Test Automation Software

- AutomationDesk
- Real-Time Testing and Real-Time Testing Observer Library
- Platform API Package and Failure Simulation Package
AutomationDesk

Powerful test authoring and automation tool

Highlights

- Graphical description of test routines
- Signal-based test authoring
- Advanced custom library concept
- Automatic report generation
- Offline test execution and development
- Certified for ISO 26262 and IEC 61508
- ASAM XIL-compliant access to simulation models and failure insertion units

Application Areas

AutomationDesk is a powerful test authoring and automation tool for testing electronic control units (ECUs). AutomationDesk users can create and edit test routines in a graphical format without requiring programming skills. AutomationDesk’s interface libraries, compliant with the ASAM standards, allow for a seamless reuse of automated tests across different development stages, such as MIL and SIL simulation with VEOS, dSPACE’s platform for PC-based simulation, and HIL simulation with SCALEXIO or any third-party simulation platform that provides a XIL API-compliant interface.

Key Benefits

AutomationDesk provides libraries containing a large number of predefined test steps, e.g., for easy access to the simulation model, a failure insertion unit (FIU), or calibration or diagnostics software. With AutomationDesk, tests can be executed 24 hours a day, seven days a week, letting engineers increase test coverage and improve ECU software quality while saving time and costs.

AutomationDesk lets you describe test routines graphically, be it test-step-based test authoring based on library elements, or signal-based test authoring based on stimulus signals and evaluation of captured signals.

AutomationDesk’s library concept, including the Framework Builder and Test Builder, helps develop and maintain templates for the test implementation – independent from the current tests.

AutomationDesk has been certified by TÜV SÜD for testing safety-related systems according to ISO 26262 and IEC 61508. The TÜV SÜD certificate confirms AutomationDesk’s suitability for developing and testing safety-related systems in automotives, commercial vehicles, aerospace and many other industries.

Functionality Overview

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Sequence Builder</td>
<td>- Graphical test development</td>
</tr>
<tr>
<td></td>
<td>- Library-based test development via drag &amp; drop</td>
</tr>
<tr>
<td></td>
<td>- Access to simulation platforms and tools from within the automation sequences</td>
</tr>
<tr>
<td>Test project management and execution</td>
<td>- Structured project overview</td>
</tr>
<tr>
<td></td>
<td>- Navigation in complex projects</td>
</tr>
<tr>
<td></td>
<td>- Managing test projects, data, results and reports</td>
</tr>
<tr>
<td></td>
<td>- Configuring test executions</td>
</tr>
<tr>
<td></td>
<td>- Debugging graphical tests</td>
</tr>
<tr>
<td></td>
<td>- Consistency checks before test execution</td>
</tr>
<tr>
<td>Functionality</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Automation libraries                | - ASAM XIL API library to access simulation platforms and failure insertion units of HiL simulators compliant to the ASAM XIL standard  
- Access to calibration tools like ControlDesk, ETAS INCA and Vector CANape via the ASAM MCD-3 MC automation interface  
- Access to diagnostics tools like ControlDesk via ASAM MCD-3 D  
- Access to MATLAB®  
- Convenient access to tools in the dSPACE tool chain  
  - ControlDesk  
  - Real-Time Test Manager  
  - ModelDesk  
  - MotionDesk |
| Open architecture                   | - Open COM API enables remote execution of tests, e.g., by test management tools  
- Edit and execute Python code or call Python scripts in AutomationDesk  
- Introduce and maintain custom libraries to develop and maintain reusable, test-domain specific blocks  
- Robust implementation of interfaces to third-party tools via COM/DCOM supported by VirtualCom feature  
- XML import/export of automation artifacts in projects and libraries |
| Test documentation                  | - Automated generation of test reports in PDF or HTML  
- Report library to define the contents of the generated report |
| Framework Builder library           | - Library that allows building up highly customizable templates as a test framework  
- Build up own templates (e.g., for test steps and tests) like in the Test Builder library  
- Templates feature implicit functions: e.g., for reporting and exception handling  
- Templates can be maintained with the AutomationDesk library mechanism |
| Evaluation library                  | - Powerful evaluation of measured signals  
- Automatic report generation, including plotted signals |
| Signal-based testing                | - Graphical description of signals for stimuli and evaluation  
- Intuitive test authoring as on a piece of paper |
| XIL API MAPort support              | - Platform-independent handling of simulation applications  
- Seamless use of simulation platforms from different vendors  
- Access to the simulation model for reading, writing, capturing, and stimulating variables  
- Stimulating time intervals of measured signals |
| XIL API EESPort support             | - Hardware-independent function blocks for Failure Insertion Units (FIU)  
- Convenient blocks for the most common FIU test scenarios  
- Base library enables using the whole range of supported EESPort functions  
- Access to dSPACE FIU systems, independently from ControlDesk  
- Advanced user interface for the error configuration in XIL EESPort Library blocks with lists for pin configurations, including validity checks  
- The stimulus duration in data file segments can be related to model variable states for signal-based tests and XIL MAPort stimuli |
| MotionDesk library                  | - Library to automate MotionDesk  
- Easy implementation of tests where online animation is involved, e.g., tests for camera based ADAS systems |
| Variable mapping                    | - Mapping Editor  
  - Clear allocation of variables, signals, parameters and variants  
  - Import/export of ASAM XIL mapping XMLs  
  - Create and manage mappings in AutomationDesk  
- Mapping Viewer  
  - Overview window with information on current variable mapping  
  - Filter and search functionalities allow for a quick localization of variables for the test description  
  - User interface for an interactive configuration of automation blocks via drag & drop  
  - Exchangeable mapping definitions compliant with ASAM XIL |
| Watcher-based switching of error sets | - XIL EESPort software trigger blocks enable to switch error sets according to model variable states |
Certified for Safety-Related Applications According to ISO 26262 and IEC 61508

AutomationDesk 5.0, like its predecessor AutomationDesk 4.1, has been certified by TÜV SÜD for testing safety-related systems according to ISO 26262 and IEC 61508. The TÜV SÜD certificate confirms AutomationDesk’s suitability for developing and testing safety-related systems in automotives, commercial vehicles, aerospace and many other industries. It also confirms the high quality of the development process and extensive quality assurance performed for AutomationDesk 5.0.

