

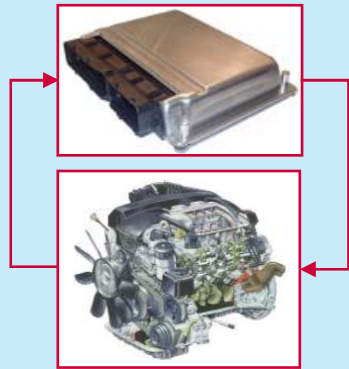
AutomationDesk – Key Features for successful  
ECU Testing



**Dr. rer. nat. Sven Burmester · Product Engineer Test and Experiment Software**  
**dSPACE GmbH · Technologiepark 25 · 33100 Paderborn**  
**automotive testing expo · 8th of may 2008**

- Introduction to ECU Testing & Hardware-in-the-Loop Simulation
- Application Examples & Benefits of Test Automation
- Key Features to maximize Benefit

ECU (Electronic Control Unit)  
controls plant

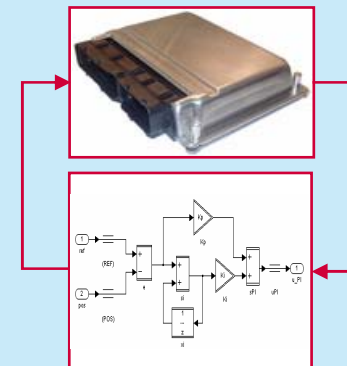


Plant (e.g. Engine)

## How to test ECU without plant?

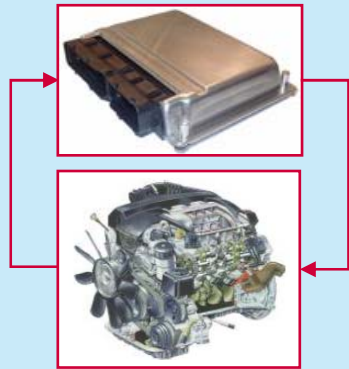
- e.g. to test ECU before finishing development of plant prototype
- e.g. to protect plant against damage in case of ECU failures
- e.g. to avoid abrasion and consumption of resources (fuel)

ECU (Electronic Control Unit)  
controls plant



Plant Model

ECU (Electronic Control Unit)  
controls plant



Plant (e.g. Engine)

## How to test ECU without plant?

- e.g. to test ECU before finishing development of plant prototype
- e.g. to protect plant against damage in case of ECU failures
- e.g. to avoid abrasion and consumption of resources (fuel)

## Hardware-in-the-Loop (HIL) Simulation!

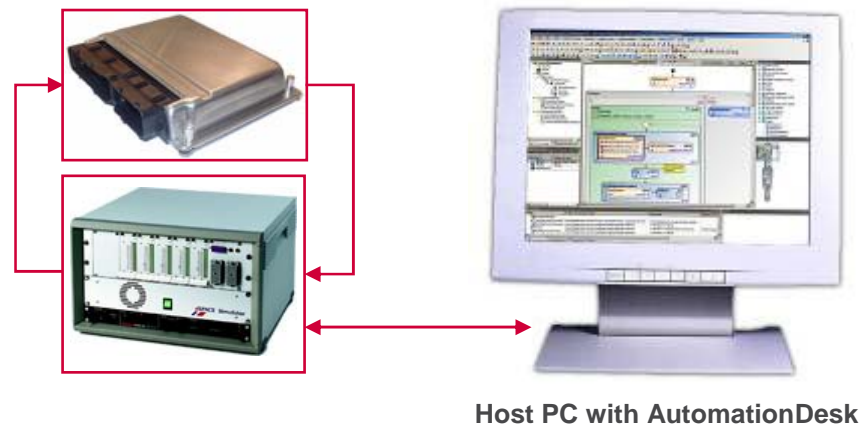
(Simulation of plant model in real-time)

