Infineon One Point Calibration Library

1.0.1

Generated by Doxygen 1.8.14

Contents

1 Infineon One Point Calibration Library **1**

1.1 Introduction 1

1.2 Usage 2

1.3 Example 2

1.4 Example 2 4

1.5 Other considerations 4

2 Data Structure Index **5**

2.1 Data Structures 5

3 File Index **7**

3.1 File List 7

4 Data Structure Documentation **9**

4.1 ANALOG_SENSOR_DATA Struct Reference 9

4.1.1 Detailed Description 9

4.1.2 Field Documentation 9

4.1.2.1 cosN_LSB 9

4.1.2.2 cosP_LSB 10

4.1.2.3 diff_X 10

4.1.2.4 diff_Y 10

4.1.2.5 sinN_LSB 10

4.1.2.6 sinP_LSB 10

4.1.2.7 uncalibratedAngle 11

4.2 CALIB_DATA Struct Reference 11

4.2.1 Detailed Description . . . 11
4.2.2 Field Documentation . . . 12
4.2.2.1 cos_ortho . . . 12
4.2.2.2 full_calibration_performed . . . 12
4.2.2.3 mean_amplitudeX . . . 12
4.2.2.4 mean_amplitudeY . . . 12
4.2.2.5 mean_offsetX . . . 12
4.2.2.6 mean_offsetY . . . 13
4.2.2.7 rotation_data_CCW . . . 13
4.2.2.8 rotation_data_CW . . . 13
4.2.2.9 sin_ortho . . . 13
4.3 ROTATION_DATA_STORE Struct Reference . . . 13
4.3.1 Detailed Description . . . 14
4.3.2 Field Documentation . . . 14
4.3.2.1 amplitudeX . . . 14
4.3.2.2 amplitudeY . . . 14
4.3.2.3 angle135found . . . 15
4.3.2.4 angle225found . . . 15
4.3.2.5 angle315found . . . 15
4.3.2.6 angle45found . . . 15
4.3.2.7 calibration_done . . . 15
4.3.2.8 magnitude135 . . . 15
4.3.2.9 magnitude45 . . . 16
4.3.2.10 maxCos . . . 16
4.3.2.11 maxSin . . . 16
4.3.2.12 minCos . . . 16
4.3.2.13 minSin . . . 16
4.3.2.14 offsetX . . . 17
4.3.2.15 offsetY . . . 17
4.3.2.16 ortho_one_dir . . . 17
4.3.2.17 X135 . . . 17
4.3.2.18 X135_corr . . . 17
4.3.2.19 X45 . . . 18
4.3.2.20 X45_corr . . . 18
4.3.2.21 Y135 . . . 18
4.3.2.22 Y135_corr . . . 18
4.3.2.23 Y45 . . . 18
4.3.2.24 Y45_corr . . . 18

5 File Documentation **19**

5.1 onecalib.c File Reference . . . 19

5.1.1 Function Documentation . . . 20

5.1.1.1 ONECALIB_CalibrationFindParam() . . . 20

5.1.1.2 ONECALIB_GetAngle() . . . 21

5.1.1.3 ONECALIB_GetCalibAngle() . . . 21

5.1.1.4 ONECALIB_GetUncalibAngle() . . . 22

5.1.1.5 ONECALIB_InitCalibData() . . . 22

5.1.1.6 ONECALIB_InitSensorData() . . . 23

5.2 onecalib.h File Reference . . . 23

5.2.1 Macro Definition Documentation . . . 25

5.2.1.1 ANGLE135 . . . 25

5.2.1.2 ANGLE135_H . . . 25

5.2.1.3 ANGLE135_L . . . 25

5.2.1.4 ANGLE180_RAD . . . 26

5.2.1.5 ANGLE225 . . . 26

5.2.1.6 ANGLE225_H . . . 26

5.2.1.7 ANGLE225_L . . . 26

5.2.1.8 ANGLE315 . . . 26

5.2.1.9 ANGLE315_H . . . 27

5.2.1.10 ANGLE315_L . . . 27

5.2.1.11 ANGLE45 . . . 27

5.2.1.12 ANGLE45_H . . . 27

5.2.1.13 ANGLE45_L . . . 27

5.2.1.14 ANGLE_TOLERANCE . . . 28

5.2.1.15 ANGLE_TOLERANCE_RAD . . . 28

5.2.1.16 DEG2RADFACTOR . . . 28

5.2.1.17 RAD2DEGFACTOR . . . 28

5.2.2 Typedef Documentation . . . 28

5.2.2.1 ANALOG_SENSOR_DATA_t . . . 28

5.2.2.2 CALIB_DATA_t . . . 28

5.2.2.3 float64_t . . . 29

5.2.2.4 ROTATION_DATA_STORE_t . . . 29

5.2.3 Function Documentation . . . 29

5.2.3.1 ONECALIB_CalibrationFindParam() . . . 29

5.2.3.2 ONECALIB_GetAngle() . . . 30

5.2.3.3 ONECALIB_GetCalibAngle() . . . 30

5.2.3.4 ONECALIB_GetUncalibAngle() . . . 31

5.2.3.5 ONECALIB_InitCalibData() . . . 31

5.2.3.6 ONECALIB_InitSensorData() . . . 32

Index **33**

Chapter 1

Infineon One Point Calibration Library

Version

1.0.1

1.1 Introduction

This library is intended to help out hardware and software engineers, system integrators and developers with the process of calibrating analog angle sensors. An analog angle sensor does not provide any angle value, only analog sine and cosine (positive and negative) output signals. The angle has to be calculated externally.

Analog angle sensors have to be calibrated before they are used in order to achieve the specified angle accuracy. This calibration process includes a compensation of offset, amplitude and non-orthogonality of the output channels in use with the help of an external microcontroller. Calibration can be done in a one time fashion at the start-up of the system for example, or in a continuous matter (dynamic calibration). This library and the examples given talk about the one-time calibration however the functions could be easily extended to support continuous calibration.

Copyright

Copyright (C) 2017 Infineon Technologies AG. All rights reserved.

Attention

Infineon Technologies AG (INFINEON) is supplying this file for use exclusively with Infineon's products. This file can be freely distributed within development tools and software supporting such microcontroller products. THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR S↩ TATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. INFINEON SHALL NOT, IN A↩ NY CIRCUMSTANCES, BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL, ASPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

Warning

It is in the responsibility of the system integrator to insure that:

- the magnet is large enough to ensure that the non-homogeneity of the magnetic field in the sensing area is negligible.
- the mechanical tolerances of the sensor-magnet assembly are minimal.
- the output amplitude together with the resolution of the ADC in use matches the requirement in angle accuracy.

