

Erriez BH1750 library for Arduino

1.0.1

Generated by Doxygen 1.8.11

Contents

| | | |
|----------|---|-----------|
| 1 | BH1750 digital light sensor library for Arduino | 1 |
| 2 | Class Index | 7 |
| 2.1 | Class List | 7 |
| 3 | File Index | 9 |
| 3.1 | File List | 9 |
| 4 | Class Documentation | 11 |
| 4.1 | BH1750 Class Reference | 11 |
| 4.1.1 | Detailed Description | 12 |
| 4.1.2 | Constructor & Destructor Documentation | 12 |
| 4.1.2.1 | BH1750(uint8_t addrPinLevel=LOW) | 12 |
| 4.1.3 | Member Function Documentation | 12 |
| 4.1.3.1 | begin(BH1750_Mode_e mode, BH1750_Resolution_e resolution) | 12 |
| 4.1.3.2 | isConversionCompleted() | 12 |
| 4.1.3.3 | read() | 13 |
| 4.1.3.4 | waitForCompletion() | 13 |
| 4.1.3.5 | writeInstruction(uint8_t instruction) | 13 |

| | |
|--------------------------------------|-----------|
| 5 File Documentation | 15 |
| 5.1 BH1750.cpp File Reference | 15 |
| 5.1.1 Detailed Description | 15 |
| 5.2 BH1750.h File Reference | 15 |
| 5.2.1 Detailed Description | 16 |
| 5.2.2 Enumeration Type Documentation | 16 |
| 5.2.2.1 BH1750_Mode_e | 16 |
| 5.2.2.2 BH1750_Resolution_e | 16 |
| 5.3 BH1750_priv.h File Reference | 16 |
| 5.3.1 Detailed Description | 17 |
| 5.3.2 Macro Definition Documentation | 17 |
| 5.3.2.1 GET_TIMEOUT | 17 |
| 5.3.2.2 IS_CONTINUES_MODE | 18 |
| 5.3.2.3 IS_INITIALIZED | 18 |
| 5.3.2.4 IS_LOW_RESOLUTION | 18 |
| 5.3.2.5 IS_ONE_TIME_MODE | 18 |
| Index | 19 |

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 [BH1750](#) - Arduino / ESP8266 boards

| BH1750 | Arduino UNO / Nano | Leonardo / Pro Micro | Mega2560 | WeMos D1 & R2 / Node MCU | WeMos LOLIN32 |
|------------------------|--------------------|----------------------|--------------|--------------------------|---------------|
| GND | GND | GND | GND | GND | GND |
| VCC | 5V (or 3.3V) | 5V (or 3.3V) | 5V (or 3.3V) | 3V3 | 3V3 |
| SDA | A4 (ANALOG pin) | 2 (DIGITAL pin) | D20 | D2 | 5 |
| SCL | A5 (ANALOG pin) | 3 (DIGITAL pin) | D21 | D1 | 4 |

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

WeMos LOLIN32 with OLED display

Change the following Wire initialization to:

```
1 {c++}
2 // WeMos LOLIN32 with OLED support
3 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

```
1 {c++}
2 #include <Wire.h>
3 #include <BH1750.h>
4
5 // ADDR line LOW/open:  I2C address 0x23 (0x46 including R/W bit) [default]
6 // ADDR line HIGH:      I2C address 0x5C (0xB8 including R/W bit)
7 BH1750 sensor(LOW);
8
9 void setup()
10 {
11     Serial.begin(115200);
12     Serial.println(F("BH1750 continues measurement high resolution example"));
13
14     // Initialize I2C bus
15     Wire.begin();
16
17     // Initialize sensor in continues mode, high 0.5 lx resolution
18     sensor.begin(ModeContinuous, ResolutionHigh);
19
20     // Start conversion
21     sensor.startConversion();
22 }
23
24 void loop()
25 {
26     uint16_t lux;
27
28     // Wait for completion (blocking busy-wait delay)
29     if (sensor.isConversionCompleted()) {
30         // Read light
31         lux = sensor.read();
32
33         // Print light
34         Serial.print(F("Light: "));
35         Serial.print(lux / 2);
36         Serial.print(F("."));
37         Serial.print(lux % 10);
38         Serial.println(F(" LUX"));
39     }
40 }
```

Output