AutomationDesk is the first hardware-in-the-loop test automation software product to be awarded such a certificate. Together with the AutomationDesk Safety Manual, the certificate gives users a valuable support for validating and qualifying safety-related systems according to ISO 26262 and IEC 61508, and for using AutomationDesk 5.0 for developing and testing them.

dSPACE Process Consulting
dSPACE Process Consulting offers comprehensive support for using dSPACE software tools in safety-critical projects. If you require a proof of suitability for your tools, dSPACE can assist you with the necessary know-how. If you need ISO 26262 certification for the software you are using, including customer-specific processes, you can rely on the services of dSPACE Process Consulting. The services range from identifying the processes required for using one individual tool to qualifying complete tool chains for highly safety-critical projects.

For more information, please see www.dspace.com/go/consulting.

IEC 61508 and ISO 26262
IEC 61508 is the internationally recognized generic standard for the development of safety-related electronic systems. ISO 26262, the standard for functional safety in passenger vehicles, replaces IEC 61508 for the automotive industry.

Order Information

<table>
<thead>
<tr>
<th>Product</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutomationDesk Basic</td>
<td>AUD_BASIC</td>
</tr>
<tr>
<td>AutomationDesk – Automation Server</td>
<td>AUS_BASIC</td>
</tr>
</tbody>
</table>
Signal-Based Test Authoring

- Intuitive test authoring as on a piece of paper
- Drag & drop to define which stimulation signals and capture signals to evaluate
- One block solution with implicit evaluation and descriptive reporting

- Wide variety of evaluation methods
- "InsideRegion" evaluation functionality: allows time- and amplitude-tolerant evaluations
- Tests are executed in real time

Signal-based test authoring lets you describe tests intuitively as if on a piece of paper. You can use a set of standard signals to define real-time stimulation and evaluation rules for captured signals. Together with the 'inside region' evaluation function, which allows time- and amplitude-tolerant signal evaluations, a powerful supplement to the given test-step-based test authoring is available.

Segment-wise stimuli or evaluation rules can be combined. Conditions for a transition between signal segments as real-time observers can be defined to check preconditions for following segments or as a dynamic wait function for synchronizing segments or for passed/failed evaluations.
NEW: XIL Framework Support for Convenient and Safe Test Automation

The Mapping Editor: Assign variables of a simulation model (left) to an alias in an editable mapping table (right). Inconsistencies are highlighted.

- New environment for defining test bench configurations, including validity checks and highlighting of inconsistencies
- Convenient definition and validation of test-bench-specific configurations independently of test configurations
- Initialization of XIL ports and XIL mappings in one step
- The configuration of tests with lists of only valid XIL framework parameters makes incorrect entries impossible

User Interface of AutomationDesk

Ribbons
Work-phase-related sets of differently sized tool bars.

Start Page
Start page with links to valuable information and often-used commands.

Output
Colors and hyperlinks in the Output Viewer improve readability of messages.

Favorites
Define your view of important blocks in a separate view.
Test Automation Software Overview

<table>
<thead>
<tr>
<th>AutomationDesk</th>
<th>AutomationDesk Basic Version</th>
<th>AutomationDesk – Automation Server Basic Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform support</td>
<td>Platform API Package</td>
<td></td>
</tr>
<tr>
<td>Additional functionality</td>
<td>Failure Simulation Package</td>
<td>Real-Time Testing</td>
</tr>
<tr>
<td></td>
<td>Real-Time Testing Observer Library</td>
<td></td>
</tr>
</tbody>
</table>

The modular packaging of dSPACE’s Test Automation Software lets you select the packages you need for your automation tasks, and add other modules later on if necessary.

**AutomationDesk Basic (p. 2)**
- Full version of AutomationDesk

**AutomationDesk – Automation Server (p. 19)**
A tool variant of AutomationDesk (without a graphical user interface) for executing implemented AutomationDesk tests:
- Parameterization and execution of existing tests
- For test management applications such as SYNECT Test Management, batch scripts in Python, and customized operator UI applications

**Platform API Package (p. 30)**
- XIL API MAPort support in .NET (successor of the HIL API): ASAM XIL API-compliant interface to all dSPACE simulation platforms
- Platform Management API: Interface for configuring dSPACE platforms

**Failure Simulation Package (p. 32)**
- XIL API EESPort in .NET: Programming interface to all dSPACE and third-party failure insertion units in compliance with the ASAM XIL API EESPort standard
- ASAM XIL EESPort-compliant access via graphical user interface in ControlDesk

**Real-Time Testing (p. 20)**
- Python-based real-time test automation
- 100% reproducible real-time tests executed synchronously with the simulation model
- Easy integration into user-defined test frameworks (AutomationDesk or others)
- Support of VEOS, dSPACE’s platform for PC-based offline simulation

**Real-Time Testing Observer Library (p. 27)**
- Real-time validation for supplementing classic test methods
- Greater test depth for safety-critical applications
- Optimal integration and cross-platform use
Graphical Setup of Test Sequences

Test-Step-Based Test Authoring
AutomationDesk's Sequence Builder provides a graphical, UML-compliant environment for developing automation sequences. The Sequence Builder and its overview panel even help you keep track of very large sequence structures. AutomationDesk combines and integrates graphical and Python-based test development. Typically, graphical development is used to describe the control flow and to access devices such as hardware or other software. Python scripts are used for implementing algorithms or for user-specific extensions.

