

---

**LwOW**

**Tilen MAJERLE**

**Nov 15, 2020**



# CONTENTS

- 1 Features 3**
- 2 Requirements 5**
- 3 Contribute 7**
- 4 License 9**
- 5 Table of contents 11**
  - 5.1 Getting started . . . . . 11
  - 5.2 User manual . . . . . 13
  - 5.3 API reference . . . . . 29
  - 5.4 Examples and demos . . . . . 46
- Index 49**



Welcome to the documentation for version 3.0.1.

LwOW is lightweight, platform independent library for Onewire protocol for embedded systems. Its primary focus is UART hardware for physical communication for sensors and other slaves.

*[Download library](#)* *[Getting started](#)* *[Open Github](#)*



## **FEATURES**

- Written in ANSI C99
- Platform independent, uses custom low-level layer for device drivers
- 1-Wire protocol fits UART specifications at 9600 and 115200 bauds
- Hardware is responsible for timing characteristics
  - Allows DMA on the high-performance microcontrollers
- Different device drivers included
  - DS18x20 temperature sensor is natively supported
- Works with operating system due to hardware timing management
  - Separate thread-safe API is available
- API for device scan, reading and writing single bits
- User friendly MIT license





## REQUIREMENTS

- C compiler
- Platform dependent drivers
- Few *kB* of non-volatile memory



## CONTRIBUTE

Fresh contributions are always welcome. Simple instructions to proceed:

1. Fork Github repository
2. Respect `C style & coding rules` used by the library
3. Create a pull request to `develop` branch with new features or bug fixes

Alternatively you may:

1. Report a bug
2. Ask for a feature request



## LICENSE

MIT License

Copyright (c) 2020 Tilen MAJERLE

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to **do** so, subject to the following **conditions**:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



## TABLE OF CONTENTS

### 5.1 Getting started

#### 5.1.1 Download library

Library is primarily hosted on [Github](#).

- Download latest release from [releases area](#) on Github
- Clone *develop* branch for latest development

#### Download from releases

All releases are available on Github [releases area](#).

#### Clone from Github

##### First-time clone

- Download and install `git` if not already
- Open console and navigate to path in the system to clone repository to. Use command `cd your_path`
- Clone repository with one of available 3 options
  - Run `git clone --recurse-submodules https://github.com/MaJerle/lwow` command to clone entire repository, including submodules
  - Run `git clone --recurse-submodules --branch develop https://github.com/MaJerle/lwow` to clone *development* branch, including submodules
  - Run `git clone --recurse-submodules --branch master https://github.com/MaJerle/lwow` to clone *latest stable* branch, including submodules
- Navigate to `examples` directory and run favourite example

## Update cloned to latest version

- Open console and navigate to path in the system where your resources repository is. Use command `cd your_path`
- Run `git pull origin master --recurse-submodules` command to pull latest changes and to fetch latest changes from submodules
- Run `git submodule foreach git pull origin master` to update & merge all submodules

---

**Note:** This is preferred option to use when you want to evaluate library and run prepared examples. Repository consists of multiple submodules which can be automatically downloaded when cloning and pulling changes from root repository.

---

### 5.1.2 Add library to project

At this point it is assumed that you have successfully download library, either cloned it or from releases page.

- Copy `lwow` folder to your project
- Add `lwow/src/include` folder to *include path* of your toolchain
- Add source files from `lwow/src/` folder to toolchain build
- Copy `lwow/src/include/lwow/lwow_opts_template.h` to project folder and rename it to `lwow_opts.h`
- Implement device drivers for UART hardware
- Build the project

### 5.1.3 Configuration file

Library comes with template config file, which can be modified according to needs. This file shall be named `lwow_opts.h` and its default template looks like the one below.

---

**Note:** Default configuration template file location: `lwow/src/include/lwow/lwow_opts_template.h`. File must be renamed to `lwow_opts.h` first and then copied to the project directory (or simply renamed in-place) where compiler include paths have access to it by using `#include "lwow_opts.h"`.

---

---

**Tip:** Check [Configuration](#) section for possible configuration settings

---

Listing 1: Config template file

```
1  /**
2   * \file          lwow_opts_template.h
3   * \brief        LwOW application configuration
4   */
5
6  /**
7   * Copyright (c) 2020 Tilen MAJERLE
8   */
```

(continues on next page)



(continued from previous page)

```

9  * Permission is hereby granted, free of charge, to any person
10 * obtaining a copy of this software and associated documentation
11 * files (the "Software"), to deal in the Software without restriction,
12 * including without limitation the rights to use, copy, modify, merge,
13 * publish, distribute, sublicense, and/or sell copies of the Software,
14 * and to permit persons to whom the Software is furnished to do so,
15 * subject to the following conditions:
16 *
17 * The above copyright notice and this permission notice shall be
18 * included in all copies or substantial portions of the Software.
19 *
20 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
21 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
22 * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
23 * AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
24 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
25 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
26 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
27 * OTHER DEALINGS IN THE SOFTWARE.
28 *
29 * This file is part of LwOW - Lightweight onewire library.
30 *
31 * Author:          Tilen MAJERLE <tilen@majerle.eu>
32 * Version:         v3.0.1
33 */
34 #ifndef LWOW_HDR_OPTS_H
35 #define LWOW_HDR_OPTS_H
36
37 /* Rename this file to "lwow_opts.h" for your application */
38
39 /*
40 * Open "include/lwow/lwow_opt.h" and
41 * copy & replace here settings you want to change values
42 */
43
44 #endif /* LWOW_HDR_OPTS_H */

```

## 5.2 User manual

### 5.2.1 How it works

LwOW library tends to use *UART* hardware of any microcontroller/embedded system, to generate timing clock for OneWire signals.

Nowadays embedded systems allow many UART ports, usually many more vs requirements for the final application. OneWire protocol needs precise timings and embedded systems (in 99.9%) do not have specific hardware to handle communication of this type.

Fortunately, OneWire timings match with UART timings at 9600 and 115200 bauds.

---

**Note:** Check [detailed tutorial](#) from Maxim for more information.

---

## 5.2.2 Thread safety

With default configuration, library is *not* thread safe. This means whenever it is used with operating system, user must resolve it with care.

Library has locking mechanism support for thread safety, which needs to be enabled manually.

---

**Tip:** To enable thread-safety support, parameter `LWOW_CFG_OS` must be set to 1. Please check [Configuration](#) for more information about other options.

---

After thread-safety features has been enabled, it is necessary to implement 4 low-level system functions.

---

**Tip:** System function template example is available in `lwow/src/system/` folder.

---

Example code for CMSIS-OS V2

---

**Note:** Check [System functions](#) section for function description

---

Listing 2: System functions for CMSIS-OS based operating system

```

1  /**
2   * \file          lwow_sys_cmsis_os.c
3   * \brief         System functions for CMSIS-OS based operating system
4   */
5
6  /**
7   * Copyright (c) 2020 Tilen MAJERLE
8   *
9   * Permission is hereby granted, free of charge, to any person
10  * obtaining a copy of this software and associated documentation
11  * files (the "Software"), to deal in the Software without restriction,
12  * including without limitation the rights to use, copy, modify, merge,
13  * publish, distribute, sublicense, and/or sell copies of the Software,
14  * and to permit persons to whom the Software is furnished to do so,
15  * subject to the following conditions:
16  *
17  * The above copyright notice and this permission notice shall be
18  * included in all copies or substantial portions of the Software.
19  *
20  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
21  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
22  * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
23  * AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
24  * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
25  * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
26  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
27  * OTHER DEALINGS IN THE SOFTWARE.
28  *
29  * This file is part of LwOW - Lightweight onewire library.
30  *
31  * Author:          Tilen MAJERLE <tilen@majerle.eu>
32  * Version:         v3.0.1
33  */

```

(continues on next page)

(continued from previous page)

```

34 #include "lwow/lwow.h"
35
36 #if LWOW_CFG_OS && !__DOXYGEN__
37
38 #include "cmsis_os.h"
39
40 uint8_t
41 lwow_sys_mutex_create(LWOW_CFG_OS_MUTEX_HANDLE* m, void* arg) {
42     LWOW_UNUSED(arg);
43     const osMutexAttr_t attr = {
44         .attr_bits = osMutexRecursive
45     };
46
47     *m = osMutexNew(&attr);
48     return *m != NULL;
49 }
50
51 uint8_t
52 lwow_sys_mutex_delete(LWOW_CFG_OS_MUTEX_HANDLE* m, void* arg) {
53     LWOW_UNUSED(arg);
54     return osMutexDelete(*m) == osOK;
55 }
56
57 uint8_t
58 lwow_sys_mutex_wait(LWOW_CFG_OS_MUTEX_HANDLE* m, void* arg) {
59     LWOW_UNUSED(arg);
60     return osMutexAcquire(*m, osWaitForever) == osOK;
61 }
62
63 uint8_t
64 lwow_sys_mutex_release(LWOW_CFG_OS_MUTEX_HANDLE* m, void* arg) {
65     LWOW_UNUSED(arg);
66     return osMutexRelease(*m) == osOK;
67 }
68
69 #endif /* LWOW_CFG_OS && !__DOXYGEN__ */

```

## 5.2.3 Hardware connection with sensor

To be able to successfully use sensors and other devices with embedded systems, these needs to be physically wired with embedded system (or PC).

