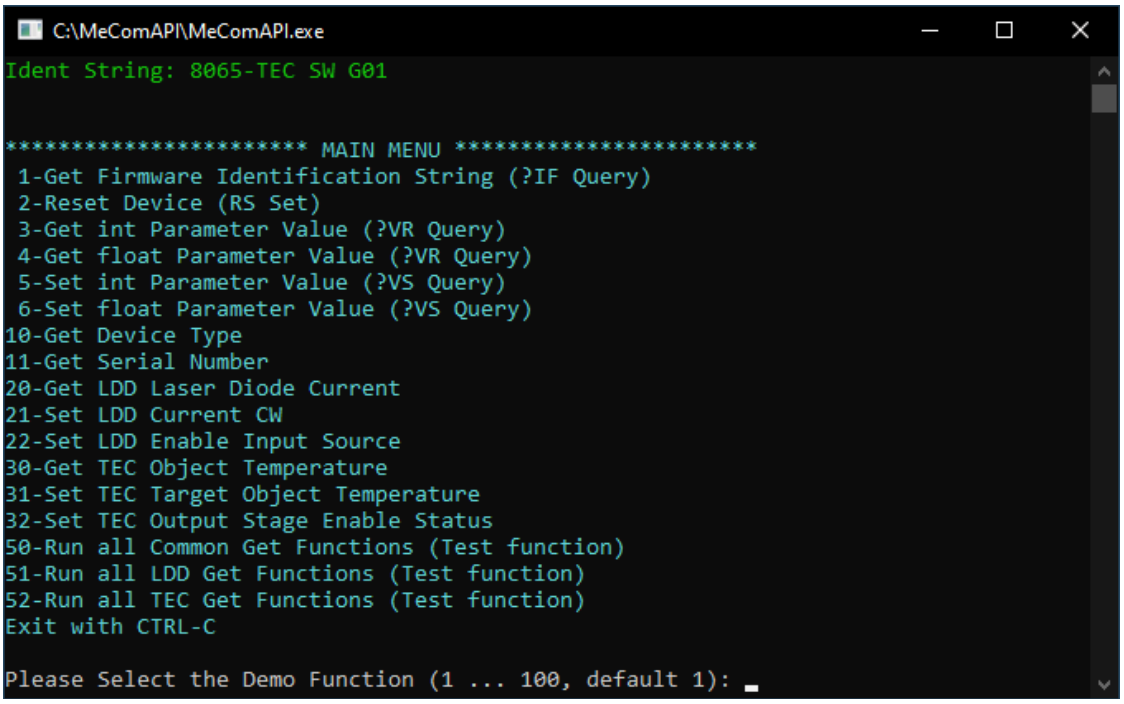
MeComAPI (C/C++)



Demo Application

Communication API for LDD / TEC Families and LTR Rack Enclosures

The package consists of a C-Code Library and a sample application.

The application exemplifies the control of LDD- and TEC-Family devices over a Serial COM Interface

---

meerstetter engineering

Member of Berndorf Group

Developed, assembled, and tested in Switzerland

Index

Meerstetter Engineering GmbH
Schulhausgasse 12
CH-3113 Rubigen
Switzerland

Phone: +41 31 529 21 00
Email: contact@meerstetter.ch

Document 5170M

Release date: 10 February 2025

1 General Description

1.1 General

- The MeComAPI provides C-code to fully control LDD / TEC - Family devices.
- The user will only need to call some simple functions to set or read parameters.
- The MeComAPI does everything that is necessary to have a reliable communication interface:
    - All parameters are predefined as macro functions
    - CRC calculations and checks
    - Sequence Number monitoring
    - Data resend on timeout and error management

1.2 Documents and Versions

This project shows the C-code implementation of the following specification documents:

- MeCom Protocol Specification 5117
- Implements most functions of the LDD and TEC Communication Specifications:
    - Laser Diode Driver Communication Protocol 5130
    - TEC Controller Communication Protocol 5136

1.3 Components

The project package consists of the following components:

- Ready to use .exe file with all available MeComAPI functions implemented in a simple Windows console Demo Application. This application has been compiled with "Microsoft Visual Studio 2022". It is necessary to have the "Microsoft .NET Framework 4.0" and "Microsoft Visual C++ 2010 x86 Redistributable Package" installed on the computer.
- Ready to use Linux Binary with all available MeComAPI functions implemented in a simple Linux Terminal Demo Application. This application has been compiled with "GCC 11.3.0". This binary was last built and tested on an "Ubuntu 22.04" operating system.
- Demo Application source code as Microsoft Visual Studio project and GNU makefile.
- The MeComAPI isolated from the Demo Application.