ECU (Electronic Control Unit)  
controls plant

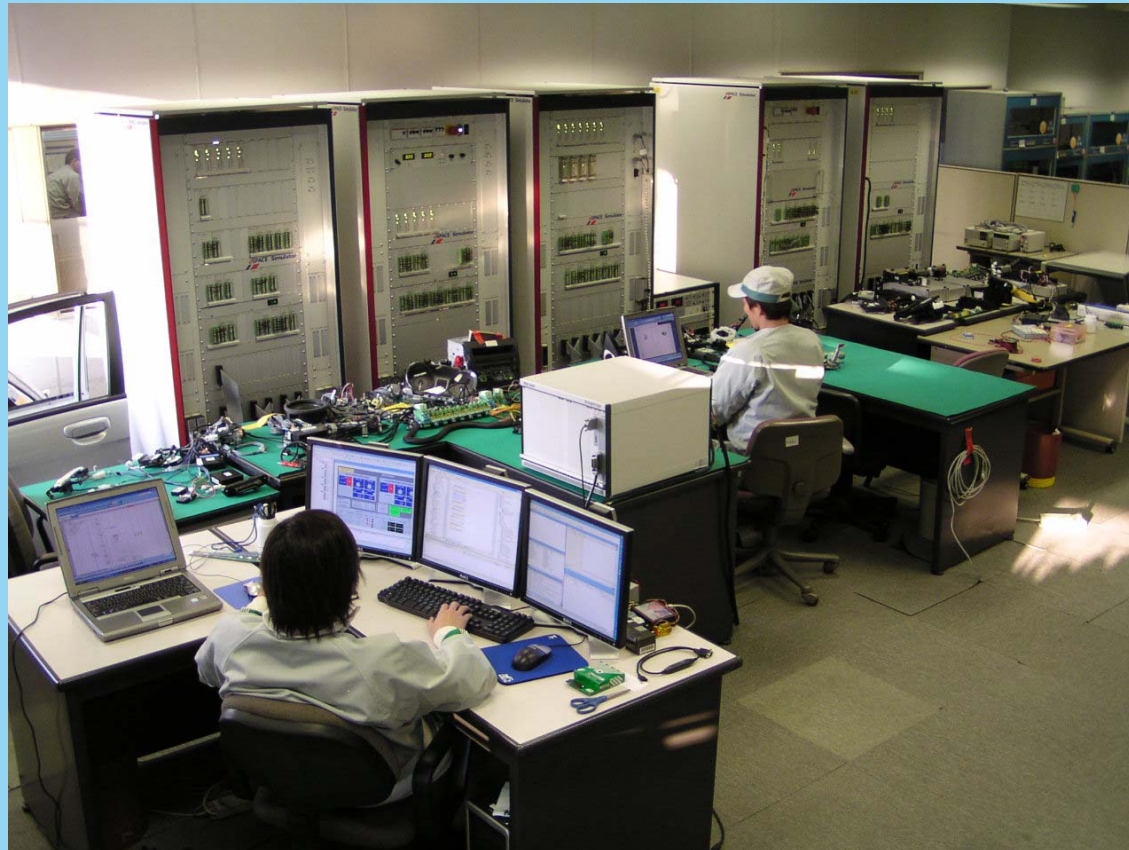


HIL Simulator

- Describing test maneuvers to test ECU
- Test maneuvers accessing HIL Simulator
- Driving test maneuvers automatically
- Automatic evaluation and reporting of test results