Target devices (usually sensors or memory devices) are connected to master host device using single wire (from here protocol name *One Wire*) for communication only. There are also voltage and ground lines, marked as *VCC* and *GND*, respectively.

At this point, we assume you are familiar with UART protocol and you understand it has 2 independent lines, one for transmitting data (*TX*) and second to receive data (*RX*).

For successful communication with sensors, bi-directional support is necessary to be implemented, but there is only 1 wire available to do so. It might sound complicated at this point.

OneWire data line is by default in *open-drain* mode. This means that:

- Any device connected to data line can at any time pull line to *GND* without fear of short circuit

- None of the devices are allowed to force high state on the line. Application must use external *pull-up* resistor to do so.

How to send data over TX pin if application cannot force high level on the line? There are 2 options:

- Configure UART TX pin to *open-drain* mode
- Use *push-pull* to *open-drain* converter using 2 mosfets and 1 resistor

Fig. 1: Push-pull to open-drain converter

Since many latest embedded systems allow you to configure TX pin to open-drain mode natively, you may consider second option instead.

Fig. 2: Embedded system with native open-drain TX pin support

**Warning:** Application must assure that TX pin is always configured to open-drain mode, either with *push-pull* to *open-drain* converter or directly configured in the system.

## TX and RX pins

Every communication starts by master initiating it. To transfer data over UART, application uses TX pin and RX pin is used to read data. With 1-Wire protocol, application needs to transfer data and read them back in real-time. This is also called *loop-back* mode.

Let's take reset sequence as an example. By specifications, UART has to be configured in 9600 bauds and master needs to send single UART byte with value of 0xF0. If there is any slave connected, slave must pull line to GND during transmission of 0xF part of byte. Master needs to identify this by using RX pin of the UART.

---

**Note:** Please check [official document on Maxim website](#) to understand why 0xF0 and 9600 bauds.

---

## 5.2.4 UART and 1-Wire timing relation

This part is explaining how UART and 1-Wire timings are connected together and what is important to take into consideration for stable and reliable communication.

1-Wire protocol specification match UART protocol specification when baudrate is configured at 115200 bauds. Going into the details about 1-Wire protocol, we can identify that:

- To send 1 logical *bit* at 1-Wire level, application needs to transmit 1 byte at UART level with speed of 115200 bauds
- To send 1 logical *byte* at 1-Wire level, application must transmit 8 bytes at UART level with speed of 115200 bauds

Fig. 3: UART byte time is equivalent to 1 bit at 1-Wire level

Timing for each bit is very clearly defined by 1-Wire specification (not purpose to go into these details) and needs to respect all low and high level states for reliable communication. Each bit at 1-Wire level starts with master pulling line low for specific amount of time. Until master initiates communication, line is in *idle* mode.

Image above shows relation between UART and 1-Wire timing. It represents transmission of 3 bits on 1-Wire level or 3 bytes at UART level. *Green* and *blue* rectangles show different times between ending of one bit and start of new bit.

---

**Note:** By 1-Wire specification, it is important to match bit timing. It is less important to match idle timings as these are not defined. Effectively this allows master to use UART to initiate byte transfer where UART takes care of proper timing.

---

Different timings (*green* vs *blue*) may happen if application uses many interrupts, but uses UART in polling mode to transmit data. This is very important for operating systems where context switch may disable interrupts. Fortunately, it is not a problem for reliable communication due to:

- When UART starts transmission, hardware takes care of timing
- If application gets preempted with more important task, 1-Wire line will be in idle state for longer time. This is not an issue by 1-Wire specification

More advanced embedded systems implement DMA controllers to support next level of transfers.

## 5.2.5 Porting guide

### Implement low-level driver

Implementation of low-level driver is an essential part. It links middleware with actual hardware design of the device.

Its implementation must provide 4 functions:

- To open/configure UART hardware
- To set UART baudrate on the fly
- To transmit/receive data over UART
- To close/de-init UART hardware

After these functions have been implemented (check below for references), driver must link these functions to single driver structure of type `lwow_ll_drv_t`, later used during instance initialization.

---

**Tip:** Check *Low-level driver* for function prototypes.

---

### Implement system functions

System functions are required only if operating system mode is enabled, with `LWOW_CFG_OS`.

Its implementation structure is not the same as for low-level driver, customer needs to implement fixed functions, with pre-defined name, starting with `ow_sys_` name.

System function must only support OS mutex management and has to provide:

- `ow_sys_mutex_create()` function to create new mutex
- `ow_sys_mutex_delete()` function to delete existing mutex
- `ow_sys_mutex_wait()` function to wait for mutex to be available
- `ow_sys_mutex_release()` function to release (give) mutex back

**Warning:** Application must define `LWOW_CFG_OS_MUTEX_HANDLE` for mutex type. This shall be done in `lwow_opts.h` file.

---

**Tip:** Check *System functions* for function prototypes.

---

### Example: Low-level driver for WIN32

Example code for low-level porting on WIN32 platform. It uses native *Windows* features to open *COM* port and read/write from/to it.

Listing 3: Actual implementation of low-level driver for WIN32

```
1  /**
2   * \file          lwow_ll_win32.c
3   * \brief         UART implementation for WIN32
4   */
5
6  /**
7   * Copyright (c) 2020 Tilen MAJERLE
8   *
9   * Permission is hereby granted, free of charge, to any person
10  * obtaining a copy of this software and associated documentation
11  * files (the "Software"), to deal in the Software without restriction,
12  * including without limitation the rights to use, copy, modify, merge,
13  * publish, distribute, sublicense, and/or sell copies of the Software,
14  * and to permit persons to whom the Software is furnished to do so,
15  * subject to the following conditions:
16  *
17  * The above copyright notice and this permission notice shall be
18  * included in all copies or substantial portions of the Software.
19  *
20  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
21  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
22  * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
23  * AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
24  * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
25  * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
26  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
27  * OTHER DEALINGS IN THE SOFTWARE.
28  *
29  * This file is part of LwOW - Lightweight onewire library.
30  *
31  * Author:          Tilen MAJERLE <tilen@majerle.eu>
32  * Version:         v3.0.1
33  */
34  #include <stdio.h>
35  #include "lwow/lwow.h"
36  #include "windows.h"
37
38  #if !__DOXYGEN__
39
40  /* Function prototypes */
41  static uint8_t init(void* arg);
```

(continues on next page)

(continued from previous page)

```

42 static uint8_t deinit(void* arg);
43 static uint8_t set_baudrate(uint32_t baud, void* arg);
44 static uint8_t transmit_receive(const uint8_t* tx, uint8_t* rx, size_t len, void* _
↳arg);
45
46 /* Win 32 LL driver for OW */
47 const lwow_ll_drv_t
48 lwow_ll_drv_win32 = {
49     .init = init,
50     .deinit = deinit,
51     .set_baudrate = set_baudrate,
52     .tx_rx = transmit_receive,
53 };
54
55 static HANDLE com_port;
56 static DCB dcb = { 0 };
57
58 static uint8_t
59 init(void* arg) {
60     dcb.DCBlength = sizeof(dcb);
61
62     /* Open virtual file as read/write */
63     com_port = CreateFile(L"\\\\.\\COM4",
64                           GENERIC_READ | GENERIC_WRITE,
65                           0,
66                           0,
67                           OPEN_EXISTING,
68                           0,
69                           NULL
70                           );
71
72     /* First read current values */
73     if (GetCommState(com_port, &dcb)) {
74         COMMTIMEOUTS timeouts;
75
76         dcb.BaudRate = 115200;
77         dcb.ByteSize = 8;
78         dcb.Parity = NOPARITY;
79         dcb.StopBits = ONESTOPBIT;
80
81         /* Try to set com port data */
82         if (!SetCommState(com_port, &dcb)) {
83             printf("Cannot get COM port\\r\\n");
84             return 0;
85         }
86
87         if (GetCommTimeouts(com_port, &timeouts)) {
88             /* Set timeout to return immediately from ReadFile function */
89             timeouts.ReadIntervalTimeout = MAXDWORD;
90             timeouts.ReadTotalTimeoutConstant = 0;
91             timeouts.ReadTotalTimeoutMultiplier = 0;
92             if (!SetCommTimeouts(com_port, &timeouts)) {
93                 printf("Cannot set COM PORT timeouts\\r\\n");
94             }
95             GetCommTimeouts(com_port, &timeouts);
96         }
97     } else {

```

(continues on next page)

