WeatherSensors Reference Manual

Generated by Doxygen 1.8.11

Sat Nov 12 2016 19:56:28

Contents

Chapter 1

WeatherSensors

Library for the Weather Sensors of the Sensors BoosterPack

Developed with embedXcode+

Author

Rei Vilo
http://embeddedcomputing.weebly.com

Date

nov. 12, 2016 19:37

Version

102

See also

ReadMe.txt for references

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Sensor_BME280 Class Reference

Class for sensor BME280.

```
#include <Sensor_BME280.h>
```

Public Member Functions

- Sensor_BME280 ()

    Constructor.
- void begin ()

    Initialisation.
- String WhoAmI ()

    Who am I?
- uint8_t get ()

    Acquire data.
- float temperature ()

    Return temperature.
- float humidity ()

    Return relative humidity.
- float pressure ()

    Return pressure, relative to current altitude.
- float absolutePressure (float altitudeMeters=50.0)

    Return absolute pressure, equivalent at sea level.
- float altitude (float seaLevelPressure=1013.250)

    Return altitude based on pressure.
- float altitude (float referencePressure=1013.250, float referenceAltitude=0.0)

    Return altitude based on reference pressure and altitude.
- void setPowerMode (uint8_t mode=LOW)

    Set power mode.

4.1.1 Detailed Description

Class for sensor BME280.

Combined humidity and pressure sensor

See also

`http://www.bosch-sensortec.com/de/homepage/products_3/environmental_↩
sensors_1/bme280/bme280_1`

4.1.2 Member Function Documentation

4.1.2.1 float Sensor_BME280::absolutePressure (float *altitudeMeters* = `50.0`)

Return absolute pressure, equivalent at sea level.

Parameters

altitudeMeters	current altitude, in meter

Returns

absolute pressure at sea level, in hPa

Note

Use conversion() for another unit

4.1.2.2 float Sensor_BME280::altitude (float *seaLevelPressure* = `1013.250`)

Return altitude based on pressure.

Parameters

seaLevelPressure	pressure at sea level, in hPa

Returns

altitude, in meter

Note

Use conversion() for another unit

4.1.2.3 float Sensor_BME280::altitude (float *referencePressure* = `1013.250`, float *referenceAltitude* = `0.0`)

Return altitude based on reference pressure and altitude.

Parameters

referencePressure	reference pressure, in hPa
referenceAltitude	reference altitude, in meter

Returns

altitude in meter

Note

The reference is a measure of the pressure at a known altitude.
Use conversion() for another unit

4.1.2.4 void Sensor_BME280::begin ( )

Initialisation.

Parameters

number	of reads

Note

See Table # of the BME280 data-sheet

4.1.2.5 uint8_t Sensor_BME280::get ( )

Acquire data.

Returns

0 if success, error code otherwise

```
do
{
    delay(100);
    result = myBME280.get();
    count++;
}
while ((result > 0) and (count < 8));
```

4.1.2.6 float Sensor_BME280::humidity ( )

Return relative humidity.

Returns

relative humidity, in %

4.1.2.7 float Sensor_BME280::pressure ( )

Return pressure, relative to current altitude.

Returns

pressure, in hPa

Note

Use conversion() for another unit

4.1.2.8 void Sensor_BME280::setPowerMode (uint8_t *mode* = `LOW`)

Set power mode.

Parameters

mode	default=LOW=sleep, HIGH=activated

4.1.2.9 float Sensor_BME280::temperature ( )

Return temperature.

Returns

temperature, in °K

Note

Use conversion() for another unit

4.1.2.10 String Sensor_BME280::WhoAmI ( )

Who am I?

Returns

Who am I? string

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

- Sensor_BME280.h
- Sensor_BME280.cpp

4.2 Sensor_OPT3001 Class Reference

Class for sensor OPT3001.

```
#include <Sensor_OPT3001.h>
```

Public Member Functions

- Sensor_OPT3001 ()

  Constructor.
- void begin (uint16_t configuration=0xc410, uint8_t interruptPin=11)

  Initialisation.
- String WhoAmI ()

  Who Am I?
- void get ()

  Acquisition.
- float light ()

  Measure.
- void setPowerMode (uint8_t mode=LOW)

  Manage power.