**First Virtual Vehicle in Japan!**



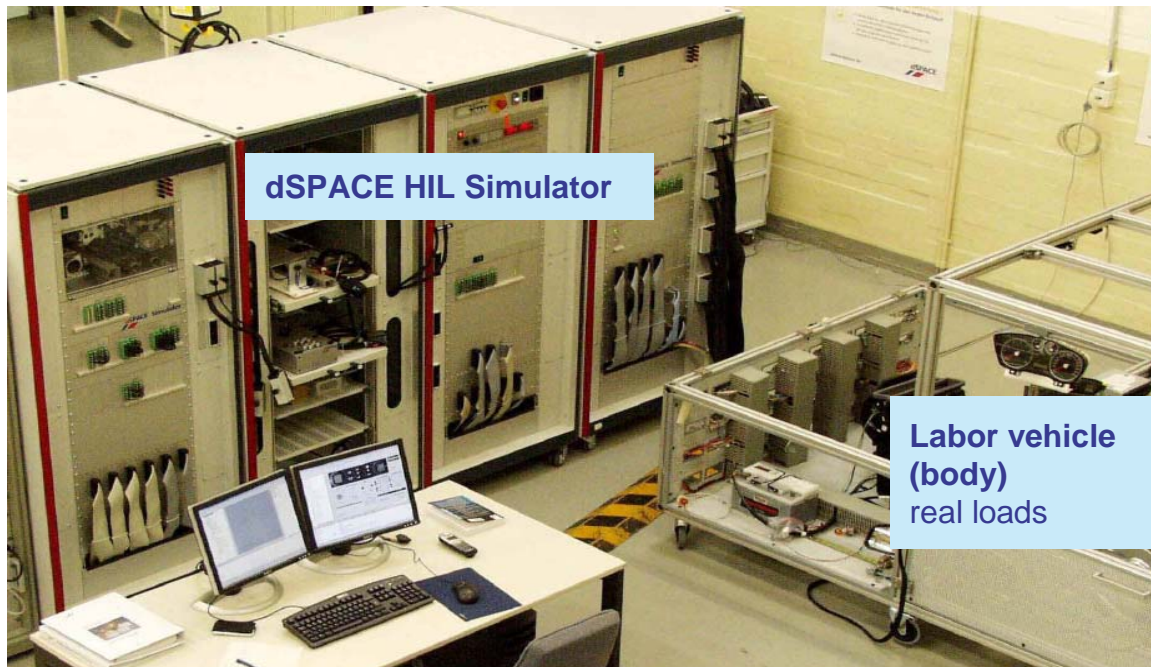
**Integration HIL**

- 5 Full-Size-Racks
- > 20 ECUs
- Body electronics & Power train
- Short start-up period
- More information: dSPACE News 3 / 2007

*“AutomationDesk is easy to use on base of Libraries.”*







dSPACE HIL Simulator

Labor vehicle (body) real loads

### HIL for Body Electronics, Power Train & Vehicle Dynamics

- > 50 ECUs
- > 10 000 test cases
- about 1800 CAN signals



Function	Number of test steps	Test duration (manually) [h]	Test duration (HIL) [h]	Improvement (factor)	Availability of test results
Door closure	937	80	10	8	1,5 days vs. 2 weeks
Window lifter	2612	100	66	1,5	2,5 days vs. 2,5 weeks
Exterior light	1300	80	5	16	1 day vs. 2 weeks
ESP	350	96	9	10,6	1,5 days vs. 2 weeks

*“Even detection of sporadic errors due to increase of number of test runs.”*

*“Reduction of complete E/E system and function testing time from 12 weeks to 1 week.”*

**HIL for body electronics, Power Train & Vehicle Dynamics**

- > 50 ECUs
- > 10 000 test cases
- about 1800 CAN signals



Function	Number of test steps	Test duration (manually) [h]	Test duration (HIL) [h]	Improvement (factor)	Availability of test results
Door closure	937	80	10	8	1,5 days vs. 2 weeks
Window lifter	2612	100	66	1,5	2,5 days vs. 2,5 weeks
Exterior light	1300	80	5	16	1 day vs. 2 weeks
ESP	350	96	9	10,6	1,5 days vs. 2 weeks



### Automated testing of steering systems

- ECU diagnostics with dSPACE CalDesk
- Coupling of AutomationDesk and Telelogic DOORS (requirements management system)

*“Coupling AutomationDesk and DOORS via dSPACE's Connect&Sync Module has greatly simplified ECU testing at ZF LS.”*

*“Visibility of constantly up-to-date test result at management level leads to high 'error remedying morale' of developers.”*



**Telelogic Technology Partner Agreement between dSPACE and Telelogic**

**How to achieve these benefits?**

**...by efficient development & execution of Tests!**

**→ Key Features for successful ECU Testing**

# Test Automation Software AutomationDesk



The screenshot displays the AutomationDesk software interface. The main window shows a sequence diagram for the 'ReadAndClearErrorMemory' sequence. The diagram starts with a 'SelectProject' block, followed by a 'TryExcept' block. The 'Try' block contains a 'Range' block with a list of ECUs, a 'ReadDiagnosticInformation' block with a 'SyncService' sub-block, an 'AddResultsToReport' block, a 'Clear' block, and a 'ReadDiagnosticInformation2' block with a 'SyncService' sub-block and an 'Evaluate' block. The 'Except' block contains a 'HandleException' block. The 'Data Object Editor' shows the 'RemoteDiagnostics.SynchronizeRead.ReadAndClearErrorMemory' sequence with data objects 'Service', 'Results', and 'OfflineResults'. The 'Found items' table lists the blocks and their paths.