(continued from previous page)

```

98     printf("Cannot get COM port info\r\n");
99 }
100
101     return 1;
102 }
103
104 uint8_t
105 deinit(void* arg) {
106     /* Disable UART peripheral */
107
108     return 1;
109 }
110
111 uint8_t
112 set_baudrate(uint32_t baud, void* arg) {
113     /* Configure UART to selected baudrate */
114     dcb.BaudRate = baud;
115
116     /* Try to set com port data */
117     if (!SetCommState(com_port, &dcb)) {
118         printf("Cannot set COM port baudrate to %u bauds\r\n", (unsigned)baud);
119         return 0;
120     }
121
122     return 1;
123 }
124
125 uint8_t
126 transmit_receive(const uint8_t* tx, uint8_t* rx, size_t len, void* arg) {
127     /* Perform data exchange */
128     size_t read = 0;
129     DWORD br;
130
131     if (com_port != NULL) {
132         /*
133          * Flush any data in RX buffer.
134          * This helps to reset communication in case of on-the-fly device management
135          * if one-or-more device(s) are added or removed.
136          *
137          * Any noise on UART level could start byte and put it to Win buffer,
138          * preventing to read aligned data from it
139          */
140         PurgeComm(com_port, PURGE_RXCLEAR | PURGE_RXABORT);
141
142         /* Write file and send data */
143         WriteFile(com_port, tx, len, &br, NULL);
144         FlushFileBuffers(com_port);
145
146         /* Read same amount of data as sent previously (loopback) */
147         do {
148             if (ReadFile(com_port, rx, (DWORD)(len - read), &br, NULL)) {
149                 read += (size_t)br;
150                 rx += (size_t)br;
151             }
152         } while (read < len);
153     }
154 }

```

(continues on next page)



(continued from previous page)

```

155     return 1;
156 }
157
158 #endif /* !__DOXYGEN__ */

```

## Example: Low-level driver for STM32

Example code for low-level porting on *STM32* platform.

Listing 4: Actual implementation of low-level driver for STM32

```

1  /**
2   * \file          lwow_ll_stm32.c
3   * \brief         Generic UART implementation for STM32 MCUs
4   */
5
6  /**
7   * Copyright (c) 2020 Tilen MAJERLE
8   *
9   * Permission is hereby granted, free of charge, to any person
10  * obtaining a copy of this software and associated documentation
11  * files (the "Software"), to deal in the Software without restriction,
12  * including without limitation the rights to use, copy, modify, merge,
13  * publish, distribute, sublicense, and/or sell copies of the Software,
14  * and to permit persons to whom the Software is furnished to do so,
15  * subject to the following conditions:
16  *
17  * The above copyright notice and this permission notice shall be
18  * included in all copies or substantial portions of the Software.
19  *
20  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
21  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
22  * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
23  * AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
24  * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
25  * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
26  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
27  * OTHER DEALINGS IN THE SOFTWARE.
28  *
29  * This file is part of LwOW - Lightweight onewire library.
30  *
31  * Author:          Tilen MAJERLE <tilen@majerle.eu>
32  * Version:         v3.0.1
33  */
34
35  /**
36   * How it works
37   *
38   * https://docs.majerle.eu/projects/lwow/en/latest/user-manual/hw\_connection.html#
39   */
40  #include "lwow/lwow.h"
41
42  #if !__DOXYGEN__
43
44  static uint8_t init(void* arg);

```

(continues on next page)

(continued from previous page)

```

45 static uint8_t deinit(void* arg);
46 static uint8_t set_baudrate(uint32_t baud, void* arg);
47 static uint8_t transmit_receive(const uint8_t* tx, uint8_t* rx, size_t len, void* _
  ↪arg);
48
49 /* STM32 LL driver for OW */
50 const lwow_ll_drv_t
51 lwow_ll_drv_stm32 = {
52     .init = init,
53     .deinit = deinit,
54     .set_baudrate = set_baudrate,
55     .tx_rx = transmit_receive,
56 };
57
58 static LL_USART_InitTypeDef
59 usart_init;
60
61 static uint8_t
62 init(void* arg) {
63     LL_GPIO_InitTypeDef gpio_init;
64
65     /* Peripheral clock enable */
66     ONEWIRE_USART_CLK_EN;
67     ONEWIRE_TX_PORT_CLK_EN;
68     ONEWIRE_RX_PORT_CLK_EN;
69
70     /* Configure GPIO pins */
71     LL_GPIO_StructInit(&gpio_init);
72     gpio_init.Mode = LL_GPIO_MODE_ALTERNATE;
73     gpio_init.Speed = LL_GPIO_SPEED_FREQ_HIGH;
74     gpio_init.OutputType = LL_GPIO_OUTPUT_OPENDRAIN;
75     gpio_init.Pull = LL_GPIO_PULL_UP;
76
77     /* TX pin */
78     gpio_init.Alternate = ONEWIRE_TX_PIN_AF;
79     gpio_init.Pin = ONEWIRE_TX_PIN;
80     LL_GPIO_Init(ONEWIRE_TX_PORT, &gpio_init);
81
82     /* RX pin */
83     gpio_init.Alternate = ONEWIRE_RX_PIN_AF;
84     gpio_init.Pin = ONEWIRE_RX_PIN;
85     LL_GPIO_Init(ONEWIRE_RX_PORT, &gpio_init);
86
87     /* Configure UART peripherals */
88     LL_USART_DeInit(ONEWIRE_USART);
89     LL_USART_StructInit(&usart_init);
90     usart_init.BaudRate = 9600;
91     usart_init.DataWidth = LL_USART_DATAWIDTH_8B;
92     usart_init.StopBits = LL_USART_STOPBITS_1;
93     usart_init.Parity = LL_USART_PARITY_NONE;
94     usart_init.TransferDirection = LL_USART_DIRECTION_TX_RX;
95     usart_init.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
96     usart_init.OverSampling = LL_USART_OVERSAMPLING_16;
97     LL_USART_Init(ONEWIRE_USART, &usart_init);
98     LL_USART_ConfigAsyncMode(ONEWIRE_USART);
99
100     LWOW_UNUSED(arg);

```

(continues on next page)

(continued from previous page)

```

101     return 1;
102 }
103
104
105 static uint8_t
106 deinit(void* arg) {
107     LL_USART_DeInit(ONEWIRE_USART);
108     LWOW_UNUSED(arg);
109     return 1;
110 }
111
112 static uint8_t
113 set_baudrate(uint32_t baud, void* arg) {
114     usart_init.BaudRate = baud;
115     LL_USART_Init(ONEWIRE_USART, &usart_init);
116     LL_USART_ConfigAsyncMode(ONEWIRE_USART);
117     LWOW_UNUSED(arg);
118
119     return 1;
120 }
121
122 static uint8_t
123 transmit_receive(const uint8_t* tx, uint8_t* rx, size_t len, void* arg) {
124     const uint8_t* t = tx;
125     uint8_t* r = rx;
126
127     /* Send byte with polling */
128     LL_USART_Enable(ONEWIRE_USART);
129     for (; len > 0; --len, ++t, ++r) {
130         LL_USART_TransmitData8(ONEWIRE_USART, *t);
131         while (!LL_USART_IsActiveFlag_TXE(ONEWIRE_USART));
132         while (!LL_USART_IsActiveFlag_RXNE(ONEWIRE_USART));
133         *r = LL_USART_ReceiveData8(ONEWIRE_USART);
134     }
135     while (!LL_USART_IsActiveFlag_TC(ONEWIRE_USART)) {}
136     LL_USART_Disable(ONEWIRE_USART);
137     LWOW_UNUSED(arg);
138     return 1;
139 }
140
141 #endif /* !__DOXYGEN__ */

```

## Example: System functions for WIN32

Listing 5: Actual implementation of system functions for WIN32

```

1  /**
2   * \file          lwow_sys_win32.c
3   * \brief         System functions for WIN32
4   */
5
6  /**
7   * Copyright (c) 2020 Tilen MAJERLE
8   *
9   * Permission is hereby granted, free of charge, to any person

```

(continues on next page)

(continued from previous page)

```

10  * obtaining a copy of this software and associated documentation
11  * files (the "Software"), to deal in the Software without restriction,
12  * including without limitation the rights to use, copy, modify, merge,
13  * publish, distribute, sublicense, and/or sell copies of the Software,
14  * and to permit persons to whom the Software is furnished to do so,
15  * subject to the following conditions:
16  *
17  * The above copyright notice and this permission notice shall be
18  * included in all copies or substantial portions of the Software.
19  *
20  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
21  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
22  * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
23  * AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
24  * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
25  * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
26  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
27  * OTHER DEALINGS IN THE SOFTWARE.
28  *
29  * This file is part of LwOW - Lightweight onewire library.
30  *
31  * Author:          Tilen MAJERLE <tilen@majerle.eu>
32  * Version:         v3.0.1
33  */
34 #include "lwow/lwow.h"
35 #include "windows.h"
36
37 #if LWOW_CFG_OS && !__DOXYGEN__
38
39 uint8_t
40 lwow_sys_mutex_create(LWOW_CFG_OS_MUTEX_HANDLE* mutex, void* arg) {
41     *mutex = CreateMutex(NULL, 0, NULL);
42     return 1;
43 }
44
45 uint8_t
46 lwow_sys_mutex_delete(LWOW_CFG_OS_MUTEX_HANDLE* mutex, void* arg) {
47     CloseHandle(*mutex);
48     *mutex = NULL;
49     return 1;
50 }
51
52 uint8_t
53 lwow_sys_mutex_wait(LWOW_CFG_OS_MUTEX_HANDLE* mutex, void* arg) {
54     return WaitForSingleObject(*mutex, INFINITE) == WAIT_OBJECT_0;
55 }
56
57 uint8_t
58 lwow_sys_mutex_release(LWOW_CFG_OS_MUTEX_HANDLE* mutex, void* arg) {
59     return ReleaseMutex(*mutex);
60 }
61
62 #endif /* LWOW_CFG_OS && !__DOXYGEN__ */

