TargetLink®

- High-quality production code generation directly from Simulink®/Stateflow®
- Powerful software design and testing features
- High-performance, native AUTOSAR support
- Certified for IEC 61508 and ISO 26262
- Numerous 3rd-party-tools for highly efficient model-based development
TargetLink®

Automatic production code generator

**Highlights**
- High-quality production code generation directly from Simulink®/Stateflow®
- Powerful software design and testing features
- High-performance, native AUTOSAR support
- Certified for IEC 61508 and ISO 26262
- Numerous 3rd-party-tools for highly efficient model-based development

**Application Area**
Model-based design has become the established development method across many industries, and production code generation is the logical step for turning models into efficient, production-ready code. TargetLink generates production code (C code) straight from the MATLAB®/Simulink/Stateflow graphical development environment. The C code generation options range from plain ANSI C code to optimized fixed- or floating-point code for AUTOSAR platforms. Versatile code configuration options ensure that the production code copes with processor constraints.

**Key Benefits**
Converting graphical models directly into production code ensures perfect consistency between model and code. With deterministic code generation by TargetLink, the same model always results in the same proven code to guarantee the highest software quality. Every step can be tested against the specification via the built-in simulation features. This allows early verification and translates directly into cost savings, for example, by avoiding expensive ECU software errors.

**Efficient Coding**
Efficiency is the key to production-quality code. Efficient code requires a minimum of execution time and resources to run on a cost-efficient embedded processor. Code generated by TargetLink is proven to be as efficient as handwritten code. Other factors also make TargetLink such a useful tool: code readability, traceable model/code dependency, and last but not least, the ability to configure the code generation to produce exactly the kind of code that is required.

**Seamless Tool Chain**
TargetLink seamlessly connects function development and code generation for the control unit or prototyping hardware. It also automatically closes the gap between the design and verification phases. The result: transparent, well-defined development processes for conventional and AUTOSAR ECUs. Calibration files, AUTOSAR software component descriptions, and comprehensive documentation are generated in addition to the production code.
# Main Features and Benefits

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code efficiency</td>
<td>Efficient fixed-point or floating-point production code directly from MATLAB/Simulink/Stateflow</td>
<td>Your specifications — models and diagrams — are directly translated into efficient C code</td>
</tr>
<tr>
<td>Code reliability</td>
<td>Consistent, deterministic translations of models into stress-tested C code</td>
<td>Errors such as typing mistakes, oversights, misunderstandings are avoided</td>
</tr>
<tr>
<td>Human readability</td>
<td>Concise, yet readable code</td>
<td>Code reviews are easy to perform</td>
</tr>
<tr>
<td>Automatic scaling</td>
<td>Intelligent scaling based on worst-case propagation of signal ranges and simulation-based scaling</td>
<td>Shortens the time-consuming and error-prone scaling process</td>
</tr>
<tr>
<td>Test mechanisms</td>
<td>Various test levels to test the production code against the specification (MIL, SIL, PIL)</td>
<td>Malfunctions are found at earliest stage</td>
</tr>
<tr>
<td>Code coverage analyses</td>
<td>Dynamic analyses of program execution to find areas that have not been run through</td>
<td>Untested code and untested model parts are detected</td>
</tr>
<tr>
<td>Incremental code generation</td>
<td>Modular code generation, i.e., for specific subsystems</td>
<td>Faster code generation, preserving approved code</td>
</tr>
<tr>
<td>Model referencing support</td>
<td>Developing models on a modular/component basis</td>
<td>Distributed development by large teams is much easier, and large models can be handled more efficiently.</td>
</tr>
<tr>
<td>Multirate code</td>
<td>Full support of multirate systems with intertask communication</td>
<td>You can already define tasks at block level</td>
</tr>
<tr>
<td>TargetLink Data Dictionary</td>
<td>Central container to handle variables, data structures, scaling formulas, tasks, functions</td>
<td>You can manage complex data to plan and structure your projects</td>
</tr>
<tr>
<td>Code generation straight from the Data Dictionary</td>
<td>Generation of code files and A2L files for Data Dictionary variables independently of their use in TargetLink models</td>
<td>Simplified software integration and integration testing for code from multiple TargetLink models/subsystems and legacy variables</td>
</tr>
<tr>
<td>TargetLink Blockset</td>
<td>The free TargetLink Blockset Stand-Alone can be used without having the Base Suite installed</td>
<td>Large workgroups can work with TargetLink models without the need for additional TargetLink licenses</td>
</tr>
<tr>
<td>Compliance with standards</td>
<td>Compliance with relevant standards such as ASAM-MCD 2MC (ASAP2), AUTOSAR, MISRA, and OSEK</td>
<td>Quality and interoperability guaranteed</td>
</tr>
<tr>
<td>AUTOSAR support</td>
<td>Support for modeling and code generation for AUTOSAR software components (SWC), and generation of SWC descriptions</td>
<td>TargetLink bridges the gap between model-based design and AUTOSAR-compliant software development</td>
</tr>
<tr>
<td>Support of OSEK/VDX-compliant operating systems</td>
<td>Support for the standardized OSEK/VDX interface and features</td>
<td>You can design multirate software that is compliant with OSEK operating systems</td>
</tr>
<tr>
<td>Calibration data generation</td>
<td>Calibration data exported as ASAM-MCD 2MC (ASAP2) file for calibration tools</td>
<td>Automated and complete process with perfect consistency between model and calibration data</td>
</tr>
<tr>
<td>Documentation</td>
<td>Automatic model and code documentation</td>
<td>Your projects are transparent and trackable</td>
</tr>
<tr>
<td>AUTOSAR software component (SWC) container exchange</td>
<td>Exchanging AUTOSAR SWC containers between TargetLink and SystemDesk®</td>
<td>Safe and convenient round trips for AUTOSAR software development</td>
</tr>
<tr>
<td>Modular development</td>
<td>Innovative interface concept for ports, measurement signals and calibration parameters</td>
<td>Easily increase software reuse across different projects</td>
</tr>
<tr>
<td>Connection to dSPACE VEOS®</td>
<td>Export virtual ECUs from TargetLink and run them in VEOS in combination with other virtual ECUs and standard dSPACE tools like ControlDesk® Next Generation</td>
<td>Easy testing and experimentation with TargetLink code</td>
</tr>
<tr>
<td>FMI support</td>
<td>Export of Functional Mock-up Units (FMUs) from TargetLink models based on the Functional Mock-up Interface (FMI) standard</td>
<td>Execution of TargetLink-generated code with offline and real-time simulators from third parties and dSPACE</td>
</tr>
</tbody>
</table>
Order Information

<table>
<thead>
<tr>
<th>Classification</th>
<th>Type</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetLink Base Suite[1)</td>
<td>Base Suite</td>
<td>TBS</td>
</tr>
<tr>
<td>Target Optimization Modules for certain processors/compilers (further details see p. 23)</td>
<td>Infineon C16x Tasking</td>
<td>TOM_C16x/Tasking</td>
</tr>
<tr>
<td></td>
<td>Infineon TriCore Tasking</td>
<td>TOM_TriCore/Tasking</td>
</tr>
<tr>
<td></td>
<td>Renesas SH-2/Renesas</td>
<td>TOM_SH2/SHC</td>
</tr>
<tr>
<td>Other modules</td>
<td>Target Simulation Module (for all supported processors)</td>
<td>TSM</td>
</tr>
<tr>
<td></td>
<td>TargetLink Module for Operating Systems – OSEK</td>
<td>TMOS_OSEK</td>
</tr>
<tr>
<td></td>
<td>TargetLink AUTOSAR Module</td>
<td>TAS</td>
</tr>
<tr>
<td></td>
<td>TargetLink Data Dictionary Manager (included with TargetLink Base Suite)</td>
<td>DSDD_MANAGER</td>
</tr>
</tbody>
</table>

[1) TargetLink is available as a 64 bit as well as a 32 bit version to be used in combination with 64 bit or 32 bit MATLAB.

