

## Lighting control system

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>README</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Module Documentation</b>	<b>11</b>
6.1	<math.h>: Mathematics . . . . .	11
6.1.1	Detailed Description . . . . .	13
6.1.2	Macro Definition Documentation . . . . .	13
6.1.2.1	__ATTR_CONST__ . . . . .	13
6.1.2.2	acosf . . . . .	13
6.1.2.3	asinf . . . . .	13
6.1.2.4	atan2f . . . . .	13
6.1.2.5	atanf . . . . .	14
6.1.2.6	cbrtf . . . . .	14
6.1.2.7	ceilf . . . . .	14
6.1.2.8	copysignf . . . . .	14

6.1.2.9	<code>cosf</code>	14
6.1.2.10	<code>coshf</code>	14
6.1.2.11	<code>expf</code>	14
6.1.2.12	<code>fabsf</code>	14
6.1.2.13	<code>fdimf</code>	15
6.1.2.14	<code>floorf</code>	15
6.1.2.15	<code>fmaf</code>	15
6.1.2.16	<code>fmaxf</code>	15
6.1.2.17	<code>fminf</code>	15
6.1.2.18	<code>fmodf</code>	15
6.1.2.19	<code>frexpf</code>	15
6.1.2.20	<code>hypotf</code>	15
6.1.2.21	<code>INFINITY</code>	16
6.1.2.22	<code>isfinitef</code>	16
6.1.2.23	<code>isinf</code>	16
6.1.2.24	<code>isnanf</code>	16
6.1.2.25	<code>ldexpf</code>	16
6.1.2.26	<code>log10f</code>	16
6.1.2.27	<code>logf</code>	16
6.1.2.28	<code>lrintf</code>	16
6.1.2.29	<code>lroundf</code>	17
6.1.2.30	<code>M_1_PI</code>	17
6.1.2.31	<code>M_2_PI</code>	17
6.1.2.32	<code>M_2_SQRTPI</code>	17
6.1.2.33	<code>M_E</code>	17
6.1.2.34	<code>M_LN10</code>	17
6.1.2.35	<code>M_LN2</code>	17
6.1.2.36	<code>M_LOG10E</code>	17
6.1.2.37	<code>M_LOG2E</code>	18
6.1.2.38	<code>M_PI</code>	18

6.1.2.39	M_PI_2	18
6.1.2.40	M_PI_4	18
6.1.2.41	M_SQRT1_2	18
6.1.2.42	M_SQRT2	18
6.1.2.43	NAN	18
6.1.2.44	powf	18
6.1.2.45	roundf	19
6.1.2.46	signbitf	19
6.1.2.47	sinf	19
6.1.2.48	sinhf	19
6.1.2.49	squaref	19
6.1.2.50	tanf	19
6.1.2.51	tanhf	19
6.1.2.52	truncf	19
6.1.3	Function Documentation	20
6.1.3.1	acos()	20
6.1.3.2	asin()	20
6.1.3.3	atan()	20
6.1.3.4	atan2()	20
6.1.3.5	cbrt()	20
6.1.3.6	ceil()	21
6.1.3.7	cos()	21
6.1.3.8	cosh()	21
6.1.3.9	exp()	21
6.1.3.10	fabs()	21
6.1.3.11	fdim()	21
6.1.3.12	floor()	22
6.1.3.13	fma()	22
6.1.3.14	fmax()	22
6.1.3.15	fmin()	22

6.1.3.16	fmod()	22
6.1.3.17	frexp()	23
6.1.3.18	hypot()	23
6.1.3.19	isinf()	23
6.1.3.20	isnan()	23
6.1.3.21	ldexp()	24
6.1.3.22	log()	24
6.1.3.23	log10()	24
6.1.3.24	lrint()	24
6.1.3.25	lround()	24
6.1.3.26	modf()	25
6.1.3.27	modff()	25
6.1.3.28	pow()	25
6.1.3.29	round()	25
6.1.3.30	signbit()	26
6.1.3.31	sin()	26
6.1.3.32	sinh()	26
6.1.3.33	sqrt()	26
6.1.3.34	sqrtf()	26
6.1.3.35	square()	26
6.1.3.36	tan()	27
6.1.3.37	tanh()	27
6.1.3.38	trunc()	27

<b>7</b>	<b>Class Documentation</b>	<b>29</b>
7.1	TwoWire Class Reference	29
7.1.1	Constructor & Destructor Documentation	30
7.1.1.1	TwoWire()	30
7.1.2	Member Function Documentation	30
7.1.2.1	available()	30
7.1.2.2	begin() [1/3]	30
7.1.2.3	begin() [2/3]	30
7.1.2.4	begin() [3/3]	30
7.1.2.5	beginTransaction() [1/2]	30
7.1.2.6	beginTransaction() [2/2]	31
7.1.2.7	end()	31
7.1.2.8	endTransmission() [1/2]	31
7.1.2.9	endTransmission() [2/2]	31
7.1.2.10	flush()	31
7.1.2.11	onReceive()	31
7.1.2.12	onRequest()	31
7.1.2.13	peek()	32
7.1.2.14	read()	32
7.1.2.15	requestFrom() [1/5]	32
7.1.2.16	requestFrom() [2/5]	32
7.1.2.17	requestFrom() [3/5]	32
7.1.2.18	requestFrom() [4/5]	32
7.1.2.19	requestFrom() [5/5]	33
7.1.2.20	setClock()	33
7.1.2.21	write() [1/6]	33
7.1.2.22	write() [2/6]	33
7.1.2.23	write() [3/6]	33
7.1.2.24	write() [4/6]	33
7.1.2.25	write() [5/6]	33
7.1.2.26	write() [6/6]	33