See also

`https://www.infineon.com/dgdl/Infineon-TLE5xxx(D)_Calibration_360_A↩ N-v02_00-AN-v02_00-EN.pdf?fileId=5546d46264a8de7e0164f09d8bfa228d`

1.2 Usage

Note

This library assumes that you have connected the analog angle sensor to an ADC and you can easily get the SINP, COSP, SINN, COSN digitized values.

The one time calibration process requires you to:

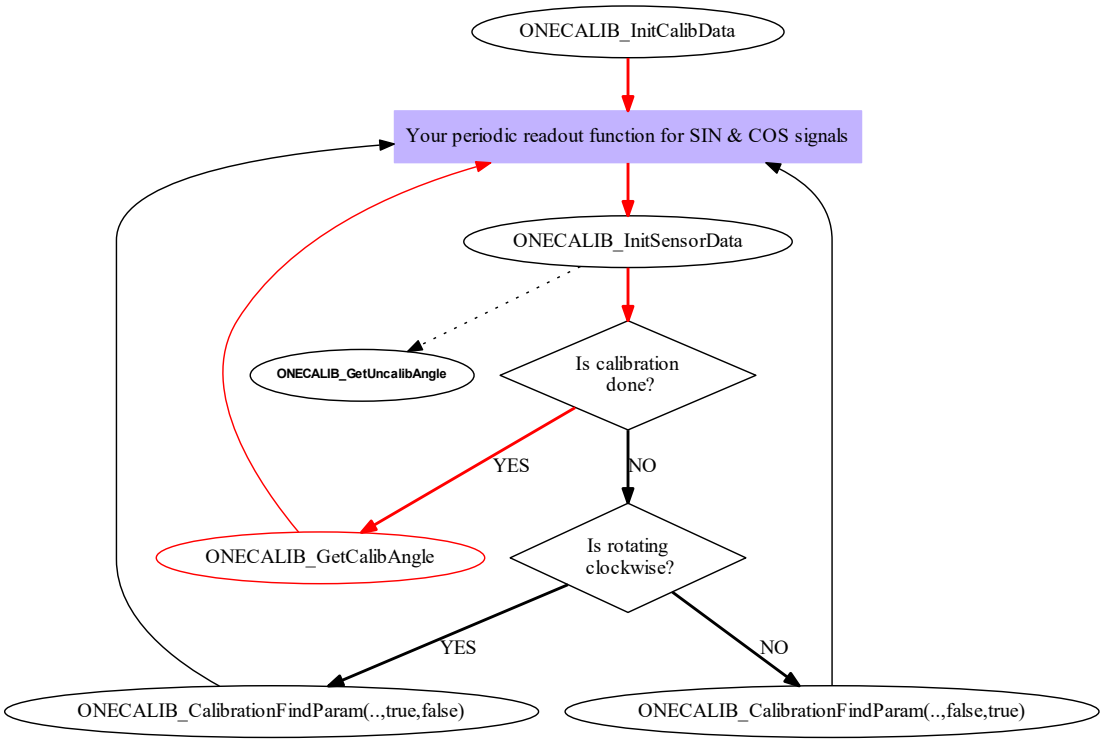
- turn the magnetic field 360 deg in one direction while acquiring SIN & COS data
- turn the magnetif field 360 deg in the opposite direction while aquiring SIN & COS data
- afterward the firmware library will compute:
  - amplitude & offset correction values for left turn
  - amplitude & offset correction values for right turn
  - mean amplitude & offset values

If your application is subjected to temperature variations you should perform dynamic/continuous calibration as the one time calibration can help reduce only the temperature independent offset.

Note

Because the calibration process is quite intensive computationally the system integrator should use either a microcontroller with an FPU or a dedicated math coprocessor/hardware CORDIC.

