Weather Sensors Reference Manual

Generated by Doxygen 1.8.16

Fri Dec 27 2019 15:31:26

Chapter 1

WeatherSensors

Library for different LaunchPad and BoosgerPack weather sensors

Developed with `embedXcode+`

Author

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

Date

12 Nov 2016

Version

103

Copyright

See also

ReadMe.txt for references

Board	Infra-Red	Temperature	Humidity	Pressure	Light

Note

List of sensors and I^2C addresses

Board	Infra-Red	Temperature	Humidity	Pressure	Light
Sensors BoosterPack	TMP007 0x40				OPT3001 0x47
		BME280 0x77	BME280 0x77	BME280 0x77	
BASS Booster↩ Pack	TMP116 0x48				OPT3001 0x44
		HDC1000 0x40	HDC1000 0x40		
CC1350 SensorTag	TMP007 0x44				OPT3001 0x45
		HDC1000 0x43	HDC1000 0x43		
		BMP280 0x77		BMP280 0x77	
CC1352 LP↩ STK		HDC2080 0x41	HDC2080 0x41		OPT3001 0x44

- **BASS** = Building Automation System Sensors
- **LPSTK** = LaunchPad SensorTag Kit

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 (uint8_t address=0x77)

  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 temperature, humidity and pressure sensor

See also

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

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Sensor_BME280()

```
Sensor_BME280::Sensor_BME280 (
            uint8_t address = 0x77 )
```

Constructor.

Parameters

address	default = BME280_SLAVE_ADDRESS

4.1.3 Member Function Documentation

4.1.3.1 absolutePressure()

```
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

4.1.3.2 altitude() [1/2]

```
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

4.1.3.3 altitude() [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

4.1.3.4 begin()

```
void Sensor_BME280::begin ( )
```

Initialisation.

Parameters

number	of reads

Note

See Table # of the BME280 data-sheet

xxxxx.011 Default = 0x00 _____.001 Humidity oversampling x1

4.1.3.5 get()

```
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));
```

4.1.3.6 humidity()

```
float Sensor_BME280::humidity ( )
```

Return relative humidity.

Returns

relative humidity, in %

4.1.3.7 pressure()

```
float Sensor_BME280::pressure ( )
```

Return pressure, relative to current altitude.

Returns

pressure, in hPa

Note

Use conversion() for another unit

4.1.3.8 setPowerMode()

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

Set power mode.

Parameters

mode	default=LOW=sleep, HIGH=activated

4.1.3.9 temperature()

```
float Sensor_BME280::temperature ( )
```

Return temperature.

Returns

temperature, in °K

Note

Use conversion() for another unit

4.1.3.10 WhoAmI()

```
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_BMP280 Class Reference

Class for sensor BMP280.

```
#include <Sensor_BMP280.h>
```

Public Member Functions

- Sensor_BMP280 (uint8_t address=0x77)

  Constructor.
- void begin ()

  Initialisation.
- String WhoAmI ()

  Who am I?
- uint8_t get ()

  Acquire data.
- float temperature ()

  Return temperature.
- 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.

4.2.1 Detailed Description

Class for sensor BMP280.

Combined humidity and pressure sensor

See also

http://www.bosch-sensortec.com/en/bst/products/all_products/bmp280

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Sensor_BMP280()

```
Sensor_BMP280::Sensor_BMP280 (
            uint8_t address = 0x77 )
```

Constructor.

Parameters

address	I2C slave address

Note

Valid addresses are 0x76..0x78

4.2.3 Member Function Documentation

4.2.3.1 absolutePressure()

```
float Sensor_BMP280::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

4.2.3.2 altitude() [1/2]

```
float Sensor_BMP280::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

4.2.3.3 altitude() [2/2]

```
float Sensor_BMP280::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

4.2.3.4 begin()

```
void Sensor_BMP280::begin ( )
```

Initialisation.

Parameters

number	of reads

Note

See Table # of the BMP280 data-sheet

4.2.3.5 get()

```
uint8_t Sensor_BMP280::get ( )
```

Acquire data.