```

## Example: System functions for CMSIS-OS

Listing 6: Actual implementation of system functions for CMSIS-OS

```

1  /**
2   * \file          lwow_sys_cmsis_os.c
3   * \brief         System functions for CMSIS-OS based operating system
4   */
5
6  /*
7   * Copyright (c) 2020 Tilen MAJERLE
8   *
9   * Permission is hereby granted, free of charge, to any person
10  * obtaining a copy of this software and associated documentation
11  * files (the "Software"), to deal in the Software without restriction,
12  * including without limitation the rights to use, copy, modify, merge,
13  * publish, distribute, sublicense, and/or sell copies of the Software,
14  * and to permit persons to whom the Software is furnished to do so,
15  * subject to the following conditions:
16  *
17  * The above copyright notice and this permission notice shall be
18  * included in all copies or substantial portions of the Software.
19  *
20  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
21  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
22  * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
23  * AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
24  * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
25  * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
26  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
27  * OTHER DEALINGS IN THE SOFTWARE.
28  *
29  * This file is part of LwOW - Lightweight onewire library.
30  *
31  * Author:          Tilen MAJERLE <tilen@majerle.eu>
32  * Version:         v3.0.1
33  */
34  #include "lwow/lwow.h"
35
36  #if LWOW_CFG_OS && !__DOXYGEN__
37
38  #include "cmsis_os.h"
39
40  uint8_t
41  lwow_sys_mutex_create(LWOW_CFG_OS_MUTEX_HANDLE* m, void* arg) {
42      LWOW_UNUSED(arg);
43      const osMutexAttr_t attr = {
44          .attr_bits = osMutexRecursive
45      };
46
47      *m = osMutexNew(&attr);
48      return *m != NULL;
49  }
50
51  uint8_t
52  lwow_sys_mutex_delete(LWOW_CFG_OS_MUTEX_HANDLE* m, void* arg) {
53      LWOW_UNUSED(arg);

```

(continues on next page)

(continued from previous page)

```

54     return osMutexDelete(*m) == osOK;
55 }
56
57 uint8_t
58 lwow_sys_mutex_wait(LWOW_CFG_OS_MUTEX_HANDLE* m, void* arg) {
59     LWOW_UNUSED(arg);
60     return osMutexAcquire(*m, osWaitForever) == osOK;
61 }
62
63 uint8_t
64 lwow_sys_mutex_release(LWOW_CFG_OS_MUTEX_HANDLE* m, void* arg) {
65     LWOW_UNUSED(arg);
66     return osMutexRelease(*m) == osOK;
67 }
68
69 #endif /* LWOW_CFG_OS && !__DOXYGEN__ */

```

## Low-Level driver for STM32 with STM32CubeMX

Specific low-level driver has been implemented for STM32 series of microcontrollers, to allow easy and simple link of LwOW library with projects generated with STM32CubeMX or STm32CubeIDE development tools.

Driver is based on HAL (Hardware Abstraction Layer) and it uses interrupt configuration to transmit/receive data. When customer starts a new project using CubeMX, it must:

- Configure specific UART IP as async mode both directions
- UART must have enabled global interrupts, to allow transmitting/receiving data using interrupts
- Application must pass pointer to UART handle when calling `ow_init` function

**Tip:** Special example has been developed to demonstrate how can application use multiple OneWire instances on multiple UART ports at the same time. It uses custom argument to determine which UART handle shall be used for data transmit. Check `/examples/stm32/` folder for actual implementation.

Listing 7: Actual implementation of low-level driver for STM32 with HAL drivers

```

1  /**
2   * \file          lwow_ll_stm32_hal.c
3   * \brief         UART driver implementation for STM32 with HAL code
4   */
5
6  /*
7   * Copyright (c) 2020 Tilen MAJERLE
8   *
9   * Permission is hereby granted, free of charge, to any person
10  * obtaining a copy of this software and associated documentation
11  * files (the "Software"), to deal in the Software without restriction,
12  * including without limitation the rights to use, copy, modify, merge,
13  * publish, distribute, sublicense, and/or sell copies of the Software,
14  * and to permit persons to whom the Software is furnished to do so,
15  * subject to the following conditions:
16  *

```

(continues on next page)

(continued from previous page)

```

17  * The above copyright notice and this permission notice shall be
18  * included in all copies or substantial portions of the Software.
19  *
20  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
21  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
22  * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
23  * AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
24  * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
25  * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
26  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
27  * OTHER DEALINGS IN THE SOFTWARE.
28  *
29  * This file is part of LwOW - Lightweight onewire library.
30  *
31  * Author:          Tilen MAJERLE <tilen@majerle.eu>
32  * Version:         v3.0.1
33  */
34
35 /*
36  * How it works (general)
37  *
38  * https://docs.majerle.eu/projects/lwow/en/latest/user-manual/hw\_connection.html#
39  *
40  * This specific driver is optimized for proejcts generated by STM32CubeMX or
41  * ↪STM32CubeIDE with HAL drivers
42  *
43  * It can be used w/ or w/o operating system and it uses interrupts & polling for
44  * ↪data receive and data transmit.
45  *
46  * Application must pass pointer to UART handle as argument to ow_init function in
47  * ↪order
48  * to link OW instance with actual UART hardware used for OW instance.
49  *
50  * To use this driver, application must:
51  * - Enable interrupt in CubeMX to allow HAL_UART_Receive_IT functionality
52  * - Use pointer to UART handle when initializing ow with ow_init
53  */
54
55 #include "lwow/lwow.h"
56 #include "main.h"                                /* Generated normally by CubeMX */
57
58 #if !__DOXYGEN__
59
60 static uint8_t init(void* arg);
61 static uint8_t deinit(void* arg);
62 static uint8_t set_baudrate(uint32_t baud, void* arg);
63 static uint8_t transmit_receive(const uint8_t* tx, uint8_t* rx, size_t len, void*
64 ↪arg);
65
66 /* STM32 LL driver for OW */
67 const lwow_ll_drv_t
68 lwow_ll_drv_stm32_hal = {
69     .init = init,
70     .deinit = deinit,
71     .set_baudrate = set_baudrate,
72     .tx_rx = transmit_receive,
73 };
74
75 static uint8_t

```

(continues on next page)

```

70 init(void* arg) {
71     UART_HandleTypeDef* huart = arg;
72
73     LWOW_ASSERT0("arg != NULL", arg != NULL);
74
75     /* Initialize UART */
76     HAL_UART_DeInit(huart);
77     return HAL_UART_Init(huart) == HAL_OK;
78 }
79
80 static uint8_t
81 deinit(void* arg) {
82     UART_HandleTypeDef* huart = arg;
83
84     LWOW_ASSERT0("arg != NULL", arg != NULL);
85
86     return HAL_UART_DeInit(huart);
87 }
88
89 static uint8_t
90 set_baudrate(uint32_t baud, void* arg) {
91     UART_HandleTypeDef* huart = arg;
92
93     LWOW_ASSERT0("arg != NULL", arg != NULL);
94
95     huart->Init.BaudRate = baud;
96     return init(huart);
97 }
98
99 static uint8_t
100 transmit_receive(const uint8_t* tx, uint8_t* rx, size_t len, void* arg) {
101     UART_HandleTypeDef* huart = arg;
102     uint32_t start;
103
104     LWOW_ASSERT0("arg != NULL", arg != NULL);
105
106     /* Get current HAL tick */
107     start = HAL_GetTick();
108
109     /* Start RX in interrupt mode */
110     HAL_UART_Receive_IT(huart, rx, len);
111
112     /* Process TX in polling mode */
113     HAL_UART_Transmit(huart, (void*)tx, len, 100);
114
115     /* Wait RX to finish */
116     while (huart->RxState != HAL_UART_STATE_READY) {
117         if (HAL_GetTick() - start > 100) {
118             return 0;
119         }
120     }
121
122     return 1;
123 }
124
125 #endif /* !__DOXYGEN__ */

```



## 5.3 API reference

List of all the modules:

### 5.3.1 LwOW

*group* **LWOW**

Lightweight onewire.

**Note** Functions with `_raw` suffix do not implement locking mechanism when used with operating system.

#### Defines

**LWOW\_UNUSED** (*x*)

Unused variable macro

**LWOW\_ASSERT** (*msg, c*)

Assert check function.

It returns *lwowERRPAR* if condition check fails

#### Parameters

- [*in*] *msg*: Optional message parameter to print on failure
- [*in*] *c*: Condition to check for

**LWOW\_ASSERT0** (*msg, c*)

Assert check function with return 0

It returns 0 if condition check fails

#### Parameters

- [*in*] *msg*: Optional message parameter to print on failure
- [*in*] *c*: Condition to check for

**LWOW\_ARRAYSIZE** (*x*)

Get size of statically declared array.

