Requirements for the Module com_lib.mock_serial

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-100

Title: Implements virtual serial port to a virtual device

Description: The module should implement a virtual device and a virtual serial port able to connect to and communicate with that virtual device.

Verification Method: A

Requirement ID: REQ-FUN-110

Title: Virtual device emulation

Description: A function emulating a device, normally, should be run in a separate thread and share two queue-like objects with the virtual serial port emulator as well as a signaling event object. As long as the signal to stop event is not set it should scan the input buffer and read one byte at a time, if input is available. The delay between the bytes read-out should be defined by the baudrate calling argument. As soon as b'\x00' (ASCII code *NUL*) is received all accumulated bytes, including the zero should be placed into the output buffer one character at the time with a delay between each byte defined by the same baudrate argument; this process should block reading loop. However, if the received command is 'quit' (i.e. b'quit\x00' is accumulated), the function should set the signaling stop event and terminate without sending it back.

Verification Method: T

Requirement ID: REQ-FUN-120

Title: Virtual serial port API

Description: The object implementing the virtual serial port must provide the minimal set of the API compatible with the PySerial library, i.e. the methods *open*(), *close*(), *write*() and *read*() as well as the properties *port*, *in_waiting*, *out_waiting*, *timeout*, *write_timeout*, *baudrate* and *is_open* with the same functionality as in the mentioned library.

Verification Method: T

Requirement ID: REQ-FUN-121

Title: Instantiation of the mock serial object

Description: The class can be instantiated without arguments, in which case the default settings are applied. But the initializer must accept an arbitrary number of keyword arguments of the arbitrary names. The following keywords must be recognized: *port*, *timeout*, *write_timeout* and *baudrate* - and the respective settings must be changed according to the passed values. If the *port* is passed and its value is acceptable - the connection must be opened.

Verification Method: T

Requirement ID: REQ-FUN-122

Title: Minimal API - settings

Description: The setter / getter properties *port*, *timeout*, *write_timeout* and *baudrate* must change the respective settings of the connection in the same way as of the class **serial.Serial** / return the current values. However, assigning the proper 'mock' value to the *port* property should open the connection only if it is not yet opened; if it is already open - no action should be taken.

Verification Method: T

Requirement ID: REQ-FUN-123

Title: Minimal API - status queries

Description: The getter only properties *is_open*, *in_waiting* and *out_waiting* should tell if the connection is open, and how many bytes are currently in the incoming and outgoing buffer respectively

Verification Method: T

Requirement ID: REQ-FUN-124

Title: Data sending behaviour

Description: The method *write*() should always place all bytes of the passed bytestring into the outgoing buffer, but the further behaviour is defined by the property *write_timeout* as:

- *write_timeout* = **None**; blocking call, waits indefinetely until the outgoing buffer is emptied
- *write_timeout* = 0; non-blocking call returns immediately
- *write_timeout* > 0; waits until the outgoing buffer is emptied, but no longer than *write_timeout* - if the timeout is reached, raises **serial.SerialTimeoutException**

Unless an exception is raised, it should return the length of the passed bytestring.

Verification Method: T

Requirement ID: REQ-FUN-125

Title: Data pulling behaviour

Description: The result of the *read*() method call is defined by the amount of bytes available in the incoming buffer, value of the property *timeout* and the optional argument *size* (defaults to 1) as:

- *timeout* = **None**; blocking call, the buffer is pulled indefinitely until exactly *size* bytes are acquired
- *timeout* = 0; non-blocking call - exits almost immediately
  - if there are more than or equal to *size* bytes in the buffer, exactly *size* bytes are pulled and returned
  - otherwise all available (< *size*) are returned
- *timeout* > 0; tries to pull exactly *size* bytes from the incoming buffer and return them, but if less bytes are obtained during the *timeout* period, only the already pulled bytes are returned

Verification Method: T

Requirement ID: REQ-FUN-126

Title: Re-opening of the closed connection

Description: The user should be able to re-open the closed connection, which was implicitely closed due to an exception raised or explicitely by the user's request.

Verification Method: T

Alarms, warnings and operator messages

Requirement ID: REQ-AWM-120

Title: Improper date type for the connection settings

Description: Unacceptable data type for a connection setting passed into the initializer or assigned to the respective attribute (e.g. property) of the mock serial connection object must result in a sub-class of **TypeError**.

Verification Method: T

Requirement ID: REQ-AWM-121

Title: Improper value for the connection settings

Description: Unacceptable value of the proper data type for a connection setting passed into the initializer or assigned to the respective attribute (e.g. property) of the mock serial connection object must result in a sub-class of **ValueError** for all settings except *port*, for which **serial.SerialException** is raised.

Verification Method: T

Requirement ID: REQ-AWM-122

Title: A proper port value must be assinged before opening the connection

Description: The **serial.SerialException** is raised if the connection is opened before the valid port value is assigned

Verification Method: T

Requirement ID: REQ-AWM-123

Title: (Re-) opening of the already opened connection raises an exception

Description: An attempt to open the already opened connection must raise **serial.SerialException**.

Verification Method: T

Requirement ID: REQ-AWM-124

Title: Closing of the closed connection raises an exception

Description: An attempt to close the already closed connection must raise **serial.SerialException**.

Verification Method: T

Requirement ID: REQ-AWM-125

Title: Communication with a not opened connection raises an exception

Description: An attempt to write into, read from or query the size of the incoming or outgoing buffer of a closed / not opened connection must raise **serial.SerialException**.

Verification Method: T

Requirement ID: REQ-AWM-126

Title: Improper date type for the read() / write() methods

Description: Unacceptable data type passed into these methods must result in a sub-class of **TypeError**.

Verification Method: T

Requirement ID: REQ-AWM-127

Title: Not-positive number of bytes to read

Description: Non-positive integer passed into the read() method must result in a sub-class of **ValueError**.

Verification Method: T

Requirement ID: REQ-AWM-128

Title: Write timeout

Description: **serial.SerialTimeoutException** must be thrown upon reaching the timeout for the data sending.

Verification Method: T