Test Report on the Library com_lib

Conventions

Each test is defined following the same format. Each test receives a unique test identifier and a reference to the ID(s) of the requirements it covers (if applicable). The goal of the test is described to clarify what is to be tested. The test steps are described in brief but clear instructions. For each test it is defined what the expected results are for the test to pass. Finally, the test result is given, this can be only pass or fail.

The test format is as follows:

Test Identifier: TEST-[I/A/D/T]-XYZ

Requirement ID(s): REQ-uvw-xyz

Verification method: I/A/D/T

Test goal: Description of what is to be tested

Expected result: What test result is expected for the test to pass

Test steps: Step by step instructions on how to perform the test

Test result: PASS/FAIL

The test ID starts with the fixed prefix 'TEST'. The prefix is followed by a single letter, which defines the test type / verification method. 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 test ordering number for this object. E.g. 'TEST-T-112'. Each test type has its own counter, thus 'TEST-T-112' and 'TEST-A-112' tests 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

Tests definition (Inspection)

Test Identifier: TEST-I-000

Requirement ID(s): REQ-INT-000

Verification method: I

Test goal: Reliable dependencies.

Expected result: If any 3rd party library is used, it should be widely accepted, well maintained and not marked for deprecation in the nearest future (2 years).

Test steps: Check / review the source code.

Findings:

- The only direct external dependence is pySerial, which is actively maintained
- The other dependencies are *introspection_lib* and *codecs_lib*, which are developed, maintained and tested by the developer of this same library

Test result: PASS

Test Identifier: TEST-I-001

Requirement ID(s): REQ-UDR-000

Verification method: I

Test goal: Library's documentation.

Expected result: The requirements documentation, test reports and the user / programmer reference documentation is present. Reference documents provide 'use manuals' sufficient to work with the libary without software development experience (via frontend), and the implementation is also documented sufficiently (API reference) to ensure the maintenance and future modification of the library.

Test steps: Check / review the documentation provided with the library.

Test result: PASS

Tests definition (Analysis)

Test Identifier: TEST-A-000

Requirement ID(s): REQ-FUN-000, REQ-FUN-001

Verification method: A

Test goal: The library implements synchronous and asynchronous (virtual) serial port communication abstraction layer API.

Expected result: The funtionality is implemented and properly tested with respect to the requirements defined in the RE002 document.

Test steps: Check / review the source code. Run the unit-test suits defined in the ut002_serial_port_com.py module. Report results in the TE002 document.

Test result: PASS

Test Identifier: TEST-A-001

Requirement ID(s): REQ-FUN-002

Verification method: A

Test goal: Check that the automated serialization of the structured and indexed container data types is implemented.

Expected result: The library defines classes emulating C-like data types:

- Structures (records) with the fixed number and order of the elements (attributes) of the fixed types for each instance of the same class
- Fixed and dynamic length arrays - sequences of the same type elements

Structures may have either the fixed byte length numbers or strings as their attributes, which can be (de-) serialized (from) into machine byte-format, or another serializable structures or arrays. Arrays are either fixed or dynamic length sequences of the fixed size, serializable elements of the same data type - either numbers / strings, or another arrays or structures. Dynamic arrays can only be either top level objects, or attributes of the top level or nested structures in the final position considering the depth-first packing order.

These classes provide methods to be packed and unpacked, which can be used by the serial communication wrapper.

Test steps: Check / review the source code. Run the unit-test suits defined in the ut003_serialization.py module. Report results in the TE003 document.

Test result: PASS

Tests definition (Demonstration)

Test Identifier: TEST-D-000

Requirement ID(s): REQ-IAR-000, REQ-IAR-002

Verification method: D

Test goal: System requirements check.

Expected result: The dependencies check script (module) is present and it detects the missing dependencies, too old versions as well as improper or too old Python interpreter version. If this script does not detect problems, the library should operate as designed.

Test steps: Install the library on different PCs using different OS. Run the check module check_dependencies.py. If missing dependencies or improper Python version is detected take the corrective actions. When the dependencies check has passed - try to work with the library, e.g. retrieve some data from the DPG. See also tested_OS.

Test result: PASS

Test Identifier: TEST-D-001

Requirement ID(s): REQ-IAR-001

Verification method: D

Test goal: Multi-platform functionality.

Expected result: The library works as designed under MS Windows and GNU Linux OS, at least.

Test steps: Install the library (via *pip*) and try to work with it on different PCs using different OS (run unit-tests). Mark the tested OS + Python version in the tested_OS.md document as well as the result of the test.

Test result: PASS

Traceability

For traceability the relation between tests and requirements is summarized in the table below:

Requirement ID	Covered in test(s)	Verified [YES/NO])
REQ-FUN-000	TEST-A-000	YES
REQ-FUN-001	TEST-A-000	YES
REQ-FUN-002	TEST-A-001	YES
REQ-INT-000	TEST-I-000	YES
REQ-IAR-000	TEST-D-000	YES
REQ-IAR-001	TEST-D-001	YES
REQ-IAR-002	TEST-D-000	YES
REQ-UDR-000	TEST-I-001	YES

Software ready for production [YES/NO]	Rationale
YES	All tests are passed