

Vector Testing Solution

Version 1.1

2017-10-09

Application Note AN-IND-1-021_Vector_Testing_Solution

Author Vector Informatik GmbH
Restrictions Public Document
Abstract Introduction for Vector Testing Solution

Table of Contents

1	Introduction to Vector Testing Solution	2
2	Traceability Workflow	3
3	Test Execution with CANoe	3
3.1	Introduction	3
3.2	CANoe Testing Concept	4
3.3	Openness to other stimulation and measurement tools, and access to real-time systems.....	9
3.3.1	MATLAB/Simulink Integration	10
3.3.2	Functional Mock-up Interface (FMI)	10
3.3.3	CANoe Fast Data eXchange (FDX)	10
3.3.4	National Instruments LabVIEW	11
3.3.5	ASAM XIL API:	11
3.3.6	COM interface:	12
3.3.7	Jenkins integration:	12
3.4	Scalable Execution Environment	12
4	Test Design with vTESTstudio.....	13
4.1	Introduction	13
4.2	Test Design Languages	13
4.2.1	Coding in CAPL and C#	13
4.2.2	Table-Based Test Design	13
4.2.3	Graphical Test Design by Test Sequence Diagrams	14
4.2.4	Graphical Test Design by State Diagrams	15
4.3	Variant Handling	16
4.4	Parameterization Concept	16
4.4.1	Parameters and Test Vectors	16
4.4.2	Classification Tree Method	16
4.4.3	Parameterized Test Case Lists	16
4.4.4	Stimulation Curves	17
5	Test Interfacing with VT System	17
5.1	Introduction	17
5.2	System Concept	18
5.2.1	Main Characteristics	19
5.3	CANoe Integration	19
5.4	VT System Modules	19
6	Additional Resources	21
7	Contacts	21

1 Introduction to Vector Testing Solution

Vector Testing Solution is a set of software and hardware tools that supports testing of software and embedded systems throughout the entire product development process. It provides methods, assistance and technical approaches for many automotive, medical and avionics software and system testing missions. For most application areas that are related to testing, Vector Testing Solution provides interfaces to other systems.

Vector Testing Solution addresses:

- > Test design
- > Real-time test execution including the control of necessary hardware
- > Reporting
- > Traceability throughout the test process

Figure 1 shows an overview of Vector Testing Solution.

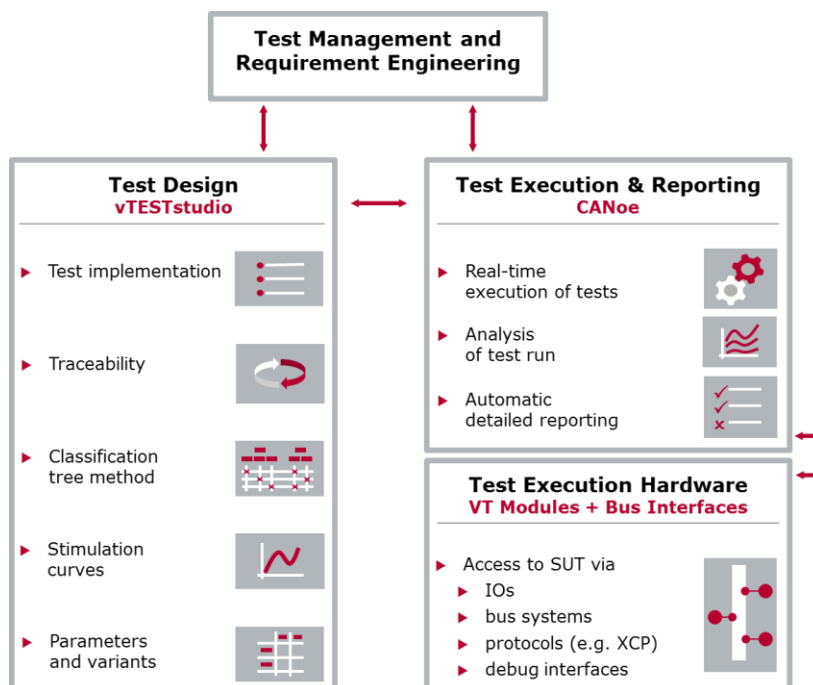


Figure 1: Overview of Vector Testing Solution

vTESTstudio is a powerful environment for test design. It maintains traceability e.g. to requirements. vTESTstudio supports variants, test parameterization, and stimulation curves. Different test design notations (graphical, tabular, and program code), depending on skills and preferences, provide freedom to test designers.

Different test design notations can be compiled by vTESTstudio into a so-called Test Unit. This allows delivering of executable tests to tester personnel. Test Units are ready-to-use files that can be further configured and executed in **CANoe**, as shown in Figure 1. Due to the integrity of Vector Testing Solution, all signals and services throughout different phases of development and the communication buses are accessible to Test Units.

The semantic elements within Test Units are **test cases**. They can have traceability information. A test case has a test result in the form of a verdict, after it is executed. Possible verdicts are none, passed, inconclusive, failed, and error in test system. Test cases are visible in CANoe at execution time and can be individually selected or deselected for the execution. Different grouping concepts are available to bundle test cases. For details of test design with vTESTstudio, please refer to chapter 4.

CANoe has access to the System Under Test (SUT) and to the test environment. In Vector Testing Solution, the SUT can refer to an ECU, a bus node, a software component or a software application. It can also be represented by a hardware/software system. The diverse analysis functionalities of CANoe enable users to analyze and monitor tests during and after the test execution. The test results and details of execution are stored in a comprehensive test report. For details on test execution with CANoe, please refer to chapter 3.

Different peripherals of the SUT (e.g. connected sensors and actuators), the test environment and further measurement systems needed for testing can be accessed and controlled directly from CANoe. The seamlessly integrated VT System is able to stimulate and measure nearly all interface types of SUT, i.e. power control, sensors, actuators, and network access. The details of VT System are provided in chapter 5.

Further utilities enrich Vector Testing Solution:

- > [DiVa](#) is an extension for automated testing of diagnostic software implementations in ECUs. Reproducible test cases are generated based on an ECU diagnostic description.
- > [VectorCAST](#) performs code-coverage analysis during functional or system tests. It can delve into software implementations in C/C++ and Ada.
- > [vVIRTUALtarget](#) allows virtual integration by running AUTOSAR ECU software directly on a simulated environment, e.g. on the developer's PC.

Besides the aforementioned tools, engineering services are provided by Vector to assist users in construction of test benches and performing test development projects.

2 Traceability Workflow

Traceability of requirements not only allows better monitoring of testing progress but also is essential in development of safety and security relevant systems. Many standards such as ISO 26262 or Automotive SPICE acquire requirements traceability to ensure the validation of critical requirements. The challenge in maintaining traceability during testing is to cope with different, in part decoupled tools and process steps.

Vector Testing Solution sustains traceability among requirements and test management, test design, and test execution. Using a connection utility, which is part of vTESTstudio, requirements and test specifications can be imported from requirements and test management tools into the test design tool vTESTstudio as trace items.

The imported trace items can be linked to test case implementations. For better conducting the test design regarding requirements coverage, various mechanisms such as a so-called traceability matrix are provided in vTESTstudio. The traceability matrix gives an overview on coverage of imported requirements by test case implementations and helps in finding the gaps.

The implemented test cases can be compiled to executable Test Units, run on a test execution tool and the corresponding hardware environment, i.e. CANoe and VT System. Finally, the verdicts of the executed test cases can be exported into requirements and test management tools. This allows assessment of test coverage from system requirements to test reports and feeds the requirements and test management tool with the test results. Currently, import/export is supported out of the box for IBM DOORS Classic, IBM DOORS NG and Rational Quality Manager, Siemens Polarion ALM, and PTC Integrity. For other tools, an open exchange format is available.

3 Test Execution with CANoe

3.1 Introduction

CANoe is the comprehensive software tool for development, testing and analysis of entire embedded software and systems, including ECU networks. It supports the entire development process. Its

versatile functions and configuration options are widely used by OEMs and suppliers worldwide. The main features of CANoe—not only limited to testing purposes—are:

- Simulation:** CANoe supports the development of embedded systems in [all product phases](#). Different components of the system under development (e.g. ECUs), represented in CANoe as nodes, can be [fully](#) or partially simulated or it can represent an existing real system. The definition of bus communications is defined by integrated data bases (e.g. DBC, Fibex, and AUTOSAR formats). In simulating the transmission behavior of each network node, [OEM-specific transmission models](#) can be used to ensure the integrity and conformance of the simulation. Furthermore, the system's transmission behavior can be recorded and replayed.
- > **Testing:** CANoe can be used for testing simulated, partly simulated and real systems and software. CANoe's Test Feature Set executes Test Units with full access to simulated¹ and connected bus participants², digital and analog I/Os³, external real-time systems⁴, and external measurement systems. This allows using all necessary variables and signals from: 1) test design, 2) system design (e.g. XCP variables), 3) test environment (e.g. sensors/actors that are simulated by or connected to VT System), and 4) external models and sources (e.g. MATLAB models or LabVIEW models). Test execution can be triggered manually or automated⁵. Moreover, a collection of predefined parameterizable test functions are provided (e.g. to monitor message cycle times and to check DTC status). During test execution, an extensive report is generated, which can be viewed and analyzed using [CANoe Test Report Viewer](#). Details of testing concept are provided in section 3.2.
 - > **Analysis:** The data flow from different sources can be analyzed in CANoe. Regardless of whether it is online or offline, the data can be filtered, either using an integrated filter dialog or for more complex filters and computations by employing CAPL programs or MATLAB models. Symbolic representation of data can be derived by embedding databases. Various presentations of data are provided to satisfy different needs, e.g. graphical representation of signals in charts, state trackers, and online bus trace windows.