Relevant Software

<table>
<thead>
<tr>
<th>Software</th>
<th>Included</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stand-alone blockset for free model exchange</td>
<td>Integrated development environment</td>
</tr>
<tr>
<td></td>
<td>Data dictionary</td>
<td>MathWorks MATLAB®/Simulink®/Stateflow®</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compiler for host simulation included in MATLAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.dspace.com/go/os_compatibility">www.dspace.com/go/os_compatibility</a></td>
</tr>
<tr>
<td></td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Compilers for processor-in-the-loop tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Target-specific compiler for processor-in-the-loop tests with Target Simulation Module</td>
</tr>
</tbody>
</table>

TargetLink Product Support Center

The TargetLink Product Support Center is the primary online resource for TargetLink users, providing information about releases, compatibility, application notes, additional utilities, TargetLink Known Problem Reports, etc. The address is www.dspace.com/tlpsc
## NEW: TargetLink 4.1

<table>
<thead>
<tr>
<th>Feature Area</th>
<th>Improvement</th>
</tr>
</thead>
</table>
| Modeling in Simulink/Stateflow       | - Multiple instantiation of referenced models, including support for model parameters, e.g., to use instance-specific parameter sets  
- Support of Simulink’s simplified initialization mode and improved support for the classic initialization mode  
- Support of initial condition (IC) structures for simplified initialization for Simulink buses  
- Simulink Signal Conversion block with TargetLink extensions, e.g., to specify structured variables  
- Bus Assignment block for simplified modeling with buses, e.g., reading and writing from/to individual bus signals  
- Buses at the Simulink/Stateflow interface and structured Stateflow variables  
- Support of structures in the Stateflow action language for simplified modeling and access to structured signals  
- Update and check capability of the subsystem frame generation in TargetLink to synchronize TargetLink ports in subsystems based on Data Dictionary signature objects  
- Support of AUTOSAR 4.2.1, including ARXML import/export, AUTOSAR-compliant code generation, etc.  
- Activation of Runtime Environment (RTE) events that let runnables detect the reasons for their activation  
- Support of port-defined argument values for more efficient server modeling  
- Support of AUTOSAR NvData (nonvolatile data) interfaces in combination with require ports, provide ports and combined provide-require ports as well as efficient mechanisms for writing to NVRAM  
- Support of AUTOSAR transformers in combination with sender-receiver communication for end-to-end communication protection and SOME/IP (Ethernet)  
- Incremental code generation for reused systems (either for incrementally generated subsystems or model referencing)  
- More efficient code for function reuse via signal inheritance from predecessor/successor blocks of reused systems  
- Code efficiency improvements, including:  
  - Dimension downgrade for vector/matrix auxiliary variables  
  - Improved moving into conditionally executed branches  
  - Improved copy propagation  
  - Generation of default keyword: "void" instead of "Void"  
  - Consolidated Data Dictionary XML import to increase robustness against XML files with minor defects and support for XML file import from older TargetLink versions  |
| AUTOSAR                              |  
| Code generation core capability      |  
| Data Dictionary and data management improvements | - Extension of the Data Dictionary’s embedded help to provide instant information on objects and properties  
- Data Dictionary views for standard TargetLink users in AUTOSAR and non-AUTOSAR use cases to hide irrelevant Data Dictionary objects and properties  
- Improved comparison of Data Dictionaries in the DD Comparison pane via filter rule sets  
- Improved Data Dictionary Message Browser for a clear indication of warnings and errors  
- Referencing code generation option sets from the TargetLink Main dialog in the Data Dictionary to use them in various models at the same time  
- Export of Functional Mock-up Units (FMI standard) from TargetLink to execute TargetLink code in third-party environments  
- Storing requirement information in the Data Dictionary to use at model elements and include in the production code as code comments  
- Separate specification of compilers for production code or MEX compilers  
- Block context menus to transfer incrementally generated subsystems into referenced models and vice versa  
- Improved MIL simulation speed for scaling-invariant systems and models with many workspace parameters  
- New TargetLink demo models for:  
  - Code with variable vector sizes (vector width)  
  - Generating reusable code by means of direct and indirect function reuse  
  - Multiple instantiation of referenced models  
  - AUTOSAR NvData interfaces and AUTOSAR transformers  
  - Data Dictionary views and Data Dictionary API  |
How Do I Get the TargetLink I Need?

**TargetLink Modules**
TargetLink is available as a base suite plus additional modules, so that you can adapt it to your needs.

---

### TargetLink Module Overview

<table>
<thead>
<tr>
<th>Module</th>
<th>License Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base Suite License</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>ANSI C coder</strong></td>
</tr>
<tr>
<td><strong>Target Simulation Module</strong></td>
<td>Freescale MPC55xx</td>
</tr>
<tr>
<td><strong>Target Optimization Modules</strong></td>
<td>Infineon C16x</td>
</tr>
<tr>
<td><strong>Automotive-Specific Modules</strong></td>
<td>OSEK/VDX module</td>
</tr>
</tbody>
</table>

1) *Useable in stand-alone mode without license.*
2) *The Data Dictionary Manager is also available as a stand-alone license, e.g., for use with the stand-alone blockset.*
3) *Selection of major microcontroller families supported. For a complete list, please refer to www.dspace.com/go/tlpil*

---

**TargetLink Base Suite**
- Highly efficient ANSI C code generation from Simulink/Stateflow
- For all microcontrollers with ANSI C compiler
- Fixed-point code, floating-point code or a mixture of both
- TargetLink Data Dictionary (p. 31)
- TargetLink Blockset (p. 11)
- Autoscaling (p. 18)
- Code coverage analysis (p. 20)
- Modular development and code generation (p. 21)

**Target Simulation Module (optional)**
- Test your generated code on the target microcontroller (for supported processors and evaluation boards see p. 7)

**TargetLink AUTOSAR Module (optional)**
- Support for the development of AUTOSAR software components (SWC) (p. 27)

---

**TargetLink Module for Operating Systems (optional)**
- Support of OSEK/VDX-compliant operating systems

**Target Optimization Modules (optional)**
- For target-specific, optimized code generation
- Uses compiler-specific language extensions and assembly macros
Supported Processors and Evaluation Boards

For processor-in-the-loop simulation, TargetLink supports the most common processors for embedded applications, especially in the automotive field.

<table>
<thead>
<tr>
<th>Processor Family</th>
<th>Compiler Supported by Target Simulation Module</th>
<th>Evaluation Boards Supported by TargetLink</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freescale MC6F83xx</td>
<td>Freescale CodeWarrior compilers</td>
<td>Freescale MC6F8367EVM</td>
</tr>
<tr>
<td>Freescale MPC55xx</td>
<td>Green Hills, Freescale CodeWarrior, GNU and Wind River compilers</td>
<td>Axiom MPC5554DEMO, Freescale MPC5561EVB, Freescale MPC5604EVB and Freescale MPC5748EVB</td>
</tr>
<tr>
<td>Freescale S12X</td>
<td>Cosmic and Freescale CodeWarrior compilers</td>
<td>Freescale EVB9S12XEP100</td>
</tr>
<tr>
<td>Infineon C16x</td>
<td>Tasking compilers</td>
<td>i+ME eCAN C167 CR</td>
</tr>
<tr>
<td>Infineon TriCore</td>
<td>Tasking and GNU compiler</td>
<td>Infineon TriBoard TC1766, Infineon TriBoard TC1767, Infineon TriBoard TC1796 and Infineon TriBoard TC275</td>
</tr>
<tr>
<td>Infineon XC2000</td>
<td>Tasking compilers</td>
<td>Infineon SK-EB XC2287</td>
</tr>
<tr>
<td>Renesas RH850</td>
<td>Green Hills compilers</td>
<td>Renesas YRH850F1L</td>
</tr>
<tr>
<td>Renesas SH-2</td>
<td>Renesas compilers</td>
<td>Renesas SDK7058 and SDK72513</td>
</tr>
<tr>
<td>Renesas V850</td>
<td>Green Hills compilers</td>
<td>Renesas AB_050_Fx4_70F4012</td>
</tr>
<tr>
<td>STMicroelectronics ARM Cortex M3</td>
<td>Keil compilers</td>
<td>Emerge-Engineering MEDKit on ARM</td>
</tr>
</tbody>
</table>

1) Only for TC275.

Some of the evaluation boards need to be modified (loader, external RAM, etc.). Please order them through dSPACE to ensure a correct board setup.

For more information on software compatibility with target compilers and evaluation boards, please refer to: www.dspace.com/go/tlpil

TargetLink Engineering Services

Our engineering portfolio includes special TargetLink customer services, for example:

- Hands-on support during introduction of TargetLink
- Support during evaluations and pilot projects
- Customer-specific TargetLink training
- Integrating TargetLink into your development processes

- Model analysis and advice
- Support for developing and integrating AUTOSAR-compliant software components
- Support for integrating the generated code in the ECU’s software environment
- Tool chain development and maintenance
- Process consulting
How Do I Work with TargetLink?