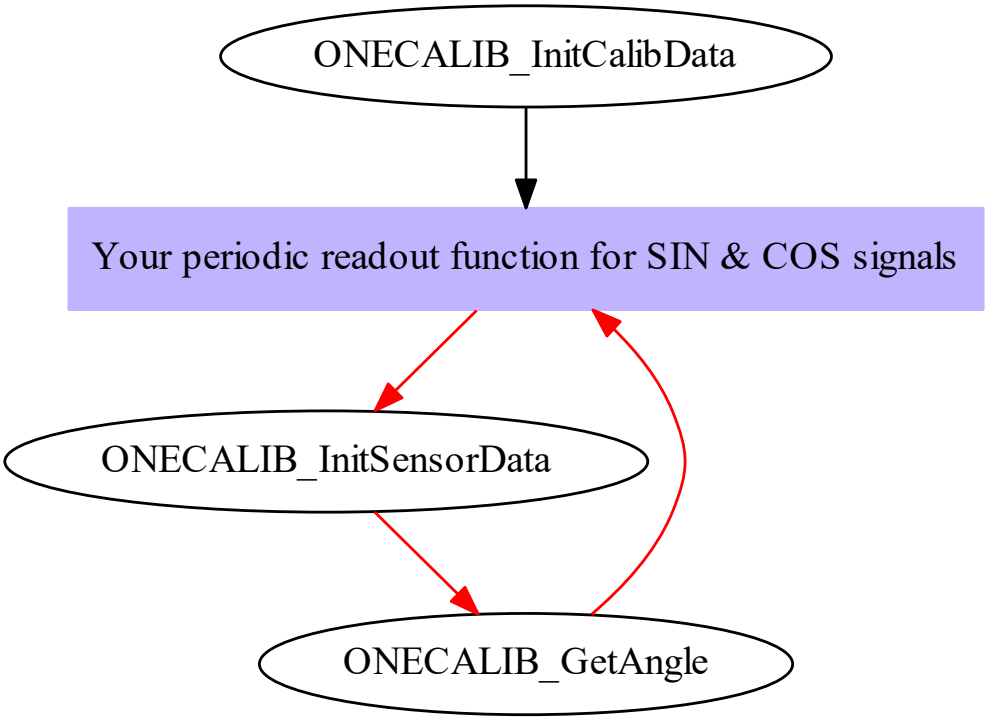
1.3 Example



Code Example 1:

```
CALIB_DATA_t calibration_store_params;

float64_t angle_read = 0;
uint32_t sinP_in_LSB = 0;
uint32_t cosP_in_LSB = 0;
uint32_t sinN_in_LSB = 0;
uint32_t cosN_in_LSB = 0;

int main(void)
{
    ONECALIB_InitCalibData(&calibration_store_params);

    while(1U)
    {
    //------------------- CALIBRATION FOR CW DIRECTION -------------------------------
            while (!calibration_store_params.rotation_data_CW.calibration_done) {
                // read sensor data
            // AnalogRead(ch) is a user defined function to read the analog voltage on P and N channels
            sinP_in_LSB = AnalogRead(ch0);
            cosP_in_LSB = AnalogRead(ch1);
            sinN_in_LSB = AnalogRead(ch2);
            cosN_in_LSB = AnalogRead(ch3);
            ONECALIB_InitSensorData(sinP_in_LSB, cosP_in_LSB, sinN_in_LSB, cosN_in_LSB);
            ONECALIB_CalibrationFindParam(&calibration_store_params, true, false);
            }

    //------------------- CALIBRATION FOR CCW DIRECTION -------------------------------
            while (!calibration_store_params.rotation_data_CCW.calibration_done) {
                // read sensor data
            // AnalogRead(ch) is a user defined function to read the analog voltage on P and N channels
            sinP_in_LSB = AnalogRead(ch0);
            cosP_in_LSB = AnalogRead(ch1);
            sinN_in_LSB = AnalogRead(ch2);
            cosN_in_LSB = AnalogRead(ch3);
            ONECALIB_InitSensorData(sinP_in_LSB, cosP_in_LSB, sinN_in_LSB, cosN_in_LSB);
            ONECALIB_CalibrationFindParam(&calibration_store_params, false, true);
            }

    //------------------- GET CALIBRATED ANGLE DATA ----------------------------------
            if (calibration_store_params.full_calibration_performed) {
                angle_read = ONECALIB_GetCalibAngle(&calibration_store_params);
            }
            // or  angle_read = ONECALIB_GetAngle();
            // Returns the calibrated or uncalibrated angle in degrees.
    }
}
```

1.4 Example 2



Code Example 2:

```
CALIB_DATA_t calibration_store_params;

float64_t differential_uncalibrated_angle_read = 0;
uint32_t sinP_in_LSB = 0;
uint32_t cosP_in_LSB = 0;
uint32_t sinN_in_LSB = 0;
uint32_t cosN_in_LSB = 0;

int main(void)
{
    ONECALIB_InitCalibData(&calibration_store_params);

    while(1U)
    {
        // read sensor data
        // AnalogRead(ch) is a user defined function to read the analog voltage on P and N channels
        sinP_in_LSB = AnalogRead(ch0);
        cosP_in_LSB = AnalogRead(ch1);
        sinN_in_LSB = AnalogRead(ch2);
        cosN_in_LSB = AnalogRead(ch3);
        ONECALIB_InitSensorData(sinP_in_LSB, cosP_in_LSB, sinN_in_LSB, cosN_in_LSB);
        differential_uncalibrated_angle_read = ONECALIB_GetAngle();
    }
}
```

1.5 Other considerations

See also

https://www.infineon.com/dgdl/Infineon-TLE5xxx(D)_Calibration_360_AN-v02_00-AN-v02_00-EN.pdf?fileId=5546d46264a8de7e0164f09d8bfa228d

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

ANALOG_SENSOR_DATA 9
CALIB_DATA 11
ROTATION_DATA_STORE 13

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

onecalib.c 19
onecalib.h 23

Chapter 4

Data Structure Documentation

4.1 ANALOG_SENSOR_DATA Struct Reference

`#include <onecalib.h>`

Data Fields

- uint32_t **cosP_LSB**
- uint32_t **cosN_LSB**
- uint32_t **sinP_LSB**
- uint32_t **sinN_LSB**
- int32_t **diff_X**
- int32_t **diff_Y**
- **float64_t uncalibratedAngle**

4.1.1 Detailed Description

Definition at line 52 of file onecalib.h.

4.1.2 Field Documentation

4.1.2.1 cosN_LSB

`uint32_t ANALOG_SENSOR_DATA::cosN_LSB`

negative cosine signal.

Definition at line 54 of file onecalib.h.

4.1.2.2 cosP_LSB

```
uint32_t ANALOG_SENSOR_DATA::cosP_LSB
```

positive cosine signal.

Definition at line 53 of file onecalib.h.

4.1.2.3 diff_X

```
int32_t ANALOG_SENSOR_DATA::diff_X
```

differential value for x-channel (cosine).

Definition at line 58 of file onecalib.h.

4.1.2.4 diff_Y

```
int32_t ANALOG_SENSOR_DATA::diff_Y
```

differential value for y-channel (sine).

Definition at line 59 of file onecalib.h.

4.1.2.5 sinN_LSB

```
uint32_t ANALOG_SENSOR_DATA::sinN_LSB
```

negative sine signal.

Definition at line 56 of file onecalib.h.

4.1.2.6 sinP_LSB

```
uint32_t ANALOG_SENSOR_DATA::sinP_LSB
```

positive sine signal.

Definition at line 55 of file onecalib.h.

4.1.2.7 uncalibratedAngle

```
float64_t ANALOG_SENSOR_DATA::uncalibratedAngle
```

differential angle.

Definition at line 61 of file onecalib.h.

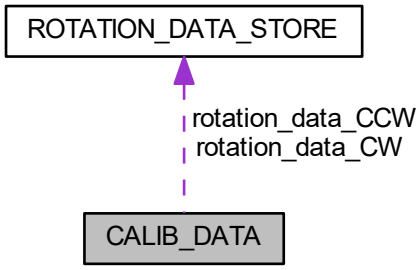
The documentation for this struct was generated from the following file:

- **onecalib.h**

4.2 CALIB_DATA Struct Reference

```
#include <onecalib.h>
```

Collaboration diagram for CALIB_DATA:



Data Fields

- **ROTATION_DATA_STORE_t rotation_data_CW**
- **ROTATION_DATA_STORE_t rotation_data_CCW**
- **float64_t mean_amplitudeX**
- **float64_t mean_amplitudeY**
- **float64_t mean_offsetX**
- **float64_t mean_offsetY**
- **float64_t sin_ortho**
- **float64_t cos_ortho**
- bool **full_calibration_performed**

4.2.1 Detailed Description

Definition at line 103 of file onecalib.h.

4.2.2 Field Documentation

4.2.2.1 cos_ortho

float64_t CALIB_DATA::cos_ortho

mean channel correction of orthogonality error cos (fi).

Definition at line 114 of file onecalib.h.

4.2.2.2 full_calibration_performed

bool CALIB_DATA::full_calibration_performed

Definition at line 116 of file onecalib.h.

4.2.2.3 mean_amplitudeX

float64_t CALIB_DATA::mean_amplitudeX

mean amplitude of COS signal from CW and CCW rotation.

Definition at line 107 of file onecalib.h.

4.2.2.4 mean_amplitudeY

float64_t CALIB_DATA::mean_amplitudeY

mean amplitude of SIN signal from CW and CCW rotation.

Definition at line 108 of file onecalib.h.