**Return** Number of array elements

#### Parameters

- [*in*] *x*: Input array

**LWOW\_CMD\_RSCRATCHPAD**

Read scratchpad command for 1-Wire devices

**LWOW\_CMD\_WSCRATCHPAD**

Write scratchpad command for 1-Wire devices

**LWOW\_CMD\_CPYSCRATCHPAD**

Copy scratchpad command for 1-Wire devices

**LWOW\_CMD\_RECEEPROM**

Read EEPROM command

**LWOW\_CMD\_RPWRSUPPLY**

Read power supply command

**LWOW\_CMD\_SEARCHROM**

Search ROM command

**LWOW\_CMD\_READROM**

Read ROM command

**LWOW\_CMD\_MATCHROM**

Match ROM command. Select device with specific ROM

**LWOW\_CMD\_SKIPROM**

Skip ROM, select all devices

**Typedefs**

```
typedef lwowr_t (*lwow_search_cb_fn) (lwow_t *const ow, const lwow_rom_t *const  
rom_id, size_t index, void *arg)
```

Search callback function implementation.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: Rom address when new device detected. Set to NULL when search finished
- [in] index: Current device index When rom\_id = NULL, value indicates number of total devices found
- [in] arg: Custom user argument

**Enums****enum *lwowr\_t***

1-Wire result enumeration

*Values:***enumerator *lwowOK***

Device returned OK

**enumerator *lwowERRPRESENCE***

Presence was not successful

**enumerator *lwowERRNODEV***

No device connected, maybe device removed during scan?

**enumerator *lwowERRTXRX***

Error while exchanging data

**enumerator *lwowERRBAUD***

Error setting baudrate

**enumerator *lwowERRPAR***

Parameter error

**enumerator *lwowERR***

General-Purpose error

## Functions

*lwowr\_t* **lwow\_init** (*lwow\_t*\***const** ow, **const** *lwow\_ll\_drv\_t*\***const** ll\_drv, void \*arg)  
Initialize OneWire instance.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

### Parameters

- [in] ow: OneWire instance
- [in] ll\_drv: Low-level driver
- [in] arg: Custom argument

void **lwow\_deinit** (*lwow\_t*\***const** ow)  
Deinitialize OneWire instance.

### Parameters

- [in] ow: OneWire instance

*lwowr\_t* **lwow\_protect** (*lwow\_t*\***const** ow, **const** uint8\_t protect)  
Protect 1-wire from concurrent access.

**Note** Used only for OS systems

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

### Parameters

- [inout] ow: 1-Wire handle
- [in] protect: Set to 1 to protect core, 0 otherwise

*lwowr\_t* **lwow\_unprotect** (*lwow\_t*\***const** ow, **const** uint8\_t protect)  
Unprotect 1-wire from concurrent access.

**Note** Used only for OS systems

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

### Parameters

- [inout] ow: 1-Wire handle
- [in] protect: Set to 1 to protect core, 0 otherwise

*lwowr\_t* **lwow\_reset\_raw** (*lwow\_t*\***const** ow)  
Reset 1-Wire bus and set connected devices to idle state.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

### Parameters

- [inout] ow: 1-Wire handle

*lwowr\_t* **lwow\_reset** (*lwow\_t*\***const** ow)  
Reset 1-Wire bus and set connected devices to idle state.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle

*lwowr\_t* **lwow\_write\_byte\_ex\_raw** (*lwow\_t*\*const ow, const uint8\_t btw, uint8\_t\*const br)

Write byte over OW and read its response.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Parameters**

- [inout] ow: 1-Wire handle
- [in] btw: Byte to write
- [out] br: Pointer to read value. Set to NULL if not used

*lwowr\_t* **lwow\_write\_byte\_ex** (*lwow\_t*\*const ow, const uint8\_t btw, uint8\_t\*const br)

Write byte over OW and read its response.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle
- [in] btw: Byte to write
- [out] br: Pointer to read value. Set to NULL if not used

*lwowr\_t* **lwow\_read\_byte\_ex\_raw** (*lwow\_t*\*const ow, uint8\_t\*const br)

Read byte from OW device.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Parameters**

- [inout] ow: 1-Wire handle
- [out] br: Pointer to save read value

*lwowr\_t* **lwow\_read\_byte\_ex** (*lwow\_t*\*const ow, uint8\_t\*const br)

Read byte from OW device.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle
- [out] br: Pointer to save read value

*lwowr\_t* **lwow\_read\_bit\_ex\_raw** (*lwow\_t*\*const ow, uint8\_t\*const br)

Read single bit from OW device.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Parameters**

- [inout] ow: 1-Wire handle
- [out] br: Pointer to save read value, either 1 or 0

*lwowr\_t* **lwow\_read\_bit\_ex** (*lwow\_t* \*const ow, uint8\_t \*const br)

Read single bit from OW device.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle
- [out] br: Pointer to save read value, either 1 or 0

*lwowr\_t* **lwow\_search\_reset\_raw** (*lwow\_t* \*const ow)

Reset search.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Parameters**

- [inout] ow: 1-Wire handle

*lwowr\_t* **lwow\_search\_reset** (*lwow\_t* \*const ow)

Reset search.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle

*lwowr\_t* **lwow\_search\_raw** (*lwow\_t* \*const ow, *lwow\_rom\_t* \*const rom\_id)

Search for devices on 1-wire bus.

**Note** To reset search and to start over, use *lwow\_search\_reset* function

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Parameters**

- [inout] ow: 1-Wire handle
- [out] rom\_id: Pointer to ROM structure to save ROM

*lwowr\_t* **lwow\_search** (*lwow\_t* \*const ow, *lwow\_rom\_t* \*const rom\_id)

Search for devices on 1-wire bus.

**Note** To reset search and to start over, use *lwow\_search\_reset* function

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle
- [out] rom\_id: Pointer to ROM structure to save ROM

*lwowr\_t* **lwow\_search\_with\_command\_raw**(*lwow\_t* \*const ow, const uint8\_t cmd,  
  *lwow\_rom\_t* \*const rom\_id)

Search for devices on 1-wire bus with custom search command.

**Note** To reset search and to start over, use *lwow\_search\_reset* function

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Parameters**

- [inout] ow: 1-Wire handle
- [in] cmd: command to use for search operation
- [out] rom\_id: Pointer to ROM structure to store address

*lwowr\_t* **lwow\_search\_with\_command**(*lwow\_t* \*const ow, const uint8\_t cmd, *lwow\_rom\_t*  
  \*const rom\_id)

Search for devices on 1-wire bus with custom search command.

**Note** To reset search and to start over, use *lwow\_search\_reset* function

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle
- [in] cmd: command to use for search operation
- [out] rom\_id: Pointer to ROM structure to store address

*lwowr\_t* **lwow\_search\_with\_command\_callback**(*lwow\_t* \*const ow, const uint8\_t cmd,  
  size\_t \*const roms\_found, const  
  *lwow\_search\_cb\_fn* func, void \*const  
  arg)

Search devices on 1-wire network by using callback function and custom search command.

When new device is detected, callback function *func* is called to notify user

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [in] ow: 1-Wire handle
- [in] cmd: 1-Wire search command
- [out] roms\_found: Output variable to save number of found devices. Set to NULL if not used
- [in] func: Callback function to call for each device
- [in] arg: Custom user argument, used in callback function

*lwowr\_t* **lwow\_search\_with\_callback** (*lwow\_t* \***const** ow, size\_t \***const** roms\_found, **const** *lwow\_search\_cb\_fn* func, void \***const** arg)

Search devices on 1-wire network by using callback function and SEARCH\_ROM 1-Wire command.

When new device is detected, callback function func is called to notify user

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

#### Parameters

- [in] ow: 1-Wire handle
- [out] roms\_found: Output variable to save number of found devices. Set to NULL if not used
- [in] func: Callback function to call for each device
- [in] arg: Custom user argument, used in callback function

*lwowr\_t* **lwow\_search\_devices\_with\_command\_raw** (*lwow\_t* \***const** ow, **const** uint8\_t cmd, *lwow\_rom\_t* \***const** rom\_id\_arr, **const** size\_t rom\_len, size\_t \***const** roms\_found)

Search for devices on 1-Wire network with command and store ROM IDs to input array.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

#### Parameters

- [in] ow: 1-Wire handle
- [in] cmd: 1-Wire search command
- [in] rom\_id\_arr: Pointer to output array to store found ROM IDs into
- [in] rom\_len: Length of input ROM array
- [out] roms\_found: Output variable to save number of found devices. Set to NULL if not used

*lwowr\_t* **lwow\_search\_devices\_with\_command** (*lwow\_t* \***const** ow, **const** uint8\_t cmd, *lwow\_rom\_t* \***const** rom\_id\_arr, **const** size\_t rom\_len, size\_t \***const** roms\_found)

Search for devices on 1-Wire network with command and store ROM IDs to input array.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

#### Parameters

- [in] ow: 1-Wire handle
- [in] cmd: 1-Wire search command
- [in] rom\_id\_arr: Pointer to output array to store found ROM IDs into
- [in] rom\_len: Length of input ROM array
- [out] roms\_found: Output variable to save number of found devices. Set to NULL if not used