Several features are especially important to support the feature categories from above:

- > **Signal stimulation:** Various mechanisms can be used for [setting signal values and variable values](#). These mechanisms are CAPL and .NET programs, [MATLAB](#) models, configurable graphical interaction panels, and Test Units; Test Units consist of test cases and test commands that can access and modify values.
- > **Diagnostics:** The integrated diagnostic feature in CANoe allows flexible utilization of the diagnostic services⁶ of an ECU. Dedicated commands in CAPL and other features like commands in Test Units and diagnostics windows can be used. External applications may configure diagnostics via COM and even communicate via COM. Diagnostic communication can be established by already supported bus systems (including DoIP) and additionally via K-Line. CANoe allows the usage of standardized transport protocols and provides OEM-specific transport protocols within dedicated OEM packages.
- > **XCP utilization:** The widely used calibration and measurement protocol Universal Measurement and Calibration Protocol (XCP) is supported by CANoe. Using XCP it is possible to access ECU internal values, for instance from within a test scenario. Besides the direct XCP communication to the ECU, it can be used together with the [Vx Hardware](#) to access deep into the ECU. This facilitates white box testing.

3.2 CANoe Testing Concept

CANoe enables its users to perform tests from black box to white box tests with arbitrary combination of simulated and real system components in different development phases. Due to tight coupling with test design ([vTESTstudio](#)) and execution hardware ([VT System](#)) and its openness to external sources

¹ i.e. remaining bus simulation

² communication e.g. via [CAN/CAN FD](#), [LIN](#), [FlexRay](#), [Ethernet](#), etc

³ e.g. via [VT System](#)

⁴ e.g. [LabVIEW](#), other HiL systems

⁵ By using COM interface

⁶ Diagnostic services are typically modeled in external diagnostic editors like ODXstudio, CANdelaStudio, etc. Basic diagnostic services can be configured also within CANoe without the need of an external editor.

(e.g. MATLAB models, LabVIEW, measurement devices, etc.), all signals and variables are accessible to tests in CANoe and at the same time to test design. Test cases can access all different sources of signals and variables for stimulation and validation in a unified manner.

Figure 2 depicts a schematic setup of a test system using CANoe and the rest of Vector Testing Solution.

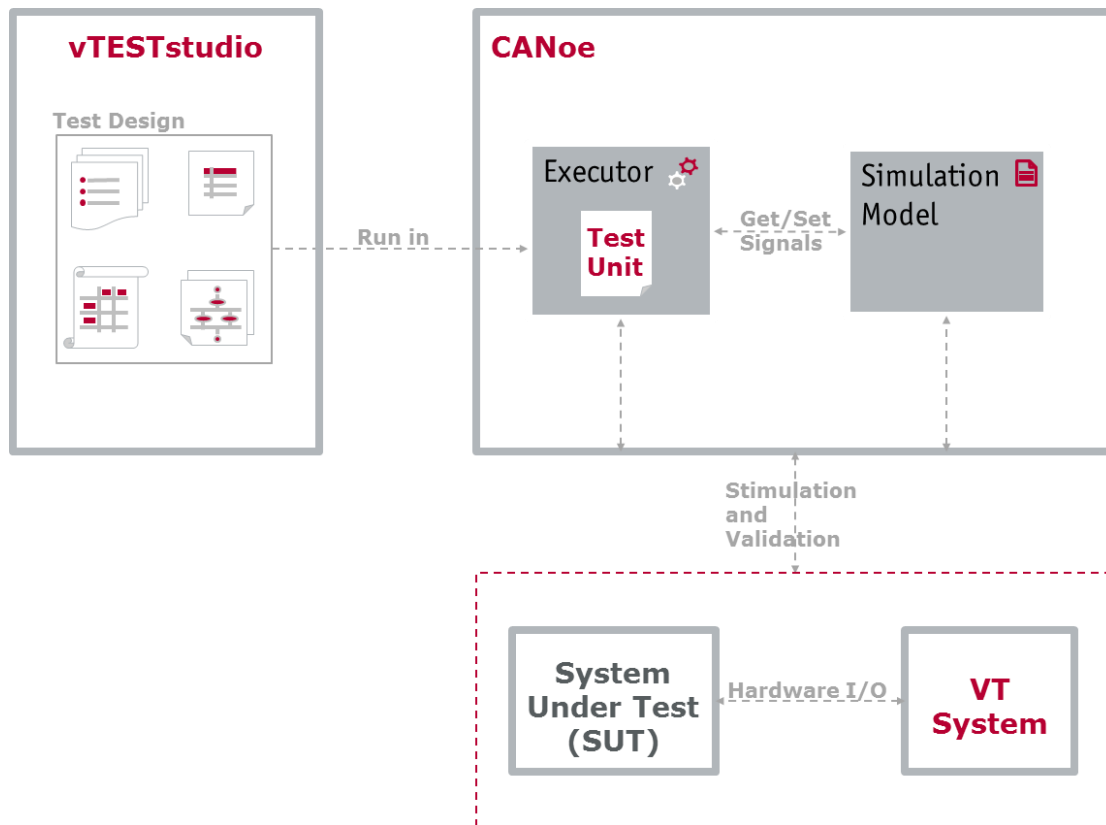


Figure 2: Schematic set-up of a basic test system using Vector Testing Solution.

Integrity and availability of signals and variables

All signals and variables from different sources are accessible in CANoe with consistent time stamps. Moreover, a unified programming environment across Vector Testing Solution is provided.

The SUT typically incorporates a real physical system. Alternatively, SUT can be a pure software component that runs within the simulation model in CANoe. Further measurement and test hardware can be accessed either directly from CANoe or as an extension to VT System. By supporting various bus communication protocols it is possible to access common bus systems out of the box.

Organization of test entities

In CANoe, Test Units are organized into so-called Test Configurations. A Test Configuration represents a set of Test Units, the activation status of test execution elements (test cases and others), and several configuration aspects. A Test Configuration can be executed as a whole. It is possible to have several Test Configurations at the same time, each of which can be executed independently. Besides configurative triggering options, it is also possible to control test execution from an external source using the COM interface (see COM interface on page 12). Different execution options of Test Configuration allow flexible control of the test execution. For instance, a Test Configuration can be set to:

- > run multiple times
- > repeat until a certain period of time is passed by
- > depend on a derived verdict, e.g. stop on verdict "failed"

Optionally, the test execution elements within a Test Configuration can be executed in a random order.

Test execution

The Test Configuration Window allows viewing and editing of a test configuration, deciding on variant alternatives, activating and deactivating test execution elements, controlling the test (start/stop/pause) and viewing progress and verdicts. The test cases (and if available their grouping elements) within a Test Unit are visible in the Test Configuration Window.

As a very unique feature, the test execution itself is done in real-time, and also the computation of the simulation model(s). It is additionally possible to monitor the current test execution online (elapsed time, verdicts) and to deeply analyze the test course by using analysis windows and the Test Trace Window. After test execution of a test configuration has been finished, the test report can be viewed. Figure 3 shows a screenshot of the Test Configuration Window.

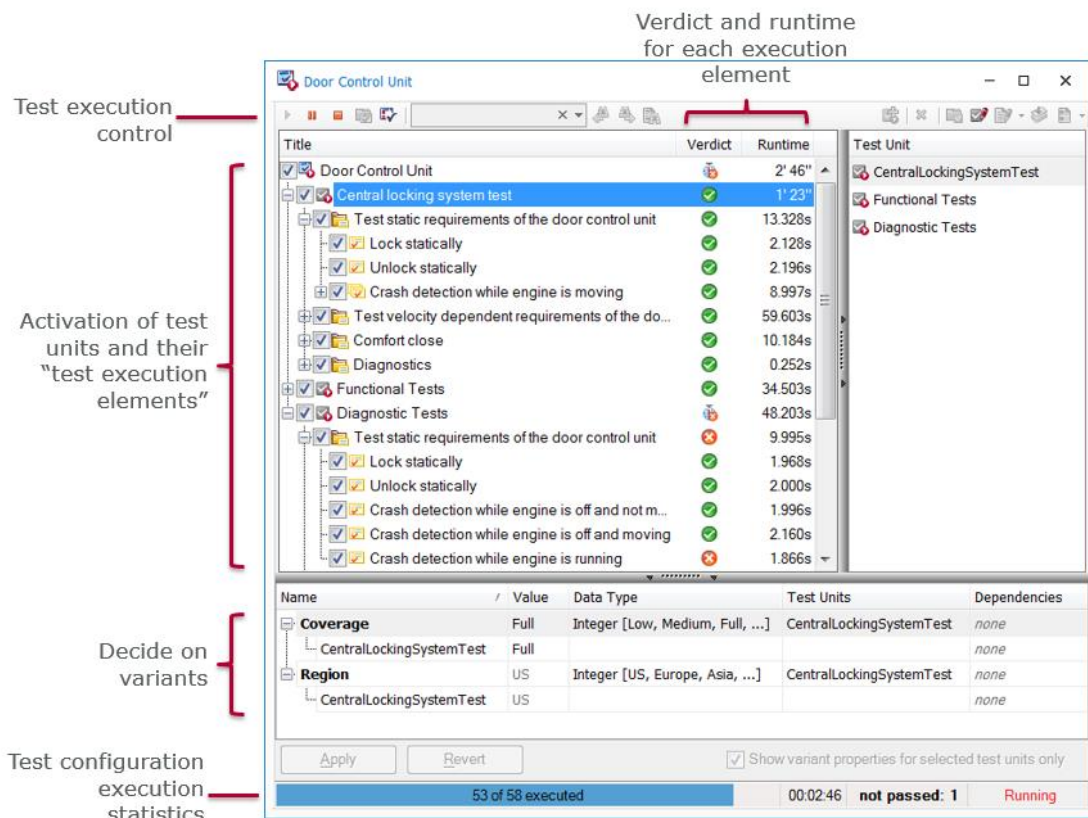


Figure 3: Test Configuration Window during Test Execution

Automated tests may include steps that the tester has to perform manually. For instance, the automated test may prompt the user to check whether a lamp is on, or if a certain value appears on an indicator. If the test implementation includes such commands, CANoe requests an action from the tester and specifically prompts the tester. In the confirmation dialog, a dedicated question is contained; further optional information such as an image or links to arbitrary resources can be included.

