ring
- **height:** Array with heights of each calibration ring
- **eval:** Array with evaluation information from a calibration stack analysis

For each ring evaluation results are available by bore_calstack_eval_t:

- **ring_processed:** Whether ring was processed successfully
- **start_rev:** Ring start revolution index, included
- **end_rev:** Ring end revolution index, included
- **inner_start_rev:** Start revolution index of evaluated inner part of ring (the outer parts of the ring shows larger diameter deviations and are therefore excluded from evaluation)
- **inner_end_rev:** Last revolution index of evaluated inner part of ring (the outer parts of the ring shows larger diameter deviations and are therefore excluded from evaluation)
- **height:** Detected ring height in um
- **inner_height:** Height of inner part of the ring used for evaluation

- **avg_diam:** Average diameter in inner part of ring
- **avg_refbary:** Average refbary position
- **avg_measbary:** Average measbary position
- **avg_center_x:** Average center x position
- **avg_center_y:** Average center y position
- **start_center_x:** Center x position at start of ring
- **start_center_y:** Center y position at start of ring
- **end_center_x:** Center x position at end of ring
- **end_center_y:** Center y position at end of ring

Using the evaluation information, a re-calibration of an existing calibration table may be computed with “boreRefCalibRecalibrateStack”.

# 5. Description of Error Codes

## 5.1 General Errors

```
// general error codes
BORE_OK                              =    0, ///< successful, no error
BORE_ERROR                           =   -1, ///< unspecified general error
BORE_ERROR_OUT_OF_MEMORY             =   -2, ///< allocation out of memory
BORE_ERROR_UNSUPPORTED               =   -3, ///< requested feature not supported
BORE_ERROR_UNINITIALIZED             =   -4, ///< interface not initialized
BORE_ERROR_RANGE                     =   -5, ///< illegal range
BORE_ERROR_INCONSISTENT              =   -6, ///< internal inconsitency
BORE_ERROR_BUFFER_OVERFLOW           =   -7, ///< buffer overflow
BORE_ERROR_ACTIVE                    =   -8, ///< not allowed while data is sent
BORE_ERROR_BUFFER_TOO_SHORT          =   -9, ///< provided buffer is too short
BORE_ERROR_FEATURE_DISABLED          =  -10, ///< software-disabled feature
```

## 5.2 Rotation Unit related Errors

```
// rotation unit related errors
BORE_ERROR_ROTATION_UNIT_UNKNOWN      = -200, ///< unknown rotation unit on connect
BORE_ERROR_ROTATION_OPEN_FAILED       = -201, ///< failed to open serial port
BORE_ERROR_ROTATION_STATE_FAILED      = -202, ///< failed to set port state
BORE_ERROR_ROTATION_TIMEOUTS_FAILED   = -203, ///< failed to set port timeouts
BORE_ERROR_ROTATION_SPEED_TOO_LOW     = -204, ///< rotation speed too low
BORE_ERROR_ROTATION_SPEED_TOO_HIGH    = -205, ///< rotation speed too high
BORE_ERROR_ROTATION_SPEED_FAILED      = -206, ///< rotation speed command failed
BORE_ERROR_ROTATION_START_FAILED      = -207, ///< rotation start command failed
BORE_ERROR_ROTATION_STOP_FAILED       = -208, ///< rotation stop command failed
BORE_ERROR_ROTATION_COMM_FAILED       = -209, ///< rotation initial communication failed
BORE_ERROR_ROTATION_SPEED_UNDEFINED   = -210, ///< no rotation speed set before start
BORE_ERROR_ROTATION_NEEDS_STOP        = -211, ///< rotation unit must be stopped
```

