Arduino Barometer Driver

Generated by Doxygen 1.8.9.1

Wed Aug 19 2015 01:07:07

Contents

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 File Index

3.1 File List

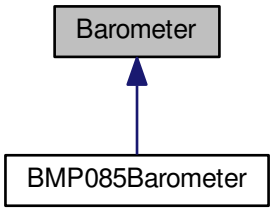
Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Barometer Class Reference

```
#include <Barometer.h>
```

Inheritance diagram for Barometer:



Public Member Functions

- virtual short getTemperature ()=0
- virtual long getPressure ()=0
- virtual float getAltitude ()=0

4.1.1 Detailed Description

Arduino - Barometer Driver.

Barometer.h

Barometer class.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 14 of file Barometer.h.

4.1.2 Member Function Documentation

4.1.2.1 virtual float Barometer::getAltitude ( ) [pure virtual]

Returns the current altitude.

Implemented in BMP085Barometer.

4.1.2.2 virtual long Barometer::getPressure ( ) [pure virtual]

Returns the current pressure.

Implemented in BMP085Barometer.

4.1.2.3 virtual short Barometer::getTemperature ( ) [pure virtual]

Returns the current temperature.

Implemented in BMP085Barometer.

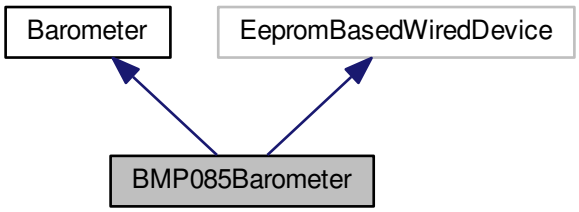
The documentation for this class was generated from the following file:

- Barometer.h

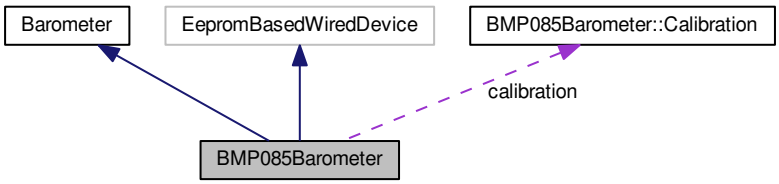
4.2 BMP085Barometer Class Reference

`#include <BMP085Barometer.h>`

Inheritance diagram for BMP085Barometer:



Collaboration diagram for BMP085Barometer:



Classes

- struct Calibration

Public Types

- enum { ULTRA_LOW_POWER = 0, STANDARD = 1, HIGH_RESOLUTION = 2, ULTRA_HIGH_RESOLU↩ TION = 3 }

Public Member Functions

- BMP085Barometer ()
- BMP085Barometer (unsigned char oversampling)
- short getTemperature ()
- long getPressure ()
- float getAltitude ()

Public Attributes

- enum BMP085Barometer:: { ... } OversamplingMode

Private Types

- enum Register {
  CALIBRATION_AC1 = 0xaa, CALIBRATION_AC2 = 0xac, CALIBRATION_AC3 = 0xae, CALIBRATION_AC4 = 0xb0,
  CALIBRATION_AC5 = 0xb2, CALIBRATION_AC6 = 0xb4, CALIBRATION_B1 = 0xb6, CALIBRATION_B2 = 0xb8,
  CALIBRATION_MB = 0xba, CALIBRATION_MC = 0xbc, CALIBRATION_MD = 0xbe, CHIP_ID = 0xd0,
  VERSION = 0xd1, SOFTWARE_RESET = 0xe0, CONTROL = 0xf4, TEMPERATURE_DATA = 0xf6,
  PRESSURE_DATA = 0xf6, READ_TEMPERATURE_COMMAND = 0x2e, READ_PRESSURE_COMMAND = 0x34 }

Private Member Functions

- void readCallibration ()
- unsigned int readUncompensatedTemperature ()
- unsigned long readUncompensatedPressure ()
- long computeB5 (unsigned int ut)

Private Attributes

- unsigned char oversampling
- Calibration calibration

Static Private Attributes

- static const float PRESSURE_AT_SEA_LEVEL = 101325 + PRESSURE_ADJUSTMENT
- static const unsigned char MAX_RETRIES_ON_READING = 10

4.2.1 Detailed Description

Definition at line 22 of file BMP085Barometer.h.