Online monitoring of test execution

In addition to the Test Configuration Window, the Test Trace Window is at hand to observe and analyze single or multiple tests during the test run and after the test has been completed. All executed actions of the tests are directly displayed in the Test Trace Window during test execution. Multiple

Test Trace Windows can be created and configured as needed. Figure 4 shows a screenshot of a Test Trace Window.

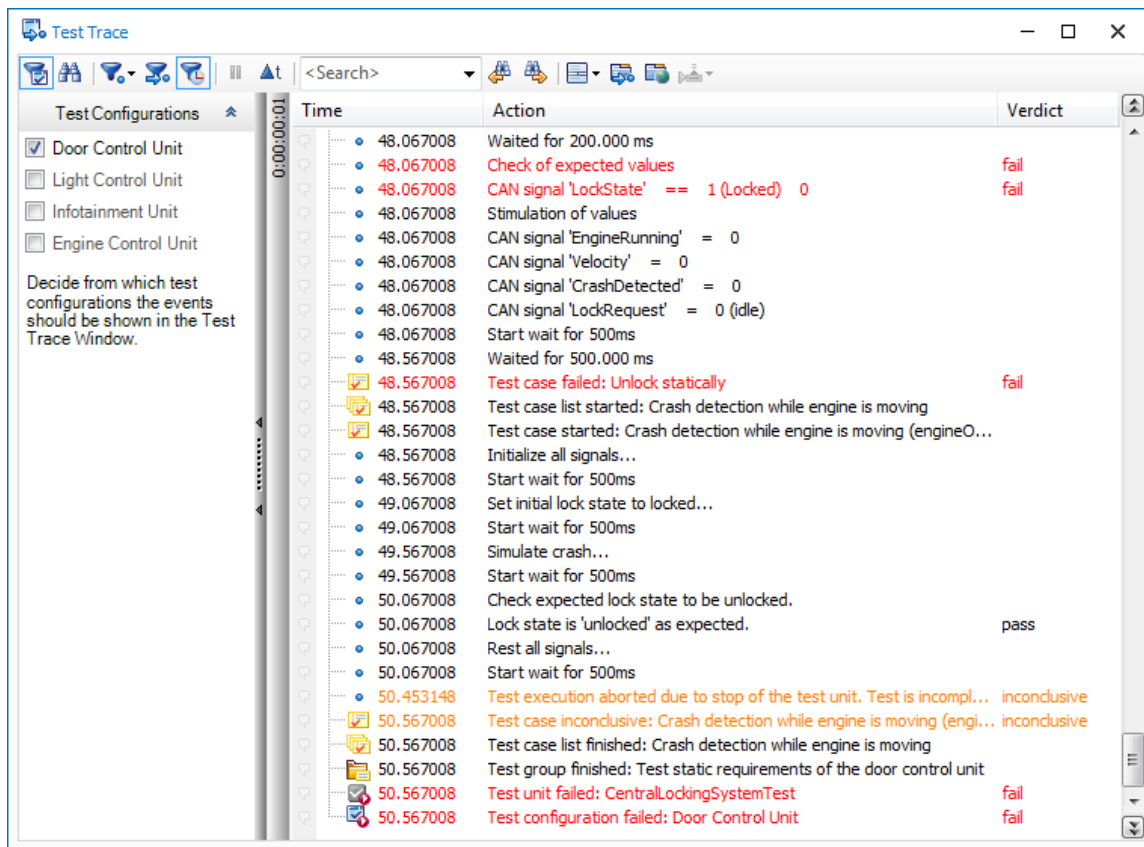


Figure 4: An example of a Test Trace Window in CANoe

One of the strengths of testing with CANoe is its powerful set of auxiliaries for online analysis of e.g. measuring devices, bus communication, and fault injection. The online analysis can be performed during and after the test run.

- > Multiple Graphics, State Trackers and Trace Windows can be set up and configured independently to visualize e.g. signals and system variables on their timely occurrence.
- > Test Traces can be used to observe and analyze test runs.
- > Communication Bus Trace Windows can be used to observe communication activities and system variable changes.
- > Graphic Panels can be designed in CANoe to modify and display the values of signals and system variables interactively by the user during tests, e.g. to inject faults. Panels are sets of graphical elements that represent switches, push buttons, control knobs, etc. The panel elements can be related to a specific signal and upon activation stimulate a certain value.

Different online analysis windows can be configured to visualize results in a time-synchronous manner. Figure 5 depicts a screenshot of an example CANoe configuration to analyze and monitor test execution online in CANoe.

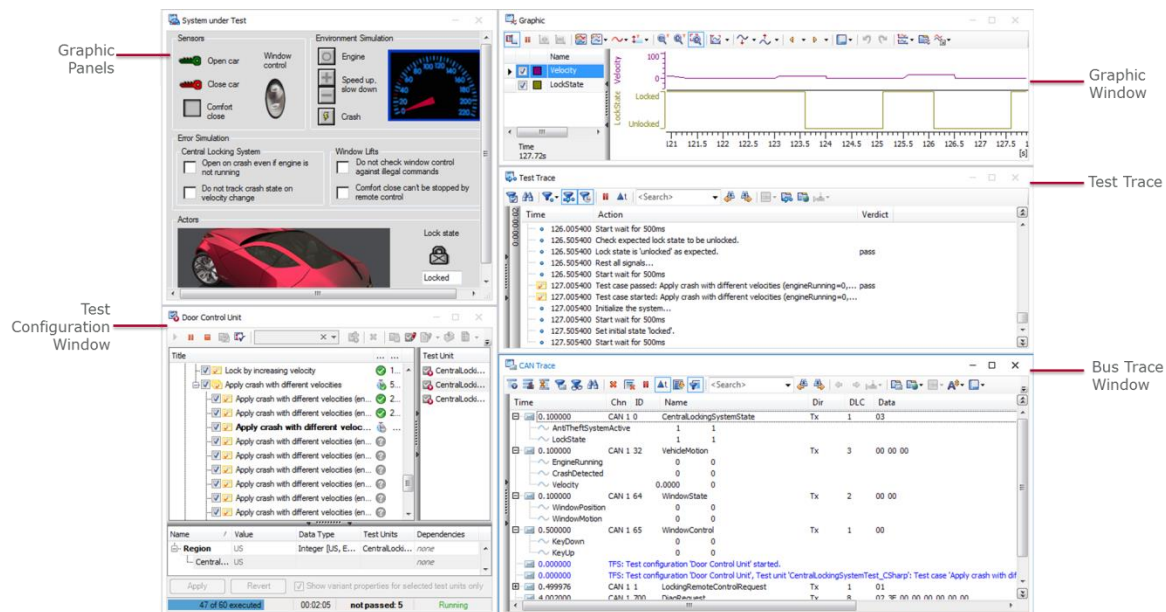


Figure 5: An example of different auxiliaries in CANoe for online analysis and monitoring of tests

Test report

During the test execution, CANoe automatically writes a comprehensive test report as a file. This file contains all information on test execution. The information contained in a test report is diverse: Global information such as tester and test environment and test-specific information such as test cases, test steps, measured values, results and verdicts are a few examples of the information contained in a test report.

After the test has been completed, the test report can be viewed and analyzed using **CANoe Test Report Viewer**. This utility is installed together with CANoe, but it can also be [downloaded](#) free of charge as a separate tool; for instance to allow stakeholders, without a CANoe installation, to view and analyze the test report.

CANoe Test Report Viewer offers various configurable views on a test report:

- > Execution Tree view shows in a hierarchical way the structure of the Test Configuration, the containing Test Units and test cases. Each element can be selected and its details can be viewed in the content area. Depending on the element selected, either a grouping element such as a Test Unit or a test case, different views and contents are presented.
- > Predefined and user-defined queries can be used to present specific information in the result view, which lists the results of the query. The elements of the result list can be selected and its details can be viewed.
- > Filtering the views can be arranged depending on various criteria such as verdict.

Using CANoe Test Report Viewer, test reports can be intuitively viewed, explored, and analyzed. The following capabilities and features make CANoe Test Report Viewer a powerful and efficient report analysis tool:

- > Easy navigation within a test report, e.g. between different information types or time stamps
- > Usability: Intuitive and efficient analysis of the test by using a selectable and navigable execution tree, multi-tab viewing, flexible querying and filtering
- > Traceability: The testing tool landscapes in different organizations are expanded over requirements engineering, test design, test execution, and test management. In order to support existing workflows and to maintain the traceability chain, it is possible to export fundamental test report information as an XML-file.
- > Queries: CANoe Test Report Viewer allows defining queries ad-hoc to search across the whole test report. Using predefined building blocks in a directed query builder, it is possible to quickly create new queries, without knowing the underlying data model in detail. CANoe Test Report Viewer comes with plenty of predefined queries from the domain of testing. Further

user-defined queries can be created and saved for later reuse. Examples of queries are “list of all test cases grouped by their verdicts” or “list of all trace items grouped by the results of their linked test cases.”