## 5.3 Controller related Errors

```
// controller related errors
BORE_ERROR_CONTROLLER_UNKNOWN         = -300, ///< unknown or no controller specified
BORE_ERROR_CONTROLLER_INSTANCE_FAILED = -301, ///< failed to create controller instance
BORE_ERROR_CONTROLLER_OPEN_FAILED     = -302, ///< failed to open controller connection
BORE_ERROR_CONTROLLER_CLOSE_FAILED    = -303, ///< failed to close controller connection
BORE_ERROR_CONTROLLER_RELEASE_FAILED  = -304, ///< failed to release controller instance
BORE_ERROR_CONTROLLER_START_FAILED    = -305, ///< failed to start data acquisition
BORE_ERROR_CONTROLLER_STOP_FAILED     = -306, ///< failed to stop data acquisition
BORE_ERROR_CONTROLLER_INITIALIZATION  = -307, ///< failed to set initial parameters
```

## 5.4 Evaluation related Errors

```
// evaluation related errors
BORE_ERROR_EVAL_TOO_FEW_VALID_POINTS  = -401, ///< too few valid points for circle fit
```

## 5.5 Offset-Calibration related Errors

```
// calibration related errors
BORE_ERROR_CALIB_UNSUCCESSFUL         = -500, ///< calibration not successful
BORE_ERROR_CALIB_OFFSET_NOT_SET       = -501, ///< offset calibration value not set
BORE_ERROR_CALIB_ANGLE                = -502, ///< angle offset reference error
BORE_ERROR_CALIB_DATA_AMBIGUOUS       = -503, ///< data not suitable for slope correction
BORE_ERROR_CALIB_DATA_NOT_LINEAR      = -504, ///< factor fit data is not linear
```

## 5.6 Reference-Calibration related Errors

```
// reference calibration related errors
BORE_ERROR_REFCALIB_OPEN_FAILED       = -600, ///< file-open of refcalib file failed
BORE_ERROR_REFCALIB_READ_FAILED       = -601, ///< wrong file format or access problem
BORE_ERROR_REFCALIB_WRITE_FAILED      = -602, ///< problem while writing
BORE_ERROR_REFCALIB_UNINITIALIZED     = -603, ///< reference calibration not loaded
BORE_ERROR_REFCALIB_ILLEGAL_MODE      = -604, ///< unknown reference calibration mode
BORE_ERROR_REFCALIB_STOP_NEEDED       = -605, ///< acquisition running, need to stop
BORE_ERROR_REFCALIB_ONREQUEST_NEEDED  = -606, ///< refcalib on request mode required
BORE_ERROR_REFCALIB_FEW_DATA          = -607, ///< too few data supplied
BORE_ERROR_REFCALIB_BOREPARAM_FAILED  = -608, ///< failed setting internal parameters
```

## 5.7 Data IO related Errors

```
// data io related errors
BORE_ERROR_DATAIO_OPEN_READ           = -700, ///< open of file to read failed
BORE_ERROR_DATAIO_OPEN_WRITE          = -701, ///< open of file to write failed
BORE_ERROR_DATAIO_READ                = -702, ///< read failed
BORE_ERROR_DATAIO_WRITE               = -703, ///< write failed
BORE_ERROR_DATAIO_FORMAT              = -704  ///< read failed because of file format
```

## 5.8 Errors related to processing Calibration Stacks

```
// calibration stack related errors
BORE_ERROR_CALSTACK_RING_MISSING      = -800, ///< ring not found in segmentation
BORE_ERROR_CALSTACK_RING_INCOMPLETE   = -801, ///< ring too small in segmentation
BORE_ERROR_CALSTACK_FEW_REVS          = -802  ///< too few revolutions inside of ring
```

# 6. SDK and Sample Applications

Software and documentation are provided as zip-file. It has to be extracted to the desired location. Below the root directory boreCONTROL-SDK the following directories are present:

- **Development:** Software Development Kit - Development\SDK
  - **Include:** boreCONTROL.h
  - **Lib:** boreCONTROL.lib, boreCONTROL.dll and MedaqLib.dll
  - **Samples:** Example projects
    boreCONTROLDemo (simple demonstration)
    boreCONTROLExtDemo (extended test program)
    boreCONTROLOffline (Evaluation of export files)
    boreCONTROLRecalib (Recalibration)
  - **Samples\Bin:** Compiled versions of boreCONTROLDemo, boreCONTROLExtDemo, boreCONTROLOffline and boreCONTROLRecalib, together with the dlls needed to execute them.
- **Documentation:** Documentation of hardware, dll, calling interface
  - **Documentation.exe:** Viewer for the included documentation
  - **What's new.txt:** Changes relative to previous versions
  - **deutsch:** Documentation boreCONTROL hardware and dll in German language
  - **english:** Documentation boreCONTROL hardware and dll in English language
  - **interface:** Automatically generated software interface documentation
  - **html:** Helper files for documentation
- **Support:**
  - **CME2:** Software to operate rotation unit and for configuration
  - **SensorFinder:** Software to find Ethernet-based controllers in network
  - **vcredist_x86:** Microsoft redistributable, to be installed in case of missing library files.