4.2.2.5 mean_offsetX

float64_t CALIB_DATA::mean_offsetX

mean offset of COS signal from CW and CCW rotation.

Definition at line 110 of file onecalib.h.

4.2.2.6 mean_offsetY

```
float64_t CALIB_DATA::mean_offsetY
```

mean offset of SIN signal from CW and CCW rotation.

Definition at line 111 of file onecalib.h.

4.2.2.7 rotation_data_CCW

```
ROTATION_DATA_STORE_t CALIB_DATA::rotation_data_CCW
```

Definition at line 105 of file onecalib.h.

4.2.2.8 rotation_data_CW

```
ROTATION_DATA_STORE_t CALIB_DATA::rotation_data_CW
```

Definition at line 104 of file onecalib.h.

Referenced by ONECALIB_CalibrationFindParam().

4.2.2.9 sin_ortho

```
float64_t CALIB_DATA::sin_ortho
```

mean channel correction of orthogonality error sin (fi).

Definition at line 113 of file onecalib.h.

The documentation for this struct was generated from the following file:

- **onecalib.h**

4.3 ROTATION_DATA_STORE Struct Reference

```
#include <onecalib.h>
```

Data Fields

- bool **angle45found**
- bool **angle135found**
- bool **angle225found**
- bool **angle315found**
- bool **calibration_done**
- int32_t **maxCos**
- int32_t **minCos**
- int32_t **maxSin**
- int32_t **minSin**
- **float64_t X45**
- **float64_t X135**
- **float64_t Y45**
- **float64_t Y135**
- **float64_t X45_corr**
- **float64_t X135_corr**
- **float64_t Y45_corr**
- **float64_t Y135_corr**
- **float64_t amplitudeX**
- **float64_t amplitudeY**
- **float64_t offsetX**
- **float64_t offsetY**
- **float64_t magnitude45**
- **float64_t magnitude135**
- **float64_t ortho_one_dir**

4.3.1 Detailed Description

Definition at line 65 of file onecalib.h.

4.3.2 Field Documentation

4.3.2.1 amplitudeX

```
float64_t ROTATION_DATA_STORE::amplitudeX
```

amplitude of COS signal.

Definition at line 90 of file onecalib.h.

4.3.2.2 amplitudeY

```
float64_t ROTATION_DATA_STORE::amplitudeY
```

amplitude of SIN signal.

Definition at line 91 of file onecalib.h.

4.3.2.3 angle135found

```
bool ROTATION_DATA_STORE::angle135found
```

Definition at line 67 of file onecalib.h.

4.3.2.4 angle225found

```
bool ROTATION_DATA_STORE::angle225found
```

Definition at line 68 of file onecalib.h.

4.3.2.5 angle315found

```
bool ROTATION_DATA_STORE::angle315found
```

Definition at line 69 of file onecalib.h.

4.3.2.6 angle45found

```
bool ROTATION_DATA_STORE::angle45found
```

Definition at line 66 of file onecalib.h.

4.3.2.7 calibration_done

```
bool ROTATION_DATA_STORE::calibration_done
```

Definition at line 71 of file onecalib.h.

Referenced by ONECALIB_CalibrationFindParam().

4.3.2.8 magnitude135

```
float64_t ROTATION_DATA_STORE::magnitude135
```

M135 value of signal.

Definition at line 97 of file onecalib.h.

4.3.2.9 magnitude45

```
float64_t ROTATION_DATA_STORE::magnitude45
```

M45 value of signal.

Definition at line 96 of file onecalib.h.

4.3.2.10 maxCos

```
int32_t ROTATION_DATA_STORE::maxCos
```

max of COS differential signal.

Definition at line 73 of file onecalib.h.

4.3.2.11 maxSin

```
int32_t ROTATION_DATA_STORE::maxSin
```

max of SIN differential signal.

Definition at line 75 of file onecalib.h.

4.3.2.12 minCos

```
int32_t ROTATION_DATA_STORE::minCos
```

min of COS differential signal.

Definition at line 74 of file onecalib.h.

4.3.2.13 minSin

```
int32_t ROTATION_DATA_STORE::minSin
```

min of SIN differential signal.

Definition at line 76 of file onecalib.h.

4.3.2.14 offsetX

```
float64_t ROTATION_DATA_STORE::offsetX
```

offset of COSP signal.

Definition at line 93 of file onecalib.h.

4.3.2.15 offsetY

```
float64_t ROTATION_DATA_STORE::offsetY
```

offset of COSN signal.

Definition at line 94 of file onecalib.h.

4.3.2.16 ortho_one_dir

```
float64_t ROTATION_DATA_STORE::ortho_one_dir
```

orthogonality error of signal.

Definition at line 99 of file onecalib.h.

4.3.2.17 X135

```
float64_t ROTATION_DATA_STORE::X135
```

COS value at 135 deg.

Definition at line 79 of file onecalib.h.

4.3.2.18 X135_corr

```
float64_t ROTATION_DATA_STORE::X135_corr
```

COS amplitude and offset corrected value at 135 deg.

Definition at line 85 of file onecalib.h.

4.3.2.19 X45

float64_t ROTATION_DATA_STORE::X45

COS value at 45 deg.

Definition at line 78 of file onecalib.h.

4.3.2.20 X45_corr

float64_t ROTATION_DATA_STORE::X45_corr

COS amplitude and offset corrected value at 45 deg.

Definition at line 84 of file onecalib.h.

4.3.2.21 Y135

float64_t ROTATION_DATA_STORE::Y135

SIN value at 135 deg.

Definition at line 82 of file onecalib.h.

4.3.2.22 Y135_corr

float64_t ROTATION_DATA_STORE::Y135_corr

SIN amplitude and offset corrected value at 135 deg.

Definition at line 88 of file onecalib.h.

4.3.2.23 Y45

float64_t ROTATION_DATA_STORE::Y45

SIN value at 45 deg.

Definition at line 81 of file onecalib.h.

4.3.2.24 Y45_corr

float64_t ROTATION_DATA_STORE::Y45_corr

SIN amplitude and offset corrected value at 45 deg.

Definition at line 87 of file onecalib.h.