- **Filters:** CANoe Test Report Viewer maintains all details of the test report. However, depending on the use case, certain information may not be needed by the user. In order to hide non-relevant information, different filters can be activated. This helps to reduce the amount of information and consequently leads to better readability and situation-specific viewing of test reports.

Figure 6 shows a screenshot of the CANoe Test Report Viewer.

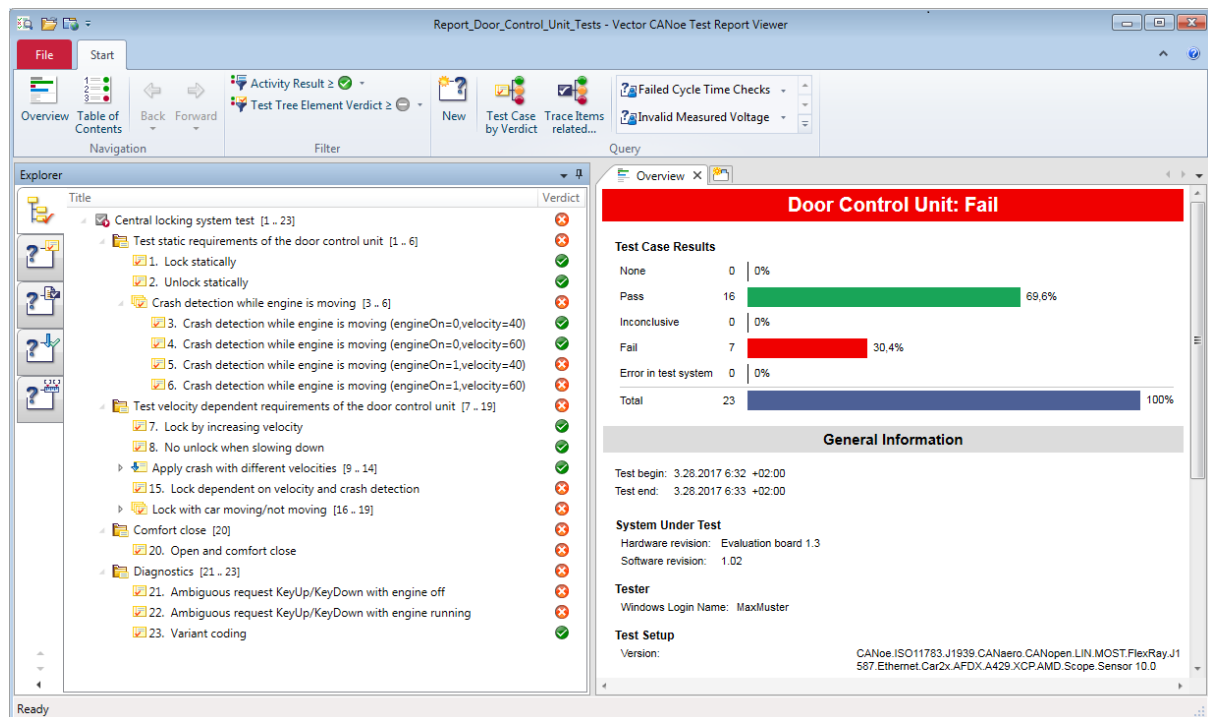


Figure 6: CANoe Test Report Viewer

3.3 Openness to other stimulation and measurement tools, and access to real-time systems

CANoe is an open environment that allows integration of numerous different sources and sinks. By implementing different communication protocols and supporting common interfaces, it is possible to e.g. embed external models, access Vector and third-party (real time) hardware systems, or control CANoe from external components. The latter is for instance needed to fully automate the testing process. Figure 7 illustrates the openness of CANoe to other (external) tools and systems.

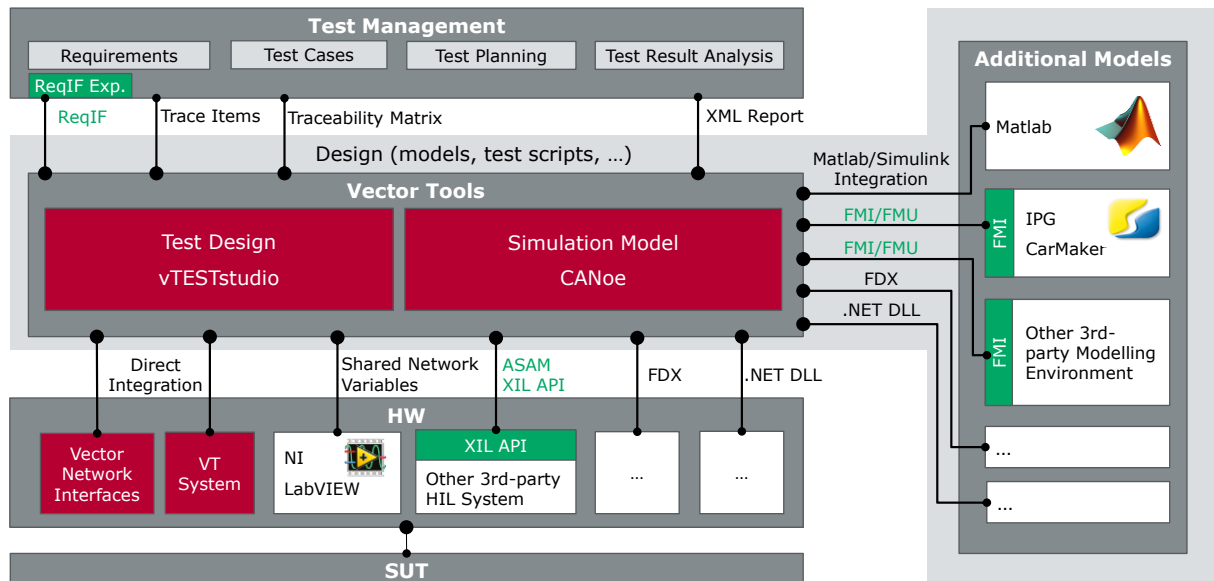


Figure 7: Openness of CANoe to other tools and systems

3.3.1 MATLAB/Simulink Integration

The purpose of this interface is to extend the node modeling capability of CANoe by adding the strength of the MATLAB/Simulink environment. MATLAB models can be used to describe an arbitrary processing of e.g. a simulated node or the test environment.

Typical use cases that rely on the integration of MATLAB/Simulink in CANoe are pre-processing and generation of signals (e.g. as stimulation or verification for tests), testing in MIL-phase, and simulation of the environment.

3.3.2 Functional Mock-up Interface (FMI)

FMI defines a standardized interface for the coupling of different complex dynamic models to build up one interconnected system, composed of those modeled components. The complex dynamic models (physical, mechanical, electrical, etc.) may refer to different components of a system with complex behavior and interconnections.

By using the **FMI interface in CANoe** it is possible to create one overall simulation environment consisting of different software models, e.g. MATLAB, LabVIEW and CarMaker. Furthermore, it can be employed for co-simulation purposes. With this functionality, the validation and testing of the interaction of the components within a heterogeneously simulated environment in CANoe can be easily done. Sample FMI configurations in CANoe simplify the application for its users.

3.3.3 CANoe Fast Data eXchange (FDX)

CANoe FDX is a protocol that can be used for easy, fast, and real-time data exchange between CANoe and other systems via Ethernet. The protocol gives other systems read and write access to a desired selection of CANoe system variables, environment variables, and bus signals. In addition, it is possible to send control commands (e.g. for starting and stopping the measurement) to CANoe or to receive status information via FDX. The other system could be a HiL system in a test bench or a PC that is to display CANoe data, for example. The protocol is based on the widely used and efficient UDP protocol standard. Thus, it is almost universally applicable. It enables users to implement specific solutions on their own.

CANoe supports specifying the protocol, and how and which signals and variables should be exchanged with external systems via FDX. For this purpose, an editor utility is provided within CANoe that allows defining XML-based description files for FDX data exchange.

3.3.4 National Instruments LabVIEW

CANoe allows out-of-the-box integration of LabVIEW based systems. There are 2 types of LabVIEW integrations:

- > Integration via LabVIEW blockset: Within a specific CANoe add-on for LabVIEW, a dedicated blockset is provided that operates the CANoe COM server. This integration can be used to control CANoe and to exchange few data.
- > Integration via LabVIEW shared network variables: Very fast integration that allows the integration of NI hardware to CANoe in real-time. This integration is appropriate for exchanging much data.

3.3.5 ASAM XIL API:

The [ASAM XIL API](#) is a standard that specifies software interfaces for the access of different XIL systems to a test system. The API defines several ports, e.g. to access model data, diagnostics and electric error simulation. By employing ASAM XIL API, the preferred test automation tool can be programmed in a standardized manner and can access information from XIL systems. Figure 8 depicts an overview on ASAM XIL API and the interaction between the test automation tool and test bench.