*lwowr\_t* **lwow\_search\_devices\_raw** (*lwow\_t* \***const** ow, *lwow\_rom\_t* \***const** rom\_id\_arr, **const** size\_t rom\_len, size\_t \***const** roms\_found)

Search for devices on 1-Wire network with default command and store ROM IDs to input array.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id\_arr: Pointer to output array to store found ROM IDs into
- [in] rom\_len: Length of input ROM array
- [out] roms\_found: Output variable to save number of found devices. Set to NULL if not used

*lwowr\_t* **lwow\_search\_devices** (*lwow\_t* \***const** ow, *lwow\_rom\_t* \***const** rom\_id\_arr, **const** size\_t rom\_len, size\_t \***const** roms\_found)

Search for devices on 1-Wire network with default command and store ROM IDs to input array.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id\_arr: Pointer to output array to store found ROM IDs into
- [in] rom\_len: Length of input ROM array
- [out] roms\_found: Output variable to save number of found devices. Set to NULL if not used

*lwowr\_t* **lwow\_match\_rom\_raw** (*lwow\_t* \***const** ow, **const** *lwow\_rom\_t* \***const** rom\_id)

Select device on 1-wire network with exact ROM number.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to match device

*lwowr\_t* **lwow\_match\_rom** (*lwow\_t* \***const** ow, **const** *lwow\_rom\_t* \***const** rom\_id)

Select device on 1-wire network with exact ROM number.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to match device

*lwowr\_t* **lwow\_skip\_rom\_raw** (*lwow\_t* \***const** ow)

Skip ROM address and select all devices on the network.



**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Parameters**

- [in] ow: 1-Wire handle

*lwowr\_t* **lwow\_skip\_rom** (*lwow\_t* \*const ow)

Skip ROM address and select all devices on the network.

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [in] ow: 1-Wire handle

uint8\_t **lwow\_crc** (const void \*const in, const size\_t len)

Calculate CRC-8 of input data.

**Return** Calculated CRC

**Note** This function is reentrant

**Parameters**

- [in] in: Input data
- [in] len: Number of bytes

uint8\_t **lwow\_write\_byte\_raw** (*lwow\_t* \*const ow, const uint8\_t b)

Write byte over 1-wire protocol.

*Deprecated:*

This function is deprecated. Use *lwow\_write\_byte\_ex\_raw* instead

**Note** This function is deprecated. Use *lwow\_write\_byte\_ex\_raw* instead

**Return** Received byte over 1-wire protocol

**Parameters**

- [inout] ow: 1-Wire handle
- [in] b: Byte to write

uint8\_t **lwow\_write\_byte** (*lwow\_t* \*const ow, const uint8\_t b)

Write byte over 1-wire protocol.

*Deprecated:*

This function is deprecated. Use *lwow\_write\_byte\_ex\_raw* instead

*Deprecated:*

This function is deprecated. Use *lwow\_write\_byte\_ex* instead

**Note** This function is deprecated. Use *lwow\_write\_byte\_ex\_raw* instead

**Return** Received byte over 1-wire protocol

**Note** This function is deprecated. Use *lwow\_write\_byte\_ex* instead

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle
- [in] b: Byte to write

uint8\_t **lwow\_read\_byte\_raw** (*lwow\_t* \*const ow)

Read next byte on 1-Wire.

*Deprecated:*

This function is deprecated. Use *lwow\_read\_byte\_ex\_raw* instead

**Note** This function is deprecated. Use *lwow\_read\_byte\_ex\_raw* instead

**Return** Byte read over 1-Wire

**Parameters**

- [inout] ow: 1-Wire handle

uint8\_t **lwow\_read\_byte** (*lwow\_t* \*const ow)

Read next byte on 1-Wire.

*Deprecated:*

This function is deprecated. Use *lwow\_read\_byte\_ex\_raw* instead

*Deprecated:*

This function is deprecated. Use *lwow\_read\_byte\_ex* instead

**Note** This function is deprecated. Use *lwow\_read\_byte\_ex\_raw* instead

**Return** Byte read over 1-Wire

**Note** This function is deprecated. Use *lwow\_read\_byte\_ex* instead

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle

uint8\_t **lwow\_read\_bit\_raw** (*lwow\_t* \*const ow)

Read single bit on 1-Wire network.

*Deprecated:*

This function is deprecated. Use *lwow\_read\_bit\_ex\_raw* instead

**Note** This function is deprecated. Use *lwow\_read\_bit\_ex\_raw* instead

**Return** Bit value

**Parameters**

- [inout] ow: 1-Wire handle

uint8\_t **lwow\_read\_bit** (*lwow\_t* \*const ow)

Read single bit on 1-Wire network.

*Deprecated:*

This function is deprecated. Use *lwow\_read\_bit\_ex\_raw* instead

*Deprecated:*

This function is deprecated. Use *lwow\_read\_bit\_ex* instead

**Note** This function is deprecated. Use *lwow\_read\_bit\_ex\_raw* instead

**Return** Bit value

**Note** This function is deprecated. Use *lwow\_read\_bit\_ex* instead

**Note** This function is thread-safe

**Parameters**

- [inout] ow: 1-Wire handle

**struct lwow\_rom\_t**  
*#include <lwow.h>* ROM structure.

**Public Members**

uint8\_t **rom**[8]  
8-bytes ROM address

**struct lwow\_t**  
*#include <lwow.h>* 1-Wire structure

**Public Members**

*lwow\_rom\_t* **rom**  
ROM address of last device found. When searching for new devices, we always need last found address, to be able to decide which way to go next time during scan.

uint8\_t **discrepancy**  
Discrepancy value on last search

void \***arg**  
User custom argument

**const** *lwow\_ll\_drv\_t* \***ll\_drv**  
Low-level functions driver

LWOW\_CFG\_OS\_MUTEX\_HANDLE **mutex**  
Mutex handle

### 5.3.2 Configuration

This is the default configuration of the middleware. When any of the settings shall be modified, it shall be done in dedicated application config *lwow\_opts.h* file.

---

**Note:** Check *Getting started* for guidelines on how to create and use configuration file.

---

*group* **LWOW\_OPT**  
OW options.

## Defines

### **LWOW\_CFG\_OS**

Enables 1 or disables 0 operating system support in the library.

**Note** When `LWOW_CFG_OS` is enabled, user must implement functions in *System functions* group.

### **LWOW\_CFG\_OS\_MUTEX\_HANDLE**

Mutex handle type.

**Note** This value must be set in case *LWOW\_CFG\_OS* is set to 1. If data type is not known to compiler, include header file with definition before you define handle type

## 5.3.3 Platform specific

List of all the modules:

### Low-level driver

#### *group* **LWOW\_LL**

Low-level device dependant functions.

#### **struct lwow\_ll\_drv\_t**

*#include <lwow.h>* 1-Wire low-level driver structure

### Public Members

`uint8_t (*init)(void *arg)`

Initialize low-level driver.

**Return** 1 on success, 0 otherwise

#### **Parameters**

- [in] `arg`: Custom argument passed to *lwow\_init* function

`uint8_t (*deinit)(void *arg)`

De-initialize low-level driver.

**Return** 1 on success, 0 otherwise

#### **Parameters**

- [in] `arg`: Custom argument passed to *lwow\_init* function

`uint8_t (*set_baudrate)(uint32_t baud, void *arg)`

Set UART baudrate.

**Return** 1 on success, 0 otherwise

#### **Parameters**

- [in] `baud`: Baudrate to set in units of bauds, normally 9600 or 115200
- [in] `arg`: Custom argument passed to *lwow\_init* function

`uint8_t (*tx_rx)(const uint8_t *tx, uint8_t *rx, size_t len, void *arg)`

Transmit and receive bytes over UART hardware (or custom implementation)

Bytes array for `tx` is already prepared to be directly transmitted over UART hardware, no data manipulation is necessary.

At the same time, library must read received data on RX port and put it to `rx` data array, one by one, up to `len` number of bytes

**Return** 1 on success, 0 otherwise

**Parameters**

- [in] `tx`: Data to transmit over UART
- [out] `rx`: Array to write received data to
- [in] `len`: Number of bytes to exchange
- [in] `arg`: Custom argument passed to [lwow\\_init](#) function

## System functions

System function are used in conjunction with thread safety. These are required when operating system is used and multiple threads want to access to the same OneWire instance.

---

**Tip:** Check [Thread safety](#) and [Porting guide](#) for instructions on how to port.

---

Below is a list of function prototypes and its implementation details.

*group* **LWOW\_SYS**

System functions when used with operating system.

## Functions

`uint8_t lwow_sys_mutex_create (LWOW_CFG_OS_MUTEX_HANDLE *mutex, void *arg)`

Create a new mutex and assign value to handle.

**Return** 1 on success, 0 otherwise

**Parameters**

- [out] `mutex`: Output variable to save mutex handle
- [in] `arg`: User argument passed on [lwow\\_init](#) function

`uint8_t lwow_sys_mutex_delete (LWOW_CFG_OS_MUTEX_HANDLE *mutex, void *arg)`

Delete existing mutex and invalidate mutex variable.

