Requirements for the Module com_lib.serial_port_com

Conventions

Requirements listed in this document are constructed according to the following structure:

Requirement ID: REQ-UVW-XYZ

Title: Title / name of the requirement

Description: Descriprion / definition of the requirement

Verification Method: I / A / T / D

The requirement ID starts with the fixed prefix 'REQ'. The prefix is followed by 3 letters abbreviation (in here 'UVW'), which defines the requiement type - e.g. 'FUN' for a functional and capability requirement, 'AWM' for an alarm, warnings and operator messages, etc. The last part of the ID is a 3-digits *hexadecimal* number (0..9|A..F), with the first digit identifing the module, the second digit identifing a class / function, and the last digit - the requirement ordering number for this object. E.g. 'REQ-FUN-112'. Each requirement type has its own counter, thus 'REQ-FUN-112' and 'REQ-AWN-112' requirements are different entities, but they refer to the same object (class or function) within the same module.

The verification method for a requirement is given by a single letter according to the table below:

Term	Definition
Inspection (I)	Control or visual verification
Analysis (A)	Verification based upon analytical evidences
Test (T)	Verification of quantitative characteristics with quantitative measurement
Demonstration (D)	Verification of operational characteristics without quantitative measurement

Functional and capability requirements

Requirement ID: REQ-FUN-200

Title: Module's functionality

Description: The module should implement funtions or classes providing the following functionality:

- Obtain a list of all serial ports, to which the actual USB devices are connected at the moment, if 2-way communication is possible with them
- Send and receive data in the asynchronous manner
- Send and receive data in the synchronous manner

NB the two modes: synchronous and asynchronous - can be mixed only if a device connected to the port always sends back response or confirmation of receipt.

Verification Method: A

Requirement ID: REQ-FUN-201

Title: Underlying dependencies and API

Description: The module should wrap the functionality of the **PySerial** library, but any other API-compatible library should be acceptable as a replacement.

Verification Method: A

Requirement ID: REQ-FUN-210

Title: Active (USB) serial ports discovery

Description: The module must provide a function to list the names (paths) of all serial ports, which can be used to communicate with the real USB devices, i.e. those ports, which have not *None* VID and PID. It should return a list of 3-elements tuples as ({port name}, {PID}, {VID})

Verification Method: T

Requirement ID: REQ-FUN-220

Title: Communication wrapper class

Description: The module should implement a class, which wraps the class **Serial** from the **pySerial** library and is capable of both the synchronous (sending + receiving in a single call) and asynchronous serial port communication. However, it should be possible for its sub-classes to use a different API-compatible class, e.g. a 'mock serial port' implementation.

Verification Method: T

Requirement ID: REQ-FUN-221

Title: (Re-) Openning the connection

Description: The connection should be opened automatically upon instantiation of the class. However, if the connection has been closed for any reason, implicitely due to timeout or data sanity checks related exceptions or explicitely by a request of the user / client, it should be possible to re-open the connection, e.g. method *open*()

Verification Method: T

Requirement ID: REQ-FUN-222

Title: Closing the connection

Description: The client of the class should be able to close the connection by explicit request, e.g. method *close*(). Such call should clear all cached data and ensure closing of the connection (via underlying object's

API), but preserve the connection settings, such as baudrate, port name, etc., so the connection can be re-established upon request. The connection should be closed implicitely before raising any exception, or if the instance of the class is to be deleted (garbage collected, etc.)

Verification Method: T

Requirement ID: REQ-FUN-223

Title: Querring the connection status and settings

Description: The class must provide property *IsOpen* to query the connection status. It must return **True** if the connection is open, and **False** otherwise. The class must also provide property *Settings* to query the connection settings, which, at least, contains keys *port*, *timeout* and *write_timeout*, and all other keyword arguments passed into the initialization method. However, even if provided as the keyword arguments the *timeout* and *write_timeout* value should be always overwriten as 0, and *port* - by the value supplied as the mandatory positional argument.

Verification Method: T

Requirement ID: REQ-FUN-224

Title: Sending data / command asynchronously

Description: The class must provide method *send*() to send data to a port. It must not await the response or even the completion of the sending, and it must return the control immediately. However, the instance of the class should keep track of the number of the calls of this method made (since the last opening of the port) and return the call's number as an integer identificator of the sent data package.