4.2.1 Detailed Description

Class for sensor OPT3001.

Digital Ambient Light Sensor (ALS) with High Precision Human Eye Response

See also

`http://www.ti.com/product/OPT3001`

4.2.2 Member Function Documentation

4.2.2.1 void Sensor_OPT3001::begin (uint16_t *configuration* = `0xc410`, uint8_t *interruptPin* = `11`)

Initialisation.

Parameters

configuration	default = 100 ms, OPT3001_100_MS or OPT3001_800_MS
interruptPin	default = 11

4.2.2.2 float Sensor_OPT3001::light ( )

Measure.

Returns

light in lux

4.2.2.3 void Sensor_OPT3001::setPowerMode (uint8_t *mode* = `LOW`)

Manage power.

Parameters

mode	LOW=default=off, HIGH=on

4.2.2.4 String Sensor_OPT3001::WhoAmI ( )

Who Am I?

Returns

name of the sensor, string

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

- Sensor_OPT3001.h
- Sensor_OPT3001.cpp

4.3 Sensor_TMP007 Class Reference

Class for sensor TMP007.

```
#include <Sensor_TMP007.h>
```

Public Member Functions

- Sensor_TMP007 ()

  Constructor.
- void begin (uint16_t totalSamples=0x0400)

  Initialisation.
- String WhoAmI ()

  Who Am I?
- void get ()

  Acquisition.
- float internalTemperature ()

  Measure.
- float externalTemperature ()

  Measure.
- void setPowerMode (uint8_t mode=LOW)

  Manage power.

4.3.1 Detailed Description

Class for sensor TMP007.

Infrared Thermopile Contactless Temperature Sensor with Integrated Math Engine

See also

`http://www.ti.com/product/TMP007`

4.3.2 Member Function Documentation

4.3.2.1 void Sensor_TMP007::begin (uint16_t *totalSamples* = `0x0400`)

Initialisation.

Parameters

totalSamples	default = 4 samples, use pre-defined constants

4.3.2.2 float Sensor_TMP007::externalTemperature ( )

Measure.

Returns

External temperature in °K

4.3.2.3 float Sensor_TMP007::internalTemperature ( )

Measure.

Returns

Internal temperature in °K

4.3.2.4 void Sensor_TMP007::setPowerMode (uint8_t *mode* = `LOW`)

Manage power.

Parameters

mode	LOW=default=off, HIGH=on

4.3.2.5 String Sensor_TMP007::WhoAmI ( )

Who Am I?

Returns

name of the sensor, string

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

- Sensor_TMP007.h
- Sensor_TMP007.cpp

4.4 unit_conversion_s Struct Reference

Units.

```
#include <Sensor_Units.h>
```

Public Attributes

- float gain

  gain
- float base

  base
- char symbol [4]

  symbol

4.4.1 Detailed Description

Units.

A unit contains gain and base for conversion based on the SI reference unit.

Note

For each set of units, all units are defined the SI reference unit

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

- Sensor_Units.h

Chapter 5

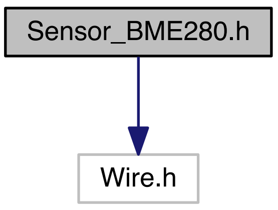
File Documentation

5.1 Sensor_BME280.h File Reference

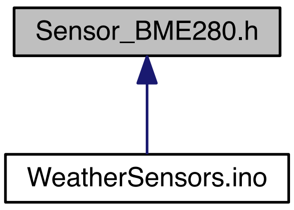
Library header for BME280 sensor.

```
#include "Wire.h"
```

Include dependency graph for Sensor_BME280.h:



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



Classes

- class Sensor_BME280

  Class for sensor BME280.

Macros

- #define Sensor_BME280_RELEASE 102

  Release.
- #define BM280_SUCCESS 0

  success
- #define BM280_ERROR 1

  error

5.1.1 Detailed Description

Library header for BME280 sensor.