The SDK is created and tested with Microsoft VS2008 under Windows 7, Service Pack 1. Please use the demo projects as basis for your own applications.

## 6.1 boreCONTROLDemo

This simple demonstration program opens a connection to the boreCONTROL hardware, starts the rotation unit and data acquisition, acquires a full circle of raw data and exports them to the file system. Then acquisition and rotation are stopped and the connection to boreCONTROL is released. Before the program is finished, the application waits for a return key press.

## 6.2 boreCONTROLExtDemo

This complex demonstration program is capable of calling most functions of boreCONTROL dll by supplying command line parameters. To start the boreCONTROLExtDemo application, open the cmd.exe console under “Start->Execute …” and change the current directory to “Development\Samples\Bin”-directory of the installation using the “cd” command.

When calling “boreCONTROLExtDemo -h”, help for using the program options is printed:

```
Usage: boreCONTROLExtDemo [options]
-----------------------------------
  boreCONTROL extended demo application with command line parameters.
  Default is IFC2461 controller at TCP/IP address 169.254.168.150
  and rotation unit at com port 1.
  If no complex command is issued, a single measurement is done.
  Options are:

  Connection options
    -ifc2451 addr     connect with ifc2451 at supplied tcpip address
```

```
   -ifc2461 addr     connect with ifc2461 at supplied tcpip address
   -ifc2471 addr     connect with ifc2471 at supplied tcpip address
   -comport value    set com port for rotation unit (default 1)
 Rotation unit and z-axis finetuning (rarely used)
   -anglezero deg    set angle zero position in degree
   -zscale value     set z scaling factor tick to um (default 0.1)
   -zoffset value    set z offset in um (default 0)
   -zzero            set z to 0 for current position
   -zsetpos value    set z to value in um for current position
 Data acquisition options
   -speed value      set rotation speed in Hz range 0.1-10 (default 2)
   -shutter value    set shutter time in um (default 200, 0 is auto)
   -rate value       set scan rate in Hz (default 2500)
 Evaluation
   -refcalib file    set reference calibration file
   -offset value     set offset for absolute measurement (default 0)
   -thres value      set intensity threshold in % (default 5)
   -uthres value     set upper intensity threshold in % (default 95)
   -baryfit          circle fit using barycenter data, not distance
 Optional reference peak acquisition before measurement
   -refrate value    set refpeak scan rate in Hz (def 2500)
   -refshutter value set refpeak  shuttertime in us (def 400, 0 auto)
 Complex commands (only one per command line)
   -info             print version information
   -dark value       calibrate dark with split position
   -angleadj thres   set zero angle by marked target, line thres (-1 auto)
   -calib value      determine offset using ring diameter in um
   -center value     center adjust mode, exit after iteration count
 Output features
   -export prefix    write data using prefix, number and .txt are added
   -wait             wait for return key press before exit
   -quiet            disable trace output
   -repeat value     number of repeats in normal evaluation mode
   -sleep seconds    seconds between acquisitions for repeat
   -v                be verbose (default off)
   -h                this help page
```

### 6.2.1 Sample calls for boreCONTROLExtDemo

The following list collects some typical usage for boreCONTROLExtDemo. Some calls need the calibration table for the currently used sensor pen as parameter. This is accomplished by the -refcalib option.