4.2.2 Member Enumeration Documentation

4.2.2.1 anonymous enum

Enumerator

ULTRA_LOW_POWER

STANDARD

HIGH_RESOLUTION

ULTRA_HIGH_RESOLUTION

Definition at line 80 of file BMP085Barometer.h.

4.2.2.2 enum BMP085Barometer::Register [private]

Enumerator

CALIBRATION_AC1

CALIBRATION_AC2

CALIBRATION_AC3

CALIBRATION_AC4

CALIBRATION_AC5
CALIBRATION_AC6
CALIBRATION_B1
CALIBRATION_B2
CALIBRATION_MB
CALIBRATION_MC
CALIBRATION_MD
CHIP_ID
VERSION
SOFTWARE_RESET
CONTROL
TEMPERATURE_DATA
PRESSURE_DATA
READ_TEMPERATURE_COMMAND
READ_PRESSURE_COMMAND

Definition at line 24 of file BMP085Barometer.h.

4.2.3 Constructor & Destructor Documentation

4.2.3.1 BMP085Barometer::BMP085Barometer ( )

Public constructor.

Definition at line 12 of file BMP085Barometer.cpp.

4.2.3.2 BMP085Barometer::BMP085Barometer (unsigned char *oversampling*)

Public constructor.

Parameters

oversampling	

Definition at line 4 of file BMP085Barometer.cpp.

4.2.4 Member Function Documentation

4.2.4.1 long BMP085Barometer::computeB5 (unsigned int *ut*) [private]

b5 is needed to measure pressure and temperature

Definition at line 120 of file BMP085Barometer.cpp.

4.2.4.2 float BMP085Barometer::getAltitude ( ) [virtual]

Altitude.

Implements Barometer.

Definition at line 61 of file BMP085Barometer.cpp.

4.2.4.3 long BMP085Barometer::getPressure ( ) [virtual]

Pressure in hPa, in steps of 1Pa (= 0.01hPa = 0.01mbar).

Implements Barometer.

Definition at line 24 of file BMP085Barometer.cpp.

4.2.4.4 short BMP085Barometer::getTemperature ( ) [virtual]

Temperature in ℃, in steps of 0.1℃.

Implements Barometer.

Definition at line 16 of file BMP085Barometer.cpp.

4.2.4.5 void BMP085Barometer::readCallibration ( ) [private]

The 176 bit EEPROM is partitioned in 11 words of 16 bit each.

These contain 11 calibration coefficients. Every sensor module has individual coefficients. Before the first calculation of temperature and pressure, the master reads out the EEPROM data. The data communication can be checked by checking that none of the words has the value 0 or 0xFFFF.

Definition at line 69 of file BMP085Barometer.cpp.

4.2.4.6 unsigned long BMP085Barometer::readUncompensatedPressure ( ) [private]

Reads uncompensated pressure value.

UP = pressure data (16 to 19 bit)

Steps:

write 0x34+(oss<<6) into reg 0xF4, wait read reg 0xF6 (MSB), 0xF7 (LSB), 0xF8 (XLSB) UP = (MSB<<16 + LSB<<8 + XLSB) >> (8-oss)

Definition at line 100 of file BMP085Barometer.cpp.

4.2.4.7 unsigned int BMP085Barometer::readUncompensatedTemperature ( ) [private]

Reads uncompensated temperature value.

UT = temperature data (16 bit)

write 0x2E into reg 0xF4, wait 4.5ms read reg 0xF6 (MSB), 0xF7 (LSB) UT = MSB << 8 + LSB

Definition at line 83 of file BMP085Barometer.cpp.

4.2.5 Member Data Documentation

4.2.5.1 Calibration BMP085Barometer::calibration [private]

Definition at line 76 of file BMP085Barometer.h.

4.2.5.2 const unsigned char BMP085Barometer::MAX_RETRIES_ON_READING = 10 [static], [private]

Definition at line 63 of file BMP085Barometer.h.