Name	Project and Sequence Path	Block Path	Reference Name
RemoteDiagnostics	RemoteDiagnostics		
ClearDiagnosticInformation	RemoteDiagnostics.System.Project.VehicleInformation.LogicalLink.Services	RemoteDiagnostics.System.Project.Vehicle...	
ReadDiagnosticInformation	RemoteDiagnostics.SynchronizeRead.ReadAndClearErrorMemory	TryExcept.Try.Range	
Service	RemoteDiagnostics.SynchronizeRead.ReadAndClearErrorMemory	TryExcept.Try.Range.Clear.SyncService	System.Project...
ReadDiagnosticInformation2	RemoteDiagnostics.SynchronizeRead.ReadAndClearErrorMemory	TryExcept.Try.Range	

# Graphical Test Development with AutomationDesk



- Graphical blocks for implementation of control flow, error handling, variant handling, ...
- Prevention of syntax errors (supervision by graphical editor)
- Python scripting for algorithms, e.g. complex algorithms, API calls, ...
- Combination & Integration of graphics & scripts.  
Experience: 40-60% graphically, 40-60% scripting

The image displays the AutomationDesk interface, which is used for graphical test development. On the left, a graphical test sequence is shown, starting with a 'SelectProject' block. This is followed by a 'TryExcept' block. The 'Try' section contains a 'Range' block with a list of project IDs, a 'ReadDiagnosticInformation' block (highlighted with a red box) containing a 'SyncService' block, an 'AddResultsToReport' block, and a 'Clear' block with a note 'Clear the fault memory'. The 'Except' section contains a 'HandleException' block. Below the 'TryExcept' block is another 'ReadDiagnosticInformation2' block containing a 'SyncService' block and an 'Evaluate' block.

On the right, a Python script is shown, which is the code generated from the graphical test sequence. The script defines a class 'RTMSequencesEvents' and implements various methods for handling events and sequences. The code is as follows:

```
34 -----  
35 # -Class: RTMSequencesEvents  
36 # -Event class to attach to real-time testing sequences events  
37 -----  
38 class RTMSequencesEvents(rttmanagerlib._IRTSequencesEvents):  
39     def __init__(self, EventSource, Events):  
40         # Call base class constructor to connect to event source  
41         rttmanagerlib._IRTSequencesEvents.__init__(self, EventSource)  
42         # Collection of all events  
43         self.Events = Events  
44     def OnError(self, Sequence):  
45         # "Method-OnError"  
46         Sequence = rttmanagerlib.IRTSequence(Sequence)  
47         Information = "Stack: %s\nType: %s\nValue: %s" % (Sequence.LastExecutionErr  
48         self.OutputEventInformation("OnError", Sequence, Information)  
49     def OnStateChanged(self, Sequence, NewState):  
50     def OnWrite(self, Sequence, Output):  
51         # "Method-OnWrite"  
52         Sequence = rttmanagerlib.IRTSequence(Sequence)  
53         self.OutputEventInformation("OnWrite", Sequence, Output)  
54     def OnRemove(self, Name):  
55     def OnCreate(self, Sequence):  
56         # "Method-OnCreate"  
57         Sequence = rttmanagerlib.IRTSequence(Sequence)  
58         self.OutputEventInformation("OnCreate", Sequence.Name, "New RTTSequence: %s"  
59     def OnResetTestEngine(self):  
60         # "Method-OnResetTestEngine"  
61         self.OutputEventInformation("OnResetTestEngine", "", "Reset test engine.")  
62     def OutputEventInformation(self, EventName, Sequence, Information):  
63         # Output the event information to stdout or trace window
```



## HIL-Simulator

- dSPACE real-time platforms, ControlDesk, electrical fault simulation units, 3<sup>rd</sup> party HILs



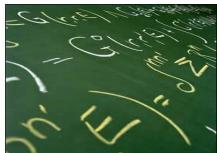
## Diagnostic tool support

- CalDesk, DTS6, DTS7, EDIABAS, VAG-Tester, DiagRA, CAESAR<sup>1)</sup>, samtec<sup>1)</sup>