Typical Steps in Generating Production Code

This diagram illustrates the workflow from model design to code implementation. It also shows that code verification based on simulation is an iterative process. The workflow is described in greater detail on the following pages.

1) valid only for fixed point software
Workflow

Control Design and Function Prototyping
Control design starts with creating a control model in the integrated design environment MATLAB/Simulink/Stateflow. Before production code generation with TargetLink, you can use dSPACE prototyping systems to carry out convenient function prototyping and validation of your new ECU control algorithms.

Using the TargetLink Block Library
To implement the control algorithms in C code, you need the TargetLink block library. TargetLink blocks contain additional data for code generation, such as the scaling information for fixed-point variables, variable classes, variable names, etc. A utility automatically replaces your Simulink controller model with blocks from the TargetLink block library. The process is reversible without any data losses. If you use the free TargetLink Blockset Stand-Alone during control design, conversion is not necessary.

Model-in-the-Loop Simulation on Host PC
Model-in-the-loop simulation (floating-point) serves as a reference for subsequent steps and provides the minima and maxima of variables as a basis for subsequent fixed-point scaling if required.
**Automatic or Manual Scaling**
If you want to generate fixed-point code, the scaling has to be specified. You can use manual scaling, simulation-based autoscaling or worst-case autoscaling. You can choose from a broad range of scaling options (p. 18) for each TargetLink block individually.

**Code Generation**
The TargetLink Base Suite generates highly efficient ANSI C code for a controller model at the click of a button.

**Verification on Host PC via Software-in-the-Loop Simulation**
By means of software-in-the-loop simulation on a host PC, you can compare the behavior of the generated code with the reference data obtained in model-in-the-loop simulation. TargetLink offers a graphical user interface, where you can select the signal histories of blocks for detailed analysis.

**Verification on Target Processor via Processor-in-the-Loop Simulation**
Using the optional Target Simulation Module (p. 6), you can execute processor-in-the-loop simulation to verify the generated code on an evaluation board equipped with the same target processor as your final ECU. Successful verification of processor-in-the-loop simulation with model-in-the-loop simulation and software-in-the-loop simulation ensures the software quality of the generated code. TargetLink also provides information on the code size, the required RAM/ROM, and the stack consumption as it evolves over time. The execution time can be displayed as well.
TargetLink Block Library for Implementation

Extended Block Functionality

Implementation-Specific Information
The Simulink® block library is very powerful in simulation tasks, providing all necessary specification features. When it comes to code generation, however, more information is needed for each block. The blocks need additional capabilities for fixed-point simulation. That is why there is a TargetLink block for each supported Simulink block. The TargetLink blocks significantly enhance the functionality of the supported Simulink blocks and have an extended dialog that allows you to enter the implementation-specific information necessary for code generation. Each block also provides a means of data logging and overflow detection. A special routine automatically prepares Simulink models for code generation with TargetLink by enhancing the Simulink blocks to TargetLink blocks.

TargetLink block dialog for entering implementation-specific information like the data type, variable name, scaling data, etc.

TargetLink Blockset Stand-Alone

Function Development with TargetLink Blocks
The TargetLink Blockset Stand-Alone is a license-free version of the TargetLink blockset which allows function developers to design their controller models directly with TargetLink. It can be used on any computer that has MATLAB/Simulink installed. With the TargetLink Blockset Stand-Alone, TargetLink models can be exchanged freely without the need for extra TargetLink licenses.

Its features include:
- Free TargetLink model exchange in workgroups
- Controller design and simulation in Simulink
- Prototyping with MathWorks® Simulink Coder™

Linking Control Design, Prototyping, and Implementation
You can use the free TargetLink Blockset Stand-Alone to design and prototype your controller without a full TargetLink installation. The TargetLink blocks can be used for rapid control prototyping on dSPACE hardware. Although the added functionality of the TargetLink blocks is not available in that case, the TargetLink Blockset Stand-Alone allows you to use the same models for rapid control prototyping and production code generation. Development iterations are thus easier to perform and less prone to error.
Function development (including rapid control prototyping) with the TargetLink Blockset Stand-Alone and software development with a full TargetLink installation.

Block Overview

TargetLink Block Library

These TargetLink blocks extend corresponding blocks in the Simulink block library.
Matrix Support
The matrix support, included as of TargetLink version 4.0, lets you combine data sets into matrices and specify matrix signals in the model. The new TargetLink-supported blocks then give you many different options to create, manipulate, or split these matrix signals. You can use matrix code generation options to translate the modelings into efficient, readable matrix signal code in your model. Additionally, you can now implement new algorithms, such as ADAS applications, state-space controls, and sensor fusion like dead reckoning, while benefiting from model-based development.

TargetLink Utility Blocks
TargetLink utility blocks provide access to specific features of TargetLink or further specify the code generation process for the model.

Multirate Blocks
The TargetLink multirate blocks make multirate operating system objects available in block diagrams. There are even OSEK-specific blocks like the CounterAlarm.

AUTOSAR Blocks
Model-based design for AUTOSAR ECUs (p. 27) is supported by blocks for structural elements such as runnables, ports, and certain communication elements.
Block Configurations

Handling Large Models with the Property Manager

If you need to change the properties of a large model, TargetLink’s Property Manager is a quicker alternative to changing the properties in the block dialogs manually. It is a graphical user interface that displays the properties of TargetLink blocks and Stateflow objects in your model, and lets you view, filter, and modify several properties simultaneously. For Stateflow objects this includes charts, events, states and data. The Property Manager provides a tree view of the model’s subsystem hierarchy, a list of blocks in each subsystem, and a configurable list of properties for each block.

Three Simulation Modes for Testing

Confirmation by Simulation

Comparing Simulation Results

Although code generators produce virtually flawless results when compared to manual programming, the generated code still needs to be tested as well as the underlying specification. TargetLink provides powerful and easy-to-use means to verify the generated code. The code tests are performed in the same simulation environment that was used to specify the underlying simulation model. Functional identity has been achieved when simulation results match. The comprehensiveness of tests can be assessed using code coverage analysis.

Test Against Specification

Verification means checking the implementation against the specification. The specification is the simulation model, as executed by Simulink and Stateflow. The executable and approved specification is used as a reference. The implementation is the generated and compiled C code, as executed by the target processor. TargetLink provides a 3-step verification process which shows at a click whether the specification and implementation are functionally identical. On the basis of a controller model, TargetLink performs simulations of the model, the generated code on the host, and the generated code on the target, without additional model modifications or preparations.
Three-Step Verification Process

Model-in-the-Loop Simulation
The first step is to record data for reference plots from the simulation model. Signals from selected blocks and state variables are automatically logged by TargetLink. The model-in-the-loop simulation captures the specified behavior of the model that is to be implemented in C code later on. The recorded signal plots act as the reference for the next verification steps. Model-in-the-loop simulation also serves other purposes. It is used for detecting overflows of integer variables, and its results are used for simulation-based autoscaling.

Software-in-the-Loop Simulation
Software-in-the-loop means the code is generated and replaces the controller blocks in the simulation model (for example, the same plant and stimulus signals). TargetLink does this automatically in the background. You still set the controller blocks, though it is the code that is executed on the host PC instead. The signal plots should be largely identical when compared to the results of model-in-the-loop simulation. If they are not, they can be analyzed to get a better understanding of the cause of the deviation and to fine-tune the fixed-point settings.

Processor-in-the-Loop Simulation
Finally, the generated code runs on an embedded processor. Code that runs correctly on the host PC can still cause trouble on the target processor. To check that this does not happen, TargetLink offers processor-in-the-loop simulation. An off-the-shelf evaluation board is connected to the host PC, and the generated code is compiled with the target compiler and downloaded to the evaluation board. TargetLink manages the communication between the host PC and the evaluation board. All these activities are automated and need no user interaction. Simulation on an evaluation board just takes two mouse clicks.
### Features and Benefits of the Simulation Concept

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Benefit</th>
</tr>
</thead>
</table>
| MIL/SIL/PIL simulation at a click | Switching from MIL to SIL or PIL simulation requires just a click           | ■ Powerful simulation environment  
■ No need for separate test models, generation of S-functions or manual insertions into test harness models |
| Integrated data logging       | Built-in data logging and result plotting for all simulation modes          | ■ No model modifications necessary  
■ Available for all simulation modes                                      |
| Direct comparison of MIL/SIL/PIL results | Automatic plotting of all simulation results in the same plot window       | ■ Display results of simulations in different modes directly and analyze deviations  
■ Direct feedback whether code matches model simulation                      |
| Detailed signal analysis and deviation plots | Zoom signals to visually inspect deviations, display constraints (e.g., defined ranges), use cursor to scroll through signal histories, display signal values numerically or plot signal deviation | ■ Get clear picture of signal behavior  
■ Especially useful for conversion from floating-point to fixed-point         |

The TargetLink block dialog lets you specify whether to log signal histories – regardless of the simulation mode.