1.4 Compatibility

- MeComAPI
    - The MeComAPI is written in ANSI C99 standard code.
    - The API code is fully hardware independent and can be used to develop PC or microcontroller applications. Only the MePort.c file holds hardware-related interface functions that may need to be adjusted.
    - The API is compatible with most operating systems, but it is also usable without the use of an operating system, e.g., on an embedded device.
- Demo Application of the MeComAPI
    - The Demo Application code which is using MeComAPI functions is mainly written in C. A few functions in the Windows-specific implementation are written in C++.

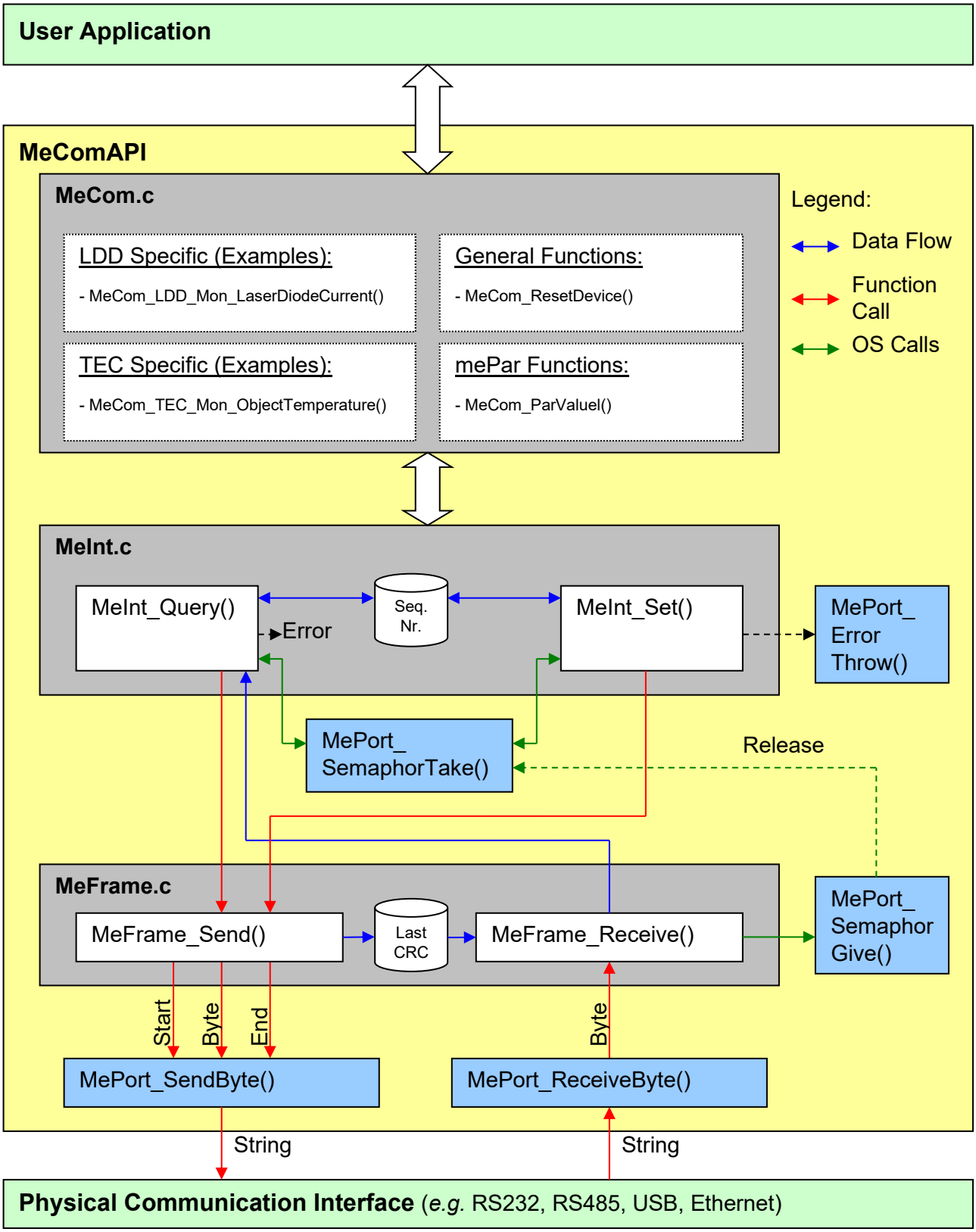
1.5 Logging

Whenever a command is sent or received by an application that is using the MeComAPI, the command string is written into a text file called "ComLog.txt" in the top-level directory of the project.

This log is mainly there for communication debugging purposes to see whether the sent commands are correct and whether a controller answers to sent commands or not. In case of no response this indicates either an issue with the sent command or a communication interruption to the controller.