Reusing Automation Tasks
If a test sequence or single test steps need to be reused for similar automation tasks in other projects or sequences, they can be stored in custom libraries. Custom libraries gradually grow to contain more and more reusable elements, so you benefit from accumulated know-how in future projects. This method speeds up the test development process and increases efficiency. Different custom libraries can be shared between various AutomationDesk installations via the file system or via a version control system.
Advanced Library Concept
The Test Builder is a ready-to-use library to comfortably set up test frameworks. It offers a ready-to-use test framework with test and test step templates, including exception/verdict handling and reporting. The Test Builder elements are implemented through the Framework Builder library.

- Ready-to-use test framework
- Test and test step templates
- Implicit reporting
- Standard HTML and PDF style sheets for detailed and brief reports
- Exception and verdict handling
- Based on the Framework Builder
- Customizable and transparent
Test Debugging and Data Comparison

Debugging Test Sequences
If an error occurs during test execution, AutomationDesk’s Debugger makes it easy to find its source. You can insert breakpoints into your test sequence and make the test execution stop automatically at these points in debug mode. You can then go through the test case step by step, analyze complex hierarchical test steps in depth or execute them as a whole. The current variable values are always visible and can be changed during the debugging process to ensure that the tests run with correct variables. Step-by-step execution supports fast, efficient error-finding because you can investigate specific potential error sources.

Disabling Tests and Test Steps
AutomationDesk makes it considerably easier to verify test sequences while they are being developed. You can disable single test steps or sequences by commenting them out. This is extremely helpful when identifying errors.

The toolbar of AutomationDesk’s Debugger.

Disabling test steps.
Evaluating Data with the Evaluation Library

With AutomationDesk’s Evaluation library, you can analyze recorded data and compare it with reference data. This helps you to find values that exceed or are below the defined limits. For example, you can specify upper and lower bounds to evaluate whether a specific signal is above or below them, outside a gap, or inside a corridor. The signal, the reference signal, the bounds, and the evaluation result are plotted and reported automatically.

Many other kinds of signal manipulation are also possible:

- Binary operations (addition, multiplication, less than, etc.)
- Calculating gradients and integrals
- Shifting, cropping, and resampling signals
- Calculating the minimum value, mean value, minimum delta x, etc.

The recorded data can be compared with the reference data.
Test Project Management in AutomationDesk

Managing Project Data
AutomationDesk’s Project Manager is the ideal way to organize the sequences, data, and results of test projects. Tests are reproducible so they can be executed repeatedly to perform regression testing, for example. Tests can be arranged according to the functions under test, the ECUs in a network, the development stages, and the users involved in a project.

Offline Test Execution & Development
AutomationDesk’s libraries can be set to offline mode, so tests that use them are executable even if the required external software or hardware is not available. The test steps then use default behavior to enable test execution. For example, tests can be executed when no hardware-in-the-loop simulator is connected to AutomationDesk. You can define the return values for the default behavior yourself, or record them in an online test execution run.

Automatic Report Generation
AutomationDesk can automatically generate report documentation based on the test results. You can select from a library of report test steps (such as AddText, AddTable, or AddPicture) to specify the report’s contents. The report begins with statistics on all the test results. You can produce a detailed report, or a brief report that helps you judge the current quality of an ECU at a glance. Additionally, there are various options for controlling the level of detail and the format of the reports. Reports are usually generated in HTML or PDF format, but almost any kind of output format can be generated with user-defined XML style sheets.

Version Control of Projects and Custom Libraries
With the AutomationDesk version control interface, you can integrate version control systems that have the Microsoft® Source Code Control (SCC) interface, such as PTC® Integrity, Apache Subversion (SVN)\(^1\), or IBM® Rational® ClearCase®, etc.

\(^1\) Requires an installation of the SCC API plug-in from PushOk Software.
Automated HIL Testing, Diagnostics & Calibration from a Single Source

- ASAM-compliant access for dSPACE and third-party tools
- No need to update test sequences during tool exchange
- Calibration and diagnostic functions based on ControlDesk tool automation
- Powerful and easy to use
- Convenient access to fault memory: e.g., Direct Read DTCs, Direct Clear DTC, Direct Clear All DTCs

Diagnostics and hardware-in-the-loop (HIL) testing are brought together in AutomationDesk. You can use it to remote-control and automate the diagnostics functionality of ControlDesk, dSPACE’s universal experiment software. This means you have a completely integrated tool chain for HIL testing, ECU calibration, measurement tasks, and diagnostics from a single source. Compatibility problems are a thing of the past.

AutomationDesk can be used to automate and remote-control ControlDesk’s ODX-based diagnostic functionality, and to automate measurement and calibration tasks. It accesses ControlDesk via the ASAM MCD-3 automation interface or, far more conveniently, via ControlDesk’s tool automation library. You can build test sequences graphically in the familiar way in AutomationDesk, using the Remote Diagnostics (COM) library or the Remote Calibration (COM) library. These are AutomationDesk libraries of test steps for automating access to ControlDesk’s diagnostics, measurement, and calibration functionalities.

Successfully integrating ECU diagnostics into the hardware-in-the-loop (HIL) testing of ECU software is becoming more and more important. AutomationDesk is a standard solution for automated HIL testing that can address a variety of software tools. These include ControlDesk from dSPACE and DTS from Softing (standard AutomationDesk), but also measurement and calibration tools which comply with the ASAM MCD-3 MC standard, like ETAS INCA or Vector CANape.

