Erriez BMP280/BME280 library for Arduino

1.0.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

BH1750 digital light sensor library for Arduino

This is a 16-bit BH1750 digital ambient light sensor on a GY-302 breakout PCB:

Arduino library features

- Measurement in LUX
- Three operation modes:
  - Continues conversion
  - One-time conversion
- Three selectable resolutions:
  - Low 4 LUX resolution (low power)
  - High 1 LUX resolution
  - High 0.5 LUX resolution
- Asynchronous and synchronous conversion

BH1750 sensor specifications

- Operating voltage: 3.3V .. 4.5V max
- Low current by power down: max 1uA
- I2C bus interface: max 400kHz
- Ambience light:
  - Range: 1 - 65535 lx
  - Deviation: +/- 20%
  - Selectable resolutions:
    - 4 lx (low resolution, max 24 ms measurement time)
    - 1 lx (mid resolution max 180 ms measurement time)
    - 0.5 lx (high resolution 180 ms measurement time)
- No additional electronic components needed

GY-302 breakout specifications

- Supply voltage: 3.3 .. 5V
- 5V tolerant I2C SCL and SDA pins
- 2 selectable I2C addresses with ADDR pin high or low/floating

Hardware

Connection Arduino UNO board - BH1750

Pins board - BH1750	VCC	GND	SDA	SCL
Arduino UNO (ATMega328 boards)	5V	GND	A4	A5
Arduino Mega2560	5V	GND	D20	D21
Arduino Leonardo	5V	GND	D2	D3
Arduino DUE (ATSAM3X8E)	3V3	GND	20	21
ESP8266	3V3	GND	GPIO4 (D2)	GPIO5 (D1)
ESP32	3V3	GND	GPIO21	GPIO22

Note: Tested ESP8266 / ESP32 boards:

- **ESP8266 boards**: ESP12E / WeMos D1 & R2 / Node MCU v2 / v3
- **ESP32 boards:** WeMos LOLIN32 / LOLIN D32

Other unlisted MCU's may work, but are not tested.

WeMos LOLIN32 with OLED display

Change the following Wire initialization to:

```
{c++}
// WeMos LOLIN32 with OLED support
Wire.begin(5, 4);
```

I2C address

- `ADDR` pin `LOW` for I2C address 0x23 (0x46 including R/W bit)
- `ADDR` pin `HIGH` for I2C address 0x5C (0xB8 including R/W bit)

Note: `ADDR` pin may be floating (open) which is the same as LOW.

Examples

Examples | Erriez BH1750:

- ContinuesMode | BH1750ContinuesAsynchronous
- ContinuesMode | BH1750ContinuesBasic
- ContinuesMode | BH1750ContinuesHighResolution
- ContinuesMode | BH1750ContinuesLowResolution
- ContinuesMode | BH1750ContinuesPowerMgt
- OneTimeMode | BH1750OneTimeBasic
- OneTimeMode| BH1750OneTimeHighResolution
- OneTimeMode| BH1750OneTimeLowResolution
- OneTimeMode| BH1750OneTimePowerMgt

Documentation

- Doxygen online HTML
- Doxygen PDF
- BH1750 chip datasheet

Example continues conversion high resolution

```
{c++}
#include <Wire.h>
#include <ErriezBH1750.h>

// ADDR line LOW/open:  I2C address 0x23 (0x46 including R/W bit) [default]
// ADDR line HIGH:      I2C address 0x5C (0xB8 including R/W bit)
BH1750 sensor(LOW);

void setup()
{
  Serial.begin(115200);
  Serial.println(F("BH1750 continues measurement high resolution example"));

  // Initialize I2C bus
  Wire.begin();

  // Initialize sensor in continues mode, high 0.5 lx resolution
  sensor.begin(ModeContinuous, ResolutionHigh);

  // Start conversion
  sensor.startConversion();
}

void loop()
{
  uint16_t lux;

  // Wait for completion (blocking busy-wait delay)
  if (sensor.isConversionCompleted()) {
    // Read light
    lux = sensor.read();

    // Print light
    Serial.print(F("Light: "));
    Serial.print(lux / 2);
    Serial.print(F("."));
    Serial.print(lux % 10);
    Serial.println(F(" LUX"));
  }
}
```

