MICROSAR

Product Information
# Table of Contents

1. MICROSMART - The Vector Solution for AUTOSAR ECU Software ................................................................. 3  
2. MICROSMART.OS ........................................................................................................................................... 9  
3. MICROSMART.COM – AUTOSAR Basic Software Modules for Communication ........................................... 12  
4. MICROSMART.CAN – AUTOSAR Basic Software Modules for CAN Communication ................................... 15  
5. MICROSMART.FR – AUTOSAR Basic Software Modules for FlexRay Communication .................................. 19  
6. MICROSMART.LIN – AUTOSAR Basic Software Modules for LIN Communication ...................................... 22  
7. MICROSMART.ETH – AUTOSAR Basic Software Modules for Ethernet-Based Communication .................. 25  
8. MICROSMART.CHARGE – Basic Software Modules for Communicating with External Infrastructure .......... 30  
9. MICROSMART AVB – Basic Software Modules for Audio/Video Communication via Ethernet ....................... 33  
10. MICROSMART.MEM – AUTOSAR Basic Software Modules for Memory Management ................................. 36  
11. MICROSMART.SYS – System-related Basic Software Modules for AUTOSAR ............................................ 39  
12. MICROSMART.DIAG – AUTOSAR-Compatible Implementation of the Diagnostic Standards UDS, OBD and J1939 .................................................................................................................. 44  
13. MICROSMART.MCAL – AUTOSAR Driver for Control of Microcontroller Peripherals ................................. 50  
14. MICROSMART.EXT – AUTOSAR Drivers for Control of External Devices ..................................................... 55  
15. MICROSMART.IO – AUTOSAR Input-Output Hardware Abstraction ............................................................. 59  
16. MICROSMART.RTE - The optimized Run-time Environment for Software Components per the AUTOSAR Standard ............................................................................................................................................... 61  
17. MICROSMART.AMD – AUTOSAR Monitoring and Debugging ...................................................................... 64  
18. MICROSMART Solutions................................................................................................................................. 68  
19. MICROSMART Safe – Functional Safety According to ISO 26262 up to ASIL D for ECU Software .................. 69  
20. MICROSMART Security – Access Security for AUTOSAR ECU’s ................................................................. 73  
21. MICROSMART.HSM – Vectors Solution for Hardware Security Modules ....................................................... 76  
22. MICROSMART Gateway – Basic Software for Gateway ECUs....................................................................... 81  
23. MICROSMART Multi-Core – The AUTOSAR Solution for Multi-Core Processors ........................................ 85  
24. MICROSMART Variant Handling – Solutions for Flexible Configurations in AUTOSAR ................................. 88  
25. MICROSMART J1939 – AUTOSAR Basic Software Modules Specially Designed for Heavy-Duty Vehicles .... 91  
26. MICROSMART vVIRTUALTarget – Virtual Integration with vVIRTUALTarget ................................................... 94  
27. MICROSMART POSIX – Connecting AUTOSAR Classic to POSIX Operating Systems ................................. 98  
28. MICROSMART.OTA – Basic Software Modules for Software Downloads .................................................. 101  
29. MICROSMART.IPC – AUTOSAR Basic Software Modules for Interprocessor Communications ..................... 103  
30. MICROSMART.SIP and MICROSMART.EIP – A Quick Start to Your AUTOSAR Project ............................. 108  
32. Additional Information ...................................................................................................................................... 116
MICROSAR is the AUTOSAR solution from Vector for your ECU software. MICROSAR consists of the MICROSAR.RTE and MICROSAR basic software modules (BSW), which cover all aspects of the AUTOSAR standard and many extensions and add-ons. Each AUTOSAR BSW module is assigned to a MICROSAR package. For detailed descriptions of individual packages and the MICROSAR.RTE, please refer to the following chapters. Vector combines and releases the BSW modules needed in individual "software integration packages" (SIP).

### Figure 1: The MICROSAR packages contain all modules of the AUTOSAR 4.3 standard

The table shows the MICROSAR packages.