4.2.5.3 unsigned char BMP085Barometer::oversampling [private]

Definition at line 74 of file BMP085Barometer.h.

4.2.5.4 enum { ... } BMP085Barometer::OversamplingMode

4.2.5.5 const float BMP085Barometer::PRESSURE_AT_SEA_LEVEL = 101325 + PRESSURE_ADJUSTMENT [static], [private]

Definition at line 61 of file BMP085Barometer.h.

The documentation for this class was generated from the following files:

- BMP085Barometer.h
- BMP085Barometer.cpp

4.3 BMP085Barometer::Calibration Struct Reference

Public Attributes

- short ac1
- short ac2
- short ac3
- unsigned short ac4
- unsigned short ac5
- unsigned short ac6
- short b1
- short b2
- short mb
- short mc
- short md

4.3.1 Detailed Description

Definition at line 46 of file BMP085Barometer.h.

4.3.2 Member Data Documentation

4.3.2.1 short BMP085Barometer::Calibration::ac1

Definition at line 47 of file BMP085Barometer.h.

4.3.2.2 short BMP085Barometer::Calibration::ac2

Definition at line 48 of file BMP085Barometer.h.

4.3.2.3 short BMP085Barometer::Calibration::ac3

Definition at line 49 of file BMP085Barometer.h.

4.3.2.4 unsigned short BMP085Barometer::Calibration::ac4

Definition at line 50 of file BMP085Barometer.h.

4.3.2.5 unsigned short BMP085Barometer::Calibration::ac5

Definition at line 51 of file BMP085Barometer.h.

4.3.2.6 unsigned short BMP085Barometer::Calibration::ac6

Definition at line 52 of file BMP085Barometer.h.

4.3.2.7 short BMP085Barometer::Calibration::b1

Definition at line 53 of file BMP085Barometer.h.

4.3.2.8 short BMP085Barometer::Calibration::b2

Definition at line 54 of file BMP085Barometer.h.

4.3.2.9 short BMP085Barometer::Calibration::mb

Definition at line 55 of file BMP085Barometer.h.

4.3.2.10 short BMP085Barometer::Calibration::mc

Definition at line 56 of file BMP085Barometer.h.

4.3.2.11 short BMP085Barometer::Calibration::md

Definition at line 57 of file BMP085Barometer.h.

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

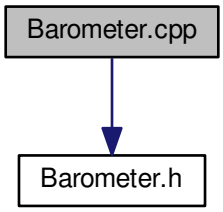
- BMP085Barometer.h

5 File Documentation

5.1 Barometer.cpp File Reference

```
#include "Barometer.h"
```

Include dependency graph for Barometer.cpp:

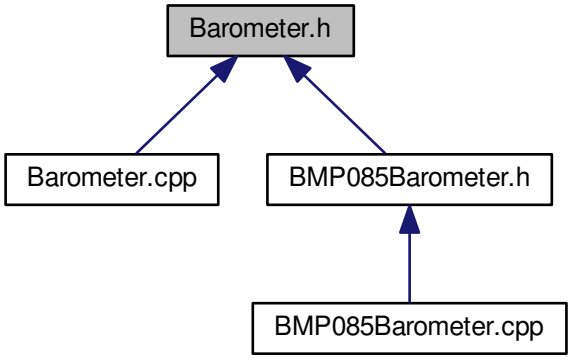


5.2 Barometer.cpp

```
00001 #include "Barometer.h"
```

5.3 Barometer.h File Reference

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



Classes

- class Barometer

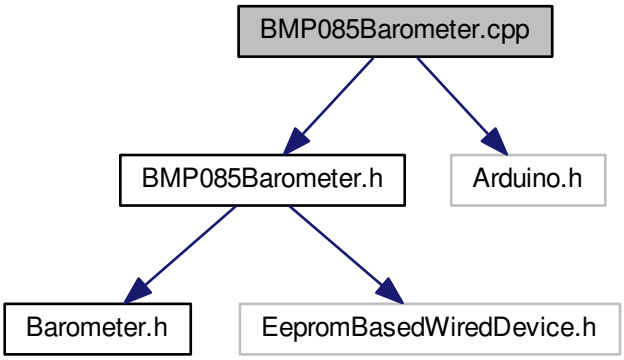
5.4 Barometer.h

```
00001
00011 #ifndef __ARDUINO_BAROMETER_DRIVER_BAROMETER_H__
00012 #define __ARDUINO_BAROMETER_DRIVER_BAROMETER_H__ 1
00013
00014 class Barometer {
00015
00016 public:
00017
00021     virtual short getTemperature() = 0;
00022
00026     virtual long getPressure() = 0;
00027
00031     virtual float getAltitude() = 0;
00032 };
00033
00034 #endif // __ARDUINO_BAROMETER_DRIVER_BAROMETER_H__
```

5.5 BMP085Barometer.cpp File Reference

```
#include "BMP085Barometer.h"
#include <Arduino.h>
```

Include dependency graph for BMP085Barometer.cpp:



5.6 BMP085Barometer.cpp

```
00001 #include "BMP085Barometer.h"
00002 #include <Arduino.h>
00003
00004 BMP085Barometer::BMP085Barometer(unsigned char oversampling)
00005         : EepromBasedWiredDevice(BMP085_ADDRESS, 0x01, EepromBasedWiredDevice::BIG_ENDIAN),
      oversampling(oversampling) {
00006
00007     // 10ms of start-up time after power-up, before first communication
00008     delay(10);
00009     readCallibration();
00010 }
00011
00012 BMP085Barometer::BMP085Barometer()
00013         : BMP085Barometer(ULTRA_LOW_POWER) {
00014 }
00015
00016 short BMP085Barometer::getTemperature() {
00017     unsigned int ut;
00018     long b5;
00019     ut = readUncompensatedTemperature();
00020     b5 = computeB5(ut);
00021     return ((b5 + 8) >> 4);
00022 }
00023
00024 long BMP085Barometer::getPressure() {
00025     unsigned int ut;
00026     long x1, x2, x3, b3, b5, b6, preasure;
00027     unsigned long up, b4, b7;
00028
00029     // b6 depends on the getTemperature be called first
00030     // to get fresher b5
00031     ut = readUncompensatedTemperature();
00032     b5 = computeB5(ut);
00033
00034     b6 = b5 - 4000;
00035     up = readUncompensatedPressure();
00036
00037     // Calculate b3
00038     x1 = (calibration.b2 * (b6 * b6) >> 12) >> 11;
00039     x2 = (calibration.ac2 * b6) >> 11;
00040     x3 = x1 + x2;
00041     b3 = (((((long) calibration.ac1) * 4 + x3) << oversampling) + 2) >> 2;
00042
00043     // Calculate b4
00044     x1 = (calibration.ac3 * b6) >> 13;
00045     x2 = (calibration.b1 * ((b6 * b6) >> 12)) >> 16;
00046     x3 = ((x1 + x2) + 2) >> 2;
00047     b4 = (calibration.ac4 * (unsigned long) (x3 + 32768)) >> 15;
00048     b7 = ((unsigned long) (up - b3) * (50000 >> oversampling));
00049     if (b7 < 0x80000000) {
00050         preasure = (b7 << 1) / b4;
```