Output

```
{c++}
BH1750 continues measurement high resolution example
Light: 15.0 LUX
Light: 31.2 LUX
Light: 385.0 LUX
Light: 575.1 LUX
Light: 667.5 LUX
```

Usage

Initialization

```
{c++}
#include <Wire.h>
#include <ErriezBH1750.h>

// ADDR line LOW/open:  I2C address 0x23 (0x46 including R/W bit) [default]
// ADDR line HIGH:      I2C address 0x5C (0xB8 including R/W bit)
BH1750 sensor(LOW);

void setup()
{
    // Initialize I2C bus
    Wire.begin();

    // Initialize sensor with a mode and resolution:
    //   Modes:
    //     ModeContinuous
    //     ModeOneTime
    //   Resolutions:
    //     ResolutionLow (4 lx resolution)
    //     ResolutionMid (1 lx resolution)
    //     ResolutionHigh (0.5 lx resolution)
    sensor.begin(mode, resolution);
}
```

Start conversion

```
{Wire.begin();```}

```c++
sensor.startConversion();
```

Wait for completion asynchronous (non-blocking)

The sensor conversion completion status can be checked asynchronously before reading the light value:

```
{c++}
bool completed = sensor.isConversionCompleted();
```

Wait for completion synchronous (blocking)

The sensor conversion completion status can be checked synchronously before reading the light value:

```
{c++}
// Wait for completion
// completed = false: Timeout or device in power-down
bool completed = sensor.waitForCompletion();
```
```

Read light value in LUX

One-time mode: The application must wait or check for a completed conversion, otherwise the sensor may return an invalid value. **Continues mode:** The application can call this function without checking completion, but is not recommended when accurate values are required.

Read sensor light value:

```
{c++}
// lux = 0: No light or not initialized
uint16_t lux = sensor.read();
```

For 4 lx low and 1 lx high resolutions:

```
{c++}
// Print low and medium resolutions
Serial.print(F("Light: "));
Serial.print(lux);
Serial.println(F(" LUX"));
```

For 0.5 lx high resolution:

```
{c++}
// Print high resolution
Serial.print(F("Light: "));
Serial.print(lux / 2);
Serial.print(F("."));
Serial.print(lux % 10);
Serial.println(F(" LUX"));
```

Power down

The device enters power down automatically after a one-time conversion.

A manual power-down in continues mode can be generated by calling:

```
{c++}
sensor.powerDown();
```

Library dependencies

- Built-in `Wire.h`

Library installation

Please refer to the `Wiki` page.

Other Arduino Libraries and Sketches from Erriez

- `Erriez Libraries and Sketches`

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 ErriezBMX280 Class Reference

BMX280 class.

`#include <ErriezBMX280.h>`

Public Member Functions

- ErriezBMX280 (uint8_t i2cAddr)
  Constructor.
- bool begin ()
  Sensor initialization.
- uint8_t getChipID ()
  Get chip ID.
- float readTemperature ()
  Read temperature.
- float readPressure ()
  Read pressure.
- float readAltitude (float seaLevel)
  Read approximate altitude.
- float readHumidity ()
  Read humidity (BME280 only)
- void setSampling (BMX280_Mode_e mode=BMX280_MODE_NORMAL, BMX280_Sampling_e tempSampling=BMX280_SAMPLING_X16, BMX280_Sampling_e pressSampling=BMX280_SAMPLING_X16, BMX280_Sampling_e humSampling=BMX280_SAMPLING_X16, BMX280_Filter_e filter=BMX280_FILTER_OFF, BMX280_Standby_e standbyDuration=BMX280_STANDBY_MS_0_5)
  Set sampling registers.
- uint8_t read8 (uint8_t reg)
  Read from 8-bit register.
- uint16_t read16 (uint8_t reg)
  Read from 16-bit register.
- uint16_t read16_LE (uint8_t reg)
  Read from 16-bit unsigned register little endian.
- int16_t readS16_LE (uint8_t reg)
  Read from 16-bit signed register little endian.
- uint32_t read24 (uint8_t reg)
  Read from 24-bit register.
- void write8 (uint8_t reg, uint8_t value)
  Write to 8-bit register.

4.1.1 Detailed Description

BMX280 class.