The documentation for this struct was generated from the following file:

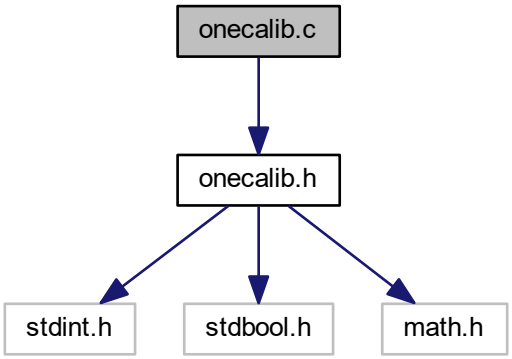
- **onecalib.h**

Chapter 5

File Documentation

5.1 onecalib.c File Reference

`#include "onecalib.h"`
Include dependency graph for onecalib.c:



Functions

- void **ONECALIB_InitSensorData** (uint32_t sinP_in_LSB, uint32_t cosP_in_LSB, uint32_t sinN_in_LSB, uint32_t cosN_in_LSB)

  Function used for ANALOG_SENSOR_DATA_t data assignment.
- void **ONECALIB_InitCalibData** (**CALIB_DATA_t** ∗calib_param)

  Function used for CALIB_DATA_t data initialization. The pointer of type CALIB_DATA_t will be used to store calibration parameters.
- **float64_t ONECALIB_GetAngle** (const **CALIB_DATA_t** ∗calib_param)

  Calculates the calibrated angle if calibration has been performed, else the uncalibrated angle.
- **float64_t ONECALIB_GetUncalibAngle** (void)

  Calculates the uncalibrated angle.

- **float64_t ONECALIB_GetCalibAngle** (const **CALIB_DATA_t** *calib_param)

  Calculates the calibrated angle.
- void **ONECALIB_CalibrationFindParam** (**CALIB_DATA_t** *calib_param, bool rotate_CW, bool rotate_CCW)

  This function is used to extract the calibration parameters. It should called until calib_param.full_calibration_performed is true. To extract the correct rotation, the input parameter rotate_CW should be true until calib_param.rotation_data_CW.calibration_done is true(while the magnet is spinning in CW direction). Same for the CCW direction. While calib_param.full_calibration_performed is false only the differential uncalibrated angle is available.

5.1.1 Function Documentation

5.1.1.1 ONECALIB_CalibrationFindParam()

```
void ONECALIB_CalibrationFindParam (
              CALIB_DATA_t * calib_param,
             bool rotate_CW,
             bool rotate_CCW )
```

This function is used to extract the calibration parameters. It should called until calib_param.full_calibration_performed is true. To extract the correct rotation, the input parameter rotate_CW should be true until calib_param.rotation_data_CW.calibration_done is true(while the magnet is spinning in CW direction). Same for the CCW direction. While calib_param.full_calibration_performed is false only the differential uncalibrated angle is available.

This function will extract the maximum, minimum voltage levels, amplitude, offset, and orthogonality. Based on the found values, it will calculate the necessary calibration parameters.

Usage example:

```
------------------- CALIBRATION FOR CW DIRECTION -------------------------------
while (!calibration_store_params.rotation_data_CW.calibration_done) {
   // read sensor data
   ONECALIB_InitSensorData(..);
   ONECALIB_CalibrationFindParam(&calibration_store_params, true,
   false);
}

------------------- CALIBRATION FOR CCW DIRECTION ------------------------------
UART_Transmit(&UART_0, message_ccw, sizeof(message_ccw) - 1);
while (!calibration_store_params.rotation_data_CCW.calibration_done) {
   // read sensor data
   ONECALIB_InitSensorData(..);
   ONECALIB_CalibrationFindParam(&calibration_store_params,
   false, true);
}

------------------- GET CALIBRATED ANGLE DATA ----------------------------------
if (calibration_store_params.full_calibration_performed) {
   angle_read = ONECALIB_GetCalibAngle(&calibration_store_params);
}
```

Parameters

in	*calib_param*	is a pointer of type CALIB_DATA_t, this structure holds the current sensor calibration parameters.
in	*rotate_CW*	is a bool that is true if the rotation is in clockwise direction.
in	*rotate_CCW*	is a bool that is true if the rotation is in counter-clockwise direction.

Returns

Nothing

Definition at line 468 of file onecalib.c.

References ROTATION_DATA_STORE::calibration_done, and CALIB_DATA::rotation_data_CW.

5.1.1.2 ONECALIB_GetAngle()

```
float64_t ONECALIB_GetAngle (
            const  CALIB_DATA_t * calib_param )
```

Calculates the calibrated angle if calibration has been performed, else the uncalibrated angle.

This function checks the stored calibration parameters and returns a angle value, no matter if the calibration procedure has been performed.

Parameters

in	*calib_param*	is a pointer of type CALIB_DATA_t, this structure holds the current sensor calibration parameters.

Returns

Returns the calibrated or uncalibrated angle in degrees.

Definition at line 372 of file onecalib.c.

5.1.1.3 ONECALIB_GetCalibAngle()

```
float64_t ONECALIB_GetCalibAngle (
            const  CALIB_DATA_t * calib_param )
```

Calculates the calibrated angle.

Don't use this if the calibration procedure has not been performed. This function dosen't check the stored calibration parameters and returns the calibrated angle value.

Parameters

in	*calib_param*	is a pointer of type CALIB_DATA_t, this structure holds the current sensor calibration parameters.

Returns

Returns the calibrated angle in degrees.

Definition at line 414 of file onecalib.c.

5.1.1.4 ONECALIB_GetUncalibAngle()

```
float64_t ONECALIB_GetUncalibAngle (
            void  )
```

Calculates the uncalibrated angle.

This function uses differential sensor data to extract the angle.

Returns

Returns the uncalibrated angle in degrees.

Definition at line 397 of file onecalib.c.

5.1.1.5 ONECALIB_InitCalibData()

```
void ONECALIB_InitCalibData (
              CALIB_DATA_t * calib_param )
```

Function used for CALIB_DATA_t data initialization. The pointer of type CALIB_DATA_t will be used to store calibration parameters.

This function must be called once before ONECALIB_CalibrationFindParam(..).

Parameters

in	*calib_param*	is a pointer of type CALIB_DATA_t.

Returns

Nothing

Definition at line 301 of file onecalib.c.

5.1.1.6 ONECALIB_InitSensorData()

```
void ONECALIB_InitSensorData (
             uint32_t sinP_in_LSB,
             uint32_t cosP_in_LSB,
             uint32_t sinN_in_LSB,
             uint32_t cosN_in_LSB )
```

Function used for ANALOG_SENSOR_DATA_t data assignment.

This function must be called after the data readout.

The initialized pointer will be used in the angle calculation and calibration process.

Parameters

in	*cosP_in_LSB*	is the sensor value in LSB corresponding to the positive cosinus.
in	*cosN_in_LSB*	is the sensor value in LSB corresponding to the negative cosinus.
in	*sinP_in_LSB*	is the sensor value in LSB corresponding to the positive sinus.
in	*sinN_in_LSB*	is the sensor value in LSB corresponding to the negative sinus.