<b>8 File Documentation</b>	<b>35</b>
8.1 D:/GIT/TheConnectedMCU_Labs/bkarachok/Final_work_IoT/light_control_system.c File Reference	35
8.1.1 Function Documentation	35
8.1.1.1 allpinslow()	35
8.1.1.2 BH1750_Read()	36
8.1.1.3 Check_Protocol()	36
8.1.1.4 Configure_BH1750()	36
8.1.1.5 loop()	36
8.1.1.6 setup()	36
8.1.2 Variable Documentation	36
8.1.2.1 BH1750_Device	36
8.1.2.2 inSerial	36
8.1.2.3 lamp	37
8.1.2.4 Lux	37
8.1.2.5 sm_mode	37
8.2 D:/GIT/TheConnectedMCU_Labs/bkarachok/Final_work_IoT/math.h File Reference	37
8.3 D:/GIT/TheConnectedMCU_Labs/bkarachok/Final_work_IoT/README.md File Reference	39
8.4 D:/GIT/TheConnectedMCU_Labs/bkarachok/Final_work_IoT/Wire.h File Reference	39
8.4.1 Macro Definition Documentation	39
8.4.1.1 BUFFER_LENGTH	39
8.4.1.2 WIRE_HAS_END	40
8.4.2 Variable Documentation	40
8.4.2.1 Wire	40
8.5 D:/GIT/TheConnectedMCU_Labs/bkarachok/Final_work_IoT/WString.h File Reference	40
<b>Index</b>	<b>41</b>



# Chapter 1

## README

In this project, a lighting control system with IoT is implemented. The device consists of a chipkit WiFire board with PIC32MZ microcontroller, bluetooth module HC-05, light sensor BH1750. The lamp uses the LD1 LED on the board. Control is carried out using the Android application on the smartphone via bluetooth. The screen displays the value of the illumination level, which is measured by the sensor. There are 2 modes of control - manual and smart. In manual mode, you can turn the lamp on and off with the buttons. To activate smart mode, you must press the Smart Mode button. In smart mode, the lamp is on if the illumination level is less than 20 lux, otherwise the lamp is turned off.

For programming, the Arduino IDE was used. To configure the Arduino IDE, you need to follow the instructions in the guides: <https://www.youtube.com/watch?v=DOEdmc57FVU> and [https://chipkit.net/wiki/index.php?title=ChipKIT\\_core](https://chipkit.net/wiki/index.php?title=ChipKIT_core). You also need to download and connect the libraries: [math.h](#), [Wire.h](#), [WString.h](#).



## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

<code>&lt;math.h&gt;</code> : Mathematics . . . . .	11
---	----



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Stream	
TwoWire . . . . .	<a href="#">29</a>



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">TwoWire</a> . . . . .	<a href="#">29</a>
-----------------------------------	--------------------





## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

D:/GIT/TheConnectedMCU_Labs/bkarachok/Final_work_IoT/ <a href="#">light_control_system.c</a> . . . . .	35
D:/GIT/TheConnectedMCU_Labs/bkarachok/Final_work_IoT/ <a href="#">math.h</a> . . . . .	37
D:/GIT/TheConnectedMCU_Labs/bkarachok/Final_work_IoT/ <a href="#">Wire.h</a> . . . . .	39
D:/GIT/TheConnectedMCU_Labs/bkarachok/Final_work_IoT/ <a href="#">WString.h</a> . . . . .	40



## Chapter 6

# Module Documentation

### 6.1 `<math.h>`: Mathematics

#### Macros

- `#define M_E 2.7182818284590452354`
- `#define M_LOG2E 1.4426950408889634074 /* log_2 e */`
- `#define M_LOG10E 0.43429448190325182765 /* log_10 e */`
- `#define M_LN2 0.69314718055994530942 /* log_e 2 */`
- `#define M_LN10 2.30258509299404568402 /* log_e 10 */`
- `#define M_PI 3.14159265358979323846 /* pi */`
- `#define M_PI_2 1.57079632679489661923 /* pi/2 */`
- `#define M_PI_4 0.78539816339744830962 /* pi/4 */`
- `#define M_1_PI 0.31830988618379067154 /* 1/pi */`
- `#define M_2_PI 0.63661977236758134308 /* 2/pi */`
- `#define M_2_SQRTPI 1.12837916709551257390 /* 2/sqrt(pi) */`
- `#define M_SQRT2 1.41421356237309504880 /* sqrt(2) */`
- `#define M_SQRT1_2 0.70710678118654752440 /* 1/sqrt(2) */`
- `#define NAN __builtin_nan("")`
- `#define INFINITY __builtin_inf()`
- `#define __ATTR_CONST__ attribute__((__const__))`
- `#define cosf cos`
- `#define sinf sin`
- `#define tanf tan`
- `#define fabsf fabs`
- `#define fmodf fmod`
- `#define cbrtf cbrt`
- `#define hypotf hypot`
- `#define squaref square`
- `#define floorf floor`
- `#define ceilf ceil`
- `#define frexpf frexp`
- `#define ldexpf ldexp`
- `#define expf exp`
- `#define coshf cosh`
- `#define sinhf sinh`
- `#define tanhf tanh`
- `#define acosf acos`

- #define [asinf](#) [asin](#)
- #define [atanf](#) [atan](#)
- #define [atan2f](#) [atan2](#)
- #define [logf](#) [log](#)
- #define [log10f](#) [log10](#)
- #define [powf](#) [pow](#)
- #define [isnanf](#) [isnan](#)
- #define [isinf](#) [isinf](#)
- #define [isfinitef](#) [isfinite](#)
- #define [copysignf](#) [copysign](#)
- #define [signbitf](#) [signbit](#)
- #define [fdimf](#) [fdim](#)
- #define [fmaf](#) [fma](#)
- #define [fmaxf](#) [fmax](#)
- #define [fminf](#) [fmin](#)
- #define [truncf](#) [trunc](#)
- #define [roundf](#) [round](#)
- #define [lroundf](#) [lround](#)
- #define [lrintf](#) [lrint](#)

## Functions

- double [cos](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [sin](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [tan](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fabs](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fmod](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [modf](#) (double \_\_x, double \*\_\_iptr)
- float [modff](#) (float \_\_x, float \*\_\_iptr)
- double [sqrt](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- float [sqrtf](#) (float) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [cbrt](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [hypot](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [square](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [floor](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [ceil](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [frexp](#) (double \_\_x, int \*\_\_pexp)
- double [ldexp](#) (double \_\_x, int \_\_exp) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [exp](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [cosh](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [sinh](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [tanh](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [acos](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [asin](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [atan](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [atan2](#) (double \_\_y, double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [log](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [log10](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [pow](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- int [isnan](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- int [isinf](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- int [signbit](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fdim](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fma](#) (double \_\_x, double \_\_y, double \_\_z) [\\_\\_ATTR\\_CONST\\_\\_](#)

- double `fmax` (double \_\_x, double \_\_y) `__ATTR_CONST__`
- double `fmin` (double \_\_x, double \_\_y) `__ATTR_CONST__`
- double `trunc` (double \_\_x) `__ATTR_CONST__`
- double `round` (double \_\_x) `__ATTR_CONST__`
- long `lround` (double \_\_x) `__ATTR_CONST__`
- long `lrint` (double \_\_x) `__ATTR_CONST__`

### 6.1.1 Detailed Description

```
#include <math.h>
```

This header file declares basic mathematics constants and functions.