- Dark level compensation has to be done once before first use after remounting the senor pen. The pen opening has to be covered or the pen has “to look into the void”. Default values for IFC2461 dark level compensation are scanrate 300 Hz and shutter time 3000 µs, same as for in-house calibration. The parameter 65 for -dark in the example below represents the position of the split pixel between reference peak and measurement peak and has to be adapted according to the SplitPixel entry in the calibration file. 300 video signals are averaged internally.

  ```
  boreCONTROLExtDemo -rate 300 -shutter 3000 -dark 65
  ```

  or simply

  ```
  boreCONTROLExtDemo -dark 65
  ```

  IFC2451 controllers require 100 Hz with 10000 µs and splitpixel position 110 instead.

- The sensor pen then is placed inside of the 6 mm diameter calibration ring and centered approximately. By using the following call, a manual centering (if available) is supported. The parameter value after option -center determines the number of repetitions when the center position changes less than 3 µm before exiting. Hint: For performing the offset calibration below, center coordinates below 10 µm are by far sufficient.

  ```
  boreCONTROLExtDemo -refcalib calfile.txt -center 10
  ```

  Providing a calibration table is necessary to generate center offsets in µm.

- The pen is now centered inside of the calibration ring. To determine the offset value for absolute measurement, a calibration measurement is performed next. The parameter value after -calib

supplies the diameter in $\mu$m of the used calibration ring. The resulting offset is expected to be well below 30 $\mu$m.

```
boreCONTROLExtDemo -refcalib calfile.txt -calib 6000.0
```

– To perform an absolute measurement with the offset determined before, boreCONTROLExtDemo is called with the offset value determined by the calibration measurement, here 2.4 as an example.

```
boreCONTROLExtDemo -refcalib calfile.txt -offset 2.4
```

– The parameters for the scanning process may be adjusted also. The following example lists the options needed for 10 repeated measurements with 1 kHz scan rate, 4 Hz rotation speed and 200 $\mu$s shutter time (single line, line feed for improved readability):

```
boreCONTROLExtDemo -refcalib calfile.txt
  -offset 2.4 -rate 1000 -speed 4 -shutter 200 -repeat 10
```

– To simplify the manual centering, boreCONTROLExtDemo allows an adjustment of the orientation of the coordinate systems according to section 2.5. While the coordinate system may be rotated using the -anglezero option, the -angleadj options does this automatically inside of a ring with 0° position marked by a dark line. The parameter after -angleadj is the extraction intensity threshold in %, -1 sets the threshold automatically. The determined angle tick offset then has to be supplied when centering or acquiring measurement data using the -anglezero option.

```
boreCONTROLExtDemo -angleadj -1
```

– For optically compensated sensor pens the option -refcalib is provided. The parameter following -refcalib provides the file name of a calibration table to load. If either one of the options -refshutter or -refrate is supplied, the compensation mode BORE_REFCALIB_ONREQUEST is used, in which the reference position is evaluated once before measurement. Otherwise the sensor is used in mode BORE_REFCALIB_ALWAYS.
Example for BORE_REFCALIB_ALWAYS:

```
boreCONTROLExtDemo -refcalib calfile.txt
```

Example for BORE_REFCALIB_ONREQUEST on dark measurement objects:

```
boreCONTROLExtDemo -refcalib calfile.txt
  -rate 1000 -shutter 1000 -refrate 2500 -refshutter 400
```

## 6.3 boreCONTROLOffline

By using boreCONTROLOffline previously exported data (boreDataIOWrite) may be analyzed again with a modified parameter set and statistical information can be generated.