<table>
<thead>
<tr>
<th>The MICROSAR Solution consists of packages that contain the BSW modules: Solution</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICROSAR AMD</td>
<td>Monitoring and debugging of application and MICROSAR BSW</td>
</tr>
<tr>
<td>MICROSAR AVB</td>
<td>Basic software modules for Audio/Video communication via Ethernet</td>
</tr>
<tr>
<td>MICROSAR MCAL</td>
<td>Driver for driving the microcontroller periphery</td>
</tr>
<tr>
<td>MICROSAR CAN</td>
<td>Basic software modules for CAN communication</td>
</tr>
<tr>
<td>MICROSAR COM</td>
<td>Basic software modules for network-independent communication and gateways</td>
</tr>
<tr>
<td>MICROSAR EXT</td>
<td>Driver for driving external chips</td>
</tr>
<tr>
<td>MICROSAR FR</td>
<td>Basic software modules for FlexRay communication</td>
</tr>
<tr>
<td>MICROSAR DIAG</td>
<td>Basic software modules for diagnostics</td>
</tr>
<tr>
<td>MICROSAR/IO</td>
<td>Interface between the microcontroller periphery and the application</td>
</tr>
</tbody>
</table>

1. MICROsar - The Vector Solution for AUTOSAR ECU Software
2. Figure 1: The MICROSAR packages contain all modules of the AUTOSAR 4.3 standard
3. The table shows the MICROSAR packages.
The MICROSAR Solution consists of packages that contain the BSW modules:

<table>
<thead>
<tr>
<th>Package</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICROSAR ETH</td>
<td>Basic software modules for Ethernet-based communication</td>
</tr>
<tr>
<td>MICROSAR.LIBS</td>
<td>AUTOSAR Libraries</td>
</tr>
<tr>
<td>MICROSAR.LIN</td>
<td>Basic software modules for LIN communication</td>
</tr>
<tr>
<td>MICROSAR.MEM</td>
<td>Basic software modules for managing nonvolatile memory</td>
</tr>
<tr>
<td>MICROSAR.OS</td>
<td>Real-time operating system according to the AUTOSAR standard</td>
</tr>
<tr>
<td>MICROSAR.RTE</td>
<td>Optimized run-time environment for software components according to the AUTOSAR standard</td>
</tr>
<tr>
<td>MICROSAR Safe (CRYPTO, HSM)</td>
<td>Safety per ISO 26262 up to ASIL D for ECU software comprises the clusters CRYPTO, HSM and single modules from other clusters.</td>
</tr>
<tr>
<td>MICROSAR.SYS</td>
<td>System-related basic software modules for AUTOSAR ECUs</td>
</tr>
<tr>
<td>MICROSAR V2G</td>
<td>Basic software modules for communication with external infrastructure</td>
</tr>
<tr>
<td>MICROSAR XCP</td>
<td>Measurement and calibration of an AUTOSAR-ECU using XCP incl. transport layer for Ethernet, FlexRay und CAN</td>
</tr>
<tr>
<td>MICROSAR.OTA</td>
<td>Over the air</td>
</tr>
</tbody>
</table>

In addition to the sub-packages, MICROSAR also includes solutions for different questions, where modules from various packages could be involved. These solutions are listed in the following table and they will be explained in detail after the description of the packages.

These solutions include:

<table>
<thead>
<tr>
<th>Package</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICROSAR Amd</td>
<td>The AMD package simplifies the testing and analysis of BSW and application functions by transferring important status information and events to tools such as CANape or CANoe. In addition, AMD offers the possibility to perform runtime measurements on the BSW and application functions. For further information, please refer to the chapter on MICROSAR Amd.</td>
</tr>
<tr>
<td>MICROSAR gw</td>
<td>Gateways connect ECUs across the vehicle’s heterogeneous network architecture. At this, a gateway must allow managing the performance trade-off between throughput, latency and resource-consumption while providing flexibility and extensibility in both function and configuration.</td>
</tr>
<tr>
<td>MICROSAR HSM</td>
<td>MICROSAR-HSM stands for including efficient cryptographic algorithm processing and acceleration as well as key management for e.g. secure boot and Secure OnBoard Communication (SecOC).</td>
</tr>
<tr>
<td>MICROSAR J1939</td>
<td>J1939 is an established communication protocol, which uses CAN for communication in commercial vehicles and generators, and through derived standards in agricultural, maritime, construction and forestry vehicles.</td>
</tr>
<tr>
<td>MICROSAR Multicore</td>
<td>The Main motivations for multi-core is to increase the available computing power and to realize partitioning for safety critical applications. Both can be achieved with the MICROSAR multi-core solution consisting of base software and tooling. The MICROSAR multi-core base software focuses on execution time efficient provision of BSW services on all cores. It is complemented with the tooling which assists in the process of deployment and runtime evaluation.</td>
</tr>
<tr>
<td>MICROSAR.OTA</td>
<td>The MICROSAR.OTA bundle contains BSW modules for processing, saving and activating software updates for your vehicle. OTA stands for ‘Over the Air’.</td>
</tr>
<tr>
<td>MICROSAR POSIX</td>
<td>MICROSAR POSIX offers solutions to operate AUTOSAR basic software under a POSIX operating system. Standard functions, e.g. diagnostics, or proven application code can be transferred to POSIX-based projects.</td>
</tr>
<tr>
<td>MICROSAR Safe</td>
<td>Functional Safety according ISO 26262. Vector offers a complete solution for your AUTOSAR control unit for the use of MICROSAR BSW in safety-relevant functions.</td>
</tr>
<tr>
<td>MICROSAR Security</td>
<td>MICROSAR Security is the Vector solution for Automotive Cyber Security. We support you with embedded software, services and tools to protect embedded systems against cyber-attacks. Protect your product effectively and efficiently and use our competence and knowledge.</td>
</tr>
<tr>
<td>MICROSAR Variant Handling</td>
<td>To save logistics costs for AUTOSAR ECUs, MICROSAR modules are available with the Identity Manager option. This option allows several configurations (e.g. left or right door) to be stored in the control unit. This enables multiple installation of an identical control unit within a series or in different series. The add-on Post-Build Loadable allows you to modify many parameters of the BSW configuration without having to recompile the ECU software. For example, routing tables or transmission modes can be changed and extended without requiring the ECU’s build environment or commissioning a new ECU variant from the supplier.</td>
</tr>
</tbody>
</table>
1.1 Application Areas

The BSW modules of the MICROSAR packages assure basic functionality of the ECU. They contain the implementations of AUTOSAR standard services you will need for your functional software. You are able to develop your functional software platform independently, because the AUTOSAR architecture follows a consistent strategy of hardware abstraction.

Modules from the MICROSAR.OS and MICROSAR.MCAL packages are hardware-dependent. Vector offers these modules for a large number of different hardware platforms and compilers, e.g. to enable quick change-out of the controller device. The operating system MICROSAR.OS is available for single core and multi-core processors. Based on its ongoing contacts with OEMs, Vector is able to offer a number of OEM-specific BSW modules and extensions such as the diagnostic modules.

You can configure all necessary MICROSAR BSW modules based on your project’s requirements, and after generation you can integrate them with the functional software. This produces a complete set of ECU software. If the functional software consists of AUTOSAR-conformant SWCs, you will need a run-time environment (Rte). The MICROSAR.RTE implements communication between the SWCs and their access to data and services from the BSW modules. Along with managing the entire flow of events and information, the MICROSAR.RTE also assures consistency in the exchange of information and coordinates accesses across core or memory protection boundaries.

ECU projects without SWC architecture (and therefore also without Rte) are optionally supported by the Vector vBre (Vector Basic Runtime Environment). vBre simplifies the BSW integration by providing configurable BSW scheduling, critical section management and creation of type definitions for service layer BSW modules which normally would be created by the RTE. Hereby the vBre speeds up and simplifies building up AUTOSAR 4 based projects which do not have a Rte.