Returns

0 if success, error code otherwise

```
do
{
    delay(100);
    result = myBMP280.get();
    count++;
}
while ((result > 0) and (count < 8));
```

4.2.3.6 pressure()

```
float Sensor_BMP280::pressure ( )
```

Return pressure, relative to current altitude.

Returns

pressure, in hPa

Note

Use conversion() for another unit

4.2.3.7 setPowerMode()

```
void Sensor_BMP280::setPowerMode (
             uint8_t mode = LOW )
```

Set power mode.

Parameters

mode	default=LOW=sleep, HIGH=activated

4.2.3.8 temperature()

```
float Sensor_BMP280::temperature ( )
```

Return temperature.

Returns

temperature, in °K

Note

Use conversion() for another unit

4.2.3.9 WhoAmI()

```
String Sensor_BMP280::WhoAmI ( )
```

Who am I?

Returns

Who am I? string

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

- Sensor_BMP280.h
- Sensor_BMP280.cpp

4.3 Sensor_HDC1000 Class Reference

Class for sensor HDC1000.

```
#include <Sensor_HDC1000.h>
```

Public Member Functions

- Sensor_HDC1000 (uint8_t address=0x43)

  Constructor.
- void begin (uint8_t configuration=0b00010101)

  Initialisation.
- void get ()

  Acquisition.
- double temperature ()

  Measure.
- double humidity ()

  Measure.
- void setPowerMode (uint8_t mode=LOW)

  Manage power.

4.3.1 Detailed Description

Class for sensor HDC1000.

Temperature and Humidity Sensor

See also

`http://www.ti.com/product/HDC1000`

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Sensor_HDC1000()

```
Sensor_HDC1000::Sensor_HDC1000 (
            uint8_t address = 0x43 )
```

Constructor.

Parameters

address	I2C slave address

Note

Valid addresses are 0x40..0x43

4.3.3 Member Function Documentation

4.3.3.1 begin()

```
void Sensor_HDC1000::begin (
             uint8_t configuration = 0b00010101 )
```

Initialisation.

Parameters

configuration	default=HDC1000_SETTINGS

4.3.3.2 humidity()

```
double Sensor_HDC1000::humidity ( )
```

Measure.

Returns

Relative humidity in %

4.3.3.3 setPowerMode()

```
void Sensor_HDC1000::setPowerMode (
             uint8_t mode = LOW )
```

Manage power.

Parameters

mode	LOW=default=off, HIGH=on

4.3.3.4 temperature()

```
double Sensor_HDC1000::temperature ( )
```

Measure.

Returns

Temperature in °K

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

- Sensor_HDC1000.h
- Sensor_HDC1000.cpp

4.4 Sensor_HDC2080 Class Reference

Class for sensor HDC2080.

```
#include <Sensor_HDC2080.h>
```

Public Member Functions

- Sensor_HDC2080 (uint8_t address=0x41)

    Constructor.
- void begin (uint8_t configuration=0b00000000, uint8_t measure=0b00000000)

    Initialisation.
- void get ()

    Acquisition.
- double temperature ()

    Measure.
- double humidity ()

    Measure.
- void **enableHeater** (void)
- void **disableHeater** (void)
- void **setLowTemp** (float temp)
- void **setHighTemp** (float temp)
- void **setHighHumidity** (float humid)
- void **setLowHumidity** (float humid)
- float **readLowHumidityThreshold** (void)
- float **readHighHumidityThreshold** (void)
- float **readLowTempThreshold** (void)
- float **readHighTempThreshold** (void)
- void **triggerMeasurement** (void)
- void **reset** (void)
- void **enableInterrupt** (void)
- void **disableInterrupt** (void)
- uint8_t **readInterruptStatus** (void)
- void **clearMaxTemp** (void)
- void **clearMaxHumidity** (void)
- float **readMaxTemp** (void)
- float **readMaxHumidity** (void)
- void **enableThresholdInterrupt** (void)
- void **disableThresholdInterrupt** (void)
- void **enableDRDYInterrupt** (void)
- void **disableDRDYInterrupt** (void)
- void **setTempRes** (int resolution)
- void **setHumidRes** (int resolution)
- void **setMeasurementMode** (int mode)
- void **setRate** (int rate)
- void **setInterruptPolarity** (int polarity)
- void **setInterruptMode** (int polarity)

4.4.1 Detailed Description

Class for sensor HDC2080.