#### Notes:

- In order to access the functions declared herein, it is usually also required to additionally link against the library `libm.a`. See also the related FAQ entry.
- Math functions do not raise exceptions and do not change the `errno` variable. Therefore the majority of them are declared with `const` attribute, for better optimization by GCC.

### 6.1.2 Macro Definition Documentation

#### 6.1.2.1 `__ATTR_CONST__`

```
#define __ATTR_CONST__ __attribute__((__const__))
```

#### 6.1.2.2 `acosf`

```
#define acosf acos
```

The alias for `acos()`.

#### 6.1.2.3 `asinf`

```
#define asinf asin
```

The alias for `asin()`.

#### 6.1.2.4 `atan2f`

```
#define atan2f atan2
```

The alias for `atan2()`.

#### 6.1.2.5 atanf

```
#define atanf atan
```

The alias for [atan\(\)](#).

#### 6.1.2.6 cbrtf

```
#define cbrtf cbrt
```

The alias for [cbrt\(\)](#).

#### 6.1.2.7 ceilf

```
#define ceilf ceil
```

The alias for [ceil\(\)](#).

#### 6.1.2.8 copysignf

```
#define copysignf copysign
```

The alias for [copysign\(\)](#).

#### 6.1.2.9 cosf

```
#define cosf cos
```

The alias for [cos\(\)](#).

#### 6.1.2.10 coshf

```
#define coshf cosh
```

The alias for [cosh\(\)](#).

#### 6.1.2.11 expf

```
#define expf exp
```

The alias for [exp\(\)](#).

#### 6.1.2.12 fabsf

```
#define fabsf fabs
```

The alias for [fabs\(\)](#).

#### 6.1.2.13 fdimf

```
#define fdimf fdim
```

The alias for `fdim()`.

#### 6.1.2.14 floorf

```
#define floorf floor
```

The alias for `floor()`.

#### 6.1.2.15 fmaf

```
#define fmaf fma
```

The alias for `fma()`.

#### 6.1.2.16 fmaxf

```
#define fmaxf fmax
```

The alias for `fmax()`.

#### 6.1.2.17 fminf

```
#define fminf fmin
```

The alias for `fmin()`.

#### 6.1.2.18 fmodf

```
#define fmodf fmod
```

The alias for `fmod()`.

#### 6.1.2.19 frexpf

```
#define frexpf frexp
```

The alias for `frexp()`.

#### 6.1.2.20 hypotf

```
#define hypotf hypot
```

The alias for `hypot()`.

#### 6.1.2.21 INFINITY

```
#define INFINITY __builtin_inf()
```

INFINITY constant.

#### 6.1.2.22 isfinitef

```
#define isfinitef isfinite
```

The alias for `isfinite()`.

#### 6.1.2.23 isinff

```
#define isinff isinf
```

The alias for `isinf()`.

#### 6.1.2.24 isnanf

```
#define isnanf isnan
```

The alias for `isnan()`.

#### 6.1.2.25 ldexpf

```
#define ldexpf ldexp
```

The alias for `ldexp()`.

#### 6.1.2.26 log10f

```
#define log10f log10
```

The alias for `log10()`.

#### 6.1.2.27 logf

```
#define logf log
```

The alias for `log()`.

#### 6.1.2.28 lrintf

```
#define lrintf lrint
```

The alias for `lrint()`.



### 6.1.2.29 lroundf

```
#define lroundf lround
```

The alias for `lround()`.

### 6.1.2.30 M\_1\_PI

```
#define M_1_PI 0.31830988618379067154 /* 1/pi */
```

The constant  $1/\pi$ .

### 6.1.2.31 M\_2\_PI

```
#define M_2_PI 0.63661977236758134308 /* 2/pi */
```

The constant  $2/\pi$ .

### 6.1.2.32 M\_2\_SQRTPI

```
#define M_2_SQRTPI 1.12837916709551257390 /* 2/sqrt(pi) */
```

The constant  $2/\sqrt{\pi}$ .

### 6.1.2.33 M\_E

```
#define M_E 2.7182818284590452354
```

The constant  $e$ .

### 6.1.2.34 M\_LN10

```
#define M_LN10 2.30258509299404568402 /* log_e 10 */
```

The natural logarithm of the 10.

### 6.1.2.35 M\_LN2

```
#define M_LN2 0.69314718055994530942 /* log_e 2 */
```

The natural logarithm of the 2.

### 6.1.2.36 M\_LOG10E

```
#define M_LOG10E 0.43429448190325182765 /* log_10 e */
```

The logarithm of the  $e$  to base 10.

#### 6.1.2.37 M\_LOG2E

```
#define M_LOG2E 1.4426950408889634074 /* log_2 e */
```

The logarithm of the  $e$  to base 2.

#### 6.1.2.38 M\_PI

```
#define M_PI 3.14159265358979323846 /* pi */
```

The constant  $\pi$ .

#### 6.1.2.39 M\_PI\_2

```
#define M_PI_2 1.57079632679489661923 /* pi/2 */
```

The constant  $\pi/2$ .

#### 6.1.2.40 M\_PI\_4

```
#define M_PI_4 0.78539816339744830962 /* pi/4 */
```

The constant  $\pi/4$ .

#### 6.1.2.41 M\_SQRT1\_2

```
#define M_SQRT1_2 0.70710678118654752440 /* 1/sqrt(2) */
```

The constant  $1/\sqrt{2}$ .

#### 6.1.2.42 M\_SQRT2

```
#define M_SQRT2 1.41421356237309504880 /* sqrt(2) */
```

The square root of 2.

#### 6.1.2.43 NAN

```
#define NAN __builtin_nan("")
```

NAN constant.

#### 6.1.2.44 powf

```
#define powf pow
```

The alias for `pow()`.

**6.1.2.45 roundf**

```
#define roundf round
```

The alias for `round()`.

**6.1.2.46 signbitf**

```
#define signbitf signbit
```

The alias for `signbit()`.

