# IoT Embedded Systems

PSampaio Ver1.0

Generated by Doxygen 1.8.16

# Chapter 1

# Module Index

## 1.1 Modules

Here is a list of all modules:

# Chapter 2

# File Index

## 2.1 File List

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

# Chapter 3

# Module Documentation

## 3.1 BH1750

This package provides the capabilities interact with the ambient light sensor BH1750.

### Enumerations

- enum BH1750ModeEnum {
  UNCONFIGURED = 0, CONTINUOUS_HIGH_RES_MODE = 0x10, CONTINUOUS_HIGH_RES_MODE_2 = 0x11, CONTINUOUS_LOW_RES_MODE = 0x13,
  ONE_TIME_HIGH_RES_MODE = 0x20, ONE_TIME_HIGH_RES_MODE_2 = 0x21, ONE_TIME_LOW_RES_MODE = 0x23 }
- enum BH1750MeasurementTimeEnum { DEFAULT_MEASUREMENT_TIME = 69, MIN_MEASUREMENT_TIME = 31, MAX_MEASUREMENT_TIME = 254 }

### Functions

- void BH1750_Init (void)

  *Initializes the BH1750 API.*
- bool BH1750_ConfigureMode (BH1750_ModeType mode)

  *Configure the operation mode.*
- bool BH1750_SetMeasurementTime (BH1750_MeasurementTimeType time)

  *Configure the measurement time.*
- bool BH1750_Ready (bool maxWait)

  *Verify if it is possible do a measures.*
- float BH1750_GetLight ()

  *Read the ambient light.*

### 3.1.1 Detailed Description

This package provides the capabilities interact with the ambient light sensor BH1750.

The sensor is connect to the microcontroller LPC1769 as show in the follow table:

| BH1750 | LPC1769 |
|---|---|
| SCL | P0.28 |
| SDA | P0.27 |
| ADDR | GND |

### 3.1.2 Enumeration Type Documentation

#### 3.1.2.1 BH1750MeasurementTimeEnum

```
enum BH1750MeasurementTimeEnum
```

**Enumerator**

| | |
|---|---|
| DEFAULT_MEASUREMENT_TIME | Default measurement time. |
| MIN_MEASUREMENT_TIME | Minimum measurement time. |
| MAX_MEASUREMENT_TIME | Maximum measurement time. |

#### 3.1.2.2 BH1750ModeEnum

```
enum BH1750ModeEnum
```

**Enumerator**

| | |
|---|---|
| UNCONFIGURED | Same as Power Down Mode |
| CONTINUOUS_HIGH_RES_MODE | Measurement at 1 lux resolution. Measurement time is approx 120ms. |
| CONTINUOUS_HIGH_RES_MODE↩ _2 | Measurement at 0.5 lux resolution. Measurement time is approx 120ms. |
| CONTINUOUS_LOW_RES_MODE | Measurement at 4 lux resolution. Measurement time is approx 16ms. |
| ONE_TIME_HIGH_RES_MODE | Measurement at 1 lux resolution. Measurement time is approx 120ms. |
| ONE_TIME_HIGH_RES_MODE_2 | Measurement at 0.5 lux resolution. Measurement time is approx 120ms. |
| ONE_TIME_LOW_RES_MODE | Measurement at 4 lux resolution. Measurement time is approx 16ms. |

### 3.1.3 Function Documentation

#### 3.1.3.1 BH1750_ConfigureMode()

```
bool BH1750_ConfigureMode (
             BH1750_ModeType mode )
```

Configure the operation mode.

**Parameters**

| | |
|---|---|
| *mode* | indicate the mode. |

**Returns**

true if success, otherwise false.

#### 3.1.3.2 BH1750_GetLight()

```
float BH1750_GetLight ( )
```

Read the ambient light.

**Returns**

Ambient light in lux.

#### 3.1.3.3 BH1750_Init()

```
void BH1750_Init (
             void  )
```

Initializes the BH1750 API.

**Returns**

None.

**Note**

This function must be called prior to any other BH1750 functions.

#### 3.1.3.4 BH1750_Ready()

```
bool BH1750_Ready (
             bool maxWait )
```

Verify if it is possible do a measures.

**Parameters**

| | |
|---|---|
| *maxWait* | indicate the measurement time. |

**Returns**

true if it is possible do a measures, otherwise false.

