Erriez BH1750 library for Arduino

1.1.1

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 BH1750 Class Reference

BH1750 class.

```
#include <ErriezBH1750.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 ErriezBH1750.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BH1750()

```
BH1750::BH1750 (
            uint8_t addrPinLevel = LOW )  [explicit]
```

Constructor.

Parameters

addrPinLevel	Sensor I2C address: ADDR pin = LOW: 0x23 (default) ADDR pin = HIGH: 0x5C

Definition at line 44 of file ErriezBH1750.cpp.

4.1.3 Member Function Documentation

4.1.3.1 begin()

```
void BH1750::begin (
            BH1750_Mode_e mode,
            BH1750_Resolution_e resolution )
```

Set mode and resolution.

Parameters

mode	ModeContinuous for continues mode Continues conversion requires more power ModeOneTime for one-time conversion mode Set in low power when conversion completed
resolution	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 ErriezBH1750.cpp.

4.1.3.2 isConversionCompleted()

```
bool BH1750::isConversionCompleted ( )
```

Wait for completion.

Returns

true: Conversion completed false: Conversion busy

Definition at line 105 of file ErriezBH1750.cpp.

4.1.3.3 read()

```
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 ErriezBH1750.cpp.

4.1.3.4 waitForCompletion()

```
bool BH1750::waitForCompletion ( )
```

Wait for completion.

Returns

true: Conversion completed false: Not initialized, or timeout

Definition at line 125 of file ErriezBH1750.cpp.

4.1.3.5 writeInstruction()

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

Write instruction to sensor.

Parameters

instruction	Sensor instruction

Definition at line 215 of file ErriezBH1750.cpp.

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

- src/ErriezBH1750.h
- src/ErriezBH1750.cpp

Chapter 5

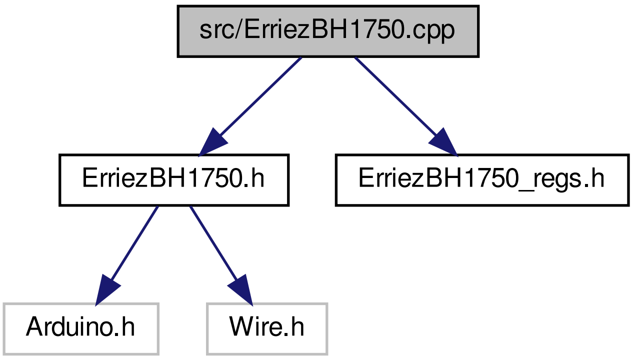
File Documentation

5.1 src/ErriezBH1750.cpp File Reference

BH1750 digital light sensor library for Arduino.

```
#include "ErriezBH1750.h"
#include "ErriezBH1750_regs.h"
```

Include dependency graph for ErriezBH1750.cpp:



5.1.1 Detailed Description

BH1750 digital light sensor library for Arduino.

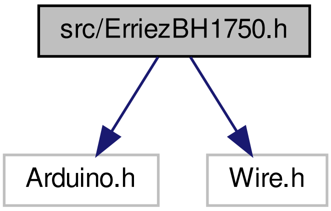
Source: `https://github.com/Erriez/ErriezBH1750` Documentation: `https://erriez.github.io/ErriezBH1750`

5.2 src/ErriezBH1750.h File Reference

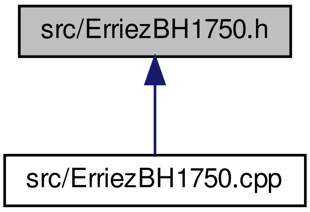
BH1750 digital light sensor library for Arduino.

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

Include dependency graph for ErriezBH1750.h:



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



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 BH1750_Mode_e

```
enum BH1750_Mode_e
```

Mode register bits.

Enumerator

ModeContinuous	Continues mode.
ModeOneTime	One-time mode.

Definition at line 40 of file ErriezBH1750.h.

5.2.2.2 BH1750_Resolution_e

```
enum BH1750_Resolution_e
```

Resolution register bits.

Enumerator

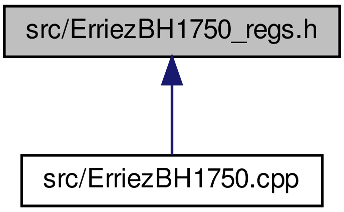
ResolutionLow	4 lx resolution
ResolutionMid	1 lx resolution
ResolutionHigh	0.5 lx resolution

Definition at line 46 of file ErriezBH1750.h.

5.3 src/ErriezBH1750_regs.h File Reference

BH1750 digital light sensor library for Arduino.

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



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.

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

5.3.2 Macro Definition Documentation

5.3.2.1 GET_TIMEOUT

```
#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 81 of file ErriezBH1750_regs.h.

5.3.2.2 IS_CONTINUES_MODE

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

Macro is continues mode enabled

Definition at line 63 of file ErriezBH1750_regs.h.

5.3.2.3 IS_INITIALIZED

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

Return if mode is set (initialized)

Definition at line 57 of file ErriezBH1750_regs.h.

5.3.2.4 IS_LOW_RESOLUTION

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

Macro is low resolution enabled from mode

Definition at line 75 of file ErriezBH1750_regs.h.

5.3.2.5 IS_ONE_TIME_MODE

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

Macro is one-time mode enabled from mode

Definition at line 69 of file ErriezBH1750_regs.h.

Index