**6.1.2.47 sinf**

```
#define sinf sin
```

The alias for `sin()`.

**6.1.2.48 sinhf**

```
#define sinhf sinh
```

The alias for `sinh()`.

**6.1.2.49 squaref**

```
#define squaref square
```

The alias for `square()`.

**6.1.2.50 tanf**

```
#define tanf tan
```

The alias for `tan()`.

**6.1.2.51 tanhf**

```
#define tanhf tanh
```

The alias for `tanh()`.

**6.1.2.52 truncf**

```
#define truncf trunc
```

The alias for `trunc()`.

## 6.1.3 Function Documentation

### 6.1.3.1 `acos()`

```
double acos (
    double __x )
```

The `acos()` function computes the principal value of the arc cosine of `__x`. The returned value is in the range  $[0, \pi]$  radians. A domain error occurs for arguments not in the range  $[-1, +1]$ .

### 6.1.3.2 `asin()`

```
double asin (
    double __x )
```

The `asin()` function computes the principal value of the arc sine of `__x`. The returned value is in the range  $[-\pi/2, \pi/2]$  radians. A domain error occurs for arguments not in the range  $[-1, +1]$ .

### 6.1.3.3 `atan()`

```
double atan (
    double __x )
```

The `atan()` function computes the principal value of the arc tangent of `__x`. The returned value is in the range  $[-\pi/2, \pi/2]$  radians.

### 6.1.3.4 `atan2()`

```
double atan2 (
    double __y,
    double __x )
```

The `atan2()` function computes the principal value of the arc tangent of `__y / __x`, using the signs of both arguments to determine the quadrant of the return value. The returned value is in the range  $[-\pi, +\pi]$  radians.

### 6.1.3.5 `cbrt()`

```
double cbrt (
    double __x )
```

The `cbrt()` function returns the cube root of `__x`.

#### 6.1.3.6 ceil()

```
double ceil (
    double __x )
```

The `ceil()` function returns the smallest integral value greater than or equal to `__x`, expressed as a floating-point number.

#### 6.1.3.7 cos()

```
double cos (
    double __x )
```

The `cos()` function returns the cosine of `__x`, measured in radians.

#### 6.1.3.8 cosh()

```
double cosh (
    double __x )
```

The `cosh()` function returns the hyperbolic cosine of `__x`.

#### 6.1.3.9 exp()

```
double exp (
    double __x )
```

The `exp()` function returns the exponential value of `__x`.

#### 6.1.3.10 fabs()

```
double fabs (
    double __x )
```

The `fabs()` function computes the absolute value of a floating-point number `__x`.

#### 6.1.3.11 fdim()

```
double fdim (
    double __x,
    double __y )
```

The `fdim()` function returns  $\max(\_\_x - \_\_y, 0)$ . If `__x` or `__y` or both are NaN, NaN is returned.

#### 6.1.3.12 floor()

```
double floor (  
    double __x )
```

The `floor()` function returns the largest integral value less than or equal to `__x`, expressed as a floating-point number.

#### 6.1.3.13 fma()

```
double fma (  
    double __x,  
    double __y,  
    double __z )
```

The `fma()` function performs floating-point multiply-add. This is the operation  $(\text{__x} * \text{__y}) + \text{__z}$ , but the intermediate result is not rounded to the destination type. This can sometimes improve the precision of a calculation.

#### 6.1.3.14 fmax()

```
double fmax (  
    double __x,  
    double __y )
```

The `fmax()` function returns the greater of the two values `__x` and `__y`. If an argument is NaN, the other argument is returned. If both arguments are NaN, NaN is returned.

#### 6.1.3.15 fmin()

```
double fmin (  
    double __x,  
    double __y )
```

The `fmin()` function returns the lesser of the two values `__x` and `__y`. If an argument is NaN, the other argument is returned. If both arguments are NaN, NaN is returned.

#### 6.1.3.16 fmod()

```
double fmod (  
    double __x,  
    double __y )
```

The function `fmod()` returns the floating-point remainder of `__x / __y`.

## 6.1.3.17 frexp()

```
double frexp (
    double __x,
    int * __pexp )
```

The `frexp()` function breaks a floating-point number into a normalized fraction and an integral power of 2. It stores the integer in the `int` object pointed to by `__pexp`.

If `__x` is a normal float point number, the `frexp()` function returns the value  $v$ , such that  $v$  has a magnitude in the interval  $[1/2, 1)$  or zero, and `__x` equals  $v$  times 2 raised to the power `__pexp`. If `__x` is zero, both parts of the result are zero. If `__x` is not a finite number, the `frexp()` returns `__x` as is and stores 0 by `__pexp`.

## Note

This implementation permits a zero pointer as a directive to skip storing the exponent.

## 6.1.3.18 hypot()

```
double hypot (
    double __x,
    double __y )
```

The `hypot()` function returns  $\sqrt{__x*__x + __y*__y}$ . This is the length of the hypotenuse of a right triangle with sides of length `__x` and `__y`, or the distance of the point  $(__x, __y)$  from the origin. Using this function instead of the direct formula is wise, since the error is much smaller. No underflow with small `__x` and `__y`. No overflow if result is in range.

## 6.1.3.19 isinf()

```
int isinf (
    double __x )
```

The function `isinf()` returns 1 if the argument `__x` is positive infinity, -1 if `__x` is negative infinity, and 0 otherwise.

## Note

The GCC 4.3 can replace this function with inline code that returns the 1 value for both infinities (gcc bug #35509).

## 6.1.3.20 isnan()

```
int isnan (
    double __x )
```

The function `isnan()` returns 1 if the argument `__x` represents a "not-a-number" (NaN) object, otherwise 0.

#### 6.1.3.21 ldexp()

```
double ldexp (
    double __x,
    int __exp )
```

The `ldexp()` function multiplies a floating-point number by an integral power of 2. It returns the value of `__x` times 2 raised to the power `__exp`.

#### 6.1.3.22 log()

```
double log (
    double __x )
```

The `log()` function returns the natural logarithm of argument `__x`.

#### 6.1.3.23 log10()

```
double log10 (
    double __x )
```

The `log10()` function returns the logarithm of argument `__x` to base 10.

#### 6.1.3.24 lrint()

```
long lrint (
    double __x )
```

The `lrint()` function rounds `__x` to the nearest integer, rounding the halfway cases to the even integer direction. (That is both 1.5 and 2.5 values are rounded to 2). This function is similar to `rint()` function, but it differs in type of return value and in that an overflow is possible.