**In Brief**

- Easy to use, intuitive, fast and frequent tests increase the software quality
- Unmatched simulation flexibility
Run-Time Analysis

Profiling the Code
Processor-in-the-loop simulation can also be used to profile the generated code. During simulation, TargetLink automatically measures execution time and stack consumption directly on the target processor. A code summary lists RAM and ROM usage for each function. These features allow you to evaluate design alternatives, such as selecting different search routines of a look-up table block. You can immediately measure the impact of the change on code efficiency. Sound implementation decisions based on accurate benchmarks become a matter of a few clicks.

Model-Code Traceability
For improved traceability and simplified code reviews, code files can be optionally generated in HTML format, with hyperlinks for navigation from model to code and vice versa at a click.
Scaling Variables

Fixed-Point Accuracy

Quick and Accurate Scaling
If TargetLink is to generate only floating-point code, scaling is not necessary. However, when fixed-point code is generated, autoscaling can be a huge time-saver. It takes away the tedious and error-prone task of manually scaling each variable and each operation in the software. What took days and weeks in the past can now be done in minutes and hours.

Overflow Detection
Another big advantage: fixed-point overflows no longer entail hours-long debugging sessions. With TargetLink, scaling can be checked during simulation. If an overflow occurs, TargetLink shows the exact location in the block diagram. The problem can be corrected right away.

Scaling Choices

Scaling Properties
TargetLink offers a two-coefficient linear scaling method, which is widely used in embedded control applications. The properties for specifying fixed-point scalings in TargetLink are:
- Data type
- Power-of-two scaling factor or arbitrary scaling factor
- Offset value
- Constraint values
- Bit safety margins
- Saturation options

Automatic Scaling
These scaling properties give ample choices to fine-tune fixed-point code to the conflicting requirements of low execution time, high computational precision, and overflow avoidance. Fixed-point scaling can be done manually by a software engineer, but in most instances it is left to the autoscaling tools from TargetLink.
Efficient Scaling Methods

Simulation-Based Autoscaling
Simulation-based autoscaling requires that the model can be simulated. This necessitates either a plant model or stimulus data for all imports. During simulation, TargetLink records the signals from each Simulink block and determines the value range in order to calculate optimum scaling for each block. The advantage of this scaling method is maximum computational precision.

Worst-Case Autoscaling
When simulation is not possible or deemed unfeasible, TargetLink can perform autoscaling based on a worst-case range calculation. This method requires value range information for imports and some specific blocks inside the model. TargetLink propagates these value ranges through the model and calculates the best scaling. The advantages of this scaling method are that it does not require a plant model and prevents overflows.

Scaling with Scaling Formulas and Type Definitions
Using scaling formulas is not actually an autoscaling method, but it is a way of scaling a model quickly. After defining a complete set of scaling properties under one name in the TargetLink Data Dictionary, you can select them in block dialogs or in the Property Manager to quickly scale signals of one kind or subsystems with the same scaling parameters.
Code Coverage Analysis

Run-Time Analyses
TargetLink offers C0 and C1 coverage analysis, also called ‘statement coverage’ and ‘decision coverage’. During simulation on host or target systems, counters record the frequency of execution for each branch of code. After simulation, a code coverage analysis report is generated with a coverage overview table and a detailed, annotated code listing with the execution count for each block. This report tells you immediately whether the tests cover everything. Code branches that have never been executed are easily identified, and test strategies can be adjusted accordingly. Thus, coverage analysis increases software quality.

<table>
<thead>
<tr>
<th>Code Coverage</th>
<th>Total Branches</th>
<th>Reached Branches</th>
<th>Unreached Branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel system</td>
<td>61.87%</td>
<td>139</td>
<td>86</td>
</tr>
<tr>
<td>Fuel rate controller</td>
<td>58.91%</td>
<td>129</td>
<td>76</td>
</tr>
<tr>
<td>Fuel rate calculator</td>
<td>56.56%</td>
<td>122</td>
<td>69</td>
</tr>
<tr>
<td>Correction redundancy</td>
<td>100%</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Airflow controller</td>
<td>100%</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Airflow calculation</td>
<td>100%</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Airflow subsystem</td>
<td>100%</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Example of the information provided by code coverage.

```c
174     | if (x <= Aux_S0) {
175     |     /* Saturation. */
176     |     return s_table[0];
177     |
178     |     } else {
179     |         /* Calculate table index. */
180     |         Aux UB_a = (UInt8) (((UInt8) ((UInt8) x) - (UInt8) Aux_S0)) >> Aux UB;
181     |         if (Aux UB_a >= ((UInt8) (map->Nx - 1))) {
182     |             return s_table[(UInt8) (map->Nx - 1)];
183     |         }
184     |     }
185     |     else {
186     |         }
187     |     }
188     |     Aux UB_b = (UInt8) (((UInt8) ((UInt8) x) - (UInt8) (Aux UB_a << Aux UB))
189     |         ) << Aux UB;
```

Annotated code listing with execution count for each code block.
Modular Development and Code Generation
Handling large models and software integration

Incremental Code Generation
Incremental code generation is available for Simulink subsystems which contain a TargetLink Function block. Each member of a development team can work on a subsystem individually and just generate code for that. TargetLink performs the necessary consistency checks when building the overall application from all the parts.

Benefits
Incremental code generation has several advantages:
- Code for individual subsystems in a model can be tested, reviewed, and frozen, while development in other subsystems continues.
- Team members can work independently on different subsystems.
- Code generation time can be significantly reduced by generating code only for subsystems that have been modified.

Model Referencing Support
TargetLink supports model referencing. Functionalities can be flexibly partitioned into several model files, versioned separately, and developed on a modular basis. Parts of models can be simulated separately and implemented individually by means of incremental code generation.

Specifying incremental code generation properties for a subsystem.

Code can be generated incrementally for referenced models.
Simplified Software Integration
TargetLink code from separate (incrementally generated) subsystems and (referenced) models can be easily integrated and tested by means of code generation from the Data Dictionary. Variables with global relevance that are shared by separate models and subsystems are specified and generated from the Data Dictionary.

Code Generation Straight from the TargetLink Data Dictionary
The Data Dictionary supports the generation of both code and A2L files for Data Dictionary variables independently of their use in TargetLink models. Variables with global relevance that are shared between different models, subsystems and referenced models should be specified in the Data Dictionary and implemented from there. Generating code files with definitions and declarations, and also A2L files for variables from the Data Dictionary, makes software integration and software integration testing easy. Moreover, all the calibration parameters for an entire ECU project can be created in one file, no matter whether they are from legacy code or used in TargetLink models.

Diff&Merge mechanisms via TargetLink Data Dictionary
TargetLink Data Dictionary has the mechanisms needed to compare different versions and display changes, e.g. old/new. You can then trace modifications back to the model to analyze their effects on it. Diff&merge mechanisms let you update interface definitions simultaneously to ensure consistency when changes are made – making your work much easier.
Code Optimization

Highly efficient production code

Optimized ANSI C Code
TargetLink generates highly efficient ANSI C code. This is achieved by various optimization techniques, which roughly fall into 3 categories:
- Standard optimizations
- Interblock optimization
- Code pattern libraries

Standard Optimizations
TargetLink uses many of the optimization techniques that are also used by C compilers: constant folding, algebraic transformations, dead code elimination, and lifespan analysis of variables, to name just a few. They are standard optimization techniques and are used by TargetLink whenever applicable.

Interblock Optimization
Interblock optimization combines the code of several blocks in one C code line. TargetLink’s interblock optimization gives the generated code a human touch, because it combines code in a very similar way to what a skilled software engineer would do. For instance, a string of arithmetical, logical, and relational blocks is usually combined in one expression. Code for calculating the inputs of a switch block is moved inside the If-Else instruction. The example shows interblock optimization at work. This technique saves execution time and ROM size, but most importantly, it reduces stack size significantly and enhances code readability considerably.

Code Pattern Libraries
A powerful block such as the 2-D Look-Up Table block or the FIR Filter block needs several lines of code. TargetLink takes them from an internal code pattern library during the code generation process, ensuring that the code for complex blocks is also highly efficient.