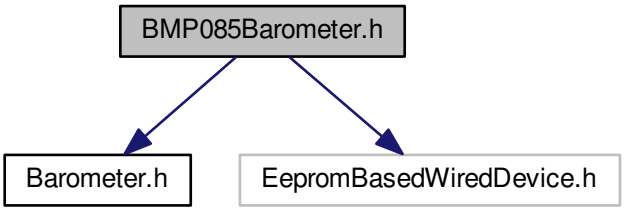
```
00051     } else {
00052         preasure = (b7 / b4) << 1;
00053     }
00054     x1 = (preasure >> 8) * (preasure >> 8);
00055     x1 = (x1 * 3038) >> 16;
00056     x2 = (-7357 * preasure) >> 16;
00057     preasure += (x1 + x2 + 3791) >> 4;
00058     return preasure;
00059 }
00060
00061 float BMP085Barometer::getAltitude() {
00062     float altitude;
00063     long pressure;
00064     pressure = getPressure();
00065     altitude = (float) 44330 * (1.0 - pow(((float) pressure /
      PRESSURE_AT_SEA_LEVEL), 0.190295));
00066     return altitude;
00067 }
00068
00069 void BMP085Barometer::readCallibration() {
00070     readBlock(CALIBRATION_AC1, (unsigned char*) &(calibration.
      ac1), 2);
00071     readBlock(CALIBRATION_AC2, (unsigned char*) &(calibration.
      ac2), 2);
00072     readBlock(CALIBRATION_AC3, (unsigned char*) &(calibration.
      ac3), 2);
00073     readBlock(CALIBRATION_AC4, (unsigned char*) &(calibration.
      ac4), 2);
00074     readBlock(CALIBRATION_AC5, (unsigned char*) &(calibration.
      ac5), 2);
00075     readBlock(CALIBRATION_AC6, (unsigned char*) &(calibration.
      ac6), 2);
00076     readBlock(CALIBRATION_B1, (unsigned char*) &(calibration.
      b1), 2);
00077     readBlock(CALIBRATION_B2, (unsigned char*) &(calibration.
      b2), 2);
00078     readBlock(CALIBRATION_MB, (unsigned char*) &(calibration.
      mb), 2);
00079     readBlock(CALIBRATION_MC, (unsigned char*) &(calibration.
      mc), 2);
00080     readBlock(CALIBRATION_MD, (unsigned char*) &(calibration.
      md), 2);
00081 }
00082
00083 unsigned int BMP085Barometer::readUncompensatedTemperature() {
00084
00085     unsigned int temperature;
00086
00087     // Write 0x2e into register 0xf4
00088     // This requests a temperature reading
00089     unsigned char temperatureRequest[] = { READ_TEMPERATURE_COMMAND };
00090     writeBlock(CONTROL, temperatureRequest, 0x01);
00091
00092     // Wait at least 4.5ms
00093     delay(5);
00094
00095     // Read two bytes from registers 0xf6 and 0xf7
00096     readBlock(TEMPERATURE_DATA, (unsigned char *) &temperature, 0x02);
00097     return temperature;
00098 }
00099
00100 unsigned long BMP085Barometer::readUncompensatedPressure() {
00101
00102     unsigned long pressure = 0;
00103
00104     // Write 0x34+(OSS<<6) into register 0xf4
00105     // Request a pressure reading w/ over-sampling setting
00106     unsigned char pressureRequest[] = { (unsigned char) (READ_PRESSURE_COMMAND + (
      oversampling << 0x06)) };
00107     writeBlock(CONTROL, pressureRequest, 0x01);
00108
00109     // Wait for conversion, delay time dependent on OSS
00110     delay(2 + (3 << oversampling));
00111
00112     // Read register 0xf6 (MSB), 0xf7 (LSB), and 0xf8 (XLSB)
00113     readBlock(PRESSURE_DATA, (unsigned char *) &pressure, 0x03);
00114
00115     // Adjust XLSB based on the over-sampling
00116     pressure >>= (8 - oversampling);
00117     return pressure;
00118 }
00119
00120 long BMP085Barometer::computeB5(unsigned int ut) {
00121     long x1, x2;
00122     x1 = (((long) ut - (long) calibration.ac6) * (long)
      calibration.ac5) >> 15;
00123     x2 = ((long) calibration.mc << 11) / (x1 + calibration.
```

```
      md);
00124     return x1 + x2;
00125 }
```

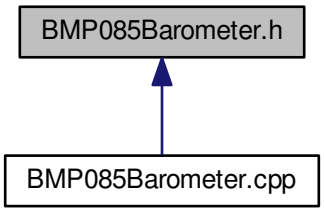
5.7 BMP085Barometer.h File Reference

```
#include <Barometer.h>
#include <EepromBasedWiredDevice.h>
```

Include dependency graph for BMP085Barometer.h:



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



Classes

- class BMP085Barometer
- struct BMP085Barometer::Calibration

Macros

- #define BMP085_ADDRESS 0x77
- #define PRESSURE_ADJUSTMENT 0

5.7.1 Macro Definition Documentation

5.7.1.1 #define BMP085_ADDRESS 0x77

Arduino - Barometer Driver.