BME280 Combined humidity and pressure sensor

Project SensorsBoosterPack
Developed with `embedXcode+`

Author

Rei Vilo
`http://embeddedcomputing.weebly.com`

Date

20/08/2015 13:43

Version

102

Copyright

(c) Rei Vilo, 2015-2016
CC = BY SA NC

See also

ReadMe.txt for references

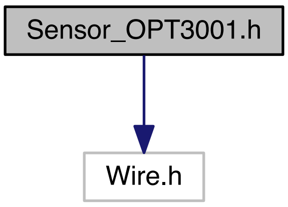
- Pressure Altimetry using the MPL3115A2
  `http://cache.freescale.com/files/sensors/doc/app_note/AN4528.pdf`

5.2 Sensor_OPT3001.h File Reference

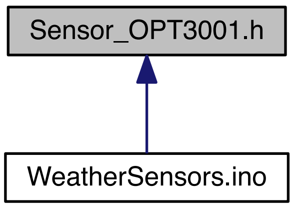
Library header for OPT3001 sensor.

`#include "Wire.h"`

Include dependency graph for Sensor_OPT3001.h:



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



Classes

- class Sensor_OPT3001

  Class for sensor OPT3001.

Macros

- #define Sensor_OPT3001_RELEASE 102

  Release.

- #define OPT3001_100_MS 0xc410

  Conversion modes.
- #define OPT3001_800_MS 0xcc10

  continuous
- #define OPT3001_INTERRUPT_PIN 11

  Conversion modes.

5.2.1 Detailed Description

Library header for OPT3001 sensor.

OPT3001 Digital Ambient Light Sensor (ALS) with High Precision Human Eye Response

Project SensorsBoosterPack
Developed with `embedXcode+`

Author

a0273900 for initial C-library
Rei Vilo for Energia adapted C++-library
`http://embeddedcomputing.weebly.com`

Date

20/08/2015 13:42

Version

102

Copyright

(c) Rei Vilo, 2015-2016
CC = BY SA NC

See also

ReadMe.txt for references

5.2.2 Macro Definition Documentation

5.2.2.1 #define OPT3001_100_MS 0xc410

Conversion modes.

continous

5.2.2.2 #define OPT3001_INTERRUPT_PIN 11

Conversion modes.

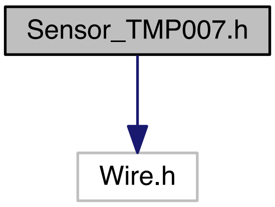
continous

5.3 Sensor_TMP007.h File Reference

Library header for TMP007 sensor.

```
#include "Wire.h"
```

Include dependency graph for Sensor_TMP007.h:



Classes

- class Sensor_TMP007

    Class for sensor TMP007.

Macros

- #define Sensor_TMP007_cpp 102

    Release.

- #define TMP007_ONE_SAMPLE 0x0000

    TMP007 constants.
- #define TMP007_TWO_SAMPLES 0x0200

    TMP007 constants.
- #define TMP007_FOUR_SAMPLES 0x0400

    TMP007 constants.
- #define TMP007_EIGHT_SAMPLES 0x0600

    TMP007 constants.
- #define TMP007_SIXTEEN_SAMPLES 0x0800

    TMP007 constants.
- #define TMP007_ONE_SAMPLE_LOW_POWER 0x0A00

    TMP007 constants.
- #define TMP007_TWO_SAMPLES_LOW_POWER 0x0C00

    TMP007 constants.
- #define TMP007_FOUR_SAMPLES_LOW_POWER 0x0E00

    TMP007 constants.

5.3.1 Detailed Description

Library header for TMP007 sensor.

TMP007 Infrared Thermopile Contactless Temperature Sensor with Integrated Math Engine

Project SensorsBoosterPack
Developed with `embedXcode+`

Author

a0273900 Rei Vilo
`http://embeddedcomputing.weebly.com`

Date

20/08/2015 13:42

Version

102

Copyright

(c) Rei Vilo, 2015-2016
CC = BY SA NC

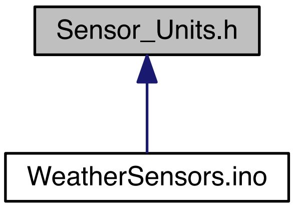
See also

ReadMe.txt for references

5.4 Sensor_Units.h File Reference

Library header.