Example of interblock optimization.
Target-Optimized Code

Optimized Assembly Code
With the optional target optimization modules (p. 23), processor- and compiler-specific language extensions can also be used for code optimizations. This further increases the efficiency of the generated code. There is a library containing highly optimized code patterns for every processor/ compiler combination supported by TargetLink. Compiler-specific instructions and even inline assembly macros are used to deliver top-notch code performance. Some features of a processor core can be used only by applying these coding techniques. This sometimes makes the difference in meeting hard real-time requirements and ROM size limits.

<table>
<thead>
<tr>
<th></th>
<th>ANSI-C</th>
<th>Optimized Assembly Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycles</td>
<td>1232</td>
<td>45</td>
</tr>
<tr>
<td>Code size</td>
<td>100 byte</td>
<td>54 byte</td>
</tr>
</tbody>
</table>

64-tap FIR filter, measured with Infineon TriCore/Tasking Compiler: Optimized code is 27 times faster than ANSI-C.

Customized Code

Code According to Customer Standards

Readability
TargetLink code is easily readable and has many comments. Unnecessary macros, function calls, cryptic naming, etc. are avoided. Comprehensive configuration options give you full control over variable, function, and file naming, as well as the flexibility to partition the code into functions and files to keep the structure logical and manageable.

Variable Classes
You also have full control of how TargetLink generates variables and and therefore of how the RAM and processor stack are utilized. You can specify the scope and storage class of a variable, as well as other C attributes like ‘volatile’ or ‘const’. Even PRAGMA directives can be generated before and after each declaration or block of declarations. To keep things simple, TargetLink bundles all these properties into ‘variable classes’. You can select these in block dialogs to save a lot of repetitive specification work.
External Code Integration
TargetLink offers a wide variety of specification options on the block diagram level for easy interfacing with external code such as device drivers, or with any other routine written in C or assembler. These are:
- Including external header files
- Using external global variables
- Using externally defined macros
- Calling imported functions
- Calling access functions or macros
- Defining a Custom Code block containing handwritten C code

Code Output Formatting
TargetLink can generate code in a format that exactly matches company-specific C code templates. Code output can be formatted via an XML configuration file and an XSL style sheet. You can define the format of file and function headers, the format of code comments, and the inclusion of specific header files. Comments can even contain Unicode, for example, for Japanese characters.

Documentation Generated Automatically

Consistent Documentation and Model
TargetLink not only generates code, it also documents what it does – keeping perfect consistency with the model and the code. An automatically generated document provides information about function interfaces and global variables, and a list of all measurable and adjustable variables, scaling parameters, code generator options and much more. Screenshots of models, subsystems, and simulation plots can also be included. Links to the generated C code are provided. You can specify the documentation you require, for example, the level of detail. Documentation can be generated in the HTML, RTF (for word processing) and PDF formats.

Production code documentation in HTML.
Automation and Process Integration

Full Access to TargetLink

Comprehensive TargetLink API

TargetLink is easy to integrate into existing development environments, because it comes with a comprehensive and fully documented API. This grants full access to all TargetLink properties and settings and allows processes to be automated, while at the same time providing options for intervention in the individual process phases. For example, ‘hook functions’ allow user tasks to be performed at all stages of the build process.

Calibration File Generation

ASAM MCD-2 MC File Format

Another important requirement for a code generator is close links with calibration systems. ECU code must be prepared for parameter fine-tuning by making calibratable or measurable variables accessible to a calibration system. TargetLink supports the generation of the standardized ASAM MCD-2 MC file format (formerly ASAP2) via the Data Dictionary to make the variables and parameters available for ECU calibration. All major calibration tools support this standard. Since the C code and the ASAM MCD-2 MC file are generated with the same data basis, they are always consistent. This eliminates another error source, and streamlines the development process.

TargetLink offers several predefined variable classes for calibratable and measurable variables. You can also specify your own classes, ensuring that each class holds suitable attributes for calibration and/or measurement.
Model-Based Design for AUTOSAR Software Components

The TargetLink AUTOSAR Module

The optional TargetLink AUTOSAR Module makes TargetLink’s modeling, simulation and code generation features available for designing AUTOSAR software components (SWCs). Developers can specify AUTOSAR structure elements, such as runnables, ports, and communication interfaces, simply at model level. Hence model-based designs are directly implemented in the form of AUTOSAR-compliant production code.

Functionality

- Model-based design for AUTOSAR software components (SWCs)
- Code generation for AUTOSAR SWCs
- Importing, exporting and merging software component descriptions
- Simulating and testing SWCs
- AUTOSAR frame model generation
- Round trips with other AUTOSAR tools based on SWC description files
- Convenient round trips with SystemDesk based on SWC container exchange

Benefits

- Efficient and easy modeling using proven workflows for AUTOSAR
- AUTOSAR-compliant code generation
- Easy migration of existing TargetLink models to AUTOSAR
- Testing and verifying SWCs in early design phases
- No tedious manual creation of software component description files
- Easy integration of TargetLink in an AUTOSAR tool chain (p. 30)

In brief: You go from model to AUTOSAR-compliant code faster.

TargetLink AUTOSAR Block Library

To design AUTOSAR SWCs, TargetLink offers the TargetLink AUTOSAR library:

- AUTOSAR runnables and operation calls are modeled using the TargetLink Function/Runnable block.
- AUTOSAR communication mechanisms like sender/receiver or client/server communication are specified in TargetLink Inport and Outport blocks.
- Specific AUTOSAR features like signal acknowledgement, signal invalidation and status signals are modeled using special ComSpec blocks.
- Optional SWC sender/receiver blocks can be used to represent AUTOSAR ports in a Simulink/TargetLink model.

TargetLink blocks for modeling AUTOSAR SWCs.
Designing AUTOSAR SWCs

To define an AUTOSAR runnable, the Function/Runnable block is applied to a modeled subsystem just as for normal TargetLink functions. This can be done either for new models designed from scratch or for legacy models with established control functions. To define how data is exchanged between the runnables of one or more SWCs, TargetLink Inports and Outports are used just as in non-AUTOSAR applications. TargetLink supports sender-receiver, interrunnable and synchronous client-server communication. Property specifications are made both on block level and via the TargetLink Data Dictionary. The same model can be used to generate both AUTOSAR-compliant code and standard TargetLink code.

Referencing AUTOSAR objects specified in the Data Dictionary from dedicated block dialogs.
AUTOSAR-Compliant Code Generation

TargetLink generates production code for AUTOSAR software components and provides all the code generation options for optimization. Modeled runnables are implemented as C functions and AUTOSAR communication mechanisms specified in TargetLink Imports/Outports are implemented as run-time environment (RTE) macros according to the AUTOSAR standard.

```c
void controller_runnable(Void)
{
   /* call of function: controller/Controller_Runnable/RteApiFunction */
   ref = Rte_IRead_PosController_rport1_ref();
   ...
   /* Sum: controller/Controller_Runnable/e */
   S12_conversion_grid{[(Int16) ref] - [(Int16) Rte_IrvIRead_PosController_InPos]};
   ...
   /* TargetLink output: controller/Controller_Runnable/OutPort */
   call of function: controller/Controller_Runnable/RteApiFunction1 */
   Rte_IWrite_PosController_pport1_upi((Int15) (S12_conversion_grid << 2));
}
```

Simulating and Testing AUTOSAR SWCs

With TargetLink, SWCs can be simulated in all three simulation modes:

- Model-in-the-loop (MIL)
- Software-in-the-loop (SIL)
- Processor-in-the-loop (PIL)

Multiple SWCs can be simulated in one simulation run. Communication between SWCs is simulated to the extent supported by the Simulink design environment; for example, there is no asynchronous client-server communication.
Importing and Exporting SWC Descriptions
To integrate the code generated for SWCs in the overall AUTOSAR software architecture, software component description files are required. These describe the structural elements such as runnables or ports that are used in an SWC. To save function designers tedious manual work, TargetLink creates the component descriptions and exports the descriptions in AUTOSAR XML format. The TargetLink Data Dictionary also lets you import and merge existing component descriptions to achieve a seamless AUTOSAR development process with a tool like SystemDesk.

AUTOSAR Frame Model Generation
Using a software component description, TargetLink automatically generates a frame model containing the relevant AUTOSAR ports and runnables. The developer can then insert the control algorithm into this model frame to obtain a complete AUTOSAR software component. This procedure makes it much easier to migrate existing models to AUTOSAR or to start modeling from scratch.