Verification Method: T

Requirement ID: REQ-FUN-225

Title: Receiving response asynchronously

Description: The class must provide method *getResponse*() to receive a response to the previously send data / command. Upon each call the incoming buffer of the serial communication underlying object must be checked, and all available bytes must be pulled from it. The pulled data should be split into packages using b'\x00' separators. All complete packages (b'\x00' separator terminated) of non-zero length should be stripted of the separator and placed into a waiting queue-like object for the further processing in the exact same order, as received. If that queue is not empty, the first waiting package should be returned, otherwise (empty queue) the **None** value should be returned. The method should also track the number of the complete packages received (since the last opening of the port) as the integer identifier of the received response, which should be returned together with the data package itself. An incomplete package should be cached until the next call.

Verification Method: T

Requirement ID: REQ-FUN-226

Title: Sending data / command synchronously

Description: The class should provide the synchronous communication method *sendSync*(). It must not only send data into a port, but also wait for the response from a device connected to it. Considering the possibility of mixing the synchronous and asynchronous data sending, the response identifier should match the send package identifier. The unclaimed responses to the previous sendings waiting in the incoming queue should be discarded. The received response data and the sending / receipt identificator should be returned.

By default the method should wait indefinetely until the response to the last sent data is received or the port is disconnected / closed. However, a positive timeout period can also be optionally passed; and an exception should be raised if the response is not received during that period.

Verification Method: T

Requirement ID: REQ-FUN-227

Title: Sending / receiving package structure

Description: The data should be written into and read-out from the port in the form of byte-string (Python type **bytes**) using b'\x00' (zero) as the package delimiter / separator. The package separator should be added before the sending and removed from the received response automatically.

Verification Method: A

Requirement ID: REQ-FUN-228

Title: Supported input / output data types

Description: The methods *send*() and *sendSync*() should accept any following data types as the input data for sending:

- (Unicode) strings (Python type **str**) - will be encoded using UTF-8 codec
- Byte-stings (type **bytes**)
- Byte arrays (type **bytearray**) - converted explicitely into byte-string before sending
- Any class instance providing method *packBytes*() - this method will be used to convert Data into bytes

By default, the methods *getResponse*() and *sendSync*() should return the received data as a byte-string, unless a different data type is explicitely requested via an optional argument, which can be:

- (Unicode) string - UTF-8 codec is applied for decoding
- Byte arrays (type **bytearray**)
- Any class instance providing class method *unpackBytes*() - the bytestring will be passed into that method to instantiate the required data type

COBS encoding / decoding must be applied to the bytestring before sending / upon receipt in order to allow proper use of b'\x00' package terminator (REQ-FUN-227).

Verification Method: T

Alarms, warnings and operator messages

Requirement ID: REQ-AWM-220

Title: Suppression of unnecessary **serial.SerialException**

Description: This exception should not be raised in the following situations:

- Attempt to (re-) open the already open port - skip requirested action silently
- Attempt to (re-) close the already closed / not yet open port - skip requirested action silently
- Attempt to read from or send into a closed port with the assigned and valid port name - silently open the port before performing the action

Verification Method: T

Requirement ID: REQ-AWM-221

Title: Situations to raise **serial.SerialException**

Description: This exception should be raised when a port referenced by the assigned name cannot be opened, which can result from:

- Explicit request from the user, i.e.:
  - Call of the method *open*() to re-open the previosuly closed (i.e. timeout, disconnect, etc.) port if the device has been disconnected
  - Port referenced by name as the argument during the class' instantiation cannot be opened
- Implicit requests from the user, i.e. an attempt to read from or write into a port, which couldn't be opened upon instantiation or was disconnected later

Verification Method: T

Requirement ID: REQ-AWM-222

Title: Treatment of a timeout situation

Description: If the response is not received during the specified time period the **serial.SerialTimeoutException** should be raised, whilst the connection should be closed and the cached data discarded.

Verification Method: T

Requirement ID: REQ-AWM-223

Title: Improper argument data type

Description: TypeError or its sub-class exception should be raised if any argument of any method is of the unaccepatable data type, including the keyword arguments defining the connection settings

Verification Method: T

Requirement ID: REQ-AWM-224

Title: Improper argument value

Description: ValueError or its sub-class exception should be raised when any of the connection settings is of the proper data type, but of an acceptable value

Verification Method: T