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



Classes

- struct unit_conversion_s

  Units.

Macros

- #define Sensor_Units_RELEASE 102

  Release.

Functions

- template<typename myType >
  float conversion (float value, myType unitFrom, myType unitTo)

  Conversion utility.
- template<typename myType >
  String symbolString (myType unit)

  Unit symbol as String.
- template<typename myType >
  char ∗ symbolChar (myType unit)

  Unit symbol as char∗.

- typedef unit_conversion_s temperature_unit_t

  Temperature units.
- const temperature_unit_t KELVIN = { 1, 0, "°K"}

  °K degree kelvin, SI reference.
- const temperature_unit_t CELSIUS = { 1, -273.15, "°C"}

  °C degree celsius.
- const temperature_unit_t FAHRENHEIT = { 1.8, -459.67, "°F"}

- typedef unit_conversion_s pressure_unit_t

  Pressure units.
- const pressure_unit_t PASCAL = { 1, 0, "Pa"}

  Pa pascal, SI reference.
- const pressure_unit_t HECTOPASCAL = { 1e-2, 0, "hPa"}

  hPa hecto pascal, SI reference
- const pressure_unit_t BAR = { 1e-5, 0, "bar"}

  bar
- const pressure_unit_t ATMOSPHERE = { 1.0 / 101325.0, 0, "atm"}

  atmosphere
- const pressure_unit_t PSI = { 0.014503773801, 0, "atm"}

- typedef unit_conversion_s altitude_unit_t

  Altitude units.
- const altitude_unit_t METRE = { 1, 0, "m"}

  m metre, SI reference
- const altitude_unit_t FOOT = { 0.3048, 0, "ft"}

- typedef unit_conversion_s light_unit_t

  Light units.
- const light_unit_t LUX = { 1, 0, "lux"}

5.4.1 Detailed Description

Library header.

Units conversion for sensors

Project SensorsBoosterPack
Developed with `embedXcode+`

Author

Rei Vilo
`http://embeddedcomputing.weebly.com`

Date

Aug 20, 2015 19:03

Version

102

Copyright

(c) Rei Vilo, 2015-2016
CC = SA BY NC

See also

ReadMe.txt for references

5.4.2 Function Documentation

5.4.2.1 template<typename myType > float conversion (float *value*, myType *unitFrom*, myType *unitTo*)

Conversion utility.

Parameters

value	input value to be converted, float
unitFrom	unit of the input value to be converted
unitTo	unit for the output converted value

Returns

output converted value, float

5.4.2.2 template< typename myType > char* symbolChar (myType *unit*)

Unit symbol as char*.

Parameters

unit	unit constant

Returns

symbol as char*

5.4.2.3 template< typename myType > String symbolString (myType *unit*)

Unit symbol as String.

Parameters

unit	unit constant

Returns

symbol as String

5.4.3 Variable Documentation

5.4.3.1 const temperature_unit_t FAHRENHEIT = { 1.8, -459.67, "°F"}

°F degree fahrenheit

5.4.3.2 const altitude_unit_t FOOT = { 0.3048, 0, "ft"}

ft foot

5.4.3.3 const light_unit_t LUX = { 1, 0, "lux"}

lux, SI reference

5.4.3.4 const pressure_unit_t PSI = { 0.014503773801, 0, "atm"}

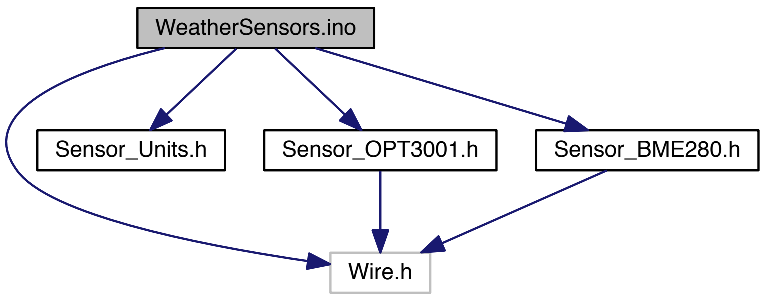
0.014503773801 pound force/square inch

5.5 WeatherSensors.ino File Reference

Main sketch.

```
#include "Wire.h"
#include "Sensor_Units.h"
#include "Sensor_OPT3001.h"
#include "Sensor_BME280.h"
```