Definition at line 134 of file ErriezBMX280.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ErriezBMX280()

```
ErriezBMX280::ErriezBMX280 (
            uint8_t i2cAddr )
```

Constructor.

Parameters

i2cAddr	I2C address

Definition at line 43 of file ErriezBMX280.cpp.

4.1.3 Member Function Documentation

4.1.3.1 begin()

```
bool ErriezBMX280::begin ( )
```

Sensor initialization.

Return values

true	BMP280 or BME280 sensor detected
false	Error: No (supported) sensor detected

Definition at line 55 of file ErriezBMX280.cpp.

4.1.3.2 getChipID()

```
uint8_t ErriezBMX280::getChipID ( )
```

Get chip ID.

Returns

Chip ID as read with begin()

Definition at line 93 of file ErriezBMX280.cpp.

4.1.3.3 read16()

```
uint16_t ErriezBMX280::read16 (
            uint8_t reg )
```

Read from 16-bit register.

Parameters

reg	Register address

Returns

16-bit register value

Definition at line 357 of file ErriezBMX280.cpp.

4.1.3.4 read16_LE()

```
uint16_t ErriezBMX280::read16_LE (
            uint8_t reg )
```

Read from 16-bit unsigned register little endian.

Parameters

reg	Register address

Returns

16-bit unsigned register value in little endian

Definition at line 329 of file ErriezBMX280.cpp.

4.1.3.5 read24()

```
uint32_t ErriezBMX280::read24 (
            uint8_t reg )
```

Read from 24-bit register.

Parameters

reg	Register address

Returns

24-bit register value

Definition at line 375 of file ErriezBMX280.cpp.

4.1.3.6 read8()

```
uint8_t ErriezBMX280::read8 (
            uint8_t reg )
```

Read from 8-bit register.

Parameters

reg	Register address

Returns

8-bit register value

Definition at line 296 of file ErriezBMX280.cpp.

4.1.3.7 readAltitude()

```
float ErriezBMX280::readAltitude (
            float seaLevel )
```

Read approximate altitude.

Parameters

seaLevel	Sea level in hPa

Returns

Altitude (float)

Definition at line 174 of file ErriezBMX280.cpp.

4.1.3.8 readHumidity()

```
float ErriezBMX280::readHumidity ( )
```

Read humidity (BME280 only)

Returns

Humidity (float)

Definition at line 187 of file ErriezBMX280.cpp.

4.1.3.9 readPressure()

```
float ErriezBMX280::readPressure ( )
```

Read pressure.

Returns

Pressure (float)

Definition at line 132 of file ErriezBMX280.cpp.

4.1.3.10 readS16_LE()

```
int16_t ErriezBMX280::readS16_LE (
            uint8_t reg )
```

Read from 16-bit signed register little endian.

Parameters

reg	Register address

Returns

16-bit signed register value in little endian

Definition at line 345 of file ErriezBMX280.cpp.

4.1.3.11 readTemperature()

```
float ErriezBMX280::readTemperature ( )
```

Read temperature.

Returns

Temperature (float)

Definition at line 104 of file ErriezBMX280.cpp.

4.1.3.12 setSampling()

```
void ErriezBMX280::setSampling (
            BMX280_Mode_e mode = BMX280_MODE_NORMAL,
            BMX280_Sampling_e tempSampling = BMX280_SAMPLING_X16,
            BMX280_Sampling_e pressSampling = BMX280_SAMPLING_X16,
            BMX280_Sampling_e humSampling = BMX280_SAMPLING_X16,
            BMX280_Filter_e filter = BMX280_FILTER_OFF,
            BMX280_Standby_e standbyDuration = BMX280_STANDBY_MS_0_5 )
```

Set sampling registers.

Parameters

mode	See BMX280_Mode_e
tempSampling	See BMX280_Sampling_e
pressSampling	See BMX280_Sampling_e
humSampling	See BMX280_Sampling_e
filter	See BMX280_Filter_e
standbyDuration	See BMX280_Standby_e

Definition at line 269 of file ErriezBMX280.cpp.

4.1.3.13 write8()

```
void ErriezBMX280::write8 (
            uint8_t reg,
            uint8_t value )
```

Write to 8-bit register.

