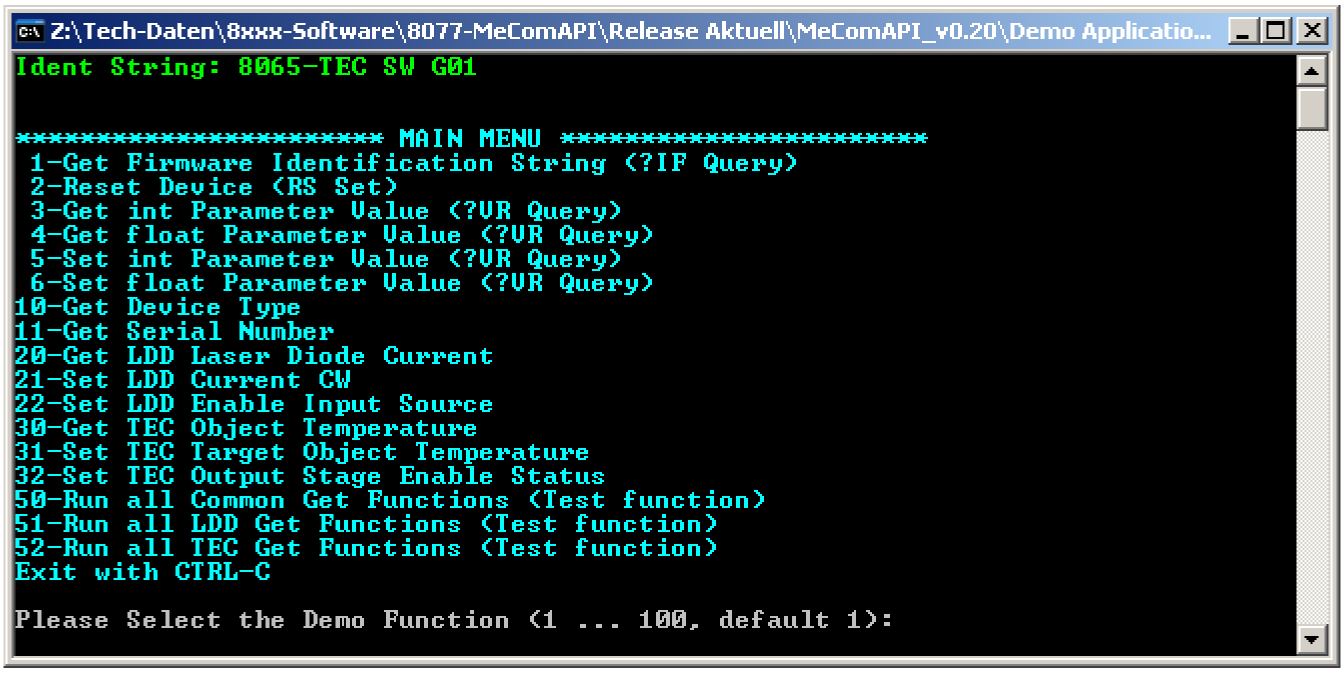
MeComAPI

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


Z:\Tech-Daten\8xxx-Software\8077-MeComAPI\Release Aktuell\MeComAPI_v0.20\Demo Applicatio...

```
Ident String: 8065-TEC SW G01


*********************** MAIN MENU ***********************
 1-Get Firmware Identification String (?IF Query)
 2-Reset Device (RS Set)
 3-Get int Parameter Value (?VR Query)
 4-Get float Parameter Value (?VR Query)
 5-Set int Parameter Value (?VR Query)
 6-Set float Parameter Value (?VR Query)
10-Get Device Type
11-Get Serial Number
20-Get LDD Laser Diode Current
21-Set LDD Current CW
22-Set LDD Enable Input Source
30-Get TEC Object Temperature
31-Set TEC Target Object Temperature
32-Set TEC Output Stage Enable Status
50-Run all Common Get Functions (Test function)
51-Run all LDD Get Functions (Test function)
52-Run all TEC Get Functions (Test function)
Exit with CTRL-C

Please Select the Demo Function (1 ... 100, default 1):
```

Demo Application

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

1.3 Components

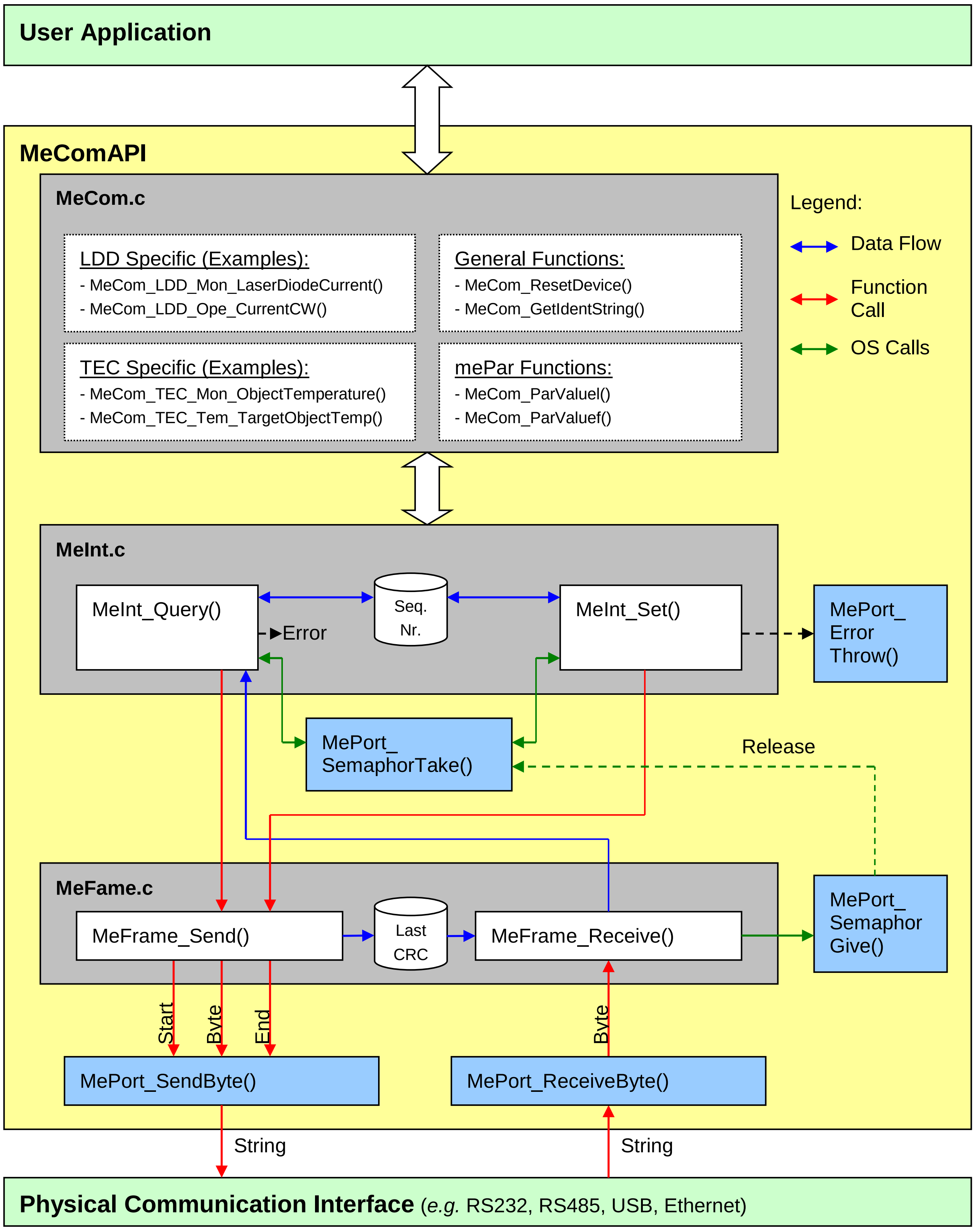
The project package consists of the following components:

- 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 2010”. 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 4.6.3”. This binary was built and tested on a “Ubuntu 12.04”.
- 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_xxx.c file holds hardware-related interface functions that may need to be adjusted.
    - The API is compatible with operating systems. There is a function which is being called while the API is waiting for a communication replay. Another function is being called if new data has been received. The API can also be used without operating system.
- Demo Application of the MeComAPI
    - The Demo Application code which is using MeComAPI functions is mainly written in C. Some other functions are written in C++.

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. Is 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 an 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 ms 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 above mentioned module global Variable is set to release the Take Function.*
        - *Linux Demo Application: See function above.*

3 Change Log

Changed	Dok	API Version	Compatible with / Change Log
25 June 2013	A	0.10	• Compatible with ○ TEC STM32 Software Version: 1.50 / 1.51
27 June 2013	B	0.20	• 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	• 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	• Compatible with ○ LDD STM32 Software Version: 1.80 ○ TEC STM32 Software Version: 1.70
4. Dez. 2013	E	0.30	• Add: Linux Demo Application
11. Feb. 2014	F	0.40	• Add: Support for 64bit Operating Systems • Compatible with ○ LDD STM32 Software Version: 1.90 ○ TEC STM32 Software Version: 1.91
25 Nov 2014	G	0.41	• Compatible with ○ LDD STM32 Software Version: 2.02 ○ TEC STM32 Software Version: 2.10
16 March 2015	H	0.42	• Compatible with ○ LDD STM32 Software Version: 2.02 ○ TEC STM32 Software Version: 2.30