Temperature and Humidity Sensor

See also

http://www.ti.com/product/HDC2080

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Sensor_HDC2080()

```
Sensor_HDC2080::Sensor_HDC2080 (
             uint8_t address = 0x41 )
```

Constructor.

Parameters

address	I2C slave address

Note

Valid addresses are 0x40..0x41, default=HDC2080_I2C_ADDRESS

4.4.3 Member Function Documentation

4.4.3.1 begin()

```
void Sensor_HDC2080::begin (
             uint8_t configuration = 0b00000000,
             uint8_t measure = 0b00000000 )
```

Initialisation.

Parameters

configuration	default=HDC2080_DEFAULT_SETTINGS
measure	default=HDC2080_MEASURE_SETTINGS

4.4.3.2 humidity()

```
double Sensor_HDC2080::humidity ( )
```

Measure.

Returns

Relative humidity in %

4.4.3.3 temperature()

```
double Sensor_HDC2080::temperature (
            void  )
```

Measure.

Returns

Temperature in ℃

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

- Sensor_HDC2080.h
- Sensor_HDC2080.cpp

4.5 Sensor_OPT3001 Class Reference

Class for sensor OPT3001.

`#include <Sensor_OPT3001.h>`

Public Member Functions

- Sensor_OPT3001 (uint8_t address=0x47)
  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.

4.5.1 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

4.5.2 Member Function Documentation

4.5.2.1 begin()

```
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

4.5.2.2 light()

```
float Sensor_OPT3001::light ( )
```

Measure.

Returns

light in lux

4.5.2.3 setPowerMode()

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

Manage power.

Parameters

mode	LOW=default=off, HIGH=on

4.5.2.4 WhoAmI()

```
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

4.6 Sensor_TMP007 Class Reference

Class for sensor TMP007.

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

Public Member Functions

- Sensor_TMP007 (uint8_t address=0x40)

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

  Initialisation.
- String WhoAmI ()

  Who Am I?
- void get ()

  Acquisition.
- float internal ()

  Measure.
- float external ()

  Measure.
- void setPowerMode (uint8_t mode=LOW)

  Manage power.

4.6.1 Detailed Description

Class for sensor TMP007.

Infrared Thermopile Contactless Temperature Sensor with Integrated Math Engine

See also

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

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Sensor_TMP007()

```
Sensor_TMP007::Sensor_TMP007 (
             uint8_t address = 0x40 )
```

Constructor.

Parameters

address	default = 0x40

4.6.3 Member Function Documentation

4.6.3.1 begin()

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

Initialisation.

Parameters

totalSamples	default = 4 samples, use pre-defined constants

4.6.3.2 external()

```
float Sensor_TMP007::external ( )
```

Measure.

Returns

External temperature in °K

4.6.3.3 internal()

```
float Sensor_TMP007::internal ( )
```

Measure.

Returns

Internal temperature in °K

4.6.3.4 setPowerMode()

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

Manage power.

Parameters

mode	LOW=default=off, HIGH=on

4.6.3.5 WhoAmI()

```
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

4.7 unit_conversion_s Struct Reference

Units.

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

Public Attributes

- float gain
  gain
- float base
  base
- char symbol [4]
  symbol

4.7.1 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

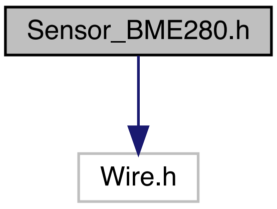
File Documentation

5.1 Sensor_BME280.h File Reference

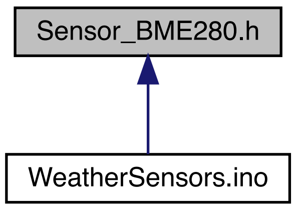
Library header for BME280 sensor.