Migrating Standard TargetLink Models to AUTOSAR
Existing, non-AUTOSAR TargetLink models can easily be migrated to AUTOSAR using the TargetLink AUTOSAR Migration Tool. It converts individual subsystems to AUTOSAR runnables and supports flexible specification of AUTOSAR properties by means of hook functions. You can configure the tool to produce precisely the kind of AUTOSAR-compliant code you require. The TargetLink AUTOSAR Migration Tool is available for download free of charge from www.dspace.com/go/tl_ar_migration

TargetLink in an AUTOSAR Tool Chain
For AUTOSAR software development, TargetLink is typically combined with an AUTOSAR architecture tool like SystemDesk. A software architecture with multiple components is specified in the architecture tool, while TargetLink is used to “fill” and implement the individual SWCs. The two tools exchange data on the basis of AUTOSAR XML files. A top-down approach (starting with the software architecture) and a bottom-up approach (starting with a TargetLink function model) can both be used for AUTOSAR round trips.
Managing Data with the TargetLink Data Dictionary

Central Data Container

**Key Features**
- Central container for managing all data required for designing a model and implementing it on an ECU
- Convenient administration of parameters, constants, signals, variables, structures, variable classes, typedefs, scaling formulas, etc.
- Tight integration with TargetLink
- Full access to code specifics
- Various import and export formats
- Full access to all TargetLink Data Dictionary objects via powerful MATLAB API

**Description**

The TargetLink Data Dictionary is a central data container holding the relevant information for a model’s design, code generation, and implementation on an ECU. Data dictionary objects can be referenced from TargetLink models. You can define and manage variables and their properties, and you can specify structured data types and use them for variable declarations. Scaling formulas can be entered and used to uniformly scale fixed-point signals and parameters in the model. You can import and export standardized or proprietary data and share the data with the calibration system.

To specify the properties of the code to be generated, Data Dictionary objects are referenced from block diagrams of the model, in this case for a specific variable.
Benefits
The TargetLink Data Dictionary is the perfect tool for defining and handling project-related code specifics, even for workgroups. It provides access to a wealth of additional information: specifics on C modules, function calls, tasks, variable classes, data variants and so forth. The data is presented in a well-organized tree and can also be accessed via an application programming interface (API). The Data Dictionary also supports common import and export formats, so that existing and proven definitions, for example, a calibration file, can be used as templates.

Data Dictionary Manager
The Data Dictionary Manager provides the user interface to the Data Dictionary for convenient administration of the data required for designing models and implementing them on an ECU. Elements such as type definitions, variables, and scaling formulas are organized clearly. Typical user interface functions are available, like copying and pasting objects, loading and saving individual data branches, and searching the entire dictionary. With its multi-edit capabilities, the Object Explorer pane provides an overview of multiple objects and selected properties, and allows the properties of multiple Data Dictionary objects such as variables to be modified simultaneously. This greatly simplifies the handling of large data volumes. The tool can be customized by using the M-script plugin mechanism for menus, context menus, and properties. Additional panes can be used for outputs generated from user scripts.
Data Structure
The TargetLink Data Dictionary has specific main areas where you can set the configuration data, the pool data for models (pre-code generation data), and the subsystem and application data (post-code generation data). Selected branches of the Data Dictionary can be loaded from separate include files, which are maintained centrally for a whole workgroup.

Share Data

Import and Export
The TargetLink Data Dictionary supports various import/export formats, including
- Variables and Simulink data objects from/to MATLAB workspace and files
- XML (Extensible Markup Language)
- ASAM MCD-2 MC (Standardized Description Data, formerly ASAP2)
- AUTOSAR software component descriptions
- OIL (OSEK Implementation Language)

Data Dictionary API
The TargetLink Data Dictionary MATLAB API gives you full access to the TargetLink Data Dictionary via MATLAB. All the functions necessary for managing data are available, for example, for creating Data Dictionary objects and defining their properties. Open API interfaces make it easy to integrate the TargetLink Data Dictionary into your company’s environment. To connect your own database to the Data Dictionary, you can develop your own export or import functionality.

Exporting Data Dictionary variables into the MATLAB workspace.
Prepare for Calibration
After generating code, TargetLink writes a detailed description of it to the TargetLink Data Dictionary, including lists of generated C modules, functions, variables, type definitions, scalings, variable classes, etc. This information is used for A2L generation when you export data to calibrate ECU parameters.

Export of A2L files from the TargetLink Data Dictionary.

TargetLink Tool Integration

- SYNECT®
- SystemDesk®
- VEOS®
- FMI standard

TargetLink Data Dictionary Connection to SYNECT
The file-based TargetLink Data Dictionary is easy to connect to SYNECT, dSPACE’s data management software, which acts as a central data base with multi-user support and integrated variant management.

Connecting the TargetLink Data Dictionary to SYNECT for signal and parameter management.
**TargetLink and SystemDesk**

TargetLink and SystemDesk are the ideal combination for developing AUTOSAR-compliant software. The two tools exchange SWC containers that hold not only ARXML files, but also source files for implementation, A2L files, and other helpful meta information. This is a reliable, transparent way to perform AUTOSAR round trips with minimum user intervention.

In addition, TargetLink users have convenient, direct access to SystemDesk’s V-ECU generation and VEOS’ simulation capabilities for evaluating and testing the behavior of TargetLink components as parts of complex systems in early development stages with constant interfaces during the different development stages.

![Diagram showing the exchange of SWC containers between TargetLink and SystemDesk](image-url)
**TargetLink Connection to VEOS®**

TargetLink code can run not only in SIL/PIL simulations in the Simulink environment, but also on dSPACE’s PC-based simulator VEOS® (figure below). This is done by wrapping code in an executable unit called a virtual ECU (V-ECU), which is generated from TargetLink models. TargetLink code can then be executed along with other V-ECUs as well as plant models on VEOS. To capture signals and adjust or test parameters, A2L files exported from TargetLink are used with dSPACE ControlDesk Next Generation, which provides access to VEOS. Thus, the standard dSPACE tools used for real-time simulation are also used for offline simulation and experimentation with TargetLink-generated code.

This provides the following benefits for TargetLink users:

- Early simulation of TargetLink code in large systems consisting of multiple virtual ECUs and plant models, including buses if required
- Convenient experimentation and testing with ControlDesk Next Generation and other dSPACE Tools

The approach shown in the figure below is intended for TargetLink users who do not work on AUTOSAR projects. For AUTOSAR projects, it is recommended to exchange AUTOSAR software components between SystemDesk and TargetLink, and generate the V-ECU from SystemDesk.

Connection of TargetLink to VEOS for non-AUTOSAR users.
**TargetLink FMI Support**

As of TargetLink 4.1, FMUs based on the FMI 2.0 for Co-Simulation standard can be exported directly from Simulink/TargetLink models. These FMUs encapsulate the TargetLink-generated production code and can therefore be integrated and simulated in all environments that suitably support the FMI standard. This can be done for real-time and non-real-time simulation platforms. As a result, software developers can create production code software in a familiar development environment and then import and reuse it in different environments without any manual effort.

Exporting Functional Mock-up Units from TargetLink to simulation environments that support FMI.
Software Quality and Standards

As a company committed to software quality, dSPACE applies a variety of methods to ensure the highest standards for its software products.

**TargetLink Certified for ISO 26262 and IEC 61508**

TargetLink is certified by TÜV SÜD (German certification authority) for use in the development of safety-related systems. TÜV confirmed that TargetLink is suitable for software development according to ISO 26262, IEC 61508 and derivative standards (such as EN 50128, which governs safety-related software on the railways). The certification was based on several areas:

- Software development process and software modification process of TargetLink
- Problem handling procedures
- Fitness for purpose in safety-related development according to IEC 61508 and ISO 26262

TÜV SÜD also approved a reference workflow providing guidance for the model-based development of safety-related software with TargetLink.

IEC 61508 is the internationally recognized generic standard for the development of safety-related electronic systems. The automotive standard ISO 26262, which is derived from IEC 61508, is the new standard for the development of safety-related systems in road vehicles.

**ISO/IEC 15504-Compliant Development Process**

ISO/IEC 15504 (also known as SPICE1) is an international standard for software processes. Its underlying concept is that a mature software product requires a mature development process. dSPACE has dedicated itself to an ISO/IEC 15504-compliant development process.

**Internal Software Quality Management**

An internal quality department, the dSPACE quality management team, proactively manages software quality at dSPACE. The team leads software improvement activities, sets internal standards, conducts internal assessments, and provides consultation services to all software groups. It acts independently and ensures that the highest product quality goals are consistently achieved and sustained.