##### Returns

The rounded long integer value. If `__x` is not a finite number or an overflow was, this realization returns the `LONG_MIN` value (0x80000000).

#### 6.1.3.25 lround()

```
long lround (
    double __x )
```

The `lround()` function rounds `__x` to the nearest integer, but rounds halfway cases away from zero (instead of to the nearest even integer). This function is similar to `round()` function, but it differs in type of return value and in that an overflow is possible.

##### Returns

The rounded long integer value. If `__x` is not a finite number or an overflow was, this realization returns the `LONG_MIN` value (0x80000000).



**6.1.3.26 modf()**

```
double modf (
    double __x,
    double * __iptr )
```

The `modf()` function breaks the argument `__x` into integral and fractional parts, each of which has the same sign as the argument. It stores the integral part as a double in the object pointed to by `__iptr`.

The `modf()` function returns the signed fractional part of `__x`.

**Note**

This implementation skips writing by zero pointer. However, the GCC 4.3 can replace this function with inline code that does not permit to use NULL address for the avoiding of storing.

**6.1.3.27 modff()**

```
float modff (
    float __x,
    float * __iptr )
```

An alias for `modf()`.

**6.1.3.28 pow()**

```
double pow (
    double __x,
    double __y )
```

The function `pow()` returns the value of `__x` to the exponent `__y`.

**6.1.3.29 round()**

```
double round (
    double __x )
```

The `round()` function rounds `__x` to the nearest integer, but rounds halfway cases away from zero (instead of to the nearest even integer). Overflow is impossible.

**Returns**

The rounded value. If `__x` is an integral or infinite, `__x` itself is returned. If `__x` is NaN, then NaN is returned.

#### 6.1.3.30 signbit()

```
int signbit (  
    double __x )
```

The `signbit()` function returns a nonzero value if the value of `__x` has its sign bit set. This is not the same as '`__x < 0.0`', because IEEE 754 floating point allows zero to be signed. The comparison '`-0.0 < 0.0`' is false, but '`signbit (-0.0)`' will return a nonzero value.

#### 6.1.3.31 sin()

```
double sin (  
    double __x )
```

The `sin()` function returns the sine of `__x`, measured in radians.

#### 6.1.3.32 sinh()

```
double sinh (  
    double __x )
```

The `sinh()` function returns the hyperbolic sine of `__x`.

#### 6.1.3.33 sqrt()

```
double sqrt (  
    double __x )
```

The `sqrt()` function returns the non-negative square root of `__x`.

#### 6.1.3.34 sqrtf()

```
float sqrtf (  
    float )
```

An alias for `sqrt()`.

#### 6.1.3.35 square()

```
double square (  
    double __x )
```

The function `square()` returns `__x * __x`.

#### Note

This function does not belong to the C standard definition.

**6.1.3.36 tan()**

```
double tan (  
    double __x )
```

The `tan()` function returns the tangent of `__x`, measured in radians.

**6.1.3.37 tanh()**

```
double tanh (  
    double __x )
```

The `tanh()` function returns the hyperbolic tangent of `__x`.

**6.1.3.38 trunc()**

```
double trunc (  
    double __x )
```

The `trunc()` function rounds `__x` to the nearest integer not larger in absolute value.



## Chapter 7

# Class Documentation

### 7.1 TwoWire Class Reference

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

Inherits Stream.

#### Public Member Functions

- [TwoWire](#) ()
- void [begin](#) ()
- void [begin](#) (uint8\_t)
- void [begin](#) (int)
- void [end](#) ()
- void [setClock](#) (uint32\_t)
- void [beginTransaction](#) (uint8\_t)
- void [beginTransaction](#) (int)
- uint8\_t [endTransmission](#) (void)
- uint8\_t [endTransmission](#) (uint8\_t)
- uint8\_t [requestFrom](#) (uint8\_t, uint8\_t)
- uint8\_t [requestFrom](#) (uint8\_t, uint8\_t, uint8\_t)
- uint8\_t [requestFrom](#) (uint8\_t, uint8\_t, uint32\_t, uint8\_t, uint8\_t)
- uint8\_t [requestFrom](#) (int, int)
- uint8\_t [requestFrom](#) (int, int, int)
- virtual size\_t [write](#) (uint8\_t)
- virtual size\_t [write](#) (const uint8\_t \*, size\_t)
- virtual int [available](#) (void)
- virtual int [read](#) (void)
- virtual int [peek](#) (void)
- virtual void [flush](#) (void)
- void [onReceive](#) (void(\*) (int))
- void [onRequest](#) (void(\*) (void))
- size\_t [write](#) (unsigned long n)
- size\_t [write](#) (long n)
- size\_t [write](#) (unsigned int n)
- size\_t [write](#) (int n)

## 7.1.1 Constructor & Destructor Documentation

### 7.1.1.1 TwoWire()

```
TwoWire::TwoWire ( )
```

## 7.1.2 Member Function Documentation

### 7.1.2.1 available()

```
virtual int TwoWire::available (
    void ) [virtual]
```

### 7.1.2.2 begin() [1/3]

```
void TwoWire::begin ( )
```

### 7.1.2.3 begin() [2/3]

```
void TwoWire::begin (
    uint8_t )
```

### 7.1.2.4 begin() [3/3]

```
void TwoWire::begin (
    int )
```

### 7.1.2.5 beginTransmission() [1/2]

```
void TwoWire::beginTransmission (
    uint8_t )
```

#### 7.1.2.6 beginTransmission() [2/2]

```
void TwoWire::beginTransmission (
    int )
```

#### 7.1.2.7 end()

```
void TwoWire::end ( )
```

#### 7.1.2.8 endTransmission() [1/2]

```
uint8_t TwoWire::endTransmission (
    void )
```

#### 7.1.2.9 endTransmission() [2/2]

```
uint8_t TwoWire::endTransmission (
    uint8_t )
```

#### 7.1.2.10 flush()

```
virtual void TwoWire::flush (
    void ) [virtual]
```

#### 7.1.2.11 onReceive()

```
void TwoWire::onReceive (
    void(*) (int) )
```

#### 7.1.2.12 onRequest()

```
void TwoWire::onRequest (
    void(*) (void) )
```

**7.1.2.13 peek()**

```
virtual int TwoWire::peek (
    void ) [virtual]
```

**7.1.2.14 read()**

```
virtual int TwoWire::read (
    void ) [virtual]
```

**7.1.2.15 requestFrom()** [1/5]

```
uint8_t TwoWire::requestFrom (
    uint8_t ,
    uint8_t )
```

**7.1.2.16 requestFrom()** [2/5]

```
uint8_t TwoWire::requestFrom (
    uint8_t ,
    uint8_t ,
    uint8_t )
```

**7.1.2.17 requestFrom()** [3/5]

```
uint8_t TwoWire::requestFrom (
    uint8_t ,
    uint8_t ,
    uint32_t ,
    uint8_t ,
    uint8_t )
```