**Return** 1 on success, 0 otherwise

**Parameters**

- [in] `mutex`: Mutex handle to remove and invalidate
- [in] `arg`: User argument passed on [lwow\\_init](#) function

`uint8_t lwow_sys_mutex_wait (LWOW_CFG_OS_MUTEX_HANDLE *mutex, void *arg)`

Wait for a mutex until ready (unlimited time)

**Return** 1 on success, 0 otherwise

**Parameters**

- [in] `mutex`: Mutex handle to wait for
- [in] `arg`: User argument passed on [lwow\\_init](#) function

uint8\_t **lwow\_sys\_mutex\_release** (LWOW\_CFG\_OS\_MUTEX\_HANDLE \*mutex, void \*arg)  
Release already locked mutex.

**Return** 1 on success, 0 otherwise

**Parameters**

- [in] mutex: Mutex handle to release
- [in] arg: User argument passed on *lwow\_init* function

## 5.3.4 Device drivers

List of all supported device drivers

### DS18x20 temperature sensor

*group* **LWOW\_DEVICE\_DS18x20**  
Device driver for DS18x20 temperature sensor.

**Note** Functions with *\_raw* suffix do not implement locking mechanism when using with operating system.

#### Defines

**LWOW\_DS18X20\_ALARM\_DISABLE**  
Disable alarm temperature

**LWOW\_DS18X20\_ALARM\_NOCHANGE**  
Do not modify current alarm settings

**LWOW\_DS18X20\_TEMP\_MIN**  
Minimum temperature

**LWOW\_DS18X20\_TEMP\_MAX**  
Maximal temperature

#### Functions

uint8\_t **lwow\_ds18x20\_start\_raw** (*lwow\_t* \*const ow, const *lwow\_rom\_t* \*const rom\_id)  
Start temperature conversion on specific (or all) devices.

**Return** 1 on success, 0 otherwise

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to start measurement for. Set to NULL to start measurement on all devices at the same time

uint8\_t **lwow\_ds18x20\_start** (*lwow\_t* \*const ow, const *lwow\_rom\_t* \*const rom\_id)  
Start temperature conversion on specific (or all) devices.

**Return** 1 on success, 0 otherwise

**Note** This function is thread-safe

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to start measurement for. Set to NULL to start measurement on all devices at the same time

uint8\_t **lwow\_ds18x20\_read\_raw**(*lwow\_t* \*const ow, const *lwow\_rom\_t* \*const rom\_id, float \*const t)  
 Read temperature previously started with *lwow\_ds18x20\_start*.

**Return** 1 on success, 0 otherwise

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to read data from
- [out] t: Pointer to output float variable to save temperature

uint8\_t **lwow\_ds18x20\_read**(*lwow\_t* \*const ow, const *lwow\_rom\_t* \*const rom\_id, float \*const t)  
 Read temperature previously started with *lwow\_ds18x20\_start*.

**Return** 1 on success, 0 otherwise

**Note** This function is thread-safe

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to read data from
- [out] t: Pointer to output float variable to save temperature

uint8\_t **lwow\_ds18x20\_set\_resolution\_raw**(*lwow\_t* \*const ow, const *lwow\_rom\_t* \*const rom\_id, const uint8\_t bits)  
 Set resolution for DS18B20 sensor.

**Note** DS18S20 has fixed 9-bit resolution

**Return** 1 on success, 0 otherwise

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to set resolution
- [in] bits: Number of resolution bits. Possible values are 9 – 12

uint8\_t **lwow\_ds18x20\_set\_resolution**(*lwow\_t* \*const ow, const *lwow\_rom\_t* \*const rom\_id, const uint8\_t bits)  
 Set resolution for DS18B20 sensor.

**Note** DS18S20 has fixed 9-bit resolution

**Return** 1 on success, 0 otherwise

**Note** This function is thread-safe

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to set resolution
- [in] bits: Number of resolution bits. Possible values are 9 – 12

```
uint8_t lwow_ds18x20_get_resolution_raw(lwow_t *const ow, const lwow_rom_t  
                                         *const rom_id)
```

Get resolution for DS18B20 device.

**Return** Resolution in units of bits (9 – 12) on success, 0 otherwise

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to get resolution from

```
uint8_t lwow_ds18x20_get_resolution(lwow_t *const ow, const lwow_rom_t *const  
                                     rom_id)
```

Get resolution for DS18B20 device.

**Return** Resolution in units of bits (9 – 12) on success, 0 otherwise

**Note** This function is thread-safe

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to get resolution from

```
uint8_t lwow_ds18x20_set_alarm_temp_raw(lwow_t *const ow, const lwow_rom_t  
                                         *const rom_id, int8_t temp_l, int8_t temp_h)
```

Set/clear temperature alarm high/low levels in units of degree Celcius.

Example usage would look something similar to:

```
//Set alarm temperature; low = 10°C, high = 30°C  
lwow_ds18x20_set_alarm_temp(&ow, dev_id, 10, 30);  
//Set alarm temperature; low = disable, high = no change  
lwow_ds18x20_set_alarm_temp(&ow, dev_id, LWOW_DS18X20_ALARM_DISABLE, LWOW_  
↪DS18X20_ALARM_NOCHANGE);  
//Set alarm temperature; low = no change, high = disable  
lwow_ds18x20_set_alarm_temp(&ow, dev_id, LWOW_DS18X20_ALARM_NOCHANGE, LWOW_  
↪DS18X20_ALARM_DISABLE);  
//Set alarm temperature; low = 10°C, high = 30°C  
lwow_ds18x20_set_alarm_temp(&ow, dev_id, 10, 30);
```

**Note** temp\_h and temp\_l are high and low temperature alarms and can accept different values:

- -55 % 125, valid temperature range
- *LWOW\_DS18X20\_ALARM\_DISABLE* to disable temperature alarm (either high or low)
- *LWOW\_DS18X20\_ALARM\_NOCHANGE* to keep current alarm temperature (either high or low)

**Return** 1 on success, 0 otherwise

**Parameters**

- [in] ow: 1-Wire handle



- [in] rom\_id: 1-Wire device address
- [in] temp\_l: Alarm low temperature
- [in] temp\_h: Alarm high temperature

`uint8_t lwow_ds18x20_set_alarm_temp(lwow_t *const ow, const lwow_rom_t *const rom_id, int8_t temp_l, int8_t temp_h)`

Set/clear temperature alarm high/low levels in units of degree Celcius.

Example usage would look something similar to:

```
//Set alarm temperature; low = 10°C, high = 30°C
lwow_ds18x20_set_alarm_temp(&ow, dev_id, 10, 30);
//Set alarm temperature; low = disable, high = no change
lwow_ds18x20_set_alarm_temp(&ow, dev_id, LWOW_DS18X20_ALARM_DISABLE, LWOW_
↳DS18X20_ALARM_NOCHANGE);
//Set alarm temperature; low = no change, high = disable
lwow_ds18x20_set_alarm_temp(&ow, dev_id, LWOW_DS18X20_ALARM_NOCHANGE, LWOW_
↳DS18X20_ALARM_DISABLE);
//Set alarm temperature; low = 10°C, high = 30°C
lwow_ds18x20_set_alarm_temp(&ow, dev_id, 10, 30);
```

**Note** temp\_h and temp\_l are high and low temperature alarms and can accept different values:

- -55 % 125, valid temperature range
- *LWOW\_DS18X20\_ALARM\_DISABLE* to disable temperature alarm (either high or low)
- *LWOW\_DS18X20\_ALARM\_NOCHANGE* to keep current alarm temperature (either high or low)

**Return** 1 on success, 0 otherwise

**Note** This function is thread-safe

#### Parameters

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address
- [in] temp\_l: Alarm low temperature
- [in] temp\_h: Alarm high temperature

`lwowr_t lwow_ds18x20_search_alarm_raw(lwow_t *const ow, lwow_rom_t *const rom_id)`

Search for DS18x20 devices with alarm flag.

**Note** To reset search, use *lwow\_search\_reset* function

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

#### Parameters

- [in] ow: 1-Wire handle
- [out] rom\_id: Pointer to 8-byte long variable to save ROM

`lwowr_t lwow_ds18x20_search_alarm(lwow_t *const ow, lwow_rom_t *const rom_id)`

Search for DS18x20 devices with alarm flag.

**Note** To reset search, use *lwow\_search\_reset* function

**Return** *lwowOK* on success, member of *lwowr\_t* otherwise

**Note** This function is thread-safe

**Parameters**

- [in] ow: 1-Wire handle
- [out] rom\_id: Pointer to 8-byte long variable to save ROM

`uint8_t lwow_ds18x20_is_b(lwow_t *const ow, const lwow_rom_t *const rom_id)`  
Check if ROM address matches DS18B20 device.

**Return** 1 on success, 0 otherwise

**Note** This function is reentrant

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to test against DS18B20

`uint8_t lwow_ds18x20_is_s(lwow_t *const ow, const lwow_rom_t *const rom_id)`  
Check if ROM address matches DS18S20 device.