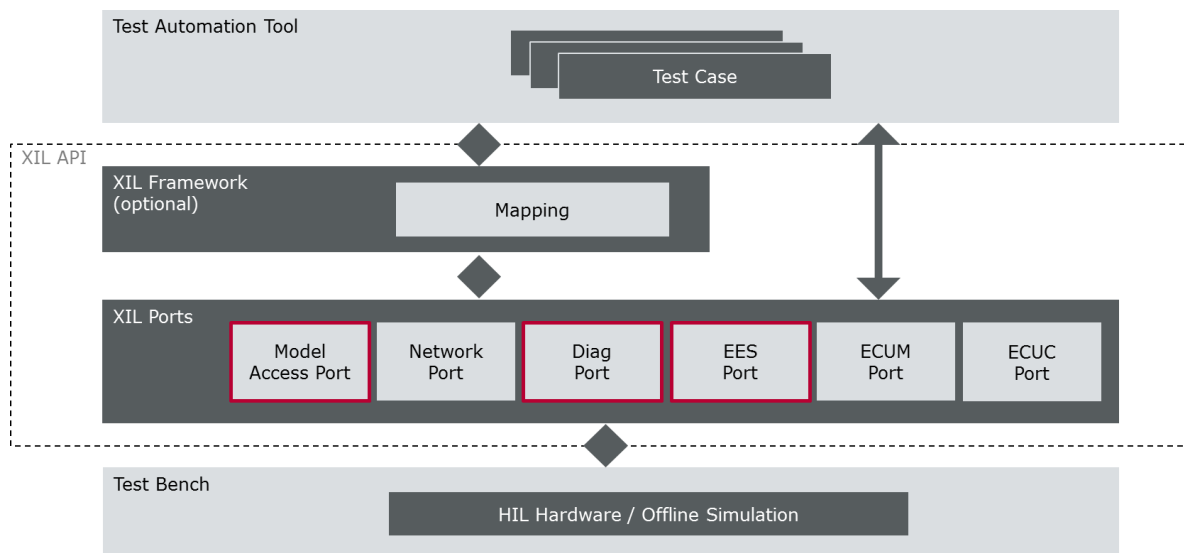


Figure 8: Overview on ASAM XIL API and the interaction between the test automation tool and test bench. The boxes with red border are provided by CANoe out of the box.

The components with red borders in Figure 8 are supported in CANoe as a server, out of the box. The Model Access Port enables you to read and write model variables using a standardized interface. Optionally, the variables can be stimulated with signal generators or can be recorded. The Diagnostic Port allows you to access the diagnostic functionality of the ECUs using a standardized interface. Finally, Electrical Error Stimulation (EES) Port enables you to generate electrical errors using a standardized interface. A variety of possible error types, such as interrupts and short circuits, are ready to use in CANoe.

There are two scenarios, where the ASAM XIL API helps. Figure 9 demonstrate these two use cases.

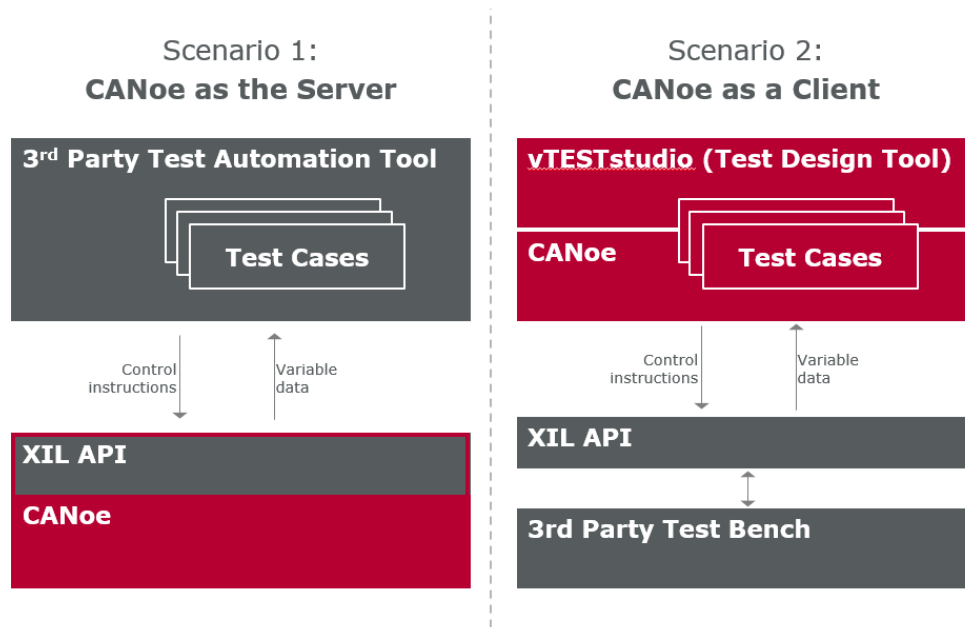


Figure 9: Possible scenarios in using the ASAM XIL API in CANoe

3.3.6 COM interface:

With the help of the COM interface it is possible to control CANoe from external programs. It allows extending CANoe to further situation-tailored logics for setup and control purposes. CANoe provides a COM server with numerous objects to access CANoe remotely, e.g. to start and to stop the measurement and to dynamically build up/modify test configurations. Therefore, the COM interface can be used to control fully automated test benches, for instance.

3.3.7 Jenkins integration:

[Jenkins](#) is a widely used continuous integration and test platform. CANoe can be controlled by Jenkins as a ready-to-use feature. Using a simple batch script returning error-level information, in Jenkins it can be decided if a job was successful or not. You can specify additional files produced by the simulation such as test reports and logging files to get a detailed view of the simulation result. The general result as well as the additional files can be directly accessed through the Jenkins web interface. If your simulation is running tests, Jenkins provides a more specific view on JUnit test results. To benefit from this feature you can provide a simple transformation from CANoe test results to JUnit test results. The general approach behind integration of Jenkins is applicable to other continuous integration systems. Since the tool integration is based on simple operating system mechanisms, you can easily adapt the approach to other tools.

3.4 Scalable Execution Environment

CANoe is available in three main variants, Full, Run, and Pex:

- > CANoe Full provides all functional features. Test configurations can be developed and tests can be run. This variant is intended for users who want to use CANoe's full functionality.
- > CANoe Run allows performing tests without the ability to change configurations. The analysis functions are available.
- > CANoe Pex (Panel Execution) allows performing tests without the analysis functionality. Pex enables low-budget test benches.

The details of each CANoe variant can be viewed [here](#).

Depending on the user-specific testing needs CANoe supports scalable execution environments. It can run on PC's, RT Racks, and on specific-purpose hardware.

4 Test Design with vTESTstudio

4.1 Introduction

vTESTstudio is a powerful development environment for creating automated tests. In order to increase efficiency in terms of test design and to simplify reusability, it provides programming-based as well as table-based and graphical test notations and test development methods.

vTESTstudio can be employed in all product development phases: from model testing to system validation. Thanks to the use of open interfaces vTESTstudio can be simply incorporated into existing tool landscapes.

By providing different test design languages vTESTstudio is suitable for multi-stakeholder and distributed teams. Corresponding editors for different test design notations allow fully-assisted implementation and review of test design. By using libraries and introducing reusability concepts, it is not only possible to better structure test design but also to accelerate implementation. The close interaction with CANoe enables easy access to the System Under Test (SUT), and to exchanging other signals, variables, and parameters.

Furthermore, special test design features such as parametrized test cases or the classification tree method are integrated to ensure optimized test coverage, while keeping the effort for test designers to a minimum. Test implementation can be done according to variants of the SUT and to test variants as well. Predefined and user-defined test commands for stimulation and verification, i.e. test case parameters, signals, etc., can be defined and assigned (per variant).

4.2 Test Design Languages

Conventional test design implementation methods, e.g. coding in CAPL or C#, are solid means to realize comprehensive test logic. However, those methods require programming skills and experience, reviewing code is tedious, and is prone to hold undetected errors. Often, depending on the involved user role, different levels of skill and requirements on test implementation exist. To accommodate users with a suitable implementation method, vTESTstudio provides different test design languages that can be used solely or in combination to build up a test. Thus, collaborative test design in a heterogeneous team with different skills and backgrounds is possible.

vTESTstudio supports CAPL, C#, test tables, test sequence diagrams, and state diagrams as test design methods. A dedicated editor is provided for each method.

4.2.1 Coding in CAPL and C#

vTESTstudio allows programming tests in CAPL and in C#. The CAPL and C# editors integrated in vTESTstudio provide the functions of modern development environments such as code completion, syntax checking, and direct access to network symbols, environment data, diagnostic symbols, and parameters.

CAPL and C# allow implementing procedural and event-triggered programs. Furthermore, they provide many predefined test functions that enable access to the values in a CANoe run-time environment. CAPL and C# provide great flexibility and freedom in implementing tests.

4.2.2 Table-Based Test Design

Using the so-called Test Table Editor, it is possible to easily define test sequences in tabular form without the need to have programming skills. Specific test commands are available for stimulating and testing the SUT. For better usability, the Test Table Editor provides different features such as:

- > Command and symbol completion while writing
- > Access to signals, environment data, diagnostic symbols, and parameters
- > Access to user-defined functions in other languages

Test Tables are simple to read through and understand. Figure 10 shows a screenshot of the Test Table Editor.