### 3.1.3.5 BH1750_SetMeasurementTime()

```
bool BH1750_SetMeasurementTime (
            BH1750_MeasurementTimeType time )
```

Configure the measurement time.

**Parameters**

| | |
|---|---|
| *time* | indicate the measurement time. |

**Returns**

true if success, otherwise false.

## 3.2 Rotary and press button.

This package provides the interface for driving the rotary and the push button.

### Typedefs

- typedef enum ButtonEnum ENCODER_ButtonValueType

  *Push button state structures definition.*

### Enumerations

- enum ButtonEnum {
  BUTTON_NOTPRESSED, BUTTON_PRESSED, BUTTON_HELD, BUTTON_RELEASE,
  BUTTON_CLICKED, BUTTON_DCLICKED }

  *Push button state structures definition.*

### Functions

- void ENCODER_Init (void)

  *Initializes Encoder.*
- ENCODER_ButtonValueType ENCODER_GetButton (void)

  *Gets the value of the push button.*
- int ENCODER_GetValue (void)

  *Gets the value of the rotary button.*

### 3.2.1 Detailed Description

This package provides the interface for driving the rotary and the push button.

The rotary data bits are connect to the microcontroller LPC1769 as show in the follow table:

| Rotary Button | LPC1769 |
|---|---|
| CLK | P0.3 |
| DT | P0.2 |
| SW | P2.13 |
| + | 3V3 |

The driver has the following behavior: i) two steps to vary in the same direction. i) one step when the inversion is performed.

After initialization its assume that the direction is the clockwise.

### 3.2.2 Enumeration Type Documentation

#### 3.2.2.1 ButtonEnum

```
enum ButtonEnum
```

Push button state structures definition.

**Enumerator**

| | |
|---|---|
| BUTTON_NOTPRESSED | Button not pressed |
| BUTTON_PRESSED | Button pressed |
| BUTTON_HELD | Button held (still pressed) |
| BUTTON_RELEASE | Button released |
| BUTTON_CLICKED | Button short pressed and released |
| BUTTON_DCLICKED | Button double short pressed and released |

### 3.2.3 Function Documentation

#### 3.2.3.1 ENCODER_GetButton()

```
ENCODER_ButtonValueType ENCODER_GetButton (
             void   )
```

Gets the value of the push button.

**Returns**

A valid state of the push button (see ENCODER_ButtonValueType

#### 3.2.3.2 ENCODER_GetValue()

```
int ENCODER_GetValue (
             void   )
```

Gets the value of the rotary button.

**Returns**

0 if not rotate. If rotate returns 1 if was clockwise or -1 if was anticlockwise

#### 3.2.3.3 ENCODER_Init()

```
void ENCODER_Init (
             void   )
```

Initializes Encoder.

**Returns**

None.

## 3.3 ESP8266 serial interface

This package provides the interface for the ESP8266 with serial interface.

### Functions

- void ESPSERIAL_Init (int baudrate)

  *Initializes the serial interface for the ESP8266.*
- int ESPSERIAL_Send (void ∗command, int size)
- int ESPSERIAL_Recv (uint8_t ∗response, int maxSize)

### 3.3.1 Detailed Description

This package provides the interface for the ESP8266 with serial interface.

### 3.3.2 Function Documentation

#### 3.3.2.1 ESPSERIAL_Init()

```
void ESPSERIAL_Init (
             int baudrate )
```

Initializes the serial interface for the ESP8266.

**Parameters**

| | |
|---|---|
| *baudrate* | Set baudrate of device. |

**Returns**

None.

#### 3.3.2.2 ESPSERIAL_Recv()

```
int ESPSERIAL_Recv (
             uint8_t ∗ response,
             int maxSize )
```

Read from the ESP8266

**Parameters**

| | |
|---|---|
| *response* | Buffer data. |
| *maxSize* | Maximum lenght expected. |

**Returns**

Number of data that was readed.

#### 3.3.2.3 ESPSERIAL_Send()

```
int ESPSERIAL_Send (
            void * command,
            int size )
```

Write to the ESP8266

**Parameters**

| | |
|---|---|
| *command* | Buffer data. |
| *size* | Lenght of the buffer. |

**Returns**

Number of data that was written.

## 3.4 Drivers

This package for drivers.

### Modules

- BH1750

  *This package provides the capabilities interact with the ambient light sensor BH1750.*
- Rotary and press button.