All Tools from One Source

By using ControlDesk and AutomationDesk, you avoid the compatibility problems that often arise when tools from different vendors have to be integrated. dSPACE offers you HIL systems plus measurement, calibration, and diagnostic tools from a single source, with no complications. The smooth interaction between AutomationDesk and ControlDesk gives you maximum convenience and functionality for successful HIL testing.

1) It is also possible to access dSPACE hardware (requires the ControlDesk standard package, please see product information “ControlDesk”).
Test Automation

Well-Planned Libraries for Reusability
To run HIL test systems efficiently in terms of time and cost, test automation has to be planned carefully. Usually thousands of relevant tests have to be defined and handled. The objective is to create basic test steps, save them to libraries and reuse them for other tests. This is achieved by generic, nonredundant basic test steps, combined with a suitable library concept. These can be reused – for example, for subsequent vehicle generations – with only a few modifications.

Building a Good Library Concept for Test Automation
dSPACE Engineering Services has years of experience in creating library concepts. So when you build your library concept, you can call on our Engineering Services to help you get started in test automation. The dSPACE engineers will help you set up a test automation infrastructure, including the library concept. You will benefit from their knowledge of the hardware and software and from their accumulated project experience. The services include:
- Defining test processes
- Implementing projects
- Resident engineers
- Creating the test template
- Creating the library concept
- Providing exemplary test implementations
- Managing data and results
- Integrating third-party software
- Connecting the test software to existing tools

Working without a library concept, gives you quick results, but is not efficient enough in the long run. The initial investments for a good library concept pays for itself in terms of efficiency and cost-savings.
Tool Interfaces

Support of the XIL API Standard

**Independent of Specific Simulation Hardware**

The ASAM XIL API V2.0 standard is the next generation of the ASAM HIL API standard. The XIL Model Access port (MAPort) supports test bench access at all stages of the function development process: MIL (model-in-the-loop), SIL (software-in-the-loop), PIL (processor-in-the-loop) and HIL (hardware-in-the-loop) simulation. The XIL Electrical Error Simulation port (EESPort) in ASAM XIL controls electrical error simulation hardware. It lets you set different types of errors.

The XIL API standardizes the interface between test hardware and software. It lets developers reuse test cases and decouples test automation software from test hardware.

You can:
- Reuse automated tests on test systems from different vendors
- Reduce training costs for employees
- Improve the transfer of know-how from one test bench to another

**Advantages for Users**

The Platform API Package for AutomationDesk supports the Version 2.0 of the XIL API standard. The dSPACE test automation software makes it easier to use tests with any HIL simulator. This increases test reuse, helping to protect investments and reduce development costs and time. Moreover, you only have to buy one program and learn how to operate it – even if you want to access different simulators.

**XIL API Support in Platform API Package**

The Platform API Package supports the MAPort implementation of the XIL API standard to access a simulation platform:
- Configure the simulation platform
- Read and write to scalar and vectorized variables
- Capture data by using complex trigger conditions
- Stimulate variables of a real-time application via a signal description set, such as the Signal Editor Module in ControlDesk

**XIL API Support in Failure Simulation Package**

- XIL API EESPort (Electrical Error Simulation Port) implemented in .NET, in compliance with the ASAM XIL API EESPort standard
- Access to all dSPACE Failure Insertion Units in automation scenarios from your own test scripts and applications
- FIU tracing, i.e., monitoring of error set switch demands via real-time variable to allow cause-and-effect analysis in FIU tests
- Watcher-based switching of error sets according to model variable states

**XIL API Vendor Switch**

- Easy switching between vendor-specific XIL API implementations
- Convenient exchange of object models from different vendors

**XIL API Convenience Library**

- Quick, convenient XIL-API-based operations – detailed XIL API background information not required
- Easy handling of read/write/capture/signal generator actions
- Easy to use blocks for failure simulation control
Interfaces to dSPACE Tools

**ControlDesk**
Convenient, comprehensive access to all ControlDesk features for:
- Measurement and calibration
- Diagnostics

**Real-Time Testing**
- Real-Time Testing library for convenient script management
- Easy integration into AutomationDesk tests

**MotionDesk**
- Remote-control MotionDesk
- Support of camera-based ADAS tests

**ModelDesk**
- ModelDesk tool automation as preparation for ADAS tests
- Remote control of ModelDesk to handle parameters, roads, maneuvers and traffic
- Open project/experiment
- Activate parameter set
- Access to parameter handling
- Modify parameters
- Download parameters
- Access to road and maneuver handling
- Activate and download roads
- Activate and download maneuvers
- Modify road and maneuver parameters (surface conditions, velocities)

Interfaces to Third-Party Tools

AutomationDesk can also integrate third-party tools, such as CANscope, CANstress, CANoe, and CANalyzer from Vector. Please contact dSPACE for details on supported third-party tools. dSPACE also has experience in connecting AutomationDesk to third-party hardware-in-the-loop systems, provided the systems support the XIL API standard properly, and offers engineering support on request. Implementation examples (e.g., VISA Access Library, AutomationDesk CANoe Library) can be found in the Test Automation Software Support Center. The entry gate is www.dspace.com/go/pscta.

**ASAM interface**
- Access to calibration tools like ControlDesk, ETAS INCA, and Vector CANape, via ASAM MCD-3 MC
- Access to diagnostics tools like ControlDesk via ASAM MCD-3 D
- Access to simulation platforms via ASAM XIL MAPort
- Access to failure simulation systems via ASAM XIL EESPort

**MATLAB**
- Access to the MATLAB Command Window
- Data exchange between AutomationDesk and MATLAB
- Remote execution of MATLAB commands
- Use of M files and MAT files
Integration of AutomationDesk in Typical Tool Chains

Connection to dSPACE SYNECT

SYNECT is a data management and collaboration software that focuses on model-based development and ECU testing. Designed to help you manage data throughout the entire development process (models, signals, parameters, tests, test results, etc.) it also handles data dependencies, versions and variants, as well as links to the underlying requirements. SYNECT Test Management is a SYNECT module which gives you a convenient connection to AutomationDesk.