Include dependency graph for WeatherSensors.ino:



Macros

- #define **USE_TMP007** 0
- #define **USE_OPT3001** 1
- #define **USE_BME280** 1

Functions

- void **setup** ()
- void **loop** ()

Variables

- Sensor_OPT3001 **myOPT3001**
- float **OPT3001_light**
- Sensor_BME280 **myBME280**
- float **BME280_pressure**
- float **BME280_temperature**
- float **BME280_humidity**
- const uint32_t **period_ms** = 10000

5.5.1 Detailed Description

Main sketch.

Demo
Developed with `embedXcode+`

Author

Rei Vilo
`http://embeddedcomputing.weebly.com`

Date

nov. 12, 2016 19:37

Version

102

Copyright

(c) Rei Vilo, 2015-2016
GNU General Public Licence

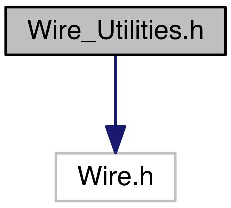
See also

ReadMe.txt for references

5.6 Wire_Utilities.h File Reference

Library header.

```
#include "Wire.h"
```
Include dependency graph for Wire_Utilities.h:



Macros

- #define **Wire_Utilities_RELEASE** 102

Functions

- void writeRegister8 (uint8_t device, uint8_t command, uint8_t data8)
  Write 1 byte.
- void writeRegister16 (uint8_t device, uint8_t command, uint16_t data16, uint8_t mode=MSBFIRST)
  Write 2 bytes.
- uint8_t readRegister8 (uint8_t device, uint8_t command)
  Read 1 byte.
- uint16_t readRegister16 (uint8_t device, uint8_t command, uint8_t mode=MSBFIRST)
  Read 2 bytes.

5.6.1 Detailed Description

Library header.

Utilities for 8- and 16-bit read and write operations

Project SensorsBoosterPack
Developed with `embedXcode+`

Author

Rei Vilo
`http://embeddedcomputing.weebly.com`

Date

20/08/2015 18:02

Version

102

Copyright

(c) Rei Vilo, 2015-2016
CC = BY SA NC

See also

ReadMe.txt for references

5.6.2 Function Documentation

5.6.2.1 uint16_t readRegister16 (uint8_t *device,* uint8_t *command,* uint8_t *mode* = `MSBFIRST`)

Read 2 bytes.

Parameters

device	I2C address, 7-bit coded
command	command or register, 8-bit
mode	default=MSBFIRST, other option=LSBFIRST

Returns

data16 value, 16-bit

Note

∗ with MSBFIRST, data16[15..8] read from command, data16[7..Ø] from command + 1
∗ with LSBFIRST, data16[7..Ø] read from command, data16[15..8] from command + 1

5.6.2.2 uint8_t readRegister8 (uint8_t *device,* uint8_t *command*)

Read 1 byte.

Parameters

device	I2C address, 7-bit coded
command	command, 8-bit

Returns

data8 value, 8-bit

5.6.2.3 void writeRegister16 (uint8_t *device,* uint8_t *command,* uint16_t *data16,* uint8_t *mode* = `MSBFIRST`)

Write 2 bytes.

Parameters

device	I2C address, 7-bit coded
command	command or register, 8-bit
data16	value, 16-bit
mode	default=MSBFIRST, other option=LSBFIRST

Note

∗ with MSBFIRST, data16[15..8] written to command, data16[7..Ø] to command + 1
∗ with LSBFIRST, data16[7..Ø] written to command, data16[15..8] to command + 1

5.6.2.4 void writeRegister8 (uint8_t *device,* uint8_t *command,* uint8_t *data8*)

Write 1 byte.

Parameters

device	I2C address, 7-bit coded
command	command or register, 8-bit
data8	value, 8-bit

Index