Returns

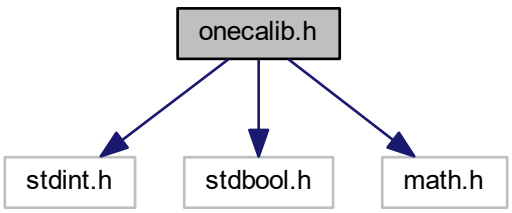
Nothing

Definition at line 242 of file onecalib.c.

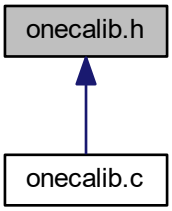
5.2 onecalib.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
```

Include dependency graph for onecalib.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **ANALOG_SENSOR_DATA**
- struct **ROTATION_DATA_STORE**
- struct **CALIB_DATA**

Macros

- #define **RAD2DEGFACTOR** (180.00 / M_PI)
- #define **DEG2RADFACTOR** (M_PI / 180.00)
- #define **ANGLE_TOLERANCE** 0.25
- #define **ANGLE_TOLERANCE_RAD** (**ANGLE_TOLERANCE** ∗ **DEG2RADFACTOR**)
- #define **ANGLE180_RAD** (180.00 ∗ **DEG2RADFACTOR**)
- #define **ANGLE45** 45.00
- #define **ANGLE135** 135.00
- #define **ANGLE225** -135.00
- #define **ANGLE315** -45.00
- #define **ANGLE45_H** ((**ANGLE45** + **ANGLE_TOLERANCE**) ∗ **DEG2RADFACTOR**)
- #define **ANGLE45_L** ((**ANGLE45** - **ANGLE_TOLERANCE**) ∗ **DEG2RADFACTOR**)
- #define **ANGLE135_H** ((**ANGLE135** + **ANGLE_TOLERANCE**) ∗ **DEG2RADFACTOR**)
- #define **ANGLE135_L** ((**ANGLE135** - **ANGLE_TOLERANCE**) ∗ **DEG2RADFACTOR**)
- #define **ANGLE225_H** ((**ANGLE225** + **ANGLE_TOLERANCE**) ∗ **DEG2RADFACTOR**)
- #define **ANGLE225_L** ((**ANGLE225** - **ANGLE_TOLERANCE**) ∗ **DEG2RADFACTOR**)
- #define **ANGLE315_H** ((**ANGLE315** + **ANGLE_TOLERANCE**) ∗ **DEG2RADFACTOR**)
- #define **ANGLE315_L** ((**ANGLE315** - **ANGLE_TOLERANCE**) ∗ **DEG2RADFACTOR**)

Typedefs

- typedef double **float64_t**
- typedef struct **ANALOG_SENSOR_DATA** **ANALOG_SENSOR_DATA_t**
- typedef struct **ROTATION_DATA_STORE** **ROTATION_DATA_STORE_t**
- typedef struct **CALIB_DATA** **CALIB_DATA_t**

Functions

- void **ONECALIB_InitSensorData** (uint32_t cosP_in_LSB, uint32_t cosN_in_LSB, uint32_t sinP_in_LSB, uint32_t sinN_in_LSB)

  Function used for ANALOG_SENSOR_DATA_t data assignment.
- void **ONECALIB_InitCalibData** (**CALIB_DATA_t** ∗param_store)

  Function used for CALIB_DATA_t data initialization. The pointer of type CALIB_DATA_t will be used to store calibration parameters.
- **float64_t ONECALIB_GetAngle** (const **CALIB_DATA_t** ∗calib_param)

  Calculates the calibrated angle if calibration has been performed, else the uncalibrated angle.
- **float64_t ONECALIB_GetUncalibAngle** (void)

  Calculates the uncalibrated angle.
- **float64_t ONECALIB_GetCalibAngle** (const **CALIB_DATA_t** ∗calib_param)

  Calculates the calibrated angle.
- void **ONECALIB_CalibrationFindParam** (**CALIB_DATA_t** ∗param_store, bool rotate_CW, bool rotate_↩ CCW)

  This function is used to extract the calibration parameters. It should called until calib_param.full_calibration_performed is true. To extract the correct rotation, the input parameter rotate_CW should be true until calib_param.rotation_↩ data_CW.calibration_done is true(while the magnet is spinning in CW direction). Same for the CCW direction. While calib_param.full_calibration_performed is false only the differential uncalibrated angle is available.

5.2.1 Macro Definition Documentation

5.2.1.1 ANGLE135

```
#define ANGLE135 135.00
```

angle position used for the calibration process.

Definition at line 37 of file onecalib.h.

5.2.1.2 ANGLE135_H

```
#define ANGLE135_H (( ANGLE135 + ANGLE_TOLERANCE) * DEG2RADFACTOR)
```

angle position limit in radians used for the calibration process.

Definition at line 43 of file onecalib.h.

5.2.1.3 ANGLE135_L

```
#define ANGLE135_L (( ANGLE135 - ANGLE_TOLERANCE) * DEG2RADFACTOR)
```

angle position limit in radians used for the calibration process.

Definition at line 44 of file onecalib.h.

5.2.1.4 ANGLE180_RAD

```
#define ANGLE180_RAD (180.00 *  DEG2RADFACTOR)
```

angle position in radians used for the calibration process.

Definition at line 34 of file onecalib.h.

5.2.1.5 ANGLE225

```
#define ANGLE225 -135.00
```

angle position used for the calibration process.

Definition at line 38 of file onecalib.h.

5.2.1.6 ANGLE225_H

```
#define ANGLE225_H (( ANGLE225 +  ANGLE_TOLERANCE) *  DEG2RADFACTOR)
```

angle position limit in radians used for the calibration process.

Definition at line 45 of file onecalib.h.

5.2.1.7 ANGLE225_L

```
#define ANGLE225_L (( ANGLE225 -  ANGLE_TOLERANCE) *  DEG2RADFACTOR)
```

angle position limit in radians used for the calibration process.

Definition at line 46 of file onecalib.h.

5.2.1.8 ANGLE315

```
#define ANGLE315 -45.00
```

angle position used for the calibration process.

Definition at line 39 of file onecalib.h.

5.2.1.9 ANGLE315_H

```
#define ANGLE315_H (( ANGLE315 + ANGLE_TOLERANCE) * DEG2RADFACTOR)
```

angle position limit in radians used for the calibration process.

Definition at line 47 of file onecalib.h.

5.2.1.10 ANGLE315_L

```
#define ANGLE315_L (( ANGLE315 - ANGLE_TOLERANCE) * DEG2RADFACTOR)
```

angle position limit in radians used for the calibration process.

Definition at line 48 of file onecalib.h.

5.2.1.11 ANGLE45