1.2 Properties

Development of the MICROSAR basic software modules is founded on the SPICE-based Vector development process for standard modules, which is based on SPICE. All MICROSAR packages offer the following features:

- Efficient memory utilization and short execution times
- Available for production use
- Available for AUTOSAR 4.x and 3.x
- Assistants and timely checks support you in achieving consistent configuration of your basic software
- Highly scalable, adaptable to your specific application
- Optimally integrated into your development process
- AUTOSAR Monitoring for testing and analysis of ECUs
- User-selectable configuration point (pre-compile, link-time or post-build)
- Supports multiple ECUs
- Optional delivery as source code
- Together with MICROSAR Safe: well-suited for safety-relevant functions (ISO 26262)

1.3 Production Use

The MICROSAR BSW modules are already being used in production projects. MICROSAR lets you benefit from Vector’s many years of experience in implementing embedded standard software. Before delivery, all MICROSAR Software modules undergo systematic integration tests for the specific conditions of your application (hardware platform, compiler, processor device, OEM, with/without Rte, etc.). Upon request, these tests may be extended to cover software modules from third-party producers (e.g. MCAL drivers).
1.4 Support of AUTOSAR 4.x and 3.x

Regardless of whether you use AUTOSAR 4.x or 3.x, from Vector you get the entire basic software from a single source. In migration of your projects, you benefit from the uniform development workflow for both AUTOSAR 4.x and 3.x:

- The configuration tools DaVinci Developer and DaVinci Configurator Pro are designed for both releases. This lets you avoid a change in tools.
- MCAL drivers from different AUTOSAR releases can be combined with MICROSAR.

In the case of a migration from AUTOSAR 3 to 4, we can support you in adapting your application software to the interfaces that were changed in the AUTOSAR 4.x standard.

Another advantage of MICROSAR lies in its many extensions in the BSW modules for AUTOSAR 3.x, which are specified in AUTOSAR 4.x. Some examples are the multi-core operating system, as well as support for J1939, XCP and Ethernet/IP, which are already available for AUTOSAR 3.x. The MICROSAR.MOST bundle is a Vector supplement to the standard and is compatible with AUTOSAR 4.x and AUTOSAR 3.x.

1.5 Consistent and Simple Configuration

With AUTOSAR, the work of manually developing or adapting the basic functionality of ECU software is replaced by configuration of the BSW modules. The intuitive, user-friendly and well-coordinated AUTOSAR tools from Vector (DaVinci) support the user in this process. Multi User Support of the DaVinci tools enables simultaneous work on a project by multiple users. The DaVinci tools require an “ECU Extract of System Description” file as input. It is also possible to produce a configuration based on commonly used network description files (DBC, FIBEX, LDF, etc.).

Early in the configuration process, all DaVinci tools check the validity of individual parameters, complex parameter groups and their interrelationships. In case of invalid configurations, the tools make recommendations for corrections, if possible.

This extension of the AUTOSAR method simplifies integration of the basic software in your ECU and reduces integration time.

![Figure 2: You use DaVinci Configurator Pro to configure the BSW modules and the Rte.](image-url)
The DaVinci tools optimally assist you in configuring the Rte and the BSW modules. In a bottom-up process, for example, the SWC service ports (including runnables) are automatically generated to match the BSW configuration. This automation relieves you of tasks that recur frequently and are prone to errors when performed manually. This saves you time and costs.

For more details on DaVinci tools from Vector, please refer to the relevant datasheets.

1.6 Scalability

In addition to fulfilling AUTOSAR requirements, the MICROSAR BSW modules also provide a number of functional extensions. The extended configuration options let you deactivate unnecessary functions to optimize the MICROSAR code for your application. This scalability makes the MICROSAR modules the optimal solution for both small and challenging applications. MICROSAR is already being implemented in a wide range of ECUs, such as steering angle sensors, door ECUs, engine ECUs, central gateways, etc. MICROSAR may also be used with other operating systems such as Linux or QNX.

1.7 User-Selectable Time Point for BSW Configuration

The configuration point of all MICROSAR basic software modules is user-selectable. You can select the configuration point for each BSW module; choices are: pre-compile, link-time or post-build.

1.8 Scope of Delivery

The standard scope of delivery includes the following components:

- Software modules
- Command line based generator (for Windows XP/Windows 7)
- BSW Module Description
- Documentation
Additional or alternative components are listed below - separately for each module. For a comfortable configuration we recommend our DaVinci Configurator Pro. For details please see the separate Product Information.