```
1 {c++}
2 BH1750 continues measurement high resolution example
3 Light: 15.0 LUX
4 Light: 31.2 LUX
5 Light: 385.0 LUX
6 Light: 575.1 LUX
7 Light: 667.5 LUX
```

Usage

Initialization

```
1 {c++}
2 #include <Wire.h>
3 #include <BH1750.h>
4
5 // ADDR line LOW/open:  I2C address 0x23 (0x46 including R/W bit) [default]
6 // ADDR line HIGH:      I2C address 0x5C (0xB8 including R/W bit)
```

```
7 BH1750 sensor(LOW);
8
9 void setup()
10 {
11     // Initialize I2C bus
12     Wire.begin();
13
14     // Initialize sensor with a mode and resolution:
15     //   Modes:
16     //     ModeContinuous
17     //     ModeOneTime
18     //   Resolutions:
19     //     ResolutionLow (4 lx resolution)
20     //     ResolutionMid (1 lx resolution)
21     //     ResolutionHigh (0.5 lx resolution)
22     sensor.begin(mode, resolution);
23 }
```

Start conversion

```
1 {Wire.begin(); ``}
2
3 ``c++
4 sensor.startConversion();
```

Wait for completion asynchronous (non-blocking)

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

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

Wait for completion synchronous (blocking)

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

```
1 {c++}
2 // Wait for completion
3 // completed = false: Timeout or device in power-down
4 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:

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

For 4 lx low and 1 lx high resolutions:

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

For 0.5 lx high resolution:

```
1 {c++}
2 // Print high resolution
3 Serial.print(F("Light: "));
4 Serial.print(lux / 2);
5 Serial.print(F("."));
6 Serial.print(lux % 10);
7 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:

```
1 {c++}  
2 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:

| | |
|--|--------------------|
| BH1750 | |
| BH1750 class | 11 |

Chapter 3

File Index

3.1 File List

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

| | | |
|---|-----------|----|
| BH1750.cpp | | |
| BH1750 digital light sensor library for Arduino | | 15 |
| BH1750.h | | |
| BH1750 digital light sensor library for Arduino | | 15 |
| BH1750_priv.h | | |
| BH1750 digital light sensor library for Arduino | | 16 |

Chapter 4

Class Documentation

4.1 BH1750 Class Reference

BH1750 class.

```
#include <BH1750.h>
```

Public Member Functions

- **BH1750** (uint8_t addrPinLevel=LOW)
Constructor.
- void **begin** (BH1750_Mode_e mode, BH1750_Resolution_e resolution)
Set mode and resolution.
- void **powerDown** ()
*Power down. Call **startConversion()** to power-up automatically.*
- void **startConversion** ()
Start conversion.
- bool **isConversionCompleted** ()
Wait for completion.
- bool **waitForCompletion** ()
Wait for completion.
- uint16_t **read** ()
Read light level asynchronous from sensor The application is responsible for wait or checking a completed conversion, otherwise the last conversion value may be returned which may not be correct. The last value is also returned when the device is in power-down.

Protected Member Functions

- void **writeInstruction** (uint8_t instruction)
Write instruction to sensor.
- void **setTimestamp** ()
Save current time + minimum delay before reading next conversion in ms.

4.1.1 Detailed Description

[BH1750](#) class.

Definition at line 53 of file BH1750.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BH1750::BH1750 (uint8_t *addrPinLevel* = LOW) [explicit]

Constructor.

Parameters

| | |
|---------------------|--|
| <i>addrPinLevel</i> | Sensor I2C address: ADDR pin = LOW: 0x23 (default) ADDR pin = HIGH: 0x5C |
|---------------------|--|

Definition at line 44 of file BH1750.cpp.

4.1.3 Member Function Documentation

4.1.3.1 void BH1750::begin (BH1750_Mode_e *mode*, BH1750_Resolution_e *resolution*)

Set mode and resolution.