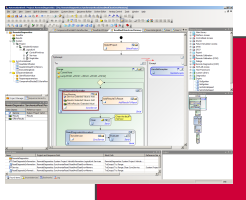
## Measurement and Calibration tool support

- CalDesk, INCA, CANape, CANoe<sup>2)</sup>, CANalyzer<sup>2)</sup>



## Calculation and Evaluation

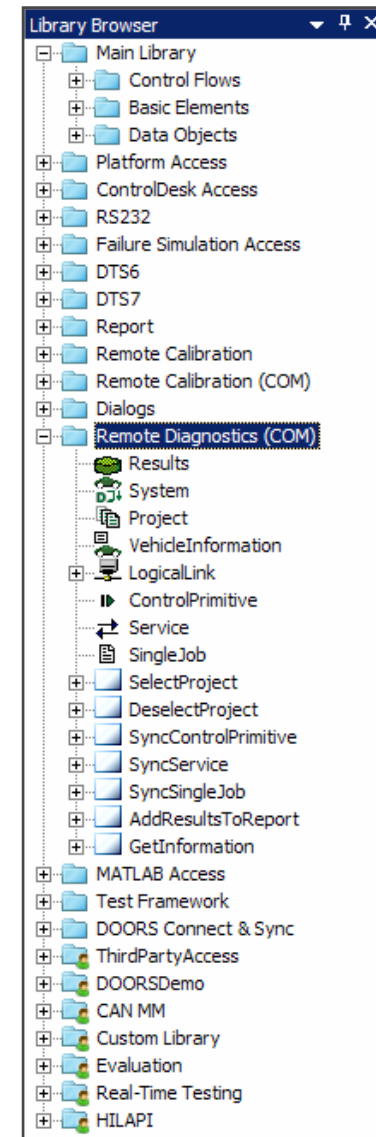
- MATLAB



## Customer specific Extension

- 3<sup>rd</sup> party hardware or software

<sup>1)</sup> on demand <sup>2)</sup> in customer projects



# Results & Reports



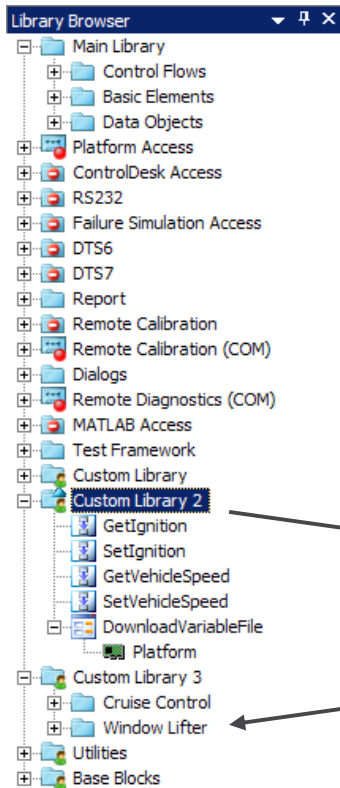
The screenshot displays the AutomationDesk interface. A window titled "AutomationDesk Block Report - Microsoft Internet Explorer" is open, showing a report with a plot titled "Sub Plots Example". The plot shows two signals: "ThrottleInput" (blue line) and "SteeringInput" (green line) over time. The y-axis is labeled "degree" and ranges from -0.8 to 1.0. The x-axis is labeled "time" and ranges from 0 to 5. Below the plot, there is a table of test steps and evaluation results.

- Automatic report generation in HTML and PDF
- Reports can contain
  - Text
  - Tables
  - Images
  - Plots
  - Hyperlinks

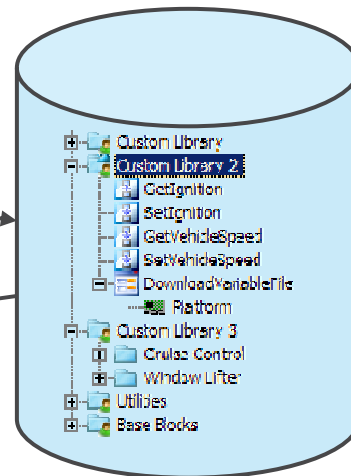
**Test Steps and Evaluation**

- 16:31:26 SetIgnitionKey (Position: Off)
- 16:31:26 GetIgnitionKey
  - CurrentPosition: Off
- 16:31:26 CheckIgnitionKey (ExpectedPosition: Off)
  - CurrentPosition: Off
  - > Result: OK