  *This package provides the interface for driving the rotary and the push button.*
- ESP8266 serial interface

  *This package provides the interface for the ESP8266 with serial interface.*
- Library information

  *This package provides the version number of the driver library.*
- Text LCD

  *This package provides the interface for driving 4-bit HD44780-based LCDs used in LPCXpresso development board based on LPC1769 from NXP.*
- LED

  *This package provides the capabilities such as on/off/toggle to the LED in LPCXpresso development board based on LPC1769 from NXP.*
- Presence detect sensor.

  *This package provides the interface for driving the presence detect sensor.*
- Real Time Clock

  *This package provides the interface for the real time clock present in the microcontroller LPC1769 from NXP.*

### 3.4.1 Detailed Description

This package for drivers.

## 3.5 Library information

This package provides the version number of the driver library.

### Functions

- int INFO_GetVersion (void)

### 3.5.1 Detailed Description

This package provides the version number of the driver library.

### 3.5.2 Function Documentation

#### 3.5.2.1 INFO_GetVersion()

```
int INFO_GetVersion (
            void  )
```

Get version number of the library.

**Returns**

Version number.

## 3.6 Text LCD

This package provides the interface for driving 4-bit HD44780-based LCDs used in LPCXpresso development board based on LPC1769 from NXP.

### Macros

- #define LCDText_LINES 2
- #define LCDText_COLUMNS 16

### Functions

- void LCDText_Init ()
- void LCDText_WriteChar (char c)
- void LCDText_WriteString (const char ∗str)
- void LCDText_WriteLine (const char ∗firstLine, const char ∗secondLine)
- void LCDText_Clear ()
- void LCDText_Locate (int line, int column)
- void LCDText_CursorOn (void)
- void LCDText_CursorOff (void)
- void LCDText_CreateChar (unsigned char location, unsigned char charmap[ ])
- void LCDText_On (void)
- void LCDText_Off (void)
- void LCDText_Printf (const char ∗fmt,...)

### 3.6.1 Detailed Description

This package provides the interface for driving 4-bit HD44780-based LCDs used in LPCXpresso development board based on LPC1769 from NXP.

The LCD data bits are connect to the microcontroller LPC1769 as show in the follow table:

| LCD | LPC1769 |
|---|---|
| D0 .. D3 | Not connected |
| D4 .. D7 | P2.0 .. P2.3 |
| EN | P0.10 |
| RS | P0.11 |
| WR | GND |

### 3.6.2 Macro Definition Documentation

#### 3.6.2.1 LCDText_COLUMNS

```
#define LCDText_COLUMNS 16
```

LCD number of columns

#### 3.6.2.2 LCDText_LINES

```
#define LCDText_LINES 2
```

LCD number of lines

### 3.6.3 Function Documentation

#### 3.6.3.1 LCDText_Clear()

```
void LCDText_Clear ( )
```

Clear the screen and locate cursor to home position (0,0)

**Returns**

None.

#### 3.6.3.2 LCDText_CreateChar()

```
void LCDText_CreateChar (
            unsigned char location,
            unsigned char charmap[] )
```

User define character.

**Parameters**

| | |
|---|---|
| *location* | The new character position, |
| *charmap* | The user defined character values. |

**Returns**

None.

#### 3.6.3.3 LCDText_CursorOff()

```
void LCDText_CursorOff (
            void )
```

Turns cursor on.

**Returns**

None.

#### 3.6.3.4 LCDText_CursorOn()

```
void LCDText_CursorOn (
             void  )
```

Turns cursor on.

**Returns**

None.

#### 3.6.3.5 LCDText_Init()

```
void LCDText_Init ( )
```

Initializes the LCD API.

**Returns**

None.

**Note**

This function must be called prior to any other LCDText functions.

#### 3.6.3.6 LCDText_Locate()

```
void LCDText_Locate (
             int line,
             int column )
```

Locate cursor to a screen line and column

**Parameters**

| | |
|---|---|
| *line* | The vertical position from the top, indexed from 0 |
| *column* | The horizontal position from the left, indexed from 0 |

**Returns**

None.

#### 3.6.3.7 LCDText_Off()

```
void LCDText_Off (
             void  )
```

Turns display off.

**Returns**

None.

#### 3.6.3.8 LCDText_On()

```
void LCDText_On (
             void  )
```

Turns display on.