**Return** 1 on success, 0 otherwise

**Note** This function is reentrant

**Parameters**

- [in] ow: 1-Wire handle
- [in] rom\_id: 1-Wire device address to test against DS18S20

## 5.4 Examples and demos

Various examples are provided for fast library evaluation on embedded systems. These are prepared and maintained for 2 platforms, but could be easily extended to more platforms:

- WIN32 examples, prepared as [Visual Studio Community](#) projects
- ARM Cortex-M examples for STM32, prepared as [STM32CubeIDE](#) GCC projects

**Warning:** Library is platform independent and can be used on any platform.

### 5.4.1 Example architectures

There are many platforms available today on a market, however supporting them all would be tough task for single person. Therefore it has been decided to support (for purpose of examples) 2 platforms only, *WIN32* and *STM32*.

## WIN32

Examples for *WIN32* are prepared as [Visual Studio Community](#) projects. You can directly open project in the IDE, compile & debug.

To run examples on this architecture, external *USB to UART* converted would be necessary. Application opens *COM port* and sends/receives data directly to there.

---

**Tip:** Push-pull to open-drain external converter might be necessary. Check [Hardware connection with sensor](#) for more information.

---

## STM32

Embedded market is supported by many vendors and STMicroelectronics is, with their *STM32* series of microcontrollers, one of the most important players. There are numerous amount of examples and topics related to this architecture.

Examples for *STM32* are natively supported with [STM32CubeIDE](#), an official development IDE from STMicroelectronics.

You can run examples on one of official development boards, available in repository examples.

Table 1: Supported development boards

Board name	Onewire settings			Debug settings		
	UART	MTX	MRX	UART	MDTX	MDRX
STM32L496G-Discovery	USART1	PB6	PG10	USART2	PA2	PD6
STM32F429ZI-Nucleo	USART1	PA9	PA10	USART3	PD8	PD9

Pins to connect to 1-Wire sensor:

- *MTX*: MCU TX pin, connected to 1-Wire network data pin (together with MCU RX pin)
- *MRX*: MCU RX pin, connected to 1-Wire network data pin (together with MCU TX pin)
  - *TX* pin is configured as open-drain and can be safely connected directly with *RX* pin

Other pins are for your information and are used for debugging purposes on board.

- *MDTX*: MCU Debug TX pin, connected via on-board ST-Link to PC
- *MDRX*: MCU Debug RX pin, connected via on-board ST-Link to PC
- Baudrate is always set to 921600 bauds

### 5.4.2 Examples list

Here is a list of all examples coming with this library.

---

**Tip:** Examples are located in `/examples/` folder in downloaded package. Check [Download library](#) section to get your package.

---

### LwOW bare-metal

Simple example, not using operating system, showing basic configuration of the library. It can be also called *bare-metal* implementation for simple applications.

### LwOW OS

LwOW library as an example when multiple threads want to access to single LwOW core.

## L

- LWOW\_ARRAYSIZE (*C macro*), 29
- LWOW\_ASSERT (*C macro*), 29
- LWOW\_ASSERT0 (*C macro*), 29
- LWOW\_CFG\_OS (*C macro*), 40
- LWOW\_CFG\_OS\_MUTEX\_HANDLE (*C macro*), 40
- LWOW\_CMD\_CPYSCRATCHPAD (*C macro*), 29
- LWOW\_CMD\_MATCHROM (*C macro*), 30
- LWOW\_CMD\_READROM (*C macro*), 30
- LWOW\_CMD\_RECEEPROM (*C macro*), 29
- LWOW\_CMD\_RPWRSUPPLY (*C macro*), 29
- LWOW\_CMD\_RSCRATCHPAD (*C macro*), 29
- LWOW\_CMD\_SEARCHROM (*C macro*), 30
- LWOW\_CMD\_SKIPROM (*C macro*), 30
- LWOW\_CMD\_WSCRATCHPAD (*C macro*), 29
- lwow\_crc (*C++ function*), 37
- lwow\_deinit (*C++ function*), 31
- LWOW\_DS18X20\_ALARM\_DISABLE (*C macro*), 42
- LWOW\_DS18X20\_ALARM\_NOCHANGE (*C macro*), 42
- lwow\_ds18x20\_get\_resolution (*C++ function*), 44
- lwow\_ds18x20\_get\_resolution\_raw (*C++ function*), 44
- lwow\_ds18x20\_is\_b (*C++ function*), 46
- lwow\_ds18x20\_is\_s (*C++ function*), 46
- lwow\_ds18x20\_read (*C++ function*), 43
- lwow\_ds18x20\_read\_raw (*C++ function*), 43
- lwow\_ds18x20\_search\_alarm (*C++ function*), 45
- lwow\_ds18x20\_search\_alarm\_raw (*C++ function*), 45
- lwow\_ds18x20\_set\_alarm\_temp (*C++ function*), 45
- lwow\_ds18x20\_set\_alarm\_temp\_raw (*C++ function*), 44
- lwow\_ds18x20\_set\_resolution (*C++ function*), 43
- lwow\_ds18x20\_set\_resolution\_raw (*C++ function*), 43
- lwow\_ds18x20\_start (*C++ function*), 42
- lwow\_ds18x20\_start\_raw (*C++ function*), 42
- LWOW\_DS18X20\_TEMP\_MAX (*C macro*), 42
- LWOW\_DS18X20\_TEMP\_MIN (*C macro*), 42
- lwow\_init (*C++ function*), 31
- lwow\_ll\_drv\_t (*C++ struct*), 40
- lwow\_ll\_drv\_t::deinit (*C++ member*), 40
- lwow\_ll\_drv\_t::init (*C++ member*), 40
- lwow\_ll\_drv\_t::set\_baudrate (*C++ member*), 40
- lwow\_ll\_drv\_t::tx\_rx (*C++ member*), 40
- lwow\_match\_rom (*C++ function*), 36
- lwow\_match\_rom\_raw (*C++ function*), 36
- lwow\_protect (*C++ function*), 31
- lwow\_read\_bit (*C++ function*), 38
- lwow\_read\_bit\_ex (*C++ function*), 33
- lwow\_read\_bit\_ex\_raw (*C++ function*), 32
- lwow\_read\_bit\_raw (*C++ function*), 38
- lwow\_read\_byte (*C++ function*), 38
- lwow\_read\_byte\_ex (*C++ function*), 32
- lwow\_read\_byte\_ex\_raw (*C++ function*), 32
- lwow\_read\_byte\_raw (*C++ function*), 38
- lwow\_reset (*C++ function*), 31
- lwow\_reset\_raw (*C++ function*), 31
- lwow\_rom\_t (*C++ struct*), 39
- lwow\_rom\_t::rom (*C++ member*), 39
- lwow\_search (*C++ function*), 33
- lwow\_search\_cb\_fn (*C++ type*), 30
- lwow\_search\_devices (*C++ function*), 36
- lwow\_search\_devices\_raw (*C++ function*), 35
- lwow\_search\_devices\_with\_command (*C++ function*), 35
- lwow\_search\_devices\_with\_command\_raw (*C++ function*), 35
- lwow\_search\_raw (*C++ function*), 33
- lwow\_search\_reset (*C++ function*), 33
- lwow\_search\_reset\_raw (*C++ function*), 33
- lwow\_search\_with\_callback (*C++ function*), 34
- lwow\_search\_with\_command (*C++ function*), 34
- lwow\_search\_with\_command\_callback (*C++ function*), 34
- lwow\_search\_with\_command\_raw (*C++ function*), 34
- lwow\_skip\_rom (*C++ function*), 37
- lwow\_skip\_rom\_raw (*C++ function*), 36
- lwow\_sys\_mutex\_create (*C++ function*), 41

`lwow_sys_mutex_delete` (C++ *function*), 41  
`lwow_sys_mutex_release` (C++ *function*), 41  
`lwow_sys_mutex_wait` (C++ *function*), 41  
`lwow_t` (C++ *struct*), 39  
`lwow_t::arg` (C++ *member*), 39  
`lwow_t::discrepancy` (C++ *member*), 39  
`lwow_t::ll_drv` (C++ *member*), 39  
`lwow_t::mutex` (C++ *member*), 39  
`lwow_t::rom` (C++ *member*), 39  
`lwow_unprotect` (C++ *function*), 31  
`LWOW_UNUSED` (C *macro*), 29  
`lwow_write_byte` (C++ *function*), 37  
`lwow_write_byte_ex` (C++ *function*), 32  
`lwow_write_byte_ex_raw` (C++ *function*), 32  
`lwow_write_byte_raw` (C++ *function*), 37  
`lwowr_t` (C++ *enum*), 30  
`lwowr_t::lwowERR` (C++ *enumerator*), 30  
`lwowr_t::lwowERRBAUD` (C++ *enumerator*), 30  
`lwowr_t::lwowERRNODEV` (C++ *enumerator*), 30  
`lwowr_t::lwowERRPAR` (C++ *enumerator*), 30  
`lwowr_t::lwowERRPRESENCE` (C++ *enumerator*),  
30  
`lwowr_t::lwowERRTXRX` (C++ *enumerator*), 30  
`lwowr_t::lwowOK` (C++ *enumerator*), 30