```
#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 103

    Release.
- #define BM280_SUCCESS 0

    success
- #define BM280_ERROR 1

    error
- #define BME280_SLAVE_ADDRESS 0x77

    Default BME280 I2C address.

- #define BME280_FORCED_MODE 0b01

    BME280 modes.
- #define BME280_SLEEP_MODE 0b00

    Sleep mode.
- #define BME280_NORMAL_MODE 0b11

    Normal mode.

5.1.1 Detailed Description

Library header for BME280 sensor.

BME280 Combined humidity and pressure sensor

Project SensorsBoosterPack
Developed with `embedXcode+`

Author

Rei Vilo

`https://embeddedcomputing.weebly.com`

Date

20 Aug 2017

Version

102

Copyright

(c) Rei Vilo, 2015-2019

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`

5.1.2 Macro Definition Documentation

5.1.2.1 BME280_FORCED_MODE

```
#define BME280_FORCED_MODE 0b01
```

BME280 modes.

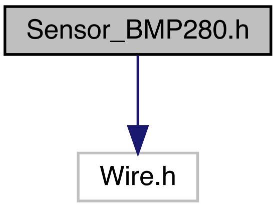
Forced mode

5.2 Sensor_BMP280.h File Reference

Library header for BMP280 sensor.

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

Include dependency graph for Sensor_BMP280.h:



Classes

- class Sensor_BMP280
  Class for sensor BMP280.

Macros

- #define Sensor_BMP280_RELEASE 102
  Release.
- #define BMP280_SLAVE_ADDRESS 0x77
  Default I2C address.
- #define BM280_SUCCESS 0
  success
- #define BM280_ERROR 1
  error

- #define BME280_FORCED_MODE 0b01

  BME280 modes.
- #define BME280_SLEEP_MODE 0b00

  Sleep mode.
- #define BME280_NORMAL_MODE 0b11

  Normal mode.

5.2.1 Detailed Description

Library header for BMP280 sensor.

BMP280 Combined humidity and pressure sensor

Project SensorsBoosterPack
Developed with `embedXcode+`

Author

Rei Vilo

`https://embeddedcomputing.weebly.com`

Date

20 Aug 2015

Version

102

Copyright

(c) Rei Vilo, 2015-2019

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`

5.2.2 Macro Definition Documentation

5.2.2.1 BME280_FORCED_MODE

```
#define BME280_FORCED_MODE 0b01
```

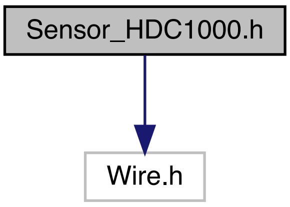
BME280 modes.

Forced mode

5.3 Sensor_HDC1000.h File Reference

Library header.

```
#include "Wire.h"
```
Include dependency graph for Sensor_HDC1000.h:



Classes

- class Sensor_HDC1000

  Class for sensor HDC1000.

Macros

- #define **Sensor_HDC1000_cpp**
- #define **HDC1000_I2C_ADDRESS** 0x43
- #define **HDC1000_RESET** 0b10000000
- #define **HDC1000_HEATER_DISABLED** 0
- #define **HDC1000_HEATER_ENABLED** 0b00100000
- #define **HDC1000_MODE_EITHER** 0
- #define **HDC1000_MODE_SEQUENCE** 0b00010000
- #define **HDC1000_TEMPERATURE_14_BITS** 0
- #define **HDC1000_TEMPERATURE_11_BITS** 0b00000100
- #define **HDC1000_HUMIDITY_14_BITS** 0
- #define **HDC1000_HUMIDITY_11_BITS** 0b00000001
- #define **HDC1000_HUMIDITY_8_BITS** 0b00000010
- #define **HDC1000_SETTINGS** 0b00010101

5.3.1 Detailed Description

Library header.