**Returns**

None.

#### 3.6.3.9 LCDText_Printf()

```
void LCDText_Printf (
             const char * fmt,
              ...  )
```

Write a formated string to the LCD

**Parameters**

| | |
|---|---|
| *fmt* | A printf-style format string, followed by the variables to use in formating the string. |

#### 3.6.3.10 LCDText_WriteChar()

```
void LCDText_WriteChar (
            char c )
```

Write a character to the LCD

**Parameters**

| | |
|---|---|
| *c* | The character to write to the display |

### 3.6.3.11 LCDText_WriteLine()

```
void LCDText_WriteLine (
            const char * firstLine,
            const char * secondLine )
```

Write a C-string to specified line of the LCD

**Parameters**

| | |
|---|---|
| *firstLine* | The C-string to write to the first of display. If NULL nothing is write. |
| *secondLine* | The C-string to write to the second of display. If NULL nothing is write. |

**Returns**

None.

### 3.6.3.12 LCDText_WriteString()

```
void LCDText_WriteString (
            const char * str )
```

Write a C-string to the LCD

**Parameters**

| | |
|---|---|
| *str* | The C-string to write to the display |

## 3.7 LED

This package provides the capabilities such as on/off/toggle to the LED in LPCXpresso development board based on LPC1769 from NXP.

### Functions

- void LED_Init (bool state)

  *Initializes the LED API.*
- bool LED_GetState (void)

  *Get LED state.*
- void LED_On (void)

  *Turn LED on.*
- void LED_Off (void)

  *Turn LED off.*
- void LED_Toggle (void)

  *Toggle LED.*

### 3.7.1 Detailed Description

This package provides the capabilities such as on/off/toggle to the LED in LPCXpresso development board based on LPC1769 from NXP.

### 3.7.2 Function Documentation

#### 3.7.2.1 LED_GetState()

```
bool LED_GetState (
             void  )
```

Get LED state.

**Returns**

status of LED: "false" indicate LED is off and "true" LED is on.

#### 3.7.2.2 LED_Init()

```
void LED_Init (
             bool state )
```

Initializes the LED API.

**Parameters**

| | |
|---|---|
| *state* | set LED state: "false" turns LED off and "true" turns LED on. |

**Returns**

None.

**Note**

This function must be called prior to any other LED functions. The LED will started in the value passed in the parameter.

#### 3.7.2.3 LED_Off()

```
void LED_Off (
            void  )
```

Turn LED off.

**Returns**

None.

#### 3.7.2.4 LED_On()

```
void LED_On (
            void  )
```

Turn LED on.

**Returns**

None.

#### 3.7.2.5 LED_Toggle()

```
void LED_Toggle (
            void  )
```

Toggle LED.

**Returns**

None.

## 3.8 Presence detect sensor.

This package provides the interface for driving the presence detect sensor.

### Functions

- void PIR_Init (void)

  *Initializes Encoder.*
- bool PIR_GetValue (void)

  *Get if was detected presence or not.*

### 3.8.1 Detailed Description

This package provides the interface for driving the presence detect sensor.

The sensor is connect to the microcontroller LPC1769 in the P2.12 pin.

### 3.8.2 Function Documentation

#### 3.8.2.1 PIR_GetValue()

```
bool PIR_GetValue (
             void  )
```

Get if was detected presence or not.

**Returns**

true if presence was detected or false if not

#### 3.8.2.2 PIR_Init()

```
void PIR_Init (
             void  )
```

Initializes Encoder.

**Returns**

None.

## 3.9 Real Time Clock

This package provides the interface for the real time clock present in the microcontroller LPC1769 from NXP.

### Functions

- void RTC_Init (struct tm ∗dateTime)

  *Initializes RTC and starts counting.*
- void RTC_InitSeconds (time_t time)

  *Initializes RTC and starts counting.*
- void RTC_GetValue (struct tm ∗dateTime)

  *Gets date and time from RTC.*
- void RTC_SetValue (struct tm ∗dateTime)

  *Sets date and time to RTC.*
- time_t RTC_GetSeconds (void)

  *Gets date and time from RTC.*
- void RTC_SetSeconds (time_t time)

  *Sets date and time from RTC.*

### 3.9.1 Detailed Description

This package provides the interface for the real time clock present in the microcontroller LPC1769 from NXP.