Parameters

reg	Register address
value	8-bit register value

Definition at line 314 of file ErriezBMX280.cpp.

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

- src/ErriezBMX280.h
- src/ErriezBMX280.cpp

Chapter 5

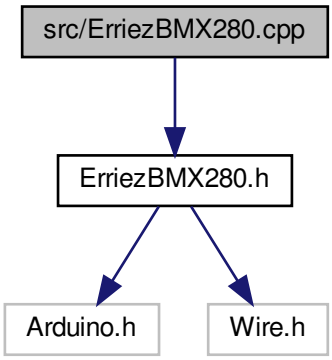
File Documentation

5.1 src/ErriezBMX280.cpp File Reference

BMP280/BME280 sensor library for Arduino.

```
#include "ErriezBMX280.h"
```

Include dependency graph for ErriezBMX280.cpp:



5.1.1 Detailed Description

BMP280/BME280 sensor library for Arduino.

BMP280 supports temperature and pressure BME280 supports temperature, pressure and humidity

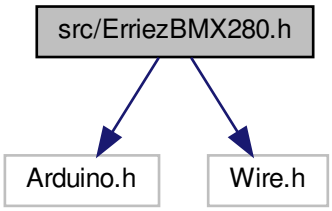
```
Source:          https://github.com/Erriez/ErriezBMX280
Documentation:   https://erriez.github.io/ErriezBMX280
```

5.2 src/ErriezBMX280.h File Reference

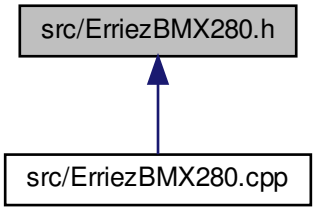
BMP280/BME280 sensor library for Arduino.

```
#include <Arduino.h>
#include <Wire.h>
```

Include dependency graph for ErriezBMX280.h:



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



Classes

- class ErriezBMX280

  BMX280 class.

Macros

- #define BMX280_I2C_ADDR 0x76

  I2C address.
- #define BMX280_I2C_ADDR_ALT 0x77

  I2C alternative address.
- #define BMX280_REG_DIG_T1 0x88

  Temperature coefficient register.

- #define BMX280_REG_DIG_T2 0x8A
  Temperature coefficient register.
- #define BMX280_REG_DIG_T3 0x8C
  Temperature coefficient register.
- #define BMX280_REG_DIG_P1 0x8E
  Pressure coefficient register.
- #define BMX280_REG_DIG_P2 0x90
  Pressure coefficient register.
- #define BMX280_REG_DIG_P3 0x92
  Pressure coefficient register.
- #define BMX280_REG_DIG_P4 0x94
  Pressure coefficient register.
- #define BMX280_REG_DIG_P5 0x96
  Pressure coefficient register.
- #define BMX280_REG_DIG_P6 0x98
  Pressure coefficient register.
- #define BMX280_REG_DIG_P7 0x9A
  Pressure coefficient register.
- #define BMX280_REG_DIG_P8 0x9C
  Pressure coefficient register.
- #define BMX280_REG_DIG_P9 0x9E
  Pressure coefficient register.
- #define BME280_REG_DIG_H1 0xA1
  Humidity coefficient register.
- #define BME280_REG_DIG_H2 0xE1
  Humidity coefficient register.
- #define BME280_REG_DIG_H3 0xE3
  Humidity coefficient register.
- #define BME280_REG_DIG_H4 0xE4
  Humidity coefficient register.
- #define BME280_REG_DIG_H5 0xE5
  Humidity coefficient register.
- #define BME280_REG_DIG_H6 0xE7
  Humidity coefficient register.
- #define BME280_REG_CHIPID 0xD0
  Chip ID register.
- #define BME280_REG_RESET 0xE0
  Reset register.
- #define BME280_REG_CTRL_HUM 0xF2
  BME280: Control humidity register.
- #define BMX280_REG_STATUS 0XF3
  Status register.
- #define BMX280_REG_CTRL_MEAS 0xF4
  Control measure register.
- #define BMX280_REG_CONFIG 0xF5
  Config register.
- #define BMX280_REG_PRESS 0xF7
  Pressure data register.
- #define BMX280_REG_TEMP 0xFA
  Temperature data register.
- #define BME280_REG_HUM 0xFD

Humidity data register.

