Use Case Study: Automated Testing of a CANopen NMT server device

Carina Heinrich (Friedrich Lütze)

CANopen is used in an ever-growing range of application fields such as railway applications, building automation, medical equipment and more. With the expansion to new application areas, the complexity of the CANopen devices increases as well. At the same time, the devices must adhere to a greater number of standards and requirements. As the complexity and variety of requirements increases, so does the need for a suitable method to ensure that the devices fulfill the chosen requirements. One way to meet this demand is thorough testing which includes hardware-in-the-loop tests. A key feature of such tests is that they are automated in order to provide reproducibility, consistent report generation and fast execution without human involvement. Moreover, fully automated tests can be used as regression tests throughout the entire development cycle. In this paper we present a use case study on how to perform automated hardware-in-the-loop tests for a CANopen NMT server device.

1. Introduction

Digitization continues to accelerate and systems grow progressively complex. As a result the functionality of used devices and their conformance to various standards needs to be ensured. To achieve this, thorough tests are necessary to comply to customer requirements, conformity to previous products or protocol, conformance to various standards and more.

There are different approaches to ensure the functionality of a device. One possible approach is to perform manual tests. Manual testing is usually quickly setup but has other disadvantages. Manual tests are often time-consuming to maintain, adapt and port to other scenarios. Manual testing is also more error-prone due to human involvement. In order to achieve reproducibility of manual tests, the test cases must be thoroughly specified and the human tester must be trained for each test set to perform them precisely. As the reports of the tests must be also manually generated, the report generation requires time and precision of the tester. As systems grow more complex, manual testing becomes hard to accomplish.

Another way to go is using automated tests that run without human involvement. Automated tests have the advantage of reproducibility, portability and adaptability [1]. Reports can be precisely generated automatically. The tests are less time-consuming as there are no humans involved. The tester has to be trained for the test environment but not for each test set. The downside to automated tests are that they are costly to setup at the beginning and there is the one-time cost of training the tester for the test setup. Moreover, not everything can be fully automated. A trade-off for such scenarios is semi-automated tests where only little human involvement is required.

One specific sub-set of tests is the conformance test. Conformance tests determine if a device adheres to a specified standard and they can either be performed manually or automated [2]. The CANopen conformance tests ensures basic CANopen functionality. This is accomplished by testing the communication processes according to the device description file, also called electronic data sheet (EDS) [3]. The CANopen conformance test checks the process data object (PDO) and service data object (SDO) communication, the network management functions and more. A conformance test is not designed to test actual device functions or issues induced during the integration of the functionality into the device firmware [3]. Therefore, the CANopen conformance test can be supplemented with additional tests covering manufacturer specific functions, deviations from the standard for legacy products and functionalities which are not part of the CANopen conformance test tool (CTT) (e.g. CANopen profiles).

Hardware-in-the-loop (HIL) tests are also a subcategory of automated tests. The scope of HIL entails the development and testing of complex real-time embedded systems. HIL tests close the gap between mere simulation and tests under real conditions [4]. Only software-based tests cannot always precisely replicate real operation conditions. HIL tests extend the computer simulation with real hardware replacing the mere emulation of the hardware under test [5]. The hardware under test is integrated in a simulated environment interacting with the simulation via sensors and actuators.

Choosing the right test approach and developing a suitable test concept involves considering different criteria. Criteria range from the effectiveness to finding defects, over implementing test cases that cover multiple faults to economical attributes such as the cost of test execution and the maintenance effort [1]. The main objective of each test concept must be determined. The attributes can then be weighted against each other, with the aim of finding the best solution for the desired and required objective.

This paper presents a use case study for implementing automated tests for a CANopen NMT service device. The first part of this paper outlines the starting point for the automated tests. The next section describes the test setup and used test concept. At the end, an overview of the implemented test cases is given and a conclusion is drawn.