### 3.9.2 Function Documentation

#### 3.9.2.1 RTC_GetSeconds()

```
time_t RTC_GetSeconds (
             void  )
```

Gets date and time from RTC.

**Returns**

A C standard time_t with the number of seconds since 01.01.1970 00:00:00

#### 3.9.2.2 RTC_GetValue()

```
void RTC_GetValue (
             struct tm ∗ dateTime )
```

Gets date and time from RTC.

**Parameters**

| | |
|---|---|
| ∗*dateTime* | A pointer to C standard structure tm to save data to. |

**Returns**

None.

#### 3.9.2.3 RTC_Init()

```
void RTC_Init (
            struct tm * dateTime )
```

Initializes RTC and starts counting.

**Parameters**

| | |
|---|---|
| *dateTime* | A pointer to C standard structure tm. |

**Note**

If you power off the LPCXpresso board the RTC will stop.

**Returns**

None.

#### 3.9.2.4 RTC_InitSeconds()

```
void RTC_InitSeconds (
            time_t time )
```

Initializes RTC and starts counting.

**Parameters**

| | |
|---|---|
| *time* | A C standard time_t value. |

**Note**

If you use RTC_Init not use this function.

**Returns**

None.

#### 3.9.2.5 RTC_SetSeconds()

```
void RTC_SetSeconds (
             time_t time )
```

Sets date and time from RTC.

**Parameters**

| | |
|---|---|
| *time* | A C standard time_t with the number of seconds since 01.01.1970 00:00:00 |

#### 3.9.2.6 RTC_SetValue()

```
void RTC_SetValue (
             struct tm * dateTime )
```

Sets date and time to RTC.

**Parameters**

| | |
|---|---|
| *∗dateTime* | A pointer to C standard structure tm with date and time |

**Returns**

None.

Chapter 4

# File Documentation

## 4.1 C:/Users/alext/Desktop/SE/Workspace/DRIVERS/inc/bh1750.h File Reference

Contains the BH1750 ambient light sensor API.

### Enumerations

- enum BH1750ModeEnum {
  UNCONFIGURED = 0, CONTINUOUS_HIGH_RES_MODE = 0x10, CONTINUOUS_HIGH_RES_MODE_2 = 0x11, CONTINUOUS_LOW_RES_MODE = 0x13,
  ONE_TIME_HIGH_RES_MODE = 0x20, ONE_TIME_HIGH_RES_MODE_2 = 0x21, ONE_TIME_LOW_RES_MODE = 0x23 }
- enum BH1750MeasurementTimeEnum { DEFAULT_MEASUREMENT_TIME = 69, MIN_MEASUREMENT_TIME = 31, MAX_MEASUREMENT_TIME = 254 }

### Functions

- void BH1750_Init (void)

  *Initializes the BH1750 API.*
- bool BH1750_ConfigureMode (BH1750_ModeType mode)

  *Configure the operation mode.*
- bool BH1750_SetMeasurementTime (BH1750_MeasurementTimeType time)

  *Configure the measurement time.*
- bool BH1750_Ready (bool maxWait)

  *Verify if it is possible do a measures.*
- float BH1750_GetLight ()

  *Read the ambient light.*

### 4.1.1 Detailed Description

Contains the BH1750 ambient light sensor API.

**Version**

1.0

**Date**

9 Out 2021

**Author**

PSampaio

Copyright(C) 2015-2022, PSampaio All rights reserved.

Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

## 4.2 C:/Users/alext/Desktop/SE/Workspace/DRIVERS/inc/encoder.h File Reference

Contains the ENCODER API.

### Typedefs

- typedef enum ButtonEnum ENCODER_ButtonValueType

  *Push button state structures definition.*

### Enumerations

- enum ButtonEnum {
  BUTTON_NOTPRESSED, BUTTON_PRESSED, BUTTON_HELD, BUTTON_RELEASE,
  BUTTON_CLICKED, BUTTON_DCLICKED }

  *Push button state structures definition.*

### Functions

- void ENCODER_Init (void)

  *Initializes Encoder.*
- ENCODER_ButtonValueType ENCODER_GetButton (void)

  *Gets the value of the push button.*
- int ENCODER_GetValue (void)

  *Gets the value of the rotary button.*

### 4.2.1 Detailed Description

Contains the ENCODER API.