HDC1000 Temperature and Humidity Sensor

Project smartWatch
Developed with `embedXcode+`

Author

ReiVilo

ReiVilo

Date

12 Mar 2016

Version

101

Copyright

(c) Rei Vilo, 2016-2019

CC = BY SA NC

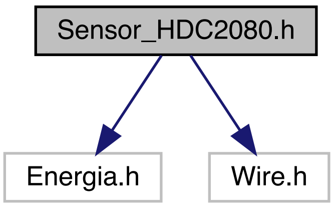
See also

ReadMe.txt for references

5.4 Sensor_HDC2080.h File Reference

Library header.

```
#include <Energia.h>
#include <Wire.h>
```
Include dependency graph for Sensor_HDC2080.h:



Classes

- class Sensor_HDC2080

  Class for sensor HDC2080.

Macros

- #define SENSOR_HDC2080_H 102

  Release.
- #define HDC2080_I2C_ADDRESS 0x41

  Default address on LPSTK.
- #define HDC2080_DEFAULT_SETTINGS 0b00000000

  Default settings.
- #define HDC2080_MEASURE_SETTINGS 0b00000000

  Default measurement settings.

- #define HDC2080_FOURTEEN_BIT 0

  Measurement resolution.
- #define HDC2080_ELEVEN_BIT 1

  7-bit
- #define HDC2080_NINE_BIT 2

  9-bit

- #define HDC2080_TEMPERATURE_AND_HUMIDITY 0

  Sensor modes.
- #define HDC2080_TEMPERATURE_ONLY 1

  temperature
- #define HDC2080_HUMIDITY_ONLY 2

  humidity
- #define HDC2080_ACTIVE_LOW 0

  interrupt output, active low
- #define HDC2080_ACTIVE_HIGH 1

  interrupt output, active high
- #define HDC2080_LEVEL_MODE 0

  interrupt output, level mode
- #define HDC2080_COMPARATOR_MODE 1

  interrupt output, comparator mode

- #define HDC2080_MANUAL 0

  Sample rate.
- #define HDC2080_TWO_MINUTES 1

  period = 2 minutes

- #define HDC2080_ONE_MINUTE 2

  period = 1 minutes
- #define HDC2080_TEN_SECONDS 3

  period = 10 seconds
- #define HDC2080_FIVE_SECONDS 4

  period = 5 seconds
- #define HDC2080_ONE_HZ 5

  period = 1 second
- #define HDC2080_TWO_HZ 6

  period = 0.5 second
- #define HDC2080_FIVE_HZ 7

  period = 0.2 second

5.4.1 Detailed Description

Library header.

Library for HDC2080 humidity and temperature sensor

Project SensorsBoosterPack
Developed with `embedXcode+`

Author

Rei Vilo

`https://embeddedcomputing.weebly.com`

Date

22 Oct 2019

Version

102

Copyright

(c) Rei Vilo, 2015-2019

CC = BY SA NC

See also

ReadMe.txt for references Brandon Fisher, August 1st 2017

5.4.2 Macro Definition Documentation

5.4.2.1 HDC2080_DEFAULT_SETTINGS

```
#define HDC2080_DEFAULT_SETTINGS 0b00000000
```

Default settings.

Values 0b01010000

- LSB b7 Sotfware reset, 0 = normal
- b6:4 Auto Measurement Mode, 101 = 1 Hz
- b3 Heater, 0 = off
- b2 Data ready interrupt, 0 = high-Z
- b2 Interrupt polarity, 0 = active low
- MSB b0 Interrupt mode, 0 = Level sensitive

Values 0b00000000

- LSB b7 Sotfware reset, 0 = normal
- b6:4 Auto Measurement Mode, 000 = manual
- b3 Heater, 0 = off
- b2 Data ready interrupt, 0 = high-Z
- b2 Interrupt polarity, 0 = active low
- MSB b0 Interrupt mode, 0 = Level sensitive

5.4.2.2 HDC2080_FOURTEEN_BIT

```
#define HDC2080_FOURTEEN_BIT 0
```

Measurement resolution.