**7.1.2.18 requestFrom()** [4/5]

```
uint8_t TwoWire::requestFrom (
    int ,
    int )
```



**7.1.2.19 requestFrom()** [5/5]

```
uint8_t TwoWire::requestFrom (
    int ,
    int ,
    int )
```

**7.1.2.20 setClock()**

```
void TwoWire::setClock (
    uint32_t )
```

**7.1.2.21 write()** [1/6]

```
virtual size_t TwoWire::write (
    uint8_t ) [virtual]
```

**7.1.2.22 write()** [2/6]

```
virtual size_t TwoWire::write (
    const uint8_t * ,
    size_t ) [virtual]
```

**7.1.2.23 write()** [3/6]

```
size_t TwoWire::write (
    unsigned long n ) [inline]
```

**7.1.2.24 write()** [4/6]

```
size_t TwoWire::write (
    long n ) [inline]
```

**7.1.2.25 write()** [5/6]

```
size_t TwoWire::write (
    unsigned int n ) [inline]
```

**7.1.2.26 write()** [6/6]

```
size_t TwoWire::write (
    int n ) [inline]
```

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

- D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_work\_IoT/[Wire.h](#)



## Chapter 8

# File Documentation

### 8.1 D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_work\_IoT/light\_control\_system.c File Reference

```
#include <Wire.h>
#include <math.h>
#include <WString.h>
```

#### Functions

- void [setup](#) ()
- void [loop](#) ()
- void [allpinslow](#) ()
- void [Check\\_Protocol](#) (char inStr[])
- unsigned int [BH1750\\_Read](#) ()
- void [Configure\\_BH1750](#) ()

#### Variables

- int [BH1750\\_Device](#) = 0x23
- unsigned int [Lux](#)
- int [lamp](#) = 13
- int [sm\\_mode](#) = 0
- char [inSerial](#) [15]

#### 8.1.1 Function Documentation

##### 8.1.1.1 [allpinslow](#)()

```
void allpinslow ( )
```

#### 8.1.1.2 BH1750\_Read()

```
unsigned int BH1750_Read ( )
```

#### 8.1.1.3 Check\_Protocol()

```
void Check_Protocol (
    char inStr[] )
```

#### 8.1.1.4 Configure\_BH1750()

```
void Configure_BH1750 ( )
```

#### 8.1.1.5 loop()

```
void loop ( )
```

#### 8.1.1.6 setup()

```
void setup ( )
```

### 8.1.2 Variable Documentation

#### 8.1.2.1 BH1750\_Device

```
int BH1750_Device = 0x23
```

#### 8.1.2.2 inSerial

```
char inSerial[15]
```

## 8.1.2.3 lamp

```
int lamp = 13
```

## 8.1.2.4 Lux

```
unsigned int Lux
```

## 8.1.2.5 sm\_mode

```
int sm_mode = 0
```

## 8.2 D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_work\_IoT/math.h File Reference

## Macros

- `#define M_E 2.7182818284590452354`
- `#define M_LOG2E 1.4426950408889634074 /* log_2 e */`
- `#define M_LOG10E 0.43429448190325182765 /* log_10 e */`
- `#define M_LN2 0.69314718055994530942 /* log_e 2 */`
- `#define M_LN10 2.30258509299404568402 /* log_e 10 */`
- `#define M_PI 3.14159265358979323846 /* pi */`
- `#define M_PI_2 1.57079632679489661923 /* pi/2 */`
- `#define M_PI_4 0.78539816339744830962 /* pi/4 */`
- `#define M_1_PI 0.31830988618379067154 /* 1/pi */`
- `#define M_2_PI 0.63661977236758134308 /* 2/pi */`
- `#define M_2_SQRTPI 1.12837916709551257390 /* 2/sqrt(pi) */`
- `#define M_SQRT2 1.41421356237309504880 /* sqrt(2) */`
- `#define M_SQRT1_2 0.70710678118654752440 /* 1/sqrt(2) */`
- `#define NAN __builtin_nan("")`
- `#define INFINITY __builtin_inf()`
- `#define __ATTR_CONST__ __attribute__((__const__))`
- `#define cosf cos`
- `#define sinf sin`
- `#define tanf tan`
- `#define fabsf fabs`
- `#define fmodf fmod`
- `#define cbrtf cbrt`
- `#define hypotf hypot`
- `#define squaref square`
- `#define floorf floor`
- `#define ceilf ceil`
- `#define frexpf frexp`
- `#define ldexpf ldexp`
- `#define expf exp`

- #define [coshf cosh](#)
- #define [sinhf sinh](#)
- #define [tanhf tanh](#)
- #define [acosf acos](#)
- #define [asinf asin](#)
- #define [atanf atan](#)
- #define [atan2f atan2](#)
- #define [logf log](#)
- #define [log10f log10](#)
- #define [powf pow](#)
- #define [isnanf isnan](#)
- #define [isinf isinf](#)
- #define [isfinitef isfinite](#)
- #define [copysignf copysign](#)
- #define [signbitf signbit](#)
- #define [fdimf fdim](#)
- #define [fmaf fma](#)
- #define [fmaxf fmax](#)
- #define [fminf fmin](#)
- #define [truncf trunc](#)
- #define [roundf round](#)
- #define [lroundf lround](#)
- #define [lrintf lrint](#)

## Functions

- double [cos](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [sin](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [tan](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fabs](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fmod](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [modf](#) (double \_\_x, double \*\_\_iptr)
- float [modff](#) (float \_\_x, float \*\_\_iptr)
- double [sqrt](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- float [sqrtf](#) (float) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [cbrt](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [hypot](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [square](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [floor](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [ceil](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [frexp](#) (double \_\_x, int \*\_\_pexp)
- double [ldexp](#) (double \_\_x, int \_\_exp) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [exp](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [cosh](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [sinh](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [tanh](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [acos](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [asin](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [atan](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [atan2](#) (double \_\_y, double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [log](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [log10](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [pow](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- int [isnan](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)