When calling “boreCONTROLOffline -h”, help for using the program options is printed:

```
Usage: boreCONTROLOffline [options] filename
-------------------------------------------
  Demonstration for boreCONTROL offline data analysis

  Parameters for typical scenarios:"
  1) Evaluate a boreCONTROL export file, append evaluation to eval.txt
     Examples: single file, directory, or file pattern
       -write eval.txt -all exportfilename.txt
       -write eval.txt -all dir
       -write eval.txt -all directory/ex_*.txt
  2) Evaluate a boreCONTROL export file with recalibration
     and (optional) offset
       -refcalib cal_file.txt -offset 10.0 exportfilename.txt
  3) Evaluate a boreCONTROL export file restricting the intensity
     range for evaluation
       -thres 10 -uthres 80 exportfilename.txt
  4) Evaluate a boreCONTROL export file after applying median filter,
     and export filtered result (export_prefix is prepended to files)
       -median 8 -export export_prefix exportfilename.txt
  5) specify diameter to compare evaluation with expected result
       -ring 5001.7 exportfilename.txt
  6) Evaluation for heavily decentered data with known angle
     positions for minmimum and maximum distance.
     Also available with automatic angle determination and sector
```

```
    size selection: 2nd example sector size 90° up to offset 50.0,
    5° for offset larger 200.0, ramp from (50.0,90°) to (200.0,5°)
      -selangle 30.0 5.0 -selangle 210.0 5.0 exportfilename.txt
      -autoangle 50.0 200.0 5 exportfilename.txt
 7) Evaluation of a set of numbered calibration ring export files
      -ring 3997  stack1single/ex0*.txt \\
      -ring 5998  stack1single/ex1*.txt \\
      -ring 7997  stack1single/ex2*.txt \\
      -ring 10007 stack1single/ex3*.txt \\
      -all -write evalstack.txt
 8) Evaluation of a set of calibration ring export files with
    diameters taken from stack information file (file format is
    BCCSI 1 nrofrings\\r\\nringdiam ringheight\\r\\n)
    Offset computation from multiple export files cutting away
    the smallest and largest diameter and averaging the rest
    2nd example with temperature specified for writing to results
    file.
      -multicut 1 -si calstackinfo.txt ex*.txt
      -multicut 1 -si calstackinfo.txt -temp 22.5 ex_temp22_5_*.txt \\
        -results results.txt

 Options are:
   -thres value      evaluation intensity threshold (5)
   -uthres value     evaluation intensity upper threshold (95)
   -refcalib file    optional reference calibration file
   -offset value     offset to add to distances (needs refcalib)
   -ring value       ring diameter in um for following files/filesets
   -baryfit          barycenter circle fit for data without distances
 Optional filters
   -median halfs     median filter, kernel 2*halfs+1
   -vmedian halfs    median filter valid points only, kernel 2*halfs+1
   -selangle ctr dev invalidate outside angle ctr+-dev (2x possible)
   -autoangle s e a  auto sector inval, start and end offset, end angle
 Multiple files processing options\n"
   -si file          stackinfo mode, generates patterns for each fileset
   -sisingle file    stackinfo mode, single set, ring auto determination
   -multicut cnt     discard cnt largest and smallest results for ring\n"
   -multimedian      use median instead of average on remaining results\n"
 Output options
   -export name      export data to disk (useful after filtering)
   -write name       append evaluation data to file
   -old              use old evaluation format for compatibility
   -calwrite name    append ring diameter and barycenters to file\n"
   -results name     append to result file, same format as AutoCalib\n"
   -temp celsius     temperature for results file\n"
 General options
   -wait             wait for key press before exit (default off)
   -quiet            disable trace output (default full output)
   -v                be verbose, also show dll version (default off)
   -h                output this help page
```

A typical usage is re-evaluation of exported data with modified calibration information:

```
boreCONTROLOffline.exe -refcalib newcalib.txt -thres 5 -uthres 95
    -write statistics.txt exportdata.txt
```

The export file “exportdata.txt” is read and the contained peak position information is converted to $\mu$m using the calibration file “newcalib.txt”. Measurement points with intensity below 5% or above 95% will be excluded from processing and the remaining points are supplied to a circle fit. The evaluation result is printed and also appended to the output file “statistics.txt” together with statistical information. The output file may be processed further using Microsoft Excel.

## 6.4 boreCONTROLRecalib

The boreCONTROL system is delivered with a calibration table for each sensor pen. In rare cases it might occur that the measurement values don’t match those at calibration time any longer. Then it is possible to adapt the existing calibration to new supporting point data acquired in gauge rings by using the boreCONTROLRecalib program.

As input boreCONTROLRecalib requires the manufacturer's calibration file including temperature compensation information and some files containing exported measurement data from a number of different (at least two) gauge rings, which should cover the measurement range used. As even gauge rings show deviations caused by production and occasionally have got contaminated spots inside, it's advantageous to provide multiple measurement data at different positions inside of every gauge ring.