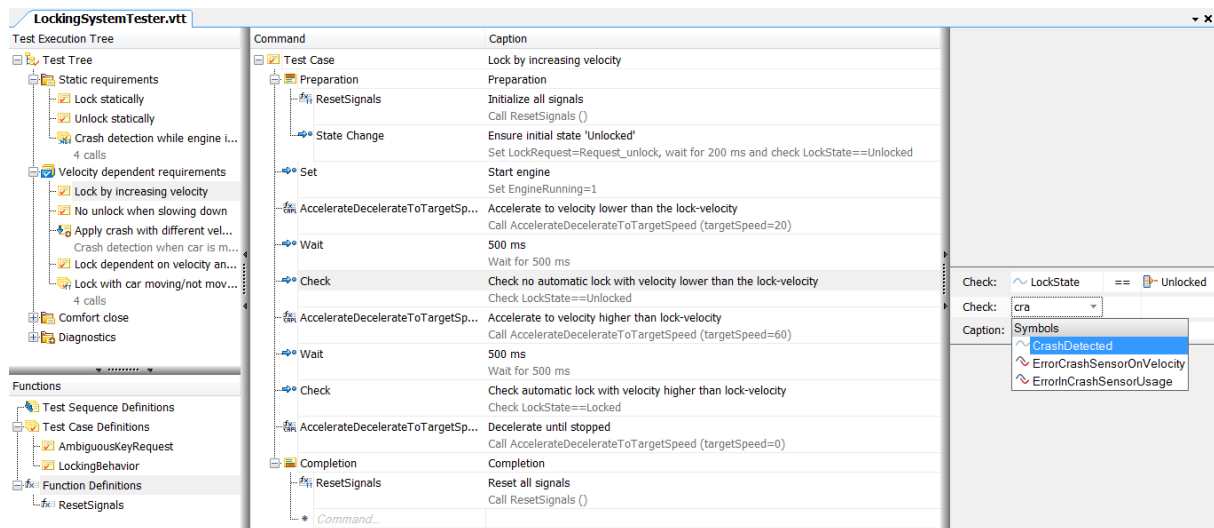


Figure 10: Test Table Editor with a tabular implementation example

4.2.3 Graphical Test Design by Test Sequence Diagrams

The Test Sequence Diagram Editor can be used to define tests in a graphical manner. The test logic is defined typically by the use of graphical notation optimized for automated tests. Out of the test sequence diagram, test cases are generated during the build process. For the realization of the test functionality within diagrams, both predefined test commands as well as user-defined functions may be used. Test sequence diagrams provide a graphical abstraction by separating control flow from test implementation. Thus, it is much easier to get an overview of the test without dealing the implementation details. Often the graphical implementation can be used to communicate the test design with management or other stakeholders that require an abstract view on test design and for reviewing purposes. Figure 11 shows a screenshot of the Test Sequence Diagram Editor.

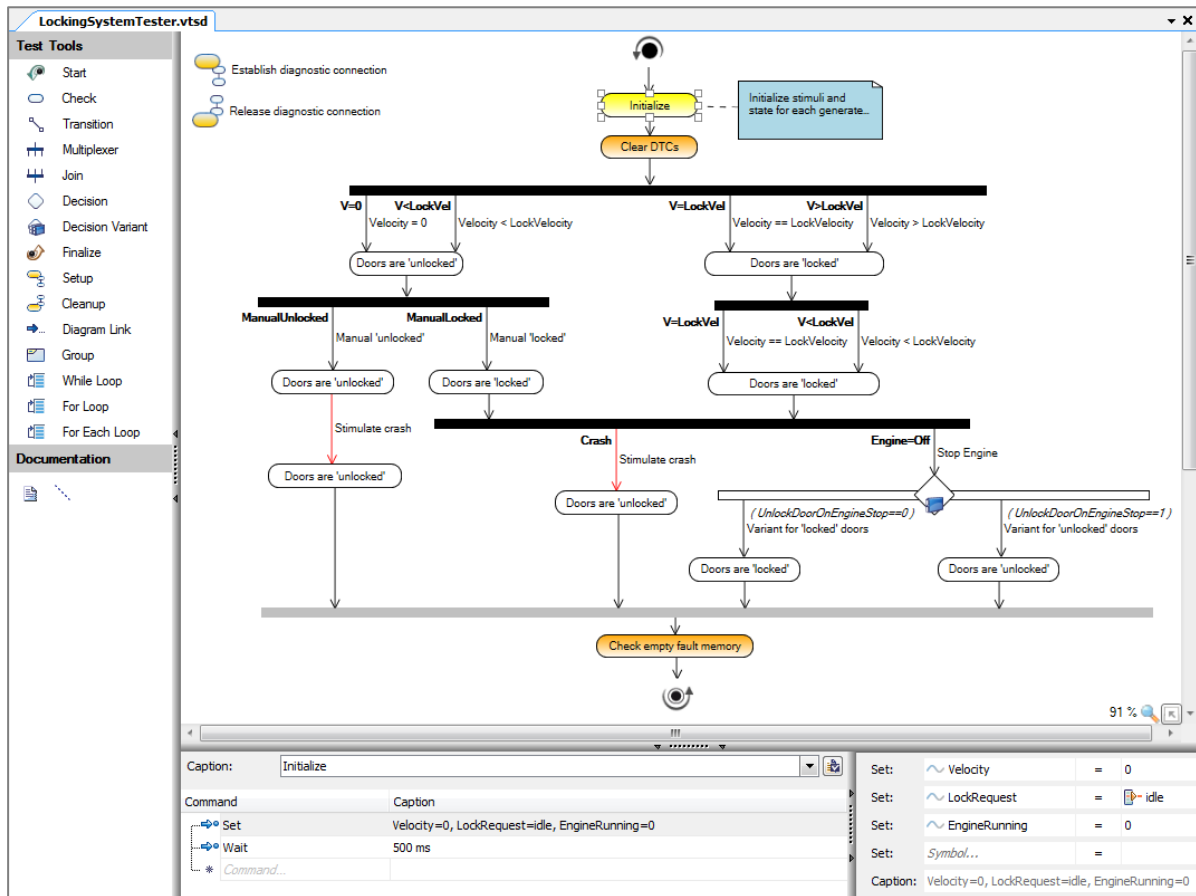


Figure 11: Graphical Test Design using a Test Sequence Diagram in vTESTstudio

4.2.4 Graphical Test Design by State Diagrams

Using the State Diagram Editor it is possible to model the expected behavior of the SUT for automatic test case generation. This test design technique offers a compact and concise way to describe what is to be tested, but not how it is to be tested. Different algorithms are supported for the generation of the test cases. State diagrams are suitable for functions that exhibit several different states and for providing transition coverage by tests. Figure 12 shows a state diagram example in vTESTstudio.

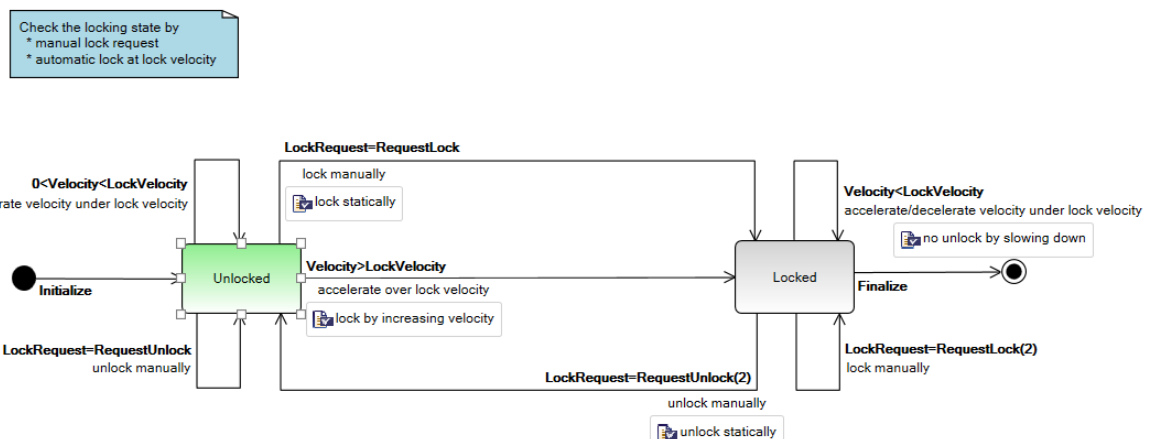


Figure 12: Graphical Test Design using State Diagram in vTESTstudio

4.3 Variant Handling

Variants of a SUT (e.g. ECU variant Europe, Asia, and North America) and test variants (e.g. test coverage full, low, and medium) can be handled in vTESTstudio. For this purpose, the so-called variant properties are used. Variant properties can be used for conditional test coding as well as for accessing variant-dependent parameter values, and for defining variant-dependent test cases or whole test groups. The value of a variant property is either already defined at design time in vTESTstudio or can be set up in CANoe prior to test execution. Variant properties can be dependent on each other, i.e. the value of one variant property (e.g. region) can determine the value of another variant property (e.g. speed limit). Regardless of the implementation editor used, after setting the variant properties test cases can be generated based on variants.

4.4 Parameterization Concept

4.4.1 Parameters and Test Vectors

The term "parameter" refers to any constant value that can be accessed within the test sequence from all implementation languages. Examples of parameters are configuration parameters for a control unit, test vectors, tolerances, etc. Parameters are defined and maintained in separate files and are accessible to test design in vTESTstudio.

4.4.2 Classification Tree Method

Using an integrated editor for the classification tree method, test case data - in terms of test vectors - can be defined. The graphical user interface supports finding the relevant input data for a test. Automatic or manual combination of all crucial input values allows defining the minimum number of required test vectors. The explicit support of boundary values additionally helps in considering critical values as minimum and maximum values of equivalence classes.