14-bit

5.4.2.3 HDC2080_MANUAL

```
#define HDC2080_MANUAL 0
```

Sample rate.

manual mode, triggered by I2C

5.4.2.4 HDC2080_MEASURE_SETTINGS

```
#define HDC2080_MEASURE_SETTINGS 0b00000000
```

Default measurement settings.

Values 0b00000000

- LSB b7:6 Temperature resolution, 0 = 14 bits
- b5:4 Humidity resolution, 0 = 14 bits
- b3 Reserved
- b2:1 Measurement configuration, 0 = Humidity + Temperature
- MSB b0 Measurement trigger, 1 = Start measurement

5.4.2.5 HDC2080_TEMPERATURE_AND_HUMIDITY

```
#define HDC2080_TEMPERATURE_AND_HUMIDITY 0
```

Sensor modes.

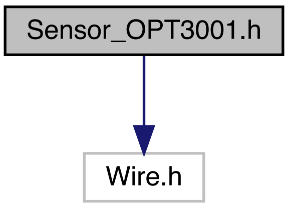
temperature and humidity

5.5 Sensor_OPT3001.h File Reference

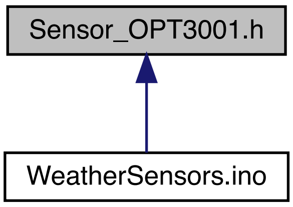
Library header for OPT3001 sensor.

```
#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_INTERRUPT_PIN 11

  Interrupt pin number.
- #define OPT3001_SLAVE_ADDRESS 0x47

  Default I2C address.

- #define OPT3001_100_MS_OFF 0xc010

  Conversion modes.
- #define OPT3001_100_MS_ONCE 0xc210

  Conversion modes.
- #define OPT3001_100_MS_CONTINUOUS 0xc410

  continous
- #define OPT3001_800_MS_ONCE 0xc810

  Conversion modes.
- #define OPT3001_800_MS_OFF 0xca10

  Conversion modes.
- #define OPT3001_800_MS_CONTINUOUS 0xcc10

  continuous

5.5.1 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

`https://embeddedcomputing.weebly.com`

Date

20 Aug 2015

Version

102

Copyright

(c) Rei Vilo, 2015-2019

CC = BY SA NC

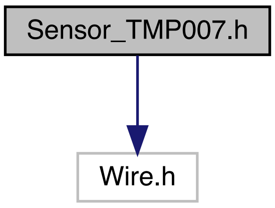
See also

ReadMe.txt for references

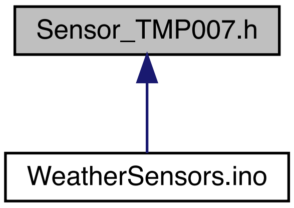
5.6 Sensor_TMP007.h File Reference

Library header for TMP007 sensor.

```
#include "Wire.h"
```
Include dependency graph for Sensor_TMP007.h:



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



Classes

- class Sensor_TMP007

  Class for sensor TMP007.

Macros

- #define Sensor_TMP007_cpp 102

  Release.

- #define TMP007_SLAVE_ADDRESS 0x40

  TMP007 constants.
- #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.

5.6.1 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

`https://embeddedcomputing.weebly.com`

Date

20 Aug 2015

Version

102

Copyright

(c) Rei Vilo, 2015-2019

CC = BY SA NC

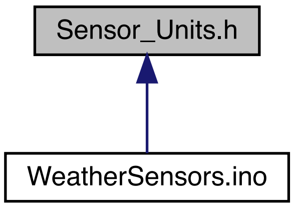
See also

ReadMe.txt for references

5.7 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.
- const temperature_unit_t CELSIUS = { 1, -273.15, "°C"}

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

  °F degree fahrenheit.

- 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
- 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"}

  0.014503773801 pound force/square inch

- typedef unit_conversion_s altitude_unit_t

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

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

  ft foot

- typedef unit_conversion_s light_unit_t

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

  lx lux

5.7.1 Detailed Description

Library header.

Units conversion for sensors

Project SensorsBoosterPack
Developed with embedXcode+

Author

Rei Vilo
https://embeddedcomputing.weebly.com

Date

20 Aug 2017

Version

102

Copyright

(c) Rei Vilo, 2015-2019

CC = SA BY NC

See also

ReadMe.txt for references

5.7.2 Typedef Documentation

5.7.2.1 altitude_unit_t