```
#define ANGLE45 45.00
```

angle position used for the calibration process.

Definition at line 36 of file onecalib.h.

5.2.1.12 ANGLE45_H

```
#define ANGLE45_H (( ANGLE45 + ANGLE_TOLERANCE) * DEG2RADFACTOR)
```

angle position limit in radians used for the calibration process.

Definition at line 41 of file onecalib.h.

5.2.1.13 ANGLE45_L

```
#define ANGLE45_L (( ANGLE45 - ANGLE_TOLERANCE) * DEG2RADFACTOR)
```

angle position limit in radians used for the calibration process.

Definition at line 42 of file onecalib.h.

5.2.1.14 ANGLE_TOLERANCE

```
#define ANGLE_TOLERANCE 0.25
```

tolerance angle used for calibration process.

Definition at line 32 of file onecalib.h.

5.2.1.15 ANGLE_TOLERANCE_RAD

```
#define ANGLE_TOLERANCE_RAD ( ANGLE_TOLERANCE *  DEG2RADFACTOR)
```

tolerance angle in radians used for the calibration process.

Definition at line 33 of file onecalib.h.

5.2.1.16 DEG2RADFACTOR

```
#define DEG2RADFACTOR (M_PI / 180.00)
```

Definition at line 31 of file onecalib.h.

5.2.1.17 RAD2DEGFACTOR

```
#define RAD2DEGFACTOR (180.00 / M_PI)
```

Definition at line 30 of file onecalib.h.

5.2.2 Typedef Documentation

5.2.2.1 ANALOG_SENSOR_DATA_t

```
typedef struct  ANALOG_SENSOR_DATA  ANALOG_SENSOR_DATA_t
```

5.2.2.2 CALIB_DATA_t

```
typedef struct  CALIB_DATA  CALIB_DATA_t
```

5.2.2.3 float64_t

```
typedef double  float64_t
```

Definition at line 50 of file onecalib.h.

5.2.2.4 ROTATION_DATA_STORE_t

```
typedef struct  ROTATION_DATA_STORE  ROTATION_DATA_STORE_t
```

5.2.3 Function Documentation

5.2.3.1 ONECALIB_CalibrationFindParam()

```
void ONECALIB_CalibrationFindParam (
             CALIB_DATA_t * calib_param,
            bool rotate_CW,
            bool rotate_CCW )
```

This function is used to extract the calibration parameters. It should called until calib_param.full_calibration_↩ performed is true. To extract the correct rotation, the input parameter rotate_CW should be true until calib_↩ param.rotation_data_CW.calibration_done is true(while the magnet is spinning in CW direction). Same for the CCW direction. While calib_param.full_calibration_performed is false only the differential uncalibrated angle is available.

This function will extract the maximum, minimum voltage levels, amplitude, offset, and orthogonality. Based on the found values, it will calculate the necessary calibration parameters.

Usage example:

```
------------------- CALIBRATION FOR CW DIRECTION -------------------------------
while (!calibration_store_params.rotation_data_CW.calibration_done) {
   // read sensor data
   ONECALIB_InitSensorData(..);
   ONECALIB_CalibrationFindParam(&calibration_store_params, true,
   false);
}

------------------- CALIBRATION FOR CCW DIRECTION ------------------------------
UART_Transmit(&UART_0, message_ccw, sizeof(message_ccw) - 1);
while (!calibration_store_params.rotation_data_CCW.calibration_done) {
   // read sensor data
   ONECALIB_InitSensorData(..);
   ONECALIB_CalibrationFindParam(&calibration_store_params,
   false, true);
}

------------------- GET CALIBRATED ANGLE DATA ----------------------------------
if (calibration_store_params.full_calibration_performed) {
   angle_read = ONECALIB_GetCalibAngle(&calibration_store_params);
}
```

Parameters

in	*calib_param*	is a pointer of type CALIB_DATA_t, this structure holds the current sensor calibration parameters.
in	*rotate_CW*	is a bool that is true if the rotation is in clockwise direction.
in	*rotate_CCW*	is a bool that is true if the rotation is in counter-clockwise direction.

Returns

Nothing

Definition at line 468 of file onecalib.c.

References ROTATION_DATA_STORE::calibration_done, and CALIB_DATA::rotation_data_CW.

5.2.3.2 ONECALIB_GetAngle()

```
float64_t ONECALIB_GetAngle (
            const  CALIB_DATA_t * calib_param )
```

Calculates the calibrated angle if calibration has been performed, else the uncalibrated angle.

This function checks the stored calibration parameters and returns a angle value, no matter if the calibration procedure has been performed.

Parameters

in	*calib_param*	is a pointer of type CALIB_DATA_t, this structure holds the current sensor calibration parameters.

Returns

Returns the calibrated or uncalibrated angle in degrees.

Definition at line 372 of file onecalib.c.

5.2.3.3 ONECALIB_GetCalibAngle()

```
float64_t ONECALIB_GetCalibAngle (
            const  CALIB_DATA_t * calib_param )
```

Calculates the calibrated angle.

Don't use this if the calibration procedure has not been performed. This function dosen't check the stored calibration parameters and returns the calibrated angle value.

Parameters

in	*calib_param*	is a pointer of type CALIB_DATA_t, this structure holds the current sensor calibration parameters.

Returns

Returns the calibrated angle in degrees.

Definition at line 414 of file onecalib.c.

5.2.3.4 ONECALIB_GetUncalibAngle()

```
float64_t ONECALIB_GetUncalibAngle (
            void  )
```

Calculates the uncalibrated angle.

This function uses differential sensor data to extract the angle.

Returns

Returns the uncalibrated angle in degrees.

Definition at line 397 of file onecalib.c.

5.2.3.5 ONECALIB_InitCalibData()

```
void ONECALIB_InitCalibData (
              CALIB_DATA_t * calib_param )
```

Function used for CALIB_DATA_t data initialization. The pointer of type CALIB_DATA_t will be used to store calibration parameters.

This function must be called once before ONECALIB_CalibrationFindParam(..).

Parameters

in	*calib_param*	is a pointer of type CALIB_DATA_t.

Returns

Nothing

Definition at line 301 of file onecalib.c.

5.2.3.6 ONECALIB_InitSensorData()