---

1) **SPICE: Software Process Improvement and Capability Determination**
MISRA C

The British MISRA\(^1\) C standard is a widely accepted C subset for projects in the automotive industry. Its aim is to define rules for avoiding common software errors that can occur when software engineers write software by hand. Most of these rules also make sense for machine-generated code. TargetLink-generated code respects the vast majority of MISRA C rules. If deviations from the MISRA C standard are a technical necessity, they are identified and well documented. dSPACE makes this document available to all TargetLink customers.

MISRA Modeling Guidelines for TargetLink

MISRA has published official MISRA modeling guidelines for TargetLink. TargetLink is the first production code generator for which such guidelines have been issued. With these guidelines, function and production code developers have an official standard that gives them support for the design and implementation of vehicle functions. The guidelines formulate requirements for modeling in TargetLink with regard to functional safety and are particularly relevant to safety-critical projects. The MISRA TargetLink modeling guidelines (MISRA AC TL) can be obtained from the official MISRA website [www.misra.org.uk](http://www.misra.org.uk).

Certified Code

With the DO-178C as the future relevant standard for the development of software in aviation, model-based design and automatic code generation will have a solid base for use in the aerospace sector. The document DO-331, Model-Based Development and Verification Supplement to DO-178C and DO-278A, which is also part of the standard, was written specifically for this. dSPACE takes this into account by providing a workflow document that explains how to use TargetLink in a model-based tool chain for DO-178C-compliant projects. The workflow document describes how to meet the individual requirements or “objectives” of DO-178C/DO-331. It focuses not just on TargetLink itself but also on a complete model-based tool chain that can contain further third-party tools, for example, from TargetLink cooperation partners such as BTC Embedded Systems, AbsInt and Model Engineering Solutions. The workflow document is thus an important contribution towards simplifying the certification of TargetLink-generated code in DO-178C-compliant applications, addressing all criticality levels up to Level A.

You can obtain the approximately 60-page document from TargetLink.Info@dspace.de.

---

\(^1\) MISRA: Motor Industry Software Reliability Association ([www.misra.org.uk](http://www.misra.org.uk))
TargetLink Partner Program

The TargetLink® Partner Program was designed for companies that develop and promote products for complementing and expanding TargetLink's functionalities. It supports customers when they launch a model-based development process and a tool chain around TargetLink.

There are two kinds of TargetLink partners:

**TargetLink Partner**
This kind of partnership offers a company or organization the possibility to cooperate closely with a market leader in embedded software development.

**TargetLink Strategic Partner**
A TargetLink Strategic Partner is a company or organization that develops and promotes a complementary product which supports TargetLink in such a manner that it goes above and beyond the normal support for comparable tools.

List of TargetLink Partners:
- www.dspace.com/go/tlpartnerprogram

Current TargetLink Partners

<table>
<thead>
<tr>
<th>TargetLink Strategic Partners</th>
<th><a href="http://www.btc-es.de">www.btc-es.de</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>BTC Embedded Systems AG</td>
<td></td>
</tr>
<tr>
<td>Model Engineering Solutions GmbH</td>
<td><a href="http://www.model-engineers.de">www.model-engineers.de</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TargetLink Partners</th>
<th><a href="http://www.absint.com">www.absint.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsInt Angewandte Informatik GmbH</td>
<td></td>
</tr>
<tr>
<td>Emerge Engineering GmbH</td>
<td><a href="http://www.emerge-engineering.de">www.emerge-engineering.de</a></td>
</tr>
<tr>
<td>PikeTec GmbH</td>
<td><a href="http://www.piketec.com">www.piketec.com</a></td>
</tr>
<tr>
<td>Reactive Systems Inc.</td>
<td><a href="http://www.reactive-systems.com">www.reactive-systems.com</a></td>
</tr>
<tr>
<td>Schaeffler Engineering GmbH</td>
<td><a href="http://www.schaeffler-engineering.com">www.schaeffler-engineering.com</a></td>
</tr>
</tbody>
</table>
Third-Party Tools and Services for TargetLink

This page provides an overview of complementary products and services that support you in establishing a model-based development process and tool chain built around TargetLink.

List of Third-Party Tools
- www.dspace.com/go/tlpartnertools

<table>
<thead>
<tr>
<th>Topic</th>
<th>Product</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements tracing</td>
<td>TargetLink® in combination with Simulink® Verification and Validation</td>
<td><a href="http://www.dspace.de">www.dspace.de</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.mathworks.de">www.mathworks.de</a></td>
</tr>
<tr>
<td>Guideline checking</td>
<td>MES Model Examiner® (MXAM)</td>
<td><a href="http://www.model-engineers.de">www.model-engineers.de</a></td>
</tr>
<tr>
<td>Model analysis and review</td>
<td>MES M-XRAY®</td>
<td><a href="http://www.model-engineers.de">www.model-engineers.de</a></td>
</tr>
<tr>
<td>Model coverage analysis</td>
<td>BTC EmbeddedTester®</td>
<td><a href="http://www.btc-es.de">www.btc-es.de</a></td>
</tr>
<tr>
<td></td>
<td>Simulink Verification and Validation</td>
<td><a href="http://www.mathworks.de">www.mathworks.de</a></td>
</tr>
<tr>
<td>Test vector generation</td>
<td>BTC EmbeddedTester</td>
<td><a href="http://www.btc-es.de">www.btc-es.de</a></td>
</tr>
<tr>
<td></td>
<td>Reactis®</td>
<td><a href="http://www.reactive-systems.com">www.reactive-systems.com</a></td>
</tr>
<tr>
<td>Model-based testing</td>
<td>BTC EmbeddedTester</td>
<td><a href="http://www.btc-es.de">www.btc-es.de</a></td>
</tr>
<tr>
<td></td>
<td>MES Test Manager® (MTest)</td>
<td><a href="http://www.model-engineers.de">www.model-engineers.de</a></td>
</tr>
<tr>
<td></td>
<td>TPT</td>
<td><a href="http://www.piketec.com">www.piketec.com</a></td>
</tr>
<tr>
<td>Formal verification</td>
<td>BTC EmbeddedSpecifier®</td>
<td><a href="http://www.btc-es.de">www.btc-es.de</a></td>
</tr>
<tr>
<td></td>
<td>BTC EmbeddedValidator®</td>
<td><a href="http://www.btc-es.de">www.btc-es.de</a></td>
</tr>
<tr>
<td>Code coverage analysis</td>
<td>BTC EmbeddedTester</td>
<td><a href="http://www.btc-es.de">www.btc-es.de</a></td>
</tr>
<tr>
<td>Run-time error analysis</td>
<td>Astrée</td>
<td><a href="http://www.absint.com">www.absint.com</a></td>
</tr>
<tr>
<td></td>
<td>Polyspace Code Prover</td>
<td><a href="http://www.mathworks.de">www.mathworks.de</a></td>
</tr>
<tr>
<td>Timing analysis</td>
<td>aIT WCET Analyzer</td>
<td><a href="http://www.absint.com">www.absint.com</a></td>
</tr>
<tr>
<td></td>
<td>TimingProfiler</td>
<td></td>
</tr>
<tr>
<td>Stack usage analysis</td>
<td>StackAnalyzer</td>
<td><a href="http://www.absint.com">www.absint.com</a></td>
</tr>
<tr>
<td>Requirements coverage analysis</td>
<td>BTC EmbeddedTester</td>
<td><a href="http://www.btc-es.de">www.btc-es.de</a></td>
</tr>
<tr>
<td>Training and development platform</td>
<td>MEDKit on ARM – Motor Control Education Kit</td>
<td><a href="http://www.emerge-engineering.de">www.emerge-engineering.de</a></td>
</tr>
<tr>
<td>ISO 26262 consulting services</td>
<td>MES ISO 26262 Consulting Services</td>
<td><a href="http://www.model-engineers.de">www.model-engineers.de</a></td>
</tr>
</tbody>
</table>
Model Compare
Comparing of Simulink®, Stateflow®, and TargetLink® models

Highlights
- Graphical display of differences
- Generation of difference reports
- Built-in support for TargetLink
- Flexible difference filters
- Review support
- Merge support for models
- Command line interface for tool automation
- 32-/64-bit MATLAB® support

Application Areas
Automatic comparison is a must whenever you work with multiple versions of a model. Model Compare from dSPACE can be used with any TargetLink, MathWorks® Simulink or Stateflow model, and also supports libraries. There is a broad range of use cases for Model Compare, for example:

- Managing different model versions or model variants
- Merging parallel development branches
- Verifying and reviewing model changes

Key Benefits
Model Compare finds all the changes in a model. Even large models can be compared in minutes, which would be practically impossible without tool support. You can use filters to focus on relevant differences and ignore unimportant ones, such as layout changes or simulation settings. The support for review sessions also enables reliable, safe, and controlled reviews of model changes. The merge support simplifies the synchronization of changes in different versions of a model.