```
typedef unit_conversion_s altitude_unit_t
```

Altitude units.

SI reference = m metre

5.7.2.2 light_unit_t

```
typedef unit_conversion_s light_unit_t
```

Light units.

SI reference = lx lux

5.7.2.3 pressure_unit_t

```
typedef unit_conversion_s pressure_unit_t
```

Pressure units.

SI reference = hPa hecto pascal

5.7.2.4 temperature_unit_t

```
typedef unit_conversion_s temperature_unit_t
```

Temperature units.

SI reference = °K degree kelvin

5.7.3 Function Documentation

5.7.3.1 conversion()

```
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

5.7.3.2 symbolChar()

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

Unit symbol as char∗.

Parameters

unit	unit constant

Returns

symbol as char∗

5.7.3.3 symbolString()

```
template<typename myType >
String symbolString (
            myType unit )
```

Unit symbol as String.

Parameters

unit	unit constant

Returns

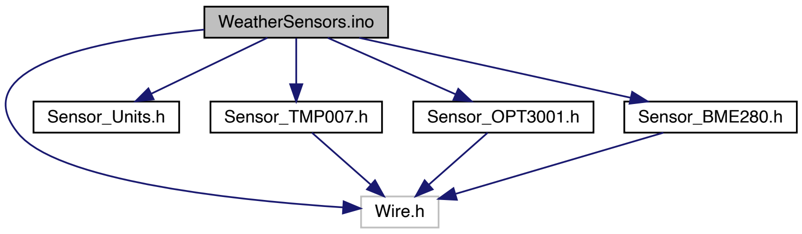
symbol as String

5.8 WeatherSensors.ino File Reference

Main sketch.

```
#include "Wire.h"
#include "Sensor_Units.h"
#include "Sensor_TMP007.h"
#include "Sensor_OPT3001.h"
#include "Sensor_BME280.h"
```

Include dependency graph for WeatherSensors.ino:



Macros

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

Functions

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

Variables

- Sensor_TMP007 **myTMP007**
- float **TMP007_internal**
- float **TMP007_external**
- Sensor_OPT3001 **myOPT3001**
- float **OPT3001_light**
- Sensor_BME280 **myBME280**
- float **BME280_pressure**
- float **BME280_temperature**
- float **BME280_humidity**
- const uint32_t **period_ms** = 10000

5.8.1 Detailed Description

Main sketch.

Example for climate sensors
Developed with `embedXcode+`

Author

Rei Vilo

`http://embeddedcomputing.weebly.com`

Date

12 Nov 2016

Version

102

Copyright

(c) Rei Vilo, 2016-2018

CC = BY SA NC

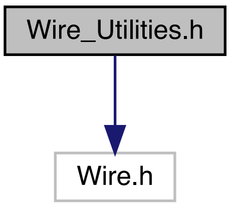
See also

ReadMe.txt for references

5.9 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.
- void delayBusy (uint32_t ms)

  Delay without yield.

5.9.1 Detailed Description

Library header.

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

Project SensorsBoosterPack
Developed with `embedXcode+`

Author

Rei Vilo

https://embeddedcomputing.weebly.com

Date

20 Aug 2015

Version

102

Copyright

(c) Rei Vilo, 2015-2019

CC = BY SA NC

See also

ReadMe.txt for references

5.9.2 Function Documentation

5.9.2.1 delayBusy()

```
void delayBusy (
             uint32_t ms )
```

Delay without yield.

Parameters

ms	period to wait for, ms

5.9.2.2 readRegister16()

```
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

5.9.2.3 readRegister8()

```
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

5.9.2.4 writeRegister16()

```
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

5.9.2.5 writeRegister8()

```
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