```
void ONECALIB_InitSensorData (
            uint32_t sinP_in_LSB,
            uint32_t cosP_in_LSB,
            uint32_t sinN_in_LSB,
            uint32_t cosN_in_LSB )
```

Function used for ANALOG_SENSOR_DATA_t data assignment.

This function must be called after the data readout.

The initialized pointer will be used in the angle calculation and calibration process.

Parameters

in	*cosP_in_LSB*	is the sensor value in LSB corresponding to the positive cosinus.
in	*cosN_in_LSB*	is the sensor value in LSB corresponding to the negative cosinus.
in	*sinP_in_LSB*	is the sensor value in LSB corresponding to the positive sinus.
in	*sinN_in_LSB*	is the sensor value in LSB corresponding to the negative sinus.

Returns

Nothing

Definition at line 242 of file onecalib.c.

Index

ANALOG_SENSOR_DATA_t
    onecalib.h, 28
ANALOG_SENSOR_DATA, 9
    cosN_LSB, 9
    cosP_LSB, 9
    diff_X, 10
    diff_Y, 10
    sinN_LSB, 10
    sinP_LSB, 10
    uncalibratedAngle, 10
ANGLE135
    onecalib.h, 25
ANGLE135_H
    onecalib.h, 25
ANGLE135_L
    onecalib.h, 25
ANGLE180_RAD
    onecalib.h, 25
ANGLE225
    onecalib.h, 26
ANGLE225_H
    onecalib.h, 26
ANGLE225_L
    onecalib.h, 26
ANGLE315
    onecalib.h, 26
ANGLE315_H
    onecalib.h, 26
ANGLE315_L
    onecalib.h, 27
ANGLE45
    onecalib.h, 27
ANGLE45_H
    onecalib.h, 27
ANGLE45_L
    onecalib.h, 27
ANGLE_TOLERANCE_RAD
    onecalib.h, 28
ANGLE_TOLERANCE
    onecalib.h, 27
amplitudeX
    ROTATION_DATA_STORE, 14
amplitudeY
    ROTATION_DATA_STORE, 14
angle135found
    ROTATION_DATA_STORE, 14
angle225found
    ROTATION_DATA_STORE, 15
angle315found
    ROTATION_DATA_STORE, 15
angle45found
    ROTATION_DATA_STORE, 15

CALIB_DATA_t
    onecalib.h, 28
CALIB_DATA, 11
    cos_ortho, 12
    full_calibration_performed, 12
    mean_amplitudeX, 12
    mean_amplitudeY, 12
    mean_offsetX, 12
    mean_offsetY, 12
    rotation_data_CCW, 13
    rotation_data_CW, 13
    sin_ortho, 13
calibration_done
    ROTATION_DATA_STORE, 15
cos_ortho
    CALIB_DATA, 12
cosN_LSB
    ANALOG_SENSOR_DATA, 9
cosP_LSB
    ANALOG_SENSOR_DATA, 9

DEG2RADFACTOR
    onecalib.h, 28
diff_X
    ANALOG_SENSOR_DATA, 10
diff_Y
    ANALOG_SENSOR_DATA, 10

float64_t
    onecalib.h, 28
full_calibration_performed
    CALIB_DATA, 12

magnitude135
    ROTATION_DATA_STORE, 15
magnitude45
    ROTATION_DATA_STORE, 15
maxCos
    ROTATION_DATA_STORE, 16
maxSin
    ROTATION_DATA_STORE, 16
mean_amplitudeX
    CALIB_DATA, 12
mean_amplitudeY
    CALIB_DATA, 12
mean_offsetX

CALIB_DATA, 12
mean_offsetY
CALIB_DATA, 12
minCos
ROTATION_DATA_STORE, 16
minSin
ROTATION_DATA_STORE, 16

ONECALIB_CalibrationFindParam
onecalib.c, 20
onecalib.h, 29
ONECALIB_GetAngle
onecalib.c, 21
onecalib.h, 30
ONECALIB_GetCalibAngle
onecalib.c, 21
onecalib.h, 30
ONECALIB_GetUncalibAngle
onecalib.c, 22
onecalib.h, 31
ONECALIB_InitCalibData
onecalib.c, 22
onecalib.h, 31
ONECALIB_InitSensorData
onecalib.c, 22
onecalib.h, 32
offsetX
ROTATION_DATA_STORE, 16
offsetY
ROTATION_DATA_STORE, 17
onecalib.c, 19
ONECALIB_CalibrationFindParam, 20
ONECALIB_GetAngle, 21
ONECALIB_GetCalibAngle, 21
ONECALIB_GetUncalibAngle, 22
ONECALIB_InitCalibData, 22
ONECALIB_InitSensorData, 22
onecalib.h, 23
ANALOG_SENSOR_DATA_t, 28
ANGLE135, 25
ANGLE135_H, 25
ANGLE135_L, 25
ANGLE180_RAD, 25
ANGLE225, 26
ANGLE225_H, 26
ANGLE225_L, 26
ANGLE315, 26
ANGLE315_H, 26
ANGLE315_L, 27
ANGLE45, 27
ANGLE45_H, 27
ANGLE45_L, 27
ANGLE_TOLERANCE_RAD, 28
ANGLE_TOLERANCE, 27
CALIB_DATA_t, 28
DEG2RADFACTOR, 28
float64_t, 28
ONECALIB_CalibrationFindParam, 29
ONECALIB_GetAngle, 30
ONECALIB_GetCalibAngle, 30
ONECALIB_GetUncalibAngle, 31
ONECALIB_InitCalibData, 31
ONECALIB_InitSensorData, 32
RAD2DEGFACTOR, 28
ROTATION_DATA_STORE_t, 29
ortho_one_dir
ROTATION_DATA_STORE, 17

RAD2DEGFACTOR
onecalib.h, 28
ROTATION_DATA_STORE_t
onecalib.h, 29
ROTATION_DATA_STORE, 13
amplitudeX, 14
amplitudeY, 14
angle135found, 14
angle225found, 15
angle315found, 15
angle45found, 15
calibration_done, 15
magnitude135, 15
magnitude45, 15
maxCos, 16
maxSin, 16
minCos, 16
minSin, 16
offsetX, 16
offsetY, 17
ortho_one_dir, 17
X135, 17
X135_corr, 17
X45, 17
X45_corr, 18
Y135, 18
Y135_corr, 18
Y45, 18
Y45_corr, 18
rotation_data_CCW
CALIB_DATA, 13
rotation_data_CW
CALIB_DATA, 13

sin_ortho
CALIB_DATA, 13
sinN_LSB
ANALOG_SENSOR_DATA, 10
sinP_LSB
ANALOG_SENSOR_DATA, 10

uncalibratedAngle
ANALOG_SENSOR_DATA, 10

X135
ROTATION_DATA_STORE, 17
X135_corr
ROTATION_DATA_STORE, 17
X45
ROTATION_DATA_STORE, 17

X45_corr
ROTATION_DATA_STORE, 18

Y135
ROTATION_DATA_STORE, 18
Y135_corr
ROTATION_DATA_STORE, 18
Y45
ROTATION_DATA_STORE, 18
Y45_corr
ROTATION_DATA_STORE, 18