**Version**

1.0

**Date**

13 set 2022

**Author**

PSampaio

Copyright(C) 2015-2023, PSampaio All rights reserved.

Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

## 4.3 C:/Users/alext/Desktop/SE/Workspace/DRIVERS/inc/espserial.h File Reference

Contains the serial ESP8266 API.

```
#include <stdint.h>
```

### Functions

- void ESPSERIAL_Init (int baudrate)

  *Initializes the serial interface for the ESP8266.*
- int ESPSERIAL_Send (void ∗command, int size)
- int ESPSERIAL_Recv (uint8_t ∗response, int maxSize)

### 4.3.1 Detailed Description

Contains the serial ESP8266 API.

**Version**

1.0

**Date**

17 Mar 2017

**Author**

PSampaio

Copyright(C) 2015-2023, PSampaio All rights reserved.

Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

## 4.4 C:/Users/alext/Desktop/SE/Workspace/DRIVERS/inc/info.h File Reference

Contains information about API version.

### Functions

- int INFO_GetVersion (void)

### 4.4.1 Detailed Description

Contains information about API version.

**Version**

1.0

**Date**

19 Mar 2023

**Author**

PSampaio

Copyright(C) 2015-2023, PSampaio All rights reserved.

Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

## 4.5 C:/Users/alext/Desktop/SE/Workspace/DRIVERS/inc/lcdtext.h File Reference

Contains the text LCD API.

### Macros

- #define LCDText_LINES 2
- #define LCDText_COLUMNS 16

## Functions

- void LCDText_Init ()
- void LCDText_WriteChar (char c)
- void LCDText_WriteString (const char ∗str)
- void LCDText_WriteLine (const char ∗firstLine, const char ∗secondLine)
- void LCDText_Clear ()
- void LCDText_Locate (int line, int column)
- void LCDText_CursorOn (void)
- void LCDText_CursorOff (void)
- void LCDText_CreateChar (unsigned char location, unsigned char charmap[ ])
- void LCDText_On (void)
- void LCDText_Off (void)
- void LCDText_Printf (const char ∗fmt,...)

### 4.5.1 Detailed Description

Contains the text LCD API.

**Version**

1.0

**Date**

30 Out 2018

**Author**

PSampaio

Copyright(C) 2015-2023, PSampaio All rights reserved.

Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

## 4.6 C:/Users/alext/Desktop/SE/Workspace/DRIVERS/inc/led.h File Reference

Contains the LED API.

## Functions

- void LED_Init (bool state)

  *Initializes the LED API.*
- bool LED_GetState (void)

  *Get LED state.*
- void LED_On (void)

  *Turn LED on.*
- void LED_Off (void)

  *Turn LED off.*
- void LED_Toggle (void)

  *Toggle LED.*

### 4.6.1 Detailed Description

Contains the LED API.

**Version**

1.0

**Date**

9 Out 2018

**Author**

PSampaio

Copyright(C) 2015-2023, PSampaio All rights reserved.

Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

## 4.7 C:/Users/alext/Desktop/SE/Workspace/DRIVERS/inc/pir.h File Reference

Contains the presence detect sensor API.

### Functions

- void PIR_Init (void)

  *Initializes Encoder.*
- bool PIR_GetValue (void)

  *Get if was detected presence or not.*

### 4.7.1 Detailed Description

Contains the presence detect sensor API.

**Version**

1.0

**Date**

13 set 2022

**Author**

PSampaio

Copyright(C) 2015-2023, PSampaio All rights reserved.

Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the products. This software is supplied "AS IS" without any warranties.

## 4.8 C:/Users/alext/Desktop/SE/Workspace/DRIVERS/inc/rtc.h File Reference

Contains the LED API.

### Functions

- void RTC_Init (struct tm ∗dateTime)

  *Initializes RTC and starts counting.*
- void RTC_InitSeconds (time_t time)

  *Initializes RTC and starts counting.*
- void RTC_GetValue (struct tm ∗dateTime)

  *Gets date and time from RTC.*
- void RTC_SetValue (struct tm ∗dateTime)

  *Sets date and time to RTC.*
- time_t RTC_GetSeconds (void)

  *Gets date and time from RTC.*
- void RTC_SetSeconds (time_t time)

  *Sets date and time from RTC.*

### 4.8.1 Detailed Description

Contains the LED API.

**Version**

1.0

**Date**

30 Out 2018

**Author**

PSampaio