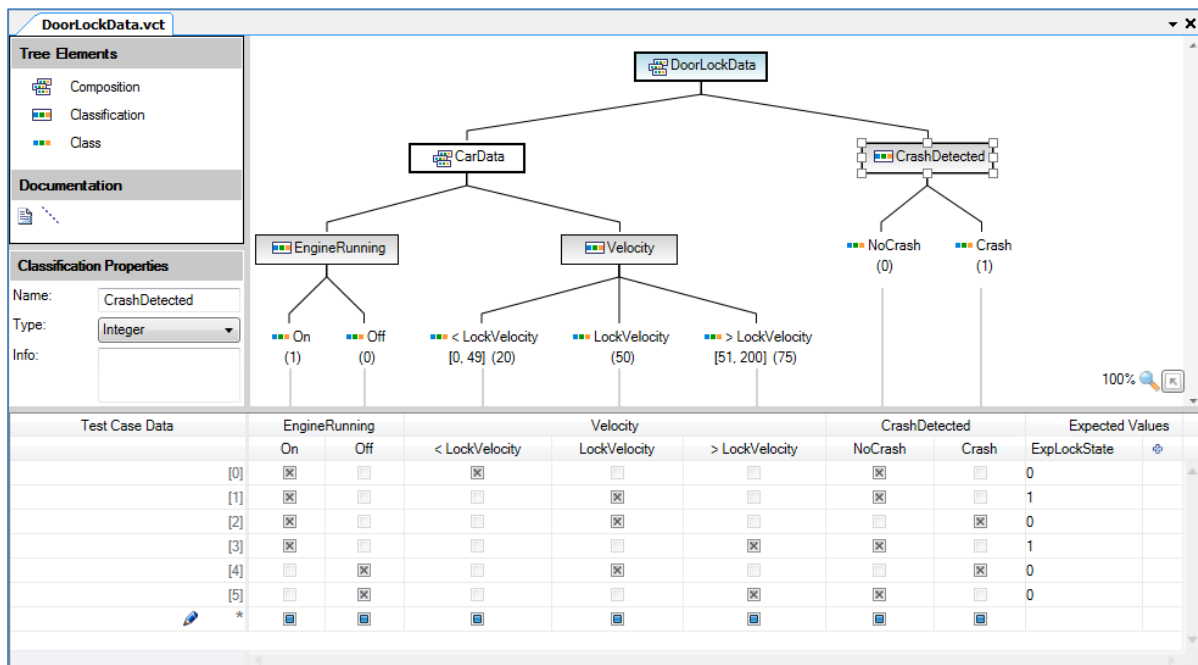


Figure 13: Classification Tree in vTESTstudio

4.4.3 Parameterized Test Case Lists

A test case can be called with different parameter values, resulting in a so-called test case list. A test case list can be defined by entering multiple values for each passed parameter of a test case definition. For test execution, vTESTstudio automatically generates a list of those test cases that either use every parameter value at least once, in pairwise combination, or in any combination of parameter

values. The individual test cases in a list are visible in CANoe and can thus be individually activated and deactivated.

4.4.4 Stimulation Curves

vTESTstudio allows definition of stimulation curves to be replayed on signals at test execution time. For this purpose, the Waveform Editor is included for defining curves for the stimulation of SUT. Predefined segment types (sinus, pulse, etc.) enable easy definition of e.g. voltage curves. Multiple curves can be synchronized easily within the same editor. Figure 14 shows a screenshot of the waveform editor.

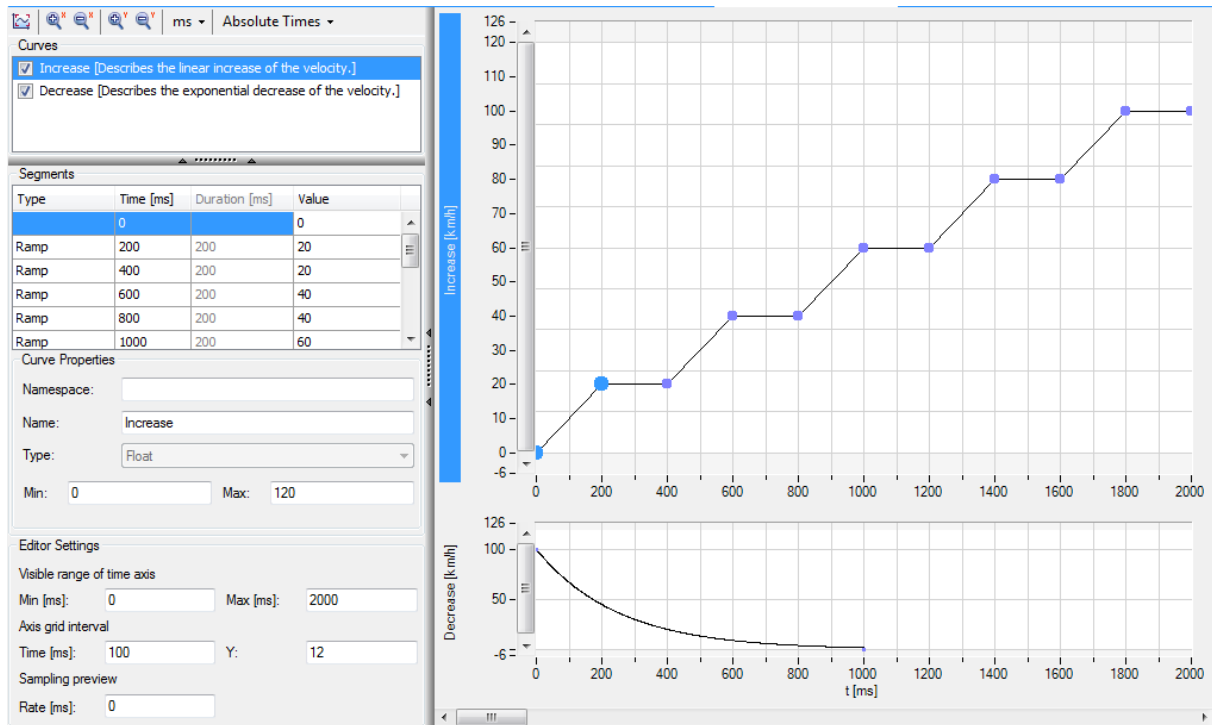


Figure 14: Waveform Editor with two synchronized wave forms

5 Test Interfacing with VT System

5.1 Introduction

A System Under Test (SUT) has numerous I/O interfaces to which actuators and sensors can be connected. Moreover, the SUT may be a bigger network of control units. A modern and flexible test system must operate these interfaces and acquire data during the test. With the VT System, Vector provides a modular and scalable test environment. It simplifies the setup of test benches and HiL test systems immensely, since it integrates all function blocks needed for a test use case to connect an I/O or bus network channel in one module. This allows fast setup of test systems and minimizes wiring effort. Examples of I/O channels are an ECU's output for controlling a headlamp or input for connecting a brightness sensor. Figure 15 illustrates a setup example of a SUT (ECU) and its interconnected I/O-components.

- ▶ ECU has to be tested in its "natural" environment
 - ▶ Other ECUs in network
 - ▶ Battery
 - ▶ Sensors
 - ▶ Actuators

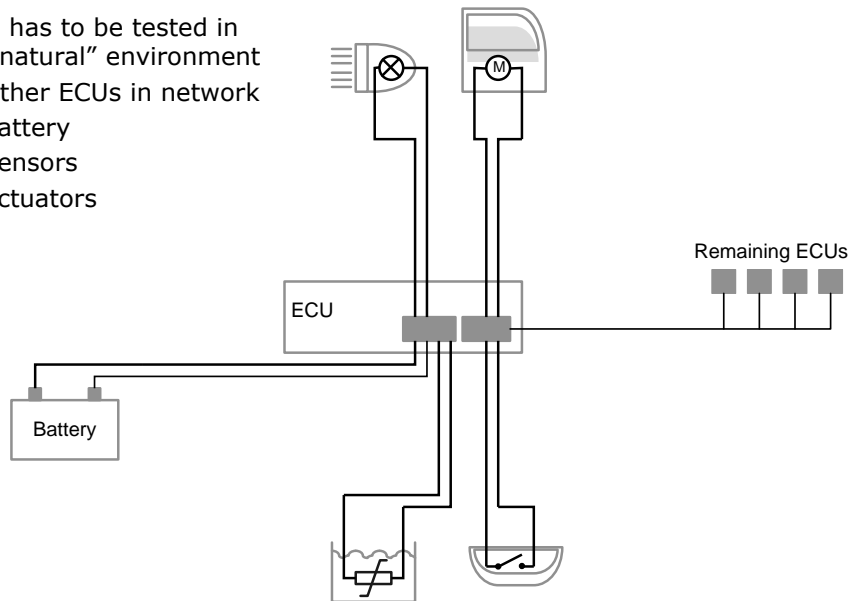


Figure 15: An example of an ECU with interconnected I/O

The VT System is configured and controlled by CANoe; thus, tests can be initiated in CANoe, with access via VT System to SUT and to all its connected I/O components. Furthermore, different modules of VT System provide various predefined testing functionalities (e.g. fault injection and power supply) out of the box.

5.2 System Concept

VT System provides an abstraction layer between the SUT (e.g. an ECU) and the test environment, i.e. all connected components to the SUT. VT System can either simulate all SUT interfaces (power supply, sensors, actuators, and network) or switch to real components. Figure 16 illustrates the main concept of VT System.

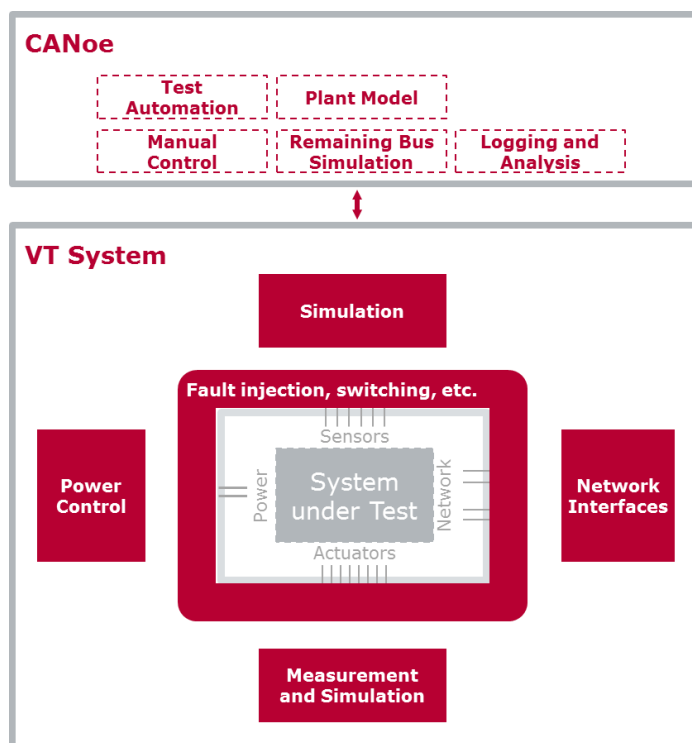


Figure 16: VT System encapsulates the System Under Test and provides all necessary interfaces for the test execution tool. The tight integration of CANoe allows a ready-to-use, holistic testing solution.