Off-the-Shelf Integration with AutomationDesk

You can use implemented test cases in AutomationDesk projects or libraries as a source for test cases in SYNECT Test Management.

Importing Requirements, Specifications, and Tests

You can import existing tests into SYNECT Test Management to define your tests and collect source data from several tools:
- Test lists from Microsoft® Excel®
- Test implementations from AutomationDesk, BTC EmbeddedTester®, MATLAB®, or Python

Test Planning and Execution

SYNECT Test Management easily integrates with test automation tools such as AutomationDesk from dSPACE and BTC EmbeddedTester®. It does this via adapters that control the test automation tool. To carry out tests, you can set up execution plans containing test cases and add tests to them. An execution plan is a list of tests that have to be executed. It can be based on various criteria: For example, it can include all your tests for one variant or for one ECU function.
Remote-Controlled Test Runs

COM Interface
AutomationDesk has a COM-based application programming interface (API) for remote-controlling and automating selected AutomationDesk functions. These are some of its typical use cases:

- Programming batch processes (with Python, Visual Basic, C++, etc.): You can program flexible control structures which skip single test sequences if necessary or execute test projects successively.

- Designing interactive user interfaces: You can create your own test UIs to reduce complexity, for example.

- Connecting other tools to AutomationDesk: For example, parameterize your test data in Microsoft® Excel® and call the AutomationDesk test sequences from Excel.

Example of a client application for test execution via the COM interface.

XML Test Exchange Format
AutomationDesk supports the import and export of projects, folders, tests, and custom libraries as XML files. Using XML to import and export test descriptions means that tests can be exchanged between different test tools. For example, to be imported from a third-party tool or from a proprietary test description, a test simply needs to be converted into AutomationDesk’s XML format.

Use Cases
XML files open up numerous new application options, such as:

- Reading and reviewing tests without AutomationDesk
- Converting tests into different file formats
- Migrating tests that were created with proprietary tools into AutomationDesk
- Generating tests from third-party tools, such as Microsoft® Excel®, and integrating them into AutomationDesk
AutomationDesk – Automation Server

AutomationDesk – Automation Server is a server variant without a graphical user interface. The Automation Server is accessed and remote-controlled via the COM/DCOM interface, so the same functions are available as for the full version of AutomationDesk. This run-time version has the advantage of reduced license costs, making it useful wherever the comprehensive functions of the full version are not required, for example, in automated test execution on hardware-in-the-loop simulators.

<table>
<thead>
<tr>
<th>AutomationDesk – Basic Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ Development environment for tests</td>
</tr>
<tr>
<td>■ Developing libraries</td>
</tr>
<tr>
<td>■ Test execution</td>
</tr>
<tr>
<td>■ Note: To access dSPACE platforms, you need the Platform API Package.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AutomationDesk – Automation Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ Tool variant without graphical user interface</td>
</tr>
<tr>
<td>■ Planning, parameterizing and executing existing tests</td>
</tr>
<tr>
<td>■ Client applications: batch scripts, Excel, operator UI, test management tools like SYNECT Test Management, etc.</td>
</tr>
</tbody>
</table>
Real-Time Testing

Python-based real-time test automation

Highlights

- 100% reproducible real-time tests executed synchronously with simulation model
- Several independent test scripts can be executed concurrently
- No model modification necessary for real-time testing
- Easy integration into user-defined test frameworks (AutomationDesk or others)
- Support of all dSPACE platforms\(^1\) including VEOS, dSPACE’s platform for PC-based offline simulation

Key Benefits

Automated testing is usually performed by executing tests on a standard PC connected to the hardware-in-the-loop (HIL) system. However, this method often cannot cope where greater timing precision is required – for example, if ECU interaction has to be captured and responded to within just milliseconds. Real-Time Testing with its Python scripts for real-time testing is the answer. The scripts run on the simulation platform, e.g., a dSPACE HIL system. They run synchronously with the model, so all test actions are performed on a real-time basis – 100% reproducibly. This opens up expanded test options with dSPACE Simulator. Reactive tests which respond to changes in model variables within the same simulation step can be implemented. Time measurements in tests are also far more precise, as there are no latencies in communication. Simulation step size is now the only limit to the maximum time resolution of measurements. Real-time test scripts do not only run on HIL systems but also on VEOS, dSPACE’s platform for PC-based offline simulation. Therefore you can reuse the test scripts seamlessly throughout the entire development process – from virtual validation to HIL simulation.

Example Use Cases for Real-Time Testing

- Testing reactivity in a range of milliseconds
- Time-precise stimulation of several model signals including a replay of measured data
- Time-precise observation of model changes
- Reliable determination of minimum and maximum values of model variables
- Python-based restbus simulation
- Exact replay of recorded bus communication (for example, CAN data replay)
- Parallel execution of multiple, independent real-time tests

Real-time-capable Python interpreter for processor board and MicroAutoBox.