1.9 Delivery of Source Code
The MICROSAR modules are delivered as source code with very few exceptions. The source code lets you make pre-compile optimizations and simplifies testing.

1.10 License and Maintenance
Vector offers you flexible licensing – individually tailored to your requirements. In the framework of a maintenance contract, you get software updates that keep you current to the latest level of development.

1.11 Additional Services
- Consulting on system design
- Extension of MICROSAR BSW modules per customer wishes
- Development of customer-specific software components (SWC)
- Support in adapting existing functional software
- Complete software integration in your ECU – including third-party software
- Migration of existing software into an AUTOSAR-based concept
- Hotline, special workshops and training courses on the topic of embedded software and AUTOSAR

1.12 The complete AUTOSAR solution from Vector
The Vector AUTOSAR solution consists of the DaVinci tools, the MICROSAR BSW and the MICROSAR.RTE. You will find information on general properties of the BSW modules from the MICROSAR packages in the following chapters. Details on functional features of the individual DaVinci Tools are found in the particular product information.

1.13 Contact and Availability
MICROSAR BSW modules are available for a large number of commonly used microcontrollers and in OEM-specific variants. You can obtain more information at www.microsar.com/availability/ or upon request.

Please find your contact person at www.vector.com/contact.

1.14 Training
In the framework of our training program, we can offer various training events and workshops for MICROSAR in our classrooms as well as at your business site.

For more information on individual training events and dates please visit: www.vector-academy.com
2 MICRO SAR.OS

MICRO SAR.OS is a pre-emptive real-time multitasking operating system with optimized properties for use on microcontrollers. Vector’s many years of experience in developing operating systems and drivers for microcontrollers are bundled into this small, robust operating system core.

Lean Hypervisor is a module to ensure a safe startup of multiple operating system partitions in a multicore processor or SoC. It initializes the system MPU and starts the different partitions.

Figure 4: MICRO SAR.OS module according to AUTOSAR 4.3

2.1 Overview of Advantages

- Small, quick, resource-economizing operating system and with short boot times
- MICRO SAR.OS is available for AUTOSAR 4.x and 3.x
- Available as multi core operating system
- Graphic configuration tool for easy configuration
- OS: Available for many 16, 32 and 64 bit microcontrollers
- OS: Available as multi-core operating system
- Lean Hypervisor: Safe startup of multiple operating system partitions
- Lean Hypervisor: Implementation according ISO 26262 ASIL D
- Lean Hypervisor: Freely selectable master core

2.2 Application Areas

MICRO SAR.OS is based on the AUTOSAR OS specification, an extension of the practice-proven OSEK/VDX-OS operating system standard. This standard was extended by Vector to include functions for time monitoring and memory protection.
The implemented high resolution timer mechanism for example offers time resolutions of less than 1ms without increasing the interrupt load. Depending on the controller, this may enable resolutions into the microseconds range. Find more information below.

MICROSAR.OS is fully conformant to the AUTOSAR OS specification and supports all scalability classes.

Protected by the System Memory Protection Unit, the operating system partitions run without the risk of mutual interference due to incorrect data changes. In this way, partitions with different ASILs can be operated in parallel.

The LeanHypervisor is implemented according to ISO26262 ASIL D, programs the system MPU during system startup and then starts the operating system partitions. Protected by the System Memory Protection Unit, the operating system partitions run without the risk of mutual interference due to faulty data changes. This allows partitions with different ASILs to be operated in parallel. The master core is freely selectable. If the hardware does not start an ASIL-compatible kernel after a reset, the protection initialization can be assigned to another ASIL-compatible kernel.

2.3 Modules and Their Add-Ons

- **Os (SC1) – Operating System Scalability Class 1**
  Real-time operating system implemented per the OSEK/VDX-OS standard and extended to include schedule tables
  - extended to include schedule tables

- **Os (SC2) – Operating System Scalability Class 2**
  Real-time operating system with time synchronization and monitoring of the time behavior of individual tasks and interrupt service routines
  - Timing protection ensures that assumptions made in the early design phase related to execution times are preserved during run-time as well. As a result, a defective application section cannot impair the execution times of other running processes.
  - Measuring execution times and interrupt blocking times of applications. These measured data can later be used as practice-based values in designing and integrating future applications.