BMP085Barometer.h

BMP085Barometer class.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 17 of file BMP085Barometer.h.

5.7.1.2 #define PRESSURE_ADJUSTMENT 0

Definition at line 20 of file BMP085Barometer.h.

5.8 BMP085Barometer.h

```
00001
00011 #ifndef __ARDUINO_BAROMETER_DRIVER_BMP085_BAROMETER_H__
00012 #define __ARDUINO_BAROMETER_DRIVER_BMP085_BAROMETER_H__ 1
00013
00014 #include <Barometer.h>
00015 #include <EepromBasedWiredDevice.h>
00016
00017 #define BMP085_ADDRESS          0x77
00018
00019 // https://www.raspberrypi.org/forums/viewtopic.php?t=64618&p=476195
00020 #define PRESSURE_ADJUSTMENT     0
00021
00022 class BMP085Barometer: public Barometer, public EepromBasedWiredDevice {
00023
00024     enum Register {
00025         CALIBRATION_AC1 = 0xaa,
00026         CALIBRATION_AC2 = 0xac,
00027         CALIBRATION_AC3 = 0xae,
00028         CALIBRATION_AC4 = 0xb0,
00029         CALIBRATION_AC5 = 0xb2,
00030         CALIBRATION_AC6 = 0xb4,
00031         CALIBRATION_B1 = 0xb6,
00032         CALIBRATION_B2 = 0xb8,
00033         CALIBRATION_MB = 0xba,
00034         CALIBRATION_MC = 0xbc,
00035         CALIBRATION_MD = 0xbe,
00036         CHIP_ID = 0xd0,
00037         VERSION = 0xd1,
00038         SOFTWARE_RESET = 0xe0,
00039         CONTROL = 0xf4,
00040         TEMPERATURE_DATA = 0xf6,
00041         PRESSURE_DATA = 0xf6,
00042         READ_TEMPERATURE_COMMAND = 0x2e,
00043         READ_PRESSURE_COMMAND = 0x34
00044     };
00045
00046     struct Calibration {
00047         short ac1;
00048         short ac2;
00049         short ac3;
00050         unsigned short ac4;
00051         unsigned short ac5;
00052         unsigned short ac6;
00053         short b1;
00054         short b2;
00055         short mb;
00056         short mc;
00057         short md;
00058     };
00059
00060     // Pressure at sea level (Pa)
00061     const static float PRESSURE_AT_SEA_LEVEL = 101325 +
      PRESSURE_ADJUSTMENT;
00062
00063     const static unsigned char MAX_RETRIES_ON_READING = 10;
00064
00065     /*
00066      * OSS selects which mode the BMP085 operates in, and can be set to either 0, 1, 2, or 3.
00067      * OSS determines how many samples the BMP085 will take before it sends over its uncompensated
```

```
00068      * pressure reading. With OSS set to 0, the BMP085 will consume the least current.
00069      * Setting OSS to 3 increases resolution, as it samples pressure eight times before
00070      * producing a reading, this comes at a cost of more power usage. If you want to change OSS,
00071      * just set it accordingly at the top of the program. Try changing OSS to 3,
00072      * does the data become more stable?
00073      */
00074     unsigned char oversampling;
00075
00076     Calibration calibration;
00077
00078 public:
00079
00080     enum {
00081         ULTRA_LOW_POWER = 0,
00082         STANDARD = 1,
00083         HIGH_RESOLUTION = 2,
00084         ULTRA_HIGH_RESOLUTION = 3
00085     } OversamplingMode;
00086
00090     BMP085Barometer();
00091
00097     BMP085Barometer(unsigned char oversampling);
00098
00102     short getTemperature();
00103
00107     long getPressure();
00108
00112     float getAltitude();
00113
00114 private:
00115
00122     void readCallibration();
00123
00133     unsigned int readUncompensatedTemperature();
00134
00146     unsigned long readUncompensatedPressure();
00147
00151     long computeB5(unsigned int ut);
00152 };
00153
00154 #endif // __ARDUINO_BAROMETER_DRIVER_BAROMETER_H__
```

Index