1) Except DS1104.
Functionality Overview

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
</table>
| Real-time testing                    | ■ Real-time test management via scripting and dedicated graphical user interface  
  ■ Real-Time Testing library in AutomationDesk for convenient script management  
  ■ Model variables can be observed and changed in every simulation step  
  ■ No model modification necessary for real-time testing |
| 100% reproducible real-time tests in Python | ■ Tests are executed synchronously with simulation model  
  ■ Real-time tests implemented in Python (user-extensible via libraries)  
  ■ Concurrent execution of several independent test scripts  
  ■ Dynamic test loading during model execution and test execution  
  ■ Read and write access to model variables in every simulation step |
| Key features                         | ■ Easy integration into user-defined test frameworks (AutomationDesk or others)  
  ■ Use of measured data such as MAT files located on host PC for real-time stimulation of model variables  
  ■ NEW: Data streaming now also supports ASAM MDF files (Version 4.x), the established ASAM measurement data format  
  ■ Python objects can be easily exchanged between real-time tests and the PC Python script  
  ■ Seamless support of all dSPACE platforms¹)  
  ■ NEW: Support of DS6001 Processor Board and simulation on a remote simulator with VEOS  
  ■ Virtual ECU (V-ECU) support  
  ■ Stimulation and monitoring of signals and variables in V-ECUs  
  ■ Sending and monitoring of CAN and CAN-FD frames (based on RTI CAN MultiMessage Blockset or Bus Manager) |

¹) Except DS1104.

Order Information

<table>
<thead>
<tr>
<th>Product</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-Time Testing</td>
<td>RTT</td>
</tr>
</tbody>
</table>

Relevant Software

<table>
<thead>
<tr>
<th>Software</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>VEOS</td>
<td></td>
</tr>
<tr>
<td>AutomationDesk Basic</td>
<td>AUD_BASIC</td>
</tr>
<tr>
<td>AutomationDesk – Automation Server</td>
<td>AUS_BASIC</td>
</tr>
<tr>
<td>Platform API Package</td>
<td>PLATFORM_API</td>
</tr>
<tr>
<td>Failure Simulation Package</td>
<td>FAILURE_SIM</td>
</tr>
<tr>
<td>Real-Time Testing Observer Library</td>
<td>RTT_OBSERVER_LIB</td>
</tr>
</tbody>
</table>
**Real-Time-Capable Python Interpreter**

A real-time Python interpreter, running on the processor board along with the model, allows the script to execute synchronously with the model. The interpreter can execute several real-time tests simultaneously and independently of one another. The tests can interact with the simulation model in real time via the memory on the processor board. You can therefore observe and influence the ECUs connected to the HIL simulator in each individual simulation step.

In the default configuration, the Python interpreter is activated on the platforms. You can formulate real-time tests using standard Python scripts from the Real-Time Testing libraries, e.g., for accessing model variables and executing several test branches within one real-time test in parallel. You can also create your own Python libraries and reuse them in several tests. The tests you have created can be loaded from the PC to the dSPACE simulation platform and executed regardless of whether another real-time test is already running. The tests run synchronously and in parallel to the engine and vehicle dynamics models, for example on a HIL system.

**Real-Time Observer**

A common test case is to detect an event generated by the real-time model. An event is specified by a condition referencing one or more real-time model variables. As soon as the event has been detected, a test action is triggered in the same sample step. In this example, the real-time script observes the speed of a simulated vehicle and reacts by performing an emergency stop maneuver immediately when the vehicle speed exceeds 80 km/h. This test case can be easily implemented by a real-time Python script. The condition is observed continuously within a While loop. The yield command pauses the test execution, which resumes in the next simulation step. This means the script can easily be time-controlled. When the speed exceeds 80 km/h, the While loop is left, and the brake maneuver is initiated by setting the associated model parameter in the same simulation step.

Managing real-time tests with the Real-Time Test Manager.
Implementing Real-Time Stimulus

In many test scenarios, recorded data for ECU stimulation has to be replayed with correct timing. For this, Python real-time scripts can perform precisely timed stimulus generation on predefined model variables. Another example is a sine generator that can be implemented by means of the standard sine functionality of Python’s math module (see scripting example below).

Recorded measurement data can also be replayed via an intelligent load mechanism, in which the Python real-time script references a recorded file (MAT file or ASAM MDF file (Version 4.x)) on the PC’s hard disk. The script links data vectors in the file to the target parameters in the model. For example, if the real-time test requires data to be replayed exactly 50 ms after a specific CAN message is received, a simple replay command in the Python script triggers the real-time-capable data transport from the PC to the real-time hardware so that the data is replayed in time. The automatic load mechanism supports the replay of large data volumes, such as test drive log files that are several hundred MB in size. Several replay processes can run simultaneously, and they can all be controlled independently of one another. In a multiprocessor system, it is even possible to stimulate on several subnodes time-synchronously.

Example of a real-time script that triggers a sine generation.
Implementing CAN Restbus Simulation

To program CAN and CAN FD restbus simulation, Real-Time Testing is conveniently integrated with the RTI CAN MultiMessage Blockset and the Bus Manager. The blockset contains an option for preparing CAN access for Real-Time Testing, so real-time scripts are able to send and receive CAN messages with freely definable CAN IDs and contents. For bus simulations configured by the Bus Manager, the real-time access is always provided. This makes it very easy to test case-specific via restbus simulation. These dynamic restbus parts (like ECU-specific CAN test messages) can be loaded and executed on the real-time hardware as required and no longer need to be a static part of the simulation model.

An example test scenario (see picture below) is the monitoring of an analog input signal. When a defined trigger threshold (e.g., 14.7 V) is exceeded, a predefined CAN message is sent cyclically every 50 ms until the value goes below the threshold again. Another application example is taking CAN communication that was recorded during a test drive and replaying it in the laboratory with correct timing.

A real-time script continuously evaluates a model variable and reacts by sending cyclic CAN TX messages.