- **Os (SC3) – Operating System Scalability Class 3**
  Real-time operating system with memory protection mechanisms on microcontrollers with suitable hardware support
  - Memory protection assures that application components do not mutually destroy data. This makes the integration of applications easier and more reliable.

- **Os (SC4) – Operating System Scalability Class 4**
  Combines scalability classes SC2 and SC3.

**Add-Ons**

The following Add-Ons are available for all OS independent of the Scalability Class (SC1 to SC4).

- **Multi-Core (symmetric):** The multi-core add-on (symmetric) is used where a multi-core system is to be developed according to the AUTOSAR specification and the cores have the same set of commands. It is based on the AUTOSAR specification 4. x, but can also be used in AUTOSAR 3. x projects.

- **Multi-Core (asymmetric):** The add-on multi-core (asymmetric) is used where a multi-core system is to be developed according to the AUTOSAR specification and where the cores have a different set of commands. It is based on the AUTOSAR specification 4. x, but can also be used in AUTOSAR 3. x projects.

- **vlhyp - Lean Hypervisor**

  In a multi-core application, the module LeanHypervisor initializes the System Memory Protection Unit (MPU) at start-up and manages the start of the cores. Each core may have its own operating system image. Any combination of POSIX, Classic or Adaptive AUTOSAR operating systems is feasible.

  Protected by the System Memory Protection Unit the various partitions can be operated free from interference regarding memory access, thus allowing to run partitions with differing ASIL in parallel.

  The master core is free selectable. Protection initialization can be assigned to an ASIL-compliant core in case that the core started after power-up or reset does not fulfill ASIL requirements.
LeanHypervisor makes partitions independent. Each partition can focus on its local memory protection requirements. In addition, partitions do not have to implement a synchronized wait state for the basic initialization. This reduction of external dependencies facilitates the development of the single partition.

LeanHypervisor is developed according to ASIL D as Safety Element out of Context (SooC). LeanHypervisor will not require CPU load after the partitions are started.

2.3.1 High Resolution Timer (HRT)

AUTOSAR operating systems usually use a periodic timer (PIT) to generate the system time. Then, the time resolution is given by the cycle time used to trigger the system counter. Each trigger corresponds to an interrupt. Generally, the integration engineer must find a trade-off between interrupt load and time resolution. Sometimes an application requires a time resolution which is better than any acceptable trade-off.

2.4 Graphical Configuration and Generation Tool

For a comfortable configuration we recommend the DaVinci Configurator Pro. It contains consistency checks and the call of the generator. The generator is implemented as a command-line tool to enable its integration in an automated development environment.

2.5 Scope of Delivery

The MICROSAR.OS product is supplied with the following components:

- Operating system core as source code
- DaVinci Configurator Base as basic editor
- Command line based generator
- BSW modules description
- Documentation
3 MICROSCAR.COM – AUTOSAR Basic Software Modules for Communication

The basic software modules (BSW) of MICROSCAR.COM include AUTOSAR services for ECU communication. These services can support any number of communication channels. They are bus-independent and are needed in every communication stack. Per AUTOSAR architecture, they handle control and full integration in the ECU software of bus-specific communication modules for CAN, CAN-FD, J1939, FR, LIN and ETH.

![Figure 5: MICROSCAR.COM modules according to AUTOSAR 4.3](image)

### 3.1 Overview of Advantages

- Code and execution time optimized by application-specific configuration
- Available for AUTOSAR 4.x and 3.x
- Contains many useful extensions beyond the AUTOSAR standard, see chapter "functions"
- Extended support for Nm coordinators, see chapter "functions"
- NM module: OSEK Nm compatibility is configurable
- Supports simultaneous operation of AUTOSAR Nm and OSEK Nm in Nm migration projects
- Very efficient signal access via function macros (for AUTOSAR 3)

### 3.2 Application