To be able to supply the relationship between data and gauge ring, the file names are required to contain the diameter of the according gauge ring in $\mu$m after the first underscore "_". Two different formats are available: Either a dot-separated floating point notation or an integer notation, optionally followed by an additional underscore and the decimal places. Some examples are:

```
export_6350.2_01.txt      (6350.2 µm, measurement count 01)
calib_6350_2_14.txt       (6350.2 µm, measurement count 14)
ring_8890_set_7.txt       (8890.0 µm, measurement count 7)
```

These data files must be provided in ascending diameter order, together with the manufacturer's calibration, as arguments for the boreCONTROLRecalib call (one line):

```
boreCONTROLRecalib.exe -c cal3mm.txt
      Ex_4999_5_000.txt  Ex_4999_5_100.txt  Ex_4999_5_200.txt
      Ex_6998_7_000.txt  Ex_6998_7_100.txt  Ex_6998_7_200.txt
      Ex_9501_0_000.txt  Ex_9501_0_100.txt  Ex_9501_0_200.txt
```

The generated calibration file is written to the current working directory as "out.txt". A different output filename can be provided using the command line option "-o", followed by the favored name.

When calling "boreCONTROLOffline –h", help for using the program options is printed:

```
Usage: boreCONTROLRecalib [options] -c calfile list-of-data-files
-----------------------------------------------------------------
  Recalibration of calibration files using calibration ring data
  Uses either single revolution data in several calibration rings
  or a complete scan through a calibration ring stack

  The calibration stack variant takes a calibration stack information
  file containing ring count, diameters and heights as parameter and
  needs a file path (with wildcard in case of multiple export files),
  or a single export files containing a complete scan, as argument(s).
  File sets with wildcard character are loaded in alphabetical order.

  The single revolution data variant needs needs a list of export files
  as parameters. It's required that the export file names contain the
  diameter of the according gauge ring after the first underscore,
    e.g. ex_6350.2_a.txt or ex_6350_2_b.txt (in micron units)
  Data files should be supplied in a sequence with ascending diameters.
  Consecutive multiple data files for one diameter are averaged before
  being used for recalibration.

  Options are:
    -h              Print this help page
    -c calfile      Calibration file to modify (required)
    -stack file     Switches to calstack mode using information from file
    -cd diff_file   Write calibration difference analysis to named file
    -o name         Name of output calfile (default: out.txt)
    -thres val      Lower evaluation threshold for data files (default 5)
    -uthres val     Upper evaluation threshold for data files (default 95)
    -parse n        Diameter is after nth underscore (default is 0 auto)
    -nowait         Do not wait before exit
```

The according recalibration functionality is also provided by the SDK for integration into customer generated applications. See boreRefCalibRecalibrate in the SDK reference manual.

Note: The newly created modified calibration is only as good as the provided data, therefore reasonable attention is mandatory for acquisition. In particular, care has to be taken for the following points:

- Well centered gauge rings (ideally below 5 $\mu$m center offset)
- A sufficient number of gauge rings used (3 or more)

- An adequate number of positions inside of a gauge ring for averaging (5 or more)
- Coverage of the used measurement range
- Well cleaned and oil-free gauge rings without rust and dirt
- Preferably all measurement point intensity values lie between 20% and 95%
- Preferably at least 95% of the measurement points are marked as valid (intensity from 5% to 99%)

# 7. Additional Hints for using boreCONTROL

Please regard the following hint when operating boreCONTROL:

Never disconnect the ethernet connection between boreCONTROL and your PC.

Never disconnect power supply of boreCONTROL while the software has an opened connection to boreCONTROL.

This may require a restart of your PC and/or controller.

MICRO-EPSILON MESSTECHNIK GmbH & Co. KG
Königbacher Str. 15 · 94496 Ortenburg / Germany
Tel. +49 (0) 8542 / 168-0 · Fax +49 (0) 8542 / 168-90
info@micro-epsilon.de · www.micro-epsilon.com

X9751240-A061097JHS