- int [isinf](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- int [signbit](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fdim](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fma](#) (double \_\_x, double \_\_y, double \_\_z) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fmax](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [fmin](#) (double \_\_x, double \_\_y) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [trunc](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- double [round](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- long [lround](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)
- long [lrint](#) (double \_\_x) [\\_\\_ATTR\\_CONST\\_\\_](#)

### 8.3 D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_work\_IoT/README.md File Reference

### 8.4 D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_work\_IoT/Wire.h File Reference

```
#include <inttypes.h>
#include "Stream.h"
```

#### Classes

- class [TwoWire](#)

#### Macros

- [#define BUFFER\\_LENGTH 32](#)
- [#define WIRE\\_HAS\\_END 1](#)

#### Variables

- [TwoWire Wire](#)

#### 8.4.1 Macro Definition Documentation

##### 8.4.1.1 BUFFER\_LENGTH

```
#define BUFFER_LENGTH 32
```

#### 8.4.1.2 WIRE\_HAS\_END

```
#define WIRE_HAS_END 1
```

### 8.4.2 Variable Documentation

#### 8.4.2.1 Wire

`TwoWire` Wire

## 8.5 D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_work\_IoT/WString.h File Reference



# Index

<math.h>: Mathematics, [11](#)

\_\_ATTR\_CONST\_\_, [13](#)

acos, [20](#)

acosf, [13](#)

asin, [20](#)

asinf, [13](#)

atan, [20](#)

atan2, [20](#)

atan2f, [13](#)

atanf, [13](#)

cbrt, [20](#)

cbrtf, [14](#)

ceil, [20](#)

ceilf, [14](#)

copysignf, [14](#)

cos, [21](#)

cosf, [14](#)

cosh, [21](#)

coshf, [14](#)

exp, [21](#)

expf, [14](#)

fabs, [21](#)

fabsf, [14](#)

fdim, [21](#)

fdimf, [14](#)

floor, [21](#)

floorf, [15](#)

fma, [22](#)

fmaf, [15](#)

fmax, [22](#)

fmaxf, [15](#)

fmin, [22](#)

fminf, [15](#)

fmod, [22](#)

fmodf, [15](#)

frexp, [22](#)

frexpf, [15](#)

hypot, [23](#)

hypotf, [15](#)

INFINITY, [15](#)

isfinitef, [16](#)

isinf, [23](#)

isinff, [16](#)

isnan, [23](#)

isnanf, [16](#)

ldexp, [23](#)

ldexpf, [16](#)

log, [24](#)

log10, [24](#)

log10f, [16](#)

logf, [16](#)

lrint, [24](#)

lrintf, [16](#)

lround, [24](#)

lroundf, [16](#)

M\_1\_PI, [17](#)

M\_2\_PI, [17](#)

M\_2\_SQRTPI, [17](#)

M\_LN10, [17](#)

M\_LN2, [17](#)

M\_LOG10E, [17](#)

M\_LOG2E, [17](#)

M\_PI\_2, [18](#)

M\_PI\_4, [18](#)

M\_PI, [18](#)

M\_SQRT1\_2, [18](#)

M\_SQRT2, [18](#)

M\_E, [17](#)

modf, [24](#)

modff, [25](#)

NAN, [18](#)

pow, [25](#)

powf, [18](#)

round, [25](#)

roundf, [18](#)

signbit, [25](#)

signbitf, [19](#)

sin, [26](#)

sinf, [19](#)

sinh, [26](#)

sinhf, [19](#)

sqrt, [26](#)

sqrtf, [26](#)

square, [26](#)

squaref, [19](#)

tan, [26](#)

tanf, [19](#)

tanh, [27](#)

tanhf, [19](#)

trunc, [27](#)

truncf, [19](#)

\_\_ATTR\_CONST\_\_

<math.h>: Mathematics, [13](#)

acos

<math.h>: Mathematics, [20](#)

acosf

<math.h>: Mathematics, [13](#)

allpinslow

- light\_control\_system.c, 35
- asin
  - <math.h>: Mathematics, 20
- asinf
  - <math.h>: Mathematics, 13
- atan
  - <math.h>: Mathematics, 20
- atan2
  - <math.h>: Mathematics, 20
- atan2f
  - <math.h>: Mathematics, 13
- atanf
  - <math.h>: Mathematics, 13
- available
  - TwoWire, 30
- BH1750\_Device
  - light\_control\_system.c, 36
- BH1750\_Read
  - light\_control\_system.c, 35
- BUFFER\_LENGTH
  - Wire.h, 39
- begin
  - TwoWire, 30
- beginTransaction
  - TwoWire, 30
- cbrt
  - <math.h>: Mathematics, 20
- cbrtf
  - <math.h>: Mathematics, 14
- ceil
  - <math.h>: Mathematics, 20
- ceilf
  - <math.h>: Mathematics, 14
- Check\_Protocol
  - light\_control\_system.c, 36
- Configure\_BH1750
  - light\_control\_system.c, 36
- copysignf
  - <math.h>: Mathematics, 14
- cos
  - <math.h>: Mathematics, 21
- cosf
  - <math.h>: Mathematics, 14
- cosh
  - <math.h>: Mathematics, 21
- coshf
  - <math.h>: Mathematics, 14
- D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_↔
  - work\_IoT/README.md, 39
- D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_↔
  - work\_IoT/WString.h, 40
- D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_↔
  - work\_IoT/Wire.h, 39
- D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_↔
  - work\_IoT/light\_control\_system.c, 35
- D:/GIT/TheConnectedMCU\_Labs/bkarachok/Final\_↔
  - work\_IoT/math.h, 37
- end
  - TwoWire, 31
- endTransmission
  - TwoWire, 31
- exp
  - <math.h>: Mathematics, 21
- expf
  - <math.h>: Mathematics, 14
- fabs
  - <math.h>: Mathematics, 21
- fabsf
  - <math.h>: Mathematics, 14
- fdim
  - <math.h>: Mathematics, 21
- fdimf
  - <math.h>: Mathematics, 14
- floor
  - <math.h>: Mathematics, 21
- floorf
  - <math.h>: Mathematics, 15
- flush
  - TwoWire, 31
- fma
  - <math.h>: Mathematics, 22
- fmaf
  - <math.h>: Mathematics, 15
- fmax
  - <math.h>: Mathematics, 22
- fmaxf
  - <math.h>: Mathematics, 15
- fmin
  - <math.h>: Mathematics, 22
- fminf
  - <math.h>: Mathematics, 15
- fmod
  - <math.h>: Mathematics, 22
- fmodf
  - <math.h>: Mathematics, 15
- frexp
  - <math.h>: Mathematics, 22
- frexpf
  - <math.h>: Mathematics, 15
- hypot
  - <math.h>: Mathematics, 23
- hypotf
  - <math.h>: Mathematics, 15
- INFINITY
  - <math.h>: Mathematics, 15
- inSerial
  - light\_control\_system.c, 36
- isfinitef
  - <math.h>: Mathematics, 16
- isinf