TargetLink Support
Model Compare has built-in support for all TargetLink blocks and properties. They are displayed and handled like ordinary Simulink/Stateflow elements, so you do not have to bother with TargetLink implementation details. Since the semantics of TargetLink properties are known, they are also intelligently handled by the predefined filter options.

Review Sessions and Merge Support
You can associate review comments to block and property differences found by the tool, including date/author tracing. Complex reviews with multiple participants are supported. Detected changes can be transferred from one model to another to merge parallel development branches or manage different model variants. With easy-to-use commands, merging models this way is much less error-prone than it would be by hand. The remaining differences between the models are constantly kept up-to-date, so that you always see the current state of your work.

1) Limited availability outside of Europe and Asia, please contact dSPACE.
# Main Features and Benefits

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for TargetLink</td>
<td>Model Compare recognizes TargetLink blocks and properties, and handles them just like built-in blocks.</td>
<td>There is no need to bother with TargetLink implementation details.</td>
</tr>
<tr>
<td>Powerful comparison algorithm</td>
<td>Models from different MATLAB versions are compared; block correspondences are detected even if the names of the blocks have changed; parameter values are compared either in unevaluated form (e.g., &quot;Kp&quot;) or in evaluated form (e.g., &quot;5.4&quot;).</td>
<td>Conversion and upgrade problems can be addressed. You get concise comparison results even if blocks were renamed. Different workspace settings can be taken into account.</td>
</tr>
<tr>
<td>Flexible filter configuration</td>
<td>Model Compare can be configured to filter out unimportant differences, e.g., layout changes or simulation options, according to a variety of criteria. A comparison can be restricted to selected subsystems.</td>
<td>You can focus on the differences and model parts that are relevant to you. This greatly improves the efficiency of a comparison.</td>
</tr>
<tr>
<td>Convenient result display</td>
<td>The comparison results are displayed in synchronized tree views, with differences indicated by customizable color schemes. A statistics window displays the number of changed, added and removed elements.</td>
<td>You can easily see which elements correspond to each other. Added, removed and modified elements as well as the number and type of changes can be seen at a glance.</td>
</tr>
<tr>
<td>Review support</td>
<td>Comments can be associated with individual blocks and properties or with the complete comparison session.</td>
<td>Reviews are performed in a reliable, controlled and safe way. You can also use the comment function to structure your own working process.</td>
</tr>
<tr>
<td>Traceability from Model Compare to model</td>
<td>You can easily highlight Simulink and Stateflow elements by selecting them in Model Compare, and show any differences directly in the compared models.</td>
<td>It is easy to view the context of a change and all the differences in a subsystem.</td>
</tr>
<tr>
<td>Traceability from model back to Model Compare</td>
<td>You can trace model elements in Simulink back to Model Compare.</td>
<td>This is helpful to identify blocks and lines in Model Compare.</td>
</tr>
<tr>
<td>Merge support</td>
<td>The commands Copy to Right, Copy to Left, and Delete can be used to transfer changes from one model to another.</td>
<td>You can merge parallel development branches and transfer changes between different model variants.</td>
</tr>
<tr>
<td>Report generation</td>
<td>You can save comparison results and associated comments as HTML, PDF and XML reports, model screenshots can be integrated in the difference reports.</td>
<td>The information can be archived and published.</td>
</tr>
<tr>
<td>Tool automation</td>
<td>You can start the comparison via the command line, and reports can be generated automatically. Reports can be saved in XML format for easy processing by external tools.</td>
<td>You can process multiple models automatically and incorporate Model Compare into your own tool chain.</td>
</tr>
</tbody>
</table>

## Order Information

<table>
<thead>
<tr>
<th>Product</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Compare&lt;sup&gt;1)&lt;/sup&gt;</td>
<td>MOC</td>
</tr>
</tbody>
</table>

## Relevant Software

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required: Operating system</td>
<td>[<a href="http://www.dspace.com/goto?os_compatibility">www.dspace.com/goto?os_compatibility</a>]</td>
<td></td>
</tr>
<tr>
<td>Required: Integrated development environment</td>
<td>MathWorks® MATLAB®, Simulink®, Stateflow®</td>
<td></td>
</tr>
<tr>
<td>Optional: Production code generator TargetLink</td>
<td>See p. 4</td>
<td></td>
</tr>
</tbody>
</table>

<sup>1)</sup> Limited availability outside of Europe and Asia, please contact dSPACE.
NEW: Model Compare 2.6

<table>
<thead>
<tr>
<th>Improvement</th>
<th>Detailed Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hook functionality for customizing block comparison</td>
<td>• Clear and accurate representation of differences also for models from other domains (HIL, RCP, etc.)</td>
</tr>
<tr>
<td>Improved search functionality</td>
<td>• The search can also be applied to property values and provides advanced find options (e.g. regular expressions)</td>
</tr>
<tr>
<td>Faster opening of model elements by double-click</td>
<td>• Faster traceability to model elements (in addition to context menu command)</td>
</tr>
<tr>
<td>Save Model Copy command</td>
<td>• Manually backing up models directly from within Model Compare, e.g. before performing merge actions</td>
</tr>
<tr>
<td>New Quick Guide and demo models</td>
<td>• Quick access to Model Compare and the frequently used features</td>
</tr>
</tbody>
</table>

Graphical Display of Differences

Model Compare shows all model differences clearly arranged in two synchronized tree views, where changed, added and removed elements are indicated by customizable colors. While navigating through the model hierarchy, all property differences can be inspected in the Property Inspector. Predefined and flexible filter configurations improve the efficiency of the comparison and let you adjust the view to your individual needs.

Differences can be traced from Model Compare directly to the Simulink/TargetLink models, where the corresponding elements are indicated by customizable colors. Thus, the differences can be easily inspected in the context of the models. In addition, elements of Simulink/TargetLink models can also be traced back to Model Compare to inspect all their differences at a glance.

Graphical display of differences in Model Compare and directly in Simulink/TargetLink models.
Merge Support
With Model Compare you can eliminate differences between two Simulink/TargetLink models by merging the two models. Changes of model elements as well as individual properties can be transferred from one model to the other by commands Copy to Right, Copy to Left or Delete, if these changes exist in only one model.
A smart line handling mechanism automatically adjusts signal lines of copied or deleted blocks. If a block is copied, the related signal lines are copied as well. If a block is deleted, the related signal lines are automatically deleted or reconnected. All merge operations are logged in the merge log window.

Report Generation
Comparison results and associated comments can be saved as difference reports in HTML, PDF and XML format. The generated reports also include all the comments that were created during a review. Thus, Model Compare’s difference reports are also a means of filing review results. As of Model Compare 2.4, there are advanced configuration options for customizing the report according to individual preferences. You can specify the subsystem level up to which screenshots are provided and the level of detail of the report.
Review Support
With Model Compare you can associate review comments with the found block and property differences or even with the complete comparison session. Time stamps as well as author information are added automatically by the tool. Thus, Model Compare supports even complex reviews with multiple participants.

Advanced Filter Options
To make your work as efficient as possible, Model Compare provides several filter options. Thus, you can focus on the differences and model parts that are relevant to you and your current work: Via display filters you can specify which model elements to show in the hierarchy display. You can use predefined filters to focus on a specific kind of difference or filter out unimportant ones such as layout changes or simulation settings. You can also define your own filters to exclude element properties or even entire model elements from the comparison. To reuse the defined filter settings in other projects, you can save them as favorites or export them as XML files.
Customizing Block Comparison

In order to make the comparison as efficient as possible also for models from other domains, Model Compare 2.6 provides a new add-on mechanism. This mechanism lets you use hook scripts to integrate block-specific knowledge in the comparison of any number of Simulink-based models. Differences in mask variables or block dialog parameters can therefore be displayed immediately, a method that has already been established for TargetLink models. Thus, Model Compare now provides a concise and efficient model comparison also for models from other domains, such as rapid control prototyping (RCP) and hardware-in-the-loop (HIL) simulation.

Via hooks, you can format the display of differences of any block dialog parameters or mask values individually. This gives you a concise and efficient model comparison for any Simulink-based models, e.g., for RCP or HIL models.