Parameters

| | |
|-------------------|---|
| <i>mode</i> | ModeContinuous for continues mode Continues conversion requires more power ModeOneTime for one-time conversion mode Set in low power when conversion completed |
| <i>resolution</i> | Resolution05Lux for high resolution (max 180ms conversion) Resolution1Lux for normal resolution (max 180ms conversion) Resolution4Lux for low resolution (max 24ms conversion, low power) |

Definition at line 66 of file BH1750.cpp.

4.1.3.2 bool BH1750::isConversionCompleted ()

Wait for completion.

Returns

true: Conversion completed false: Conversion busy

Definition at line 105 of file BH1750.cpp.

4.1.3.3 `uint16_t BH1750::read ()`

Read light level asynchronous from sensor. The application is responsible for wait or checking a completed conversion, otherwise the last conversion value may be returned which may not be correct. The last value is also returned when the device is in power-down.

Returns

Light level in lux (0..65535). In high resolution, the last digit is the remainder.

Definition at line 162 of file BH1750.cpp.

4.1.3.4 `bool BH1750::waitForCompletion ()`

Wait for completion.

Returns

true: Conversion completed false: Not initialized, or timeout

Definition at line 125 of file BH1750.cpp.

4.1.3.5 `void BH1750::writeInstruction (uint8_t instruction)` [protected]

Write instruction to sensor.

Parameters

| | |
|--------------------|--------------------|
| <i>instruction</i> | Sensor instruction |
|--------------------|--------------------|

Definition at line 215 of file BH1750.cpp.

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

- [BH1750.h](#)
- [BH1750.cpp](#)

Chapter 5

File Documentation

5.1 BH1750.cpp File Reference

BH1750 digital light sensor library for Arduino.

```
#include "BH1750.h"  
#include "BH1750_priv.h"
```

5.1.1 Detailed Description

BH1750 digital light sensor library for Arduino.

Source: <https://github.com/Erriez/ErriezBH1750> Documentation: <https://erriez.github.io/ErriezBH1750>

5.2 BH1750.h File Reference

BH1750 digital light sensor library for Arduino.

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

Classes

- class **BH1750**
BH1750 class.

Enumerations

- enum **BH1750_Mode_e** { **ModeContinuous** = 0x10, **ModeOneTime** = 0x20 }
Mode register bits.
- enum **BH1750_Resolution_e** { **ResolutionLow** = 0x03, **ResolutionMid** = 0x00, **ResolutionHigh** = 0x01 }
Resolution register bits.

5.2.1 Detailed Description

[BH1750](#) digital light sensor library for Arduino.

Source: <https://github.com/Erriez/ErriezBH1750> Documentation: <https://erriez.github.io/ErriezBH1750>

5.2.2 Enumeration Type Documentation

5.2.2.1 enum BH1750_Mode_e

Mode register bits.

Enumerator

ModeContinuous Continues mode.

ModeOneTime One-time mode.

Definition at line 40 of file BH1750.h.

5.2.2.2 enum BH1750_Resolution_e

Resolution register bits.

Enumerator

ResolutionLow 4 lx resolution

ResolutionMid 1 lx resolution

ResolutionHigh 0.5 lx resolution

Definition at line 46 of file BH1750.h.

5.3 BH1750_priv.h File Reference

[BH1750](#) digital light sensor library for Arduino.

Macros

- `#define BH1750_I2C_ADDR_L 0x23`
I2C address with ADDR pin low.
- `#define BH1750_I2C_ADDR_H 0x5C`
I2C address with ADDR pin high.
- `#define BH1750_POWER_DOWN 0x00`
Power down instruction.
- `#define BH1750_POWER_ON 0x01`
Power on instruction.
- `#define BH1750_RESET 0x07`
Reset instruction.
- `#define BH1750_MODE_MASK 0x30`
Mode mask bits.
- `#define BH1750_RES_MASK 0x03`
Mode resolution mask bits.
- `#define BH1750_CONV_TIME_L 24`
Worst case conversion timing low res.
- `#define BH1750_CONV_TIME_H 180`
Worst case conversion timing high res.
- `#define IS_INITIALIZED(mode) (((mode) & BH1750_MODE_MASK) != 0x00)`
- `#define IS_CONTINUES_MODE(mode) (((mode) & BH1750_MODE_MASK) == ModeContinuous)`
- `#define IS_ONE_TIME_MODE(mode) (((mode) & BH1750_MODE_MASK) == ModeOneTime)`
- `#define IS_LOW_RESOLUTION(mode) (((mode) & BH1750_RES_MASK) == ResolutionLow)`
- `#define GET_TIMEOUT(mode)`

5.3.1 Detailed Description

BH1750 digital light sensor library for Arduino.