- `<math.h>`: Mathematics, [23](#)
- `isinf`
  - `<math.h>`: Mathematics, [16](#)
- `isnan`
  - `<math.h>`: Mathematics, [23](#)
- `isnanf`
  - `<math.h>`: Mathematics, [16](#)
- `lamp`
  - `light_control_system.c`, [36](#)
- `ldexp`
  - `<math.h>`: Mathematics, [23](#)
- `ldexpf`
  - `<math.h>`: Mathematics, [16](#)
- `light_control_system.c`
  - `allpinslow`, [35](#)
  - `BH1750_Device`, [36](#)
  - `BH1750_Read`, [35](#)
  - `Check_Protocol`, [36](#)
  - `Configure_BH1750`, [36](#)
  - `inSerial`, [36](#)
  - `lamp`, [36](#)
  - `loop`, [36](#)
  - `Lux`, [37](#)
  - `setup`, [36](#)
  - `sm_mode`, [37](#)
- `log`
  - `<math.h>`: Mathematics, [24](#)
- `log10`
  - `<math.h>`: Mathematics, [24](#)
- `log10f`
  - `<math.h>`: Mathematics, [16](#)
- `logf`
  - `<math.h>`: Mathematics, [16](#)
- `loop`
  - `light_control_system.c`, [36](#)
- `lrint`
  - `<math.h>`: Mathematics, [24](#)
- `lrintf`
  - `<math.h>`: Mathematics, [16](#)
- `lround`
  - `<math.h>`: Mathematics, [24](#)
- `lroundf`
  - `<math.h>`: Mathematics, [16](#)
- `Lux`
  - `light_control_system.c`, [37](#)
- `M_1_PI`
  - `<math.h>`: Mathematics, [17](#)
- `M_2_PI`
  - `<math.h>`: Mathematics, [17](#)
- `M_2_SQRTPI`
  - `<math.h>`: Mathematics, [17](#)
- `M_LN10`
  - `<math.h>`: Mathematics, [17](#)
- `M_LN2`
  - `<math.h>`: Mathematics, [17](#)
- `M_LOG10E`
  - `<math.h>`: Mathematics, [17](#)
- `M_LOG2E`
  - `<math.h>`: Mathematics, [17](#)
- `M_PI_2`
  - `<math.h>`: Mathematics, [18](#)
- `M_PI_4`
  - `<math.h>`: Mathematics, [18](#)
- `M_PI`
  - `<math.h>`: Mathematics, [18](#)
- `M_SQRT1_2`
  - `<math.h>`: Mathematics, [18](#)
- `M_SQRT2`
  - `<math.h>`: Mathematics, [18](#)
- `M_E`
  - `<math.h>`: Mathematics, [17](#)
- `modf`
  - `<math.h>`: Mathematics, [24](#)
- `modff`
  - `<math.h>`: Mathematics, [25](#)
- `NAN`
  - `<math.h>`: Mathematics, [18](#)
- `onReceive`
  - `TwoWire`, [31](#)
- `onRequest`
  - `TwoWire`, [31](#)
- `peek`
  - `TwoWire`, [31](#)
- `pow`
  - `<math.h>`: Mathematics, [25](#)
- `powf`
  - `<math.h>`: Mathematics, [18](#)
- `read`
  - `TwoWire`, [32](#)
- `requestFrom`
  - `TwoWire`, [32](#)
- `round`
  - `<math.h>`: Mathematics, [25](#)
- `roundf`
  - `<math.h>`: Mathematics, [18](#)
- `setClock`
  - `TwoWire`, [33](#)
- `setup`
  - `light_control_system.c`, [36](#)
- `signbit`
  - `<math.h>`: Mathematics, [25](#)
- `signbitf`
  - `<math.h>`: Mathematics, [19](#)
- `sin`
  - `<math.h>`: Mathematics, [26](#)
- `sinf`
  - `<math.h>`: Mathematics, [19](#)
- `sinh`
  - `<math.h>`: Mathematics, [26](#)
- `sinhf`
  - `<math.h>`: Mathematics, [19](#)

- sm\_mode
  - light\_control\_system.c, [37](#)
- sqrt
  - <math.h>: Mathematics, [26](#)
- sqrtf
  - <math.h>: Mathematics, [26](#)
- square
  - <math.h>: Mathematics, [26](#)
- squaref
  - <math.h>: Mathematics, [19](#)
- tan
  - <math.h>: Mathematics, [26](#)
- tanf
  - <math.h>: Mathematics, [19](#)
- tanh
  - <math.h>: Mathematics, [27](#)
- tanhf
  - <math.h>: Mathematics, [19](#)
- trunc
  - <math.h>: Mathematics, [27](#)
- truncf
  - <math.h>: Mathematics, [19](#)
- TwoWire, [29](#)
  - available, [30](#)
  - begin, [30](#)
  - beginTransmission, [30](#)
  - end, [31](#)
  - endTransmission, [31](#)
  - flush, [31](#)
  - onReceive, [31](#)
  - onRequest, [31](#)
  - peek, [31](#)
  - read, [32](#)
  - requestFrom, [32](#)
  - setClock, [33](#)
  - TwoWire, [30](#)
  - write, [33](#)
- WIRE\_HAS\_END
  - Wire.h, [39](#)
- Wire
  - Wire.h, [40](#)
- Wire.h
  - BUFFER\_LENGTH, [39](#)
  - WIRE\_HAS\_END, [39](#)
  - Wire, [40](#)
- write
  - TwoWire, [33](#)