The responsible functions for this logging functionality can be found in the "Source/Diagnostics/Log.c" source file. Since these functions are used in the "Source/ComPort/ComPort.c" and "Source/TcpSocket/TcpSocket.c" files the user should look into whether they want this functionality in their system or not. This is especially important when implementing the MeComAPI in the context of an embedded system.

2 Function Diagram of the MeComAPI



2.1 Diagram Description

2.1.1 MeComAPI Block Description

- MeCom.c
  - TOP Level. Contains functions to be called by the User Application.
  - The Payload of the Query and Set command is formed and passed to the lower-level function.
  - The returning Data is interpreted and given back to the User Application.
- MeInt.c
  - Connection-oriented Level. Its functions are being called by MeCom.c functions.
  - Adds a sequence number to the Payload and passes the data to MeFrame.c functions.
  - Calls the Semaphore function to wait for an answer.
  - Resends the Data up to 2 times if a timeout has occurred.
  - Checks the Sequence Number, Address and Type of a received Frame.
  - Tells the MeCom.c function if the received data is correct or not.
  - Throws an Error if 3 timeouts have been expired or a Server Error code has been received.
- MeFrame.c
  - MeFrame_Send() is being called by the MeInt.c Level. Adds the checksum and forms the frame of the given Data and passes every single byte to the Port Send function.
  - MeFrame_Receice() receives Bytes from the Port Receive function, checks the checksum and extracts the data from the received frame. Gives the Data up to the MeInt.c Level.

2.1.2 Interfaces to the User Application

The MeComAPI has basically only 2 points at which the user should interact. It should not be necessary to modify any functions of the "private" folder.

- The first type of interactions are function calls of TOP Level functions such as "MeCom_ResetDevice()".
    - All TOP Level functions are published in MeCom.h. This file contains macro functions for most of the parameters of the LDD / TEC.
    - These functions are blocking and do only return when the communication timeout has been expired (default after 3 trials of 100ms) or the correct answer has been received.
      Blocking: Check MePort_SemaphorTake() and MePort_SemaphorGive() descriptions.
    - They do return 1 on success and 0 if an error has occurred.
- The second interaction points are MePort functions (shown in blue). These are the low-level interface functions to the user system. All these functions are located in the MePort.c file.
    - MePort_SendByte()
        - Is being called for every single byte which should be sent to the Physical Communication Interface.
        - This function gets additional information that identifies first and last bytes of a frame. This can be useful to form a string or to enable or disable an RS485 TX interface.
        - *Demo Application: A string is formed and then passed to the Com Port send function.*
    - MePort_ReceiveByte()
        - Must be called if new data has been received on the Physical Communication Interface.
        - Usually, this function is being called from an interrupt service routine.
        - This function receives a string and passes every single byte to the Frame Level functions. The function prototype can be modified, to receive just one Byte.
        - *Demo Application: A string is being received from the Com Port.*
    - MePort_ErrorThrow()
        - Is being called if an error has occurred.
        - The user can add some error management code.
        - *Demo Application: Message print to the console.*
    - MePort_SemaphorTake()
        - This function is being called after the Query or Set string has been sent to the Physical Communication Interface. The timeout variable in milliseconds is given to this function.
        - The function does only return if the given timeout has been expired or the function is being released with the MePort_SemaphorGive() function.
        - The user should add its Operation System Semaphore functions for optimal performance.
        - *Windows Demo Application: Only the standard Windows Sleep(10) function is being called. And a module global variable is being polled to check if the MePort_SemaphorGive() function has been called.*

        - *Linux Demo Application: A Mutex and a Condition Variable is being used to check if the MePort_SemaphorGive() function has been called.*
    - MePort_SemaphorGive()
        - This Function is being called if a complete and correct Frame has been received from the Physical Communication Interface.
        - The user should add its Operation System Semaphore functions.
        - *Windows Demo Application: Only the module global Variable mentioned above is set to release the Take Function.*
        - *Linux Demo Application: See function above.*

3 Demo Application Compilation

3.1 Windows

On Windows the demo application can be compiled with .NET Framework v4.0 using the included Visual Studio solution ("8077-MeComAPI.sln").

Use the "Win32" platform configuration to compile the program. x64 is not configured and might not be compatible with certain libraries used in the project.

3.1.1 Requirements

- Visual Studio 2010 or higher (solution file is currently on version 2022)
- .NET Framework v4.0 or higher
- C++ x64/x86 BuildTools

3.2 UNIX-like (*nix)

On *nix operating systems the demo application can be compiled using the included "makefile".