# Multi User Support



- Multiple Custom Flows Libraries can be used at a time
- Re-use of Custom Libraries through Import and Export
- **Version Management Interface** for Custom Libraries

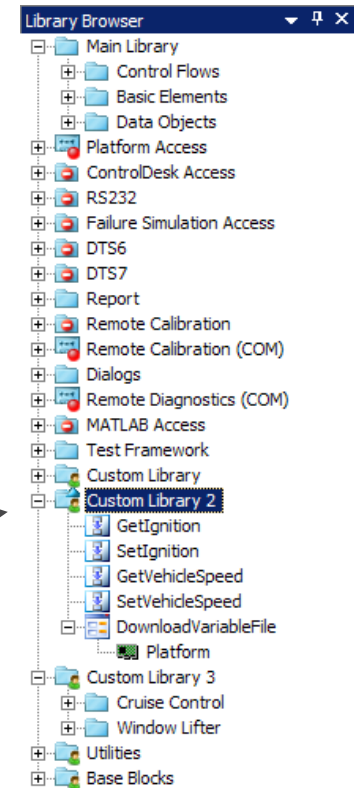


Check-In()

Check-Out()

Get()

Check-In()



Version Management Tool



# Bookmarks

fast navigation by double clicking

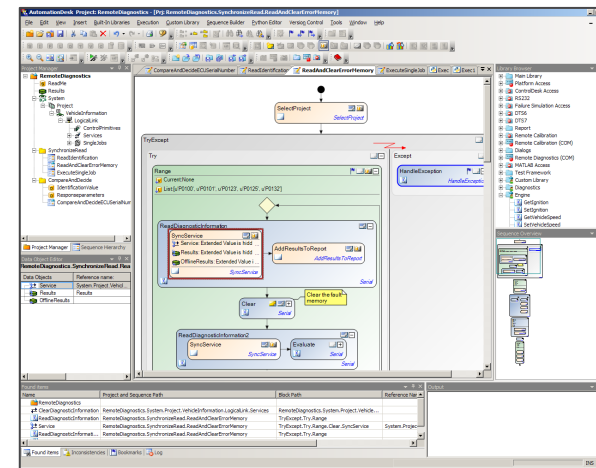
list of all available bookmarks

Name	Project and Sequence Path	Block Path
ChangeVariableforMeasurement	Remote Calibration(COM) CalDesk.Examples.Measurement.Dynamic_Measurement	For.MeasurementDynamic
DeselectProject	Remote Calibration(COM) CalDesk.Examples.Measurement.Dynamic_Measurement	
SwitchOnline	Remote Calibration(COM) CalDesk.Examples.Measurement.Dynamic_Measurement	

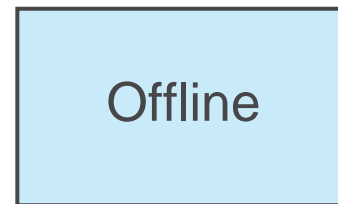
# Offline Test Execution



- Execution of test sequences to “test the test”
- Execution possible without “real HIL”
- Access to real-time model, failure insertion units, MATLAB, remote calibration, and ECU diagnostics are “redirected”
- Instead, blocks return a “dummy” value
  - Test development „offline“ at the developer’s desk
  - HIL-Simulator can be used full-time for automated testing



Execute()



## AutomationDesk – Key Features for successful ECU Testing



- Process integration capabilities, Openness, Flexibility
- HIL Test-Automation-Tool with most installations worldwide
- Worldwide customer and user base
- Worldwide sales, professional training
- Competent services and support in Test-Automation, HIL simulation, and diagnostics from one source – dSPACE
- Continuous enhancements and new versions



## Test Automation 2.0



### Important Notice

© Copyright 2008, dSPACE GmbH. All rights reserved.

Brand names or product names are trademarks or registered trademarks of their respective companies or organizations.