5.2.1 Main Characteristics

The predominant characteristic of VT System is its integrated interface to the SUT (e.g. ECU). By embedding functions such as signal conditioning and processing and predefined test procedures - and by facilitating relay switches and the necessary I/O hardware - it simplifies testing and minimizes wiring effort.

Numerous **dedicated test function blocks** make testing efficient. Examples of the test function blocks in VT System are decade resistor for sensor simulation, controllable electronic load for load simulation, relays for short circuit and open load, and specialized measurement capabilities like root mean square or wide range current measurement.

The **modularity** of the VT System components allows customizable setups and situation-tailored solutions. Various independent I/O modules for desktop housings or 19" racks can be ordered according to needs. The connectivity over Ethernet to regular PCs makes the VT System versatile in its application domain. By supporting, for instance, electrical characteristics for ECU tests, it integrates domain-specific functionalities to ease the testing process.

5.3 CANoe Integration

All I/O signals with accurate time stamps are available in CANoe as system variables. The I/O signals can be directly accessed by automated tests. The availability of I/O signals can be used to write tests that include I/O channels and to set up different utilities in CANoe, e.g. analysis windows, panels and logs. Furthermore, it is possible to manually operate SUT from CANoe. For this purpose a VT System-specific GUI is available in CANoe. Figure 17 shows a screenshot of the dedicated GUI to control VT System and consequently the interconnected SUT.

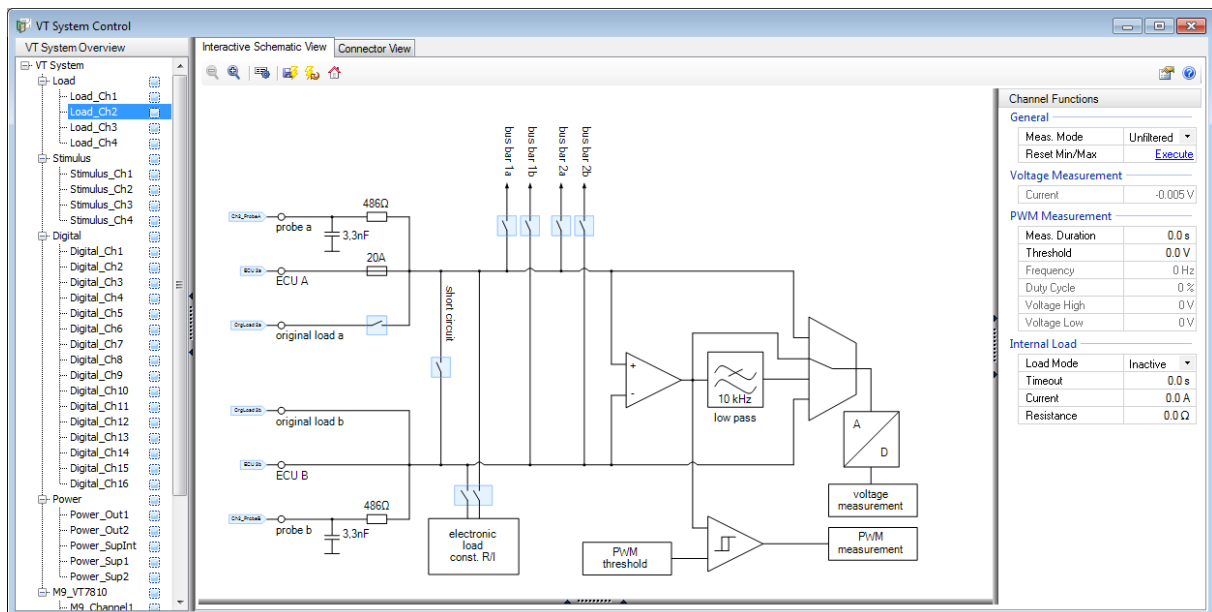


Figure 17: The dedicated GUI to manually control the System Under Test that is connected to VT System

A video on HiL tests using VT System and CANoe can be viewed [here](#).

5.4 VT System Modules

Various VT System modules have been developed, driven by the domain-specific needs in testing ECUs and other embedded systems. The set of domain-specific modules grows along with emerging technologies and trends. Generic modules extend the set of VT System modules to provide users with the freedom to realize specific test benches and environments. This is, for instance, done by simple configuration of those modules. Figure 18 shows an example of a VT System module. It highlights the common parts that are included in different modules.

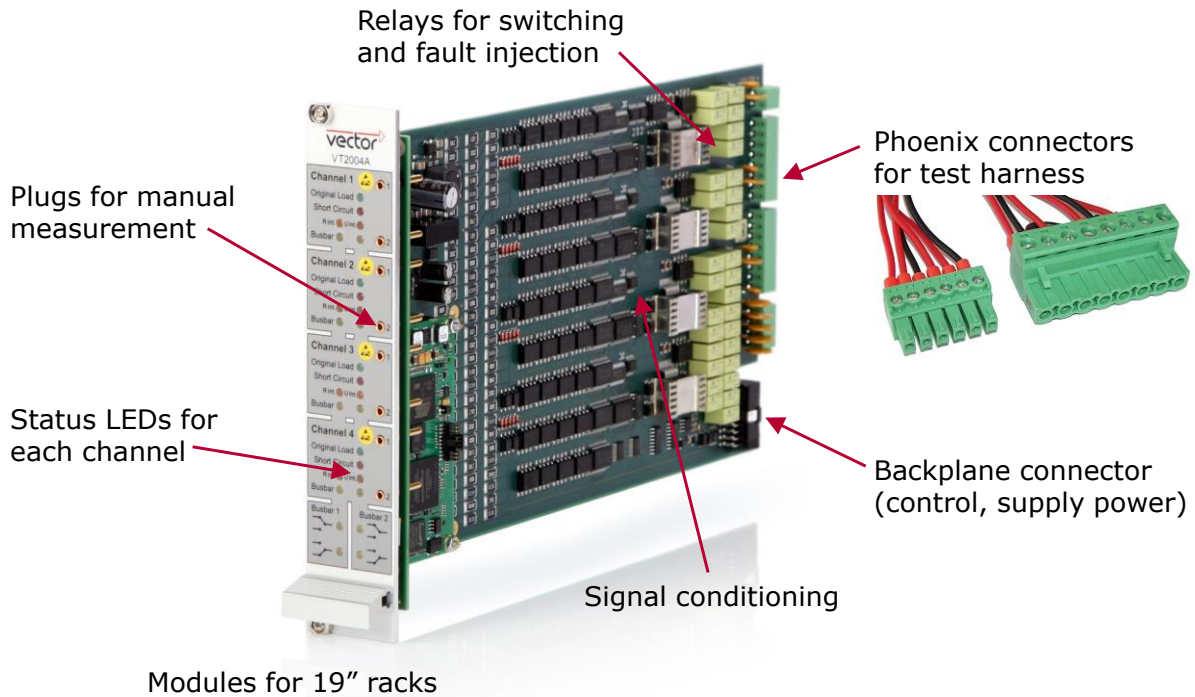


Figure 18: Structure of a VT System module. The marked components are common between different module types.

Depending on their functionality and the covered use case, different modules can be categorized as modules that simulate environment, modules that can be used for measurement purposes, and modules to induce fault injection:

- > Environment simulation:
 - Simulation of loads, e.g. by electronic load
 - Sensor simulation by decade resistor, PWM, or voltage signals
 - Relays to optionally switch to original loads and sensors
- > Measurement:
 - Measurement of ECU output voltage, PWM, high/low levels, etc.
 - Current measurement of supply voltage in a wide range, e.g. to detect power saving modes
- > Fault injection:
 - Short circuits between signal wires or ECU pin to ground/Vbatt
 - Line interruption (open load/broken sensor)
 - Simulation of faulty loads and sensor values
 - Switch of the bus termination resistance

In addition to the off-the-shelf VT System modules, there are two ways to extend VT System functionality which can be used individually or combined:

- > Customer-specific electronics can be attached to a special VT System module, which makes them visible and useable in the CANoe environment as if they were a standard module. The required workflow is highly standardized and prepared ready-to-use.
- > Customer-specific functionality can be executed locally on a number of modules in a user programmable FPGA. The execution of these programs generates no extra load to the CANoe executor and has full-speed access to the module-specific electronics. [User-specific VT System functionality](#) can be coded natively in VHDL or modeled with Matlab/Simulink and compiled to the user FPGA.

More details on VT System modules can be viewed using an interactive graphic [here](#).

6 Additional Resources

Further information, success stories, demos, case studies, and technical descriptions on Vector Testing Solution can be found [here](#). General information on Vector Testing Solution can be viewed [here](#).

7 Contacts

For a full list of all Vector locations and addresses worldwide, please visit <http://vector.com/contact/>.