1. Import Real-Time Testing libraries (RTI CAN MultiMessage Library, etc.).
2. Generate the variable object of the voltage to be monitored.
3. Select the CAN send controller.
4. Define the send message with the real-time test.
5. Help function for precise time measurement (for later transmission of CAN messages at exact time intervals).
6. Real-time test sequence: Check the voltage once per simulation step and react by sending a cyclic CAN message.

Implementation of a CAN test scenario.
Implementing Real-Time Tests for Multiprocessor Systems

Today, HIL integration tests are executed on multiprocessor (MP) systems where submodels for the engine, transmission, restbus simulation, etc., are distributed across several processor boards to ensure real-time capability. Real-Time Testing provides transparent variable access in the MP system so that real-time scripts can be implemented independently of the structure of the HIL model. The communication channels needed between the processor boards are set up dynamically by the Python test scripts during model run time, instead of being created statically in advance by the modeler. Local and remote variables can both be accessed during testing: Specifying the unique MP variable path is enough. This ensures that the real-time tests can be executed on different MP nodes without modifications. They are implemented once and run everywhere.

VEOS and Virtual ECU Support

Real-Time Testing (RTT) supports VEOS, dSPACE’s PC-based simulation platform, and can also be used in virtual ECUs (V-ECUs). With the help of RTT, VEOS users are able to monitor and stimulate signals and variables in V-ECUs time-synchronously. This feature comes as a useful supplement to VEOS’ numerous functionalities, which among others include early function tests and increased test capabilities compared to testing with real ECUs.

Real-Time Testing is also available for VEOS, dSPACE’s PC-based simulation platform.
Real-Time Test Library in AutomationDesk

AutomationDesk includes a block library for convenient handling of real-time test sequences. The library lets you automate your management of Real-Time Testing (RTT) tasks with the data objects and test steps of the AutomationDesk Real-Time Testing library. This is the key to integrating Real-Time Testing into your existing graphical test environment.
Real-Time Testing Observer Library

Monitor safety-critical requirements continuously in real time

Highlights
- Real-time validation for supplementing classic test methods
- Greater test depth for safety-critical applications
- Optimal integration and cross-platform use

Application Areas
Safety-critical applications are common in many kinds of fields, such as automotive, aerospace, medical engineering, and manufacturing. Even when complexity is high, the functionality needs to be guaranteed and formally validated as, for example, recommended by the ISO 26262 standard for testing the functional safety of road vehicles. By using the Real-Time Testing (RTT) Observer Library, you can use requirement observers that were created with BTC EmbeddedSpeciﬁer on dSPACE platforms to continuously monitor your requirements.

Key Benefits
By integrating the RTT Observer Library and BTC EmbeddedSpeciﬁer in the dSPACE tool chain, you can supplement any existing model-in-the-loop (MIL), software-in-the-loop (SIL) or hardware-in-the-loop (HIL) environment with observers to continuously validate safety-critical requirements in real time. This combination of test methods increases the achievable test depth enormously, within the same amount of time. The optimal integration into the dSPACE tool chain for HIL and virtual validation lets you work comfortably in a familiar environment.

Efficient Tool Chain
This solution for continuously monitoring safety-critical requirements uses a combination of various tools, which can vary from application to application. This involves any of the following tools (see also p. 29):
- **Real-Time Testing Observer Library** – Take requirement observers generated with BTC EmbeddedSpeciﬁer® and execute them on dSPACE platforms.
- **BTC EmbeddedSpeciﬁer** – Transfer informal requirements into a formal representation and then into requirement observers, even without expert know-how. This formal, tool-supported method also increases the quality of the requirements.

- **dSPACE platform** – SCALEXIO, DS1006-based HIL simulators, or VEOS – the right platform for each test phase.
- **ControlDesk** – The universal, modular experiment and instrumentation software. Requirement observer-speciﬁc layouts and the ability to load, start, and stop observers lets you monitor the state of each requirement throughout the entire execution. If any requirement is violated, this is shown directly in the layout.
- **AutomationDesk** – The powerful tool for test authoring and automation. The monitored requirements are checked in line with the deﬁned test workﬂow. The test reports also include the results from the observers, so it is easy to trace the reasons for any requirement violations.
Functionality Overview

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
</table>
| General       | - Execution of requirement observers, generated by BTC EmbeddedSpecifier, on dSPACE platforms  
                 - Continuous validation of requirements during test scenarios  
                 - Requirement observers can be added to the running model execution as independent routines. Recompilations of the model are not necessary.  
| Setup package (additionally provided with RTT Observer Library license) | - ControlDesk layout generator  
                 - AutomationDesk test template  
                 - BTC Observer Simulink® block  
                 - Product documentation  
                 - Demo files  
| Supported dSPACE platforms | - SCALEXIO, SCALEXIO MC/MP\(^1\)  
                 - DS1006, DS1006 MC/MP\(^1\)  
                 - VEOS, VEOS MC\(^1\) |

Order Information

<table>
<thead>
<tr>
<th>Product</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-Time Testing Observer Library</td>
<td>RTT_OBSERVER_LIB</td>
</tr>
</tbody>
</table>

Relevant Software and Hardware

<table>
<thead>
<tr>
<th>Software</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Included</td>
<td>Setup package (see table “Functionality Overview”)</td>
</tr>
<tr>
<td>Required</td>
<td>BTC EmbeddedSpecifier(^6)</td>
</tr>
<tr>
<td>Optional</td>
<td>ControlDesk</td>
</tr>
<tr>
<td></td>
<td>AutomationDesk</td>
</tr>
<tr>
<td></td>
<td>VEOS(^1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>SCALEXIO(^1)-based real-time simulation system</td>
</tr>
<tr>
<td></td>
<td>DS1006(^1)-based real-time simulation system</td>
</tr>
</tbody>
</table>

\(^1\) Multicore and multi-processing support limited to observers that operate on core-local variables.
Tool Chain Overview

Validating safety-critical requirements on a dSPACE platform permanently with generated, real-time-capable observers.