- #define CHIP_ID_BMP280 0x58

  BMP280 chip ID.
- #define CHIP_ID_BME280 0x60

  BME280 chip ID.
- #define RESET_KEY 0xB6

  Reset value for reset register.
- #define STATUS_IM_UPDATE 0

  im_update bit in status register

Enumerations

- enum BMX280_Mode_e { BMX280_MODE_SLEEP = 0b00, BMX280_MODE_FORCED = 0b01, BMX280_MODE_NORMAL = 0b11 }

  Sleep mode bits ctrl_meas register.
- enum BMX280_Sampling_e { BMX280_SAMPLING_NONE = 0b000, BMX280_SAMPLING_X1 = 0b001, BMX280_SAMPLING_X2 = 0b010, BMX280_SAMPLING_X4 = 0b011, BMX280_SAMPLING_X8 = 0b100, BMX280_SAMPLING_X16 = 0b101 }

  Sampling bits registers ctrl_hum, ctrl_meas.
- enum BMX280_Filter_e { BMX280_FILTER_OFF = 0b000, BMX280_FILTER_X2 = 0b001, BMX280_FILTER_X4 = 0b010, BMX280_FILTER_X8 = 0b011, BMX280_FILTER_X16 = 0b100 }

  Filter bits config register.
- enum BMX280_Standby_e { BMX280_STANDBY_MS_0_5 = 0b000, BMX280_STANDBY_MS_10 = 0b110, BMX280_STANDBY_MS_20 = 0b111, BMX280_STANDBY_MS_62_5 = 0b001, BMX280_STANDBY_MS_125 = 0b010, BMX280_STANDBY_MS_250 = 0b011, BMX280_STANDBY_MS_500 = 0b100, BMX280_STANDBY_MS_1000 = 0b101 }

  Standby duration bits config register.

5.2.1 Detailed Description

BMP280/BME280 sensor library for Arduino.

BMP280 supports temperature and pressure BME280 supports temperature, pressure and humidity

```
Source:          https://github.com/Erriez/ErriezBMX280
Documentation:   https://erriez.github.io/ErriezBMX280
```

5.2.2 Enumeration Type Documentation

5.2.2.1 BMX280_Filter_e

```
enum BMX280_Filter_e
```

Filter bits config register.

Enumerator

BMX280_FILTER_OFF	Filter off.
BMX280_FILTER_X2	x2 Filter
BMX280_FILTER_X4	x4 Filter
BMX280_FILTER_X8	x8 Filter
BMX280_FILTER_X16	x16 Filter

Definition at line 109 of file ErriezBMX280.h.

5.2.2.2 BMX280_Mode_e

`enum BMX280_Mode_e`

Sleep mode bits ctrl_meas register.

Enumerator

BMX280_MODE_SLEEP	Sleep mode.
BMX280_MODE_FORCED	Forced mode.
BMX280_MODE_NORMAL	Normal mode.

Definition at line 88 of file ErriezBMX280.h.

5.2.2.3 BMX280_Sampling_e

`enum BMX280_Sampling_e`

Sampling bits registers ctrl_hum, ctrl_meas.

Enumerator

BMX280_SAMPLING_NONE	Sampling disabled.
BMX280_SAMPLING_X1	x1 Sampling
BMX280_SAMPLING_X2	x2 Sampling
BMX280_SAMPLING_X4	x4 Sampling
BMX280_SAMPLING_X8	x8 Sampling
BMX280_SAMPLING_X16	x16 Sampling

Definition at line 97 of file ErriezBMX280.h.

5.2.2.4 BMX280_Standby_e

```
enum BMX280_Standby_e
```

Standby duration bits config register.

Enumerator

BMX280_STANDBY_MS_0_5	0.5m standby
BMX280_STANDBY_MS_10	10ms standby
BMX280_STANDBY_MS_20	20ms standby
BMX280_STANDBY_MS_62↩ _5	62.5 standby
BMX280_STANDBY_MS_125	125ms standby
BMX280_STANDBY_MS_250	250ms standby
BMX280_STANDBY_MS_500	500ms standby
BMX280_STANDBY_MS_1000	1s standby

Definition at line 120 of file ErriezBMX280.h.

Index