Note: Compilation and functionality of the demo application using WSL (Windows Subsystem for Linux) was tested and should work without any issues. Bear in mind that to use serial ports on WSL, you will likely have to adjust the "`DEVICE`" definition in the "Source/ComPort/ComPort.h" file from "`/dev/ttyUSB`" to "`/dev/ttyS`".

3.2.1 Requirements

- GCC compiler (last tested with v11.3.0, should work down to at least v4.6.3)
- GNU Make (last tested with v4.3, but earlier versions should work as well)

3.2.2 Linux Example (tested on Ubuntu 22.04)

3.2.2.1 Install Prerequisites

```
> sudo apt-get install gcc
> sudo apt-get install make
```

3.2.2.2 Compile and run the Demo Application

```
> make
> ./MeComAPI
```

To clean up the files generated by "make" you can run the "`make clean`" command.

3.2.3 macOS Example (tested on Ventura 13.2)

3.2.3.1 Install Prerequisites

In this example Homebrew is used to install the prerequisites but other package managers should work fine as well.

```
> brew install gcc
> brew install make
```

3.2.3.2 Compile and run the Demo Application

```
> make
> ./MeComAPI
```

3.3 Arduino

To use the API on an Arduino device, import the "MeComAPI.zip" file in the Arduino IDE. Note that the regular demo application is not available on Arduino systems, use the minimal example located in "example.cpp" instead.

Be aware that certain Arduino models only have one USB host controller. In this case the Arduino cannot communicate via MeCom while connected to a computer, assuming both are using a USB or RS232 connection.

The MeComAPI on Arduino devices works mostly the same as on other systems, the most notable differences are that there is no error message output, and internally semaphores were replaced with synchronous code.

The zip archive contains unchanged MeComAPI code, however, the file and folder structure has been modified to be compatible with the Arduino IDE. The following steps can be used to produce the Arduino library package:

1. Create a folder called "MeComAPI".
2. Copy the "library.properties" file into the "MeComAPI" folder.
3. Create a "src" folder.
4. Copy the "MeComAPI.h" file into the "src" folder.
5. Copy the "Source/MeComAPI" folder into the "src" folder.
6. Copy the Source/ArduinoSerial folder into the "src" folder.
7. Change all ".c" file extensions to ".cpp".
8. Archive the "MeComAPI" folder into a ZIP file. Make sure you archive the folder itself, not just the files inside of it.

A Change History

Date of change	Doc/ Version	API Version	Changed/ Approved	Compatible with / Change Log
25 June 2013	A	0.10	ML	• Compatible with: ○ TEC STM32 Software Version: 1.50 / 1.51

27 June 2013	B	0.20	ML	• Compatible with:
				  ○ LDD STM32 Software Version: 1.50
				  ○ TEC STM32 Software Version: 1.50 / 1.51
				• Add: Com Ports 10-50

14 August 2013	C	0.21	ML	• Compatible with:
				  ○ LDD STM32 Software Version: 1.60
				  ○ TEC STM32 Software Version: 1.60
				• Bug: ComPort Log File Access violation problem solved

16 October 2013	D	0.22	ML	• Compatible with:
				  ○ LDD STM32 Software Version: 1.80
				  ○ TEC STM32 Software Version: 1.70

4 December 2013	E	0.30	TB	• Add: Linux Demo Application

11 February 2014	F	0.40	TB	• Add: Support for 64bit Operating Systems
				• Compatible with:
				  ○ LDD STM32 Software Version: 1.90
				  ○ TEC STM32 Software Version: 1.91

25 November 2014	G	0.41	ML	• Compatible with:
				  ○ LDD STM32 Software Version: 2.02
				  ○ TEC STM32 Software Version: 2.10

16 March 2015	H	0.42	ML	• Compatible with:
				  ○ LDD STM32 Software Version: 2.02
				  ○ TEC STM32 Software Version: 2.30

17 March 2023	I	0.50	XF	• Add: TCP Socket Client to communicate over Ethernet • Mod: Combined Windows- and Linux-specific source files to simplify the project structure • Add: Information regarding the included logging functionality in the documentation • Add: Demo application compilation instructions in the documentation
01 May 2024	J	0.60	SC / XF	• Add: Support for Arduino devices
03 June 2024	K	0.61	SC / XF	• Bug: Arduino ESP32 compile error fixed
24 September 2024	L	0.62	XF / ML	• Add: Trigger Save to Flash (SP) command. When a parameter is changed it is now only ever changed in the volatile memory of the device until a SP command is sent to it. Before this change parameter changes were periodically saved to the flash.
24 January 2025	M	0.63	SC / XF	• Bug: Arduino data receive function did not check for the correct End-of-Frame byte • Bug: Arduino data receive function timeout did not work as expected • Add: Support for compiling to newer Arduino boards