[BH1750_priv.h](#)

Source: <https://github.com/Erriez/ErriezBH1750>
Documentation: <https://erriez.github.io/ErriezBH1750>

5.3.2 Macro Definition Documentation

5.3.2.1 `#define GET_TIMEOUT(mode)`

Value:

```
(( (mode) & BH1750_RES_MASK) == ResolutionLow) ? \
    BH1750_CONV_TIME_L :
    BH1750_CONV_TIME_H)
```

Macro low/high resolution timeout from mode

Definition at line 83 of file BH1750_priv.h.

5.3.2.2 `#define IS_CONTINUES_MODE(mode) (((mode) & BH1750_MODE_MASK) == ModeContinuous)`

Macro is continues mode enabled

Definition at line 65 of file BH1750_priv.h.

5.3.2.3 `#define IS_INITIALIZED(mode) (((mode) & BH1750_MODE_MASK) != 0x00)`

Return if mode is set (initialized)

Definition at line 59 of file BH1750_priv.h.

5.3.2.4 `#define IS_LOW_RESOLUTION(mode) (((mode) & BH1750_RES_MASK) == ResolutionLow)`

Macro is low resolution enabled from mode

Definition at line 77 of file BH1750_priv.h.

5.3.2.5 `#define IS_ONE_TIME_MODE(mode) (((mode) & BH1750_MODE_MASK) == ModeOneTime)`

Macro is one-time mode enabled from mode

Definition at line 71 of file BH1750_priv.h.

Index

- BH1750, [11](#)
 - BH1750, [12](#)
 - begin, [12](#)
 - isConversionCompleted, [12](#)
 - read, [12](#)
 - waitForCompletion, [13](#)
 - writeInstruction, [13](#)
- BH1750.cpp, [15](#)
- BH1750.h, [15](#)
 - BH1750_Mode_e, [16](#)
 - BH1750_Resolution_e, [16](#)
 - ModeContinuous, [16](#)
 - ModeOneTime, [16](#)
 - ResolutionHigh, [16](#)
 - ResolutionLow, [16](#)
 - ResolutionMid, [16](#)
- BH1750_Mode_e
 - BH1750.h, [16](#)
- BH1750_Resolution_e
 - BH1750.h, [16](#)
- BH1750_priv.h, [16](#)
 - GET_TIMEOUT, [17](#)
 - IS_CONTINUES_MODE, [17](#)
 - IS_INITIALIZED, [18](#)
 - IS_LOW_RESOLUTION, [18](#)
 - IS_ONE_TIME_MODE, [18](#)
- begin
 - BH1750, [12](#)
- GET_TIMEOUT
 - BH1750_priv.h, [17](#)
- IS_CONTINUES_MODE
 - BH1750_priv.h, [17](#)
- IS_INITIALIZED
 - BH1750_priv.h, [18](#)
- IS_LOW_RESOLUTION
 - BH1750_priv.h, [18](#)
- IS_ONE_TIME_MODE
 - BH1750_priv.h, [18](#)
- isConversionCompleted
 - BH1750, [12](#)
- ModeContinuous
 - BH1750.h, [16](#)
- ModeOneTime
 - BH1750.h, [16](#)
- read
 - BH1750, [12](#)
- ResolutionHigh
 - BH1750.h, [16](#)
- ResolutionLow
 - BH1750.h, [16](#)
- ResolutionMid
 - BH1750.h, [16](#)
- waitForCompletion
 - BH1750, [13](#)
- writeInstruction
 - BH1750, [13](#)