Example: Integration in ControlDesk

Full overview at a glance: Requirement observers can be started, stopped, loaded, and reset. Compliance with requirements (green), non-compliance (red), and not yet evaluated observers due to unmatched preconditions (grey) are visible immediately.
Platform API Package

Application programming interfaces for connecting to dSPACE platforms

Highlights

- Package containing automation libraries for accessing dSPACE systems
- High-level libraries for convenient platform management and simulation model access
- Access to dSPACE real-time platforms and dSPACE VEOS
- Support of Python or any .NET-compliant language
- Compliant to ASAM XIL API standard

Key Benefits

- The Platform API Package is a set of libraries to access and manage dSPACE real-time platforms and dSPACE VEOS. It provides a convenient way to download, start and stop models and also grants high-level access to the model variables for reading, writing, stimulating, capturing, etc.
- Create your own test scripts and test automation applications to access parameters and signals in your simulation
- Automation interface to MATLAB®, e.g., for data processing

XIL API support in .NET

- Programming interface that complies with the ASAM XIL API standard for accessing the simulation model
- MAPort support including read, write, and capture access plus stimulus support for most dSPACE platforms
- Port management: e.g., for loading applications and setting the simulation state
- XIL framework variable support, including unit conversion, supports advanced variable mapping features. This allows an improved decoupling of test scripts from simulation models, hence a better reuse of test scripts.

Platform Management API

- Programming interface for downloading, starting and stopping a model running on a dSPACE real-time platform or dSPACE VEOS

MATLABLIB2

- Remote-control MATLAB to use its powerful mathematical function, e.g., for signal processing
### Order Information

<table>
<thead>
<tr>
<th>Product</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform API Package</td>
<td>PLATFORM_API</td>
</tr>
</tbody>
</table>

### Relevant Software

<table>
<thead>
<tr>
<th>Software</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>AutomationDesk Basic</td>
<td>AUD_BASIC</td>
</tr>
<tr>
<td>AutomationDesk – Automation Server</td>
<td>AUS_BASIC</td>
</tr>
<tr>
<td>Failure Simulation Package</td>
<td>FAILURE_SIM</td>
</tr>
<tr>
<td>Real-Time Testing</td>
<td>RIT</td>
</tr>
</tbody>
</table>
Failure Simulation Package

Interface to dSPACE Failure Insertion Units

Highlights

- Automation libraries and graphical user interface for dSPACE Failure Insertion Units (FIUs)
- ASAM XIL API EESPort-compliant implementation
- ControlDesk front end for interactive FIU handling
- Easy-to-use AutomationDesk libraries
- Flexible integration into user-defined test frameworks based on Python, C# or any other .NET-compliant language

Key Benefits

The Failure Simulation Package gives you quick and easy access to ASAM XIL-compliant failure simulation units. It provides API access according to the ASAM XIL API Electrical Error Simulation port (EESPort) and uniform access to all dSPACE failure simulation systems. High-level libraries for convenient failure simulation access are available in .NET and can be used in Python, C#, and any other .NET-compliant programming language.

If you use dSPACE ControlDesk, the Failure Simulation Package offers an ASAM XIL API EESPort-compliant graphical user interface component that is available when the Failure Simulation Package is installed.

ASAM XIL-Compliant

With the Failure Simulation Package, the features of the ASAM XIL standard are available in dSPACE products.

The Failure Simulation Package user interface in ControlDesk.
**Module Overview**

**XIL API EESPort in .NET**
- Programming interface for accessing Failure Insertion Units
- Uniform support of all dSPACE Failure Insertion Units
- Automated configuration and switching of XIL API error configurations
- Monitoring of error set switch demands via real-time variable allows cause-and-effect analysis in Failure Insertion Unit tests
- ASAM XIL API EESPort-compliant software trigger support enables fast and precisely timed switching of error sets according to model variable states
**XIL API EESPort GUI in ControlDesk**

- Successor to ControlDesk’s Failure Simulation Module
- Interactive graphical configuration of error configurations and error sets compliant to ASAM AE XIL API standard (EESPort)
- Requires ControlDesk
- **NEW**: Grouping and filtering with meaningful icons in ControlDesk’s EESPort Configurations controlbar
- **NEW**: Lists of valid parameters and highlighting of invalid configurations in ControlDesk

The XIL API EESPort GUI not only provides for a convenient control of electrical failure insertion hardware, it also allows for an interactive graphical configuration of error sets and error configurations directly from ControlDesk, which acts as a XIL API EESPort client. Through compliance with the ASAM XIL API standard, this solution offers uniform support for all dSPACE SCALEXIO and PHS Failure Insertion Units (FIU).

Definition of various error sets with the XIL API EESPort user interface in ControlDesk.
**AutomationDesk Client**

- Convenient and easy-to-use AutomationDesk library blocks for FIU automation
- Uniform function blocks for any FIU that complies with the ASAM XIL standard
- **NEW**: User interface for configuration in AutomationDesk with lists of valid parameters and validation checks

**Order Information**

<table>
<thead>
<tr>
<th>Product</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Simulation Package</td>
<td>FAILURE_SIM</td>
</tr>
</tbody>
</table>

**Relevant Software**

<table>
<thead>
<tr>
<th>Software</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>AutomationDesk</td>
<td>AUD_BASIC</td>
</tr>
<tr>
<td>ControlDesk</td>
<td>Please see the ControlDesk product information.</td>
</tr>
<tr>
<td>Platform API Package</td>
<td>PLATFORM_API</td>
</tr>
</tbody>
</table>