2. Starting point

2.1. Device under test (DUT)

The starting point for developing automated tests is the request to redesign a LÜTZE TRANSPORTATION CANopen device called the DIOLINE CANopen bus coupler (shown in Figure 1) [6, 7]. The DIOLINE CANopen bus coupler is a CANopen compact station with local- bus expansion for use on rail vehicles. The main task of the DIOLINE CANopen bus coupler is to map CANopen to the local-bus and vice versa. The bus coupler was developed in the year 2000 and comes with a delivery guarantee of 30 years reaching to the year of 2030. The implementation is mainly based on the specifications CiA 301 (draft standard version 4.0.1) and CiA 401 (draft standard version V2.0).

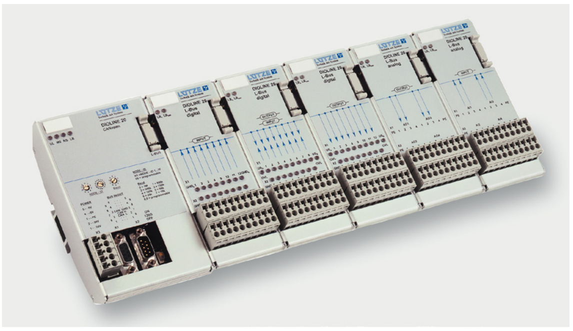


Figure 1 DUT: DIOLINE CANopen bus coupler by LÜTZE TRANSPORTATION (displayed to the left with DIOLINE I/O modules to the right).

The redesigned product is required to be form, fit and function compatible with the old product which means among other things that the external behavior of both products must be the same. The DIOLINE CANopen bus coupler has two external interfaces: the CANopen protocol and the L-Bus 1 protocol. The L-Bus 1 protocol is a Lütze proprietary bus protocol used in the DIOLINE 20 I/O system for rail vehicles. An L-Bus 1 conformance test is performed to ensure that the product adheres to the specification. Besides, the CANopen communication must be tested to make sure that the product behaves as required.

2.2. Used approach

Automated tests in the form of the CTT by CiA already exist, which are used to ensure with a reasonable degree of confidence the conformance to the CANopen standard [2, 8]. Since the CANopen conformance test is not designed to test actual device functions or the integration of the CANopen functionality into the device firmware, additional tests are performed to supplement the CANopen conformance test [3].

The objectives of the additional implemented automated tests are:

- providing a supplement to existing CANopen and L-Bus 1 conformance tests to test additional features.
- supplying tests for legacy systems with deviations to the current standard.
- ensuring correct functionality after modification during development (regression test).

The main focus of the implemented tests is guaranteeing the same CANopen behavior to the exterior. Part of the L-Bus communication is implicitly tested as well. Automated tests are chosen where possible for the purpose of benefiting from all the mentioned advantages. Some minor parts, including hardware manipulation such as rotary switches, are hard to automate and are, therefore, semi-automated with minimal user involvement.

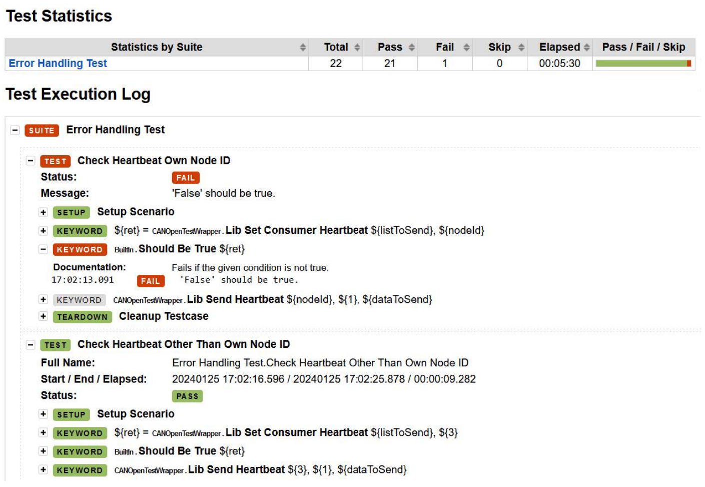
3. Test setup

For the sake of performing automated tests for the mentioned DUT we need a testing framework and a system to interact with the DUT.

3.1. Testing framework

As a testing framework we use the Robot Framework [9]. The Robot Framework is a generic open source automation tool funded by the non-profit Robot Framework Foundation [10] and released under Apache 2.0 license. The Robot Framework is known for being flexible, easy to use and its extensible nature [11]. It offers a good report generation, allows libraries from various languages to be imported and has a user-friendly syntax. Figure 2 shows an excerpt from the test report, generated by the Robot Framework with one failing test.

In our test setup most functionality is implemented in custom Python libraries and linked to the test cases of the Robot Framework. Certainly, any other testing framework can be used as well.



Test Statistics

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Error Handling Test	22	21	1	0	00:05:30	🟩🟥

Test Execution Log

```
− SUITE Error Handling Test
  − TEST Check Heartbeat Own Node ID
    Status:                 FAIL
    Message:                'False' should be true.
    + SETUP Setup Scenario
    + KEYWORD ${ret} = CANOpenTestWrapper.Lib Set Consumer Heartbeat ${listToSend}, ${nodeId}
    − KEYWORD BuiltIn.Should Be True ${ret}
      Documentation:        Fails if the given condition is not true.
      17:02:13.091    FAIL   'False' should be true.
    + KEYWORD CANOpenTestWrapper.Lib Send Heartbeat ${nodeId}, ${1}, ${dataToSend}
    + TEARDOWN Cleanup Testcase
  − TEST Check Heartbeat Other Than Own Node ID
    Full Name:              Error Handling Test.Check Heartbeat Other Than Own Node ID
    Start / End / Elapsed:  20240125 17:02:16.596 / 20240125 17:02:25.878 / 00:00:09.282
    Status:                 PASS
    + SETUP Setup Scenario
    + KEYWORD ${ret} = CANOpenTestWrapper.Lib Set Consumer Heartbeat ${listToSend}, ${3}
    + KEYWORD BuiltIn.Should Be True ${ret}
    + KEYWORD CANOpenTestWrapper.Lib Send Heartbeat ${3}, ${1}, ${dataToSend}
```

Figure 2: Test report generated by Robot Framework with one failing test.

3.2. Interaction framework

The first part of the chosen interaction framework is the CAN manager. We use the PCAN-USB Pro FD by PEAK and its corresponding PCAN-Basic API [12]. The PCAN-Basic API is provided by PEAK in different languages including Python. Any other CAN manager with suitable API can be used as well.

There are different categories of functionality to be tested. The first category does not require any additional interaction framework beside the CAN manager. These tests cover manufacture specific functions such as writing to the EEPROM via SDO and storing and loading manufacturer specific objects such as node ID and baud rate.

The second category of functionalities to be tested requires an additional interaction framework. For the purpose of testing CANopen functionalities such as different transmission types for PDO, inhibit, event time or error handling, a way to stimulate and monitor the DUT is required. The DIOLINE CANopen bus coupler can be stimulated and monitored using its I/O modules which are connected via the L-Bus 1 bus.

In order to distinguish possible errors, we add another I/O system to read and write the I/O modules connected to the CANopen bus coupler.

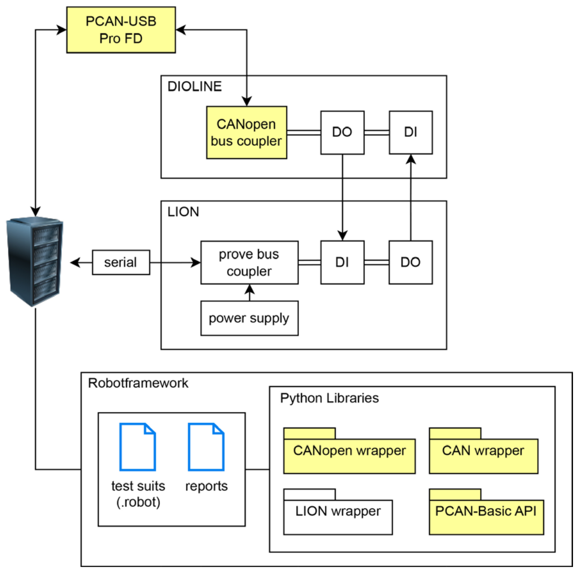


Figure 3: Test concept of the automated tests for the DIOLINE CANopen bus coupler using the Robot Framework and the LION I/O system.

We integrate the LÜTZE Input Output Network (LION) for this purpose [13, 14]. The LION system has been designed for use on rail vehicles in applications up to SIL2. In our test setup, the LION I/O modules are accessed using a special LION prove bus coupler. The special LION prove bus coupler provides a serial interface with commands to read from and write to the LION I/O modules and reset their local communication bus.

As a shell implementation to interact with the serial interface of the LION prove bus coupler, we make use of the shellmatta and an already available wrapper around the shellmatta with its transport layer [15]. Shellmatta is a tiny and flexible shell implementation for embedded devices.

Any other I/O system is suited as well. Depending on the DUT another interaction framework might be used.

3.3. Test concept

Figure 3 shows the automated test concept for the DIOLINE CANopen bus coupler using the Robot Framework, the PCAN-USB Pro FD and the LION I/O system.

The test cases are organized in test suits and implemented in robot files. The tests use custom Python libraries including a library for the communication with the LION prove bus coupler, the PCAN-Basic API from Peak and a CAN and CANopen wrapper surrounding the PCAN-Basic API. The Robot Framework runs the test suits and generates one report file per test suit.

The DUT (the DIOLINE CANopen bus coupler) is connected to the computer via the PCAN-USB Pro FD communicating via CANopen. A DIOLINE digital in- and a DIOLINE digital output module is connected to the DUT via the DIOLINE protocol (L-Bus 1). The DIOLINE digital I/O modules are stimulated and monitored via the LION digital I/O modules which communicate with the prove bus coupler. The prove bus coupler is connected to the computer via a serial communication.

4. Test cases

In the Robot Framework tests are organized as test suits which are made up of test cases. We organized our tests in four tests suits ranging from SDO, PDO for digital modules, PDO for analog modules and error handling. The four test suits with the corresponding implemented tests are shown in Table 1.

Table 1: Test suits with test cases for the automated HIL tests

Robot Framework Test Suits		
Test suit	**Test cases**	**Interaction system**
SDO	- read/write for manufacturer SDO (e.g. password protected write), - save/ load/ clear of manufacturer parameters	- CAN manager
PDO digital	- inhibit & event time, - transmission types (e.g. RTR RPDO only allowed in old product)	- CAN manager, - DIOLINE digital I/O, - LION digital I/O
PDO analog	- inhibit & event time, - transmission types, - delta for analog modules (with various transmission types)	- CAN manager, - DIOLINE analog I/O
Error handling	- error behavior (for input, output, communication error + freeze on), - emergency messages, - life guarding & heartbeat (e.g. setting heartbeat node ID to own ID only works for old product)	- CAN manager - DIOLINE digital I/O - DIOLINE analog I/O - LION digital I/O

An excerpt of the test report generated by the Robot Framework for the test suit error handling is shown in Figure 2. The failing test indicates that it is not allowed in the redesigned product to set the heartbeat ID to the own node ID.

Each test case is lead with a test setup and followed by a test tear down function. The test case setup function includes network management (NMT) features such as resetting the communication, starting the node, switching back to pre-operational and checks of NMT features such as boot-up message or receiving PDOs at switch to operational. Besides, the test case setup function sets communication parameters to chosen defaults (e.g. event and inhibit time, transmission type of PDO).

The test setup and tear down functions also handle resetting and cleaning up the used libraries (e.g. LION library).

5. Conclusion

The objective of our test concept was to detect deviations for the DIOLINE CANopen bus coupler compared to the previous product and to detect deviations during the development cycle (regression test). We used an approach with automated test for the DIOLINE CANopen bus coupler using the Robot Framework, the PCAN-USB Pro FD and the LION I/O system.

The automated tests were quick and easy to run. Hence, the tests could be run frequently without much effort. This led to finding deviations earlier and, consequently, less effort in debugging and fixing the deviation. At the same time the confidence in the system increased as there were continual tests ensuring the functionality. The tests detected the majority of deviations from the previous product and during the development cycle. The costs and time to perform the tests were low, the training effort for the test environment was also low. The implementation of the tests took a reasonably small amount of time and the tests could be adapted and ported in a reasonable amount of time. The tests, therefore, fulfilled the defined objective.

The tests, of course, did not cover all functionality and could be extended in the future. This could include adding a framework to automatically monitor the LEDs and modify the rotary switches. The test suits also did not cover CAN error handling using error injections or any explicit stress tests. Another extension would be the integration of the test setup onto a server.

Abbreviations

CTT	CANopen conformance test tool.
DUT	device under test.
EDS	electronic data sheet.
HIL	hardware-in-the-loop.
LION	LÜTZE Input Output Network.
NMT	network management.
PDO	process data object.
SDO	service data object.

References

[1] M. Fewster and D. Graham, Software test automation. Addison-Wesley Reading, 1999.

[2] P. I. Christoph Wosnitza, Gabriel Moyano, “Interoperability challenges for CAN FD/PN transceivers: Lessons learned from CAN high speed interoperability tests,” in Proceedings of the 16th international CAN Conference, (Nuremberg, Germany), CiA, 2017.

[3] M. Schwager, “New test concept for improving CANopen device quality,” in Proceedings of the 17th international CAN Conference, CiA, 2020.

[4] B. Lu, X. Wu, H. Figueroa, and A. Monti, “A low-cost real-time hardware-in-the-loop testing approach of power electronics controls,” IEEE Transactions on Industrial Electronics, vol. 54, no. 2, 2007.

[5] P. Sarhadi and S. Yousefpour, “State of the art: hardware in the loop modeling and simulation with its applications in design, development and implementation of system and control software,” International Journal of Dynamics and Control, vol. 3, 2015.

[6] Lütze Transportation, DIOLINE 20. Available at https://www.luetze-transportation. com/products/control-technology/dioline-20 [Accessed: 10.01.2023].

[7] Lütze Transportation, Lütze Rail Technology, DIOLINE - Train Control Management System. Available at https://www.luetze-transportation.com/downloads/file/dioline-train-control-management-system [Accessed: 10.01.2023].

[8] CAN in AUTOMATION (CiA), CANopen conformance test plan - Part 1: CiA 301 testing, Version 1.1.0, 2009.

[9] Robot Framework Foundation, Robot Framework. Available at https://robotframework.org/ [Accessed: 10.01.2023].

[10] S. Bisht, Robot framework test automation. Packt Publishing Ltd, 2013.

[11] S. Stresnjak and Z. Hocenski, “Usage of robot framework in automation of functional test regression,” in Proc. 6th Int. Conf. Softw. Eng. Adv.(ICSEA), pp. 30–34, 2011.

[12] Peak System, PCAN-USB Pro FD. Available at https://www.peak-system.com/ PCAN-USB-Pro-FD.366.0.html?&L=1 [Accessed: 10.01.2023].

[13] Lütze Transportation, LION - Safety-related remote I/O system. Available at https://www.luetze-transportation.com/products/control-technology/io-system [Accessed: 10.01.2023].

[14] Lütze Transportation, Lütze Rail Technology, LION - Train Control Management System. Available at https://www. luetze-transportation.com/downloads/file/lion-train-control-management-system [Accessed: 10.01.2023].

[15] Shellmatta. Available at https://git.shimatta.net/shimatta/shellmatta [Accessed: 10.01.2023].

Carina Heinrich
Friedrich Lütze
Bruckwiesenstr. 17-19
DE-71384 Weinstadt
carina.heinrich@luetze.de
www.luetze.de