UD000 User Reference for the Library com_lib

Scope

This document provides only the overview of the general design, functionality and components of the library. For the more detailed reference information, please, consult the corresponding documents:

- Module *mock_serial* - UD001 - only for developers, implements a mock serial connection to a mock device
- Module *serial_port_com* - UD002
- Module *serialization* - UD003

Design and Functionality

This library implements *atomic* sending and receiving of data over serial port connection using zero-terminated packages - the COBS-encoded bytestrings with the added b'\x00' byte (module *serial_port_com*). The supported modes of operation are:

- Asynchronous communication
    - Uni-directional
        - Only sending with the responses being ignored / not claimed, even if any is issued
        - Only listening to an external data provider
    - Bi-directional - sending packages and checking for the incoming data / acquiring the received data in an arbitrary order
- Synchronous communication - bi-directional by definition, when for each sending the response is waited for and reclaimed - the other side MUST send a response to each received data package
- Mixing the synchronous and asynchronous sending and receiving - the other side MUST send a response to each received data package

Note, that in the mixed mode some responses to the asynchronous sendings may be lost depending on the sequence of the send / received requests made, whereas for each synchronous sending the retrieval of the response is guaranteed.

The synchronous send and receive can be performed in the blocking mode or in the timeouted mode. In the blocking mode the control is not returned to the caller until the respose is received, however long it may take. In the timeouted mode the connection is automatically closed and an exception is raised.

Different data types are acceptable as the input data to be send, and the received bytestring data can be converted into that same data types:

- Bytestring (Python type **bytes**)
- Byte arrays (Python type **bytearray**)
- Unicode strings (Python type **str**) - the 'utf-8' codec is used for the **str** <-> **bytes** conversion
- Auto-serializable data types (user defined classes), which must have
    - instance method *packBytes*(), which is called without an argument and returns a bytestring
    - class method *unpackBytes*() accepting a single bytestring argument and returning any value

The module *serialization* provides template classes to construct such auto-serializable classes for:

* C *struct* -like structured data storage (fields access by name)
* Fixed length arrays - sequences of the same type elements accessible by an index - with the same length for all instances of such class
* Dynamic length arrays - similar, but each instance may have a different length, which is defined during the instantation, and cannot be changed afterwards

See DE000, UD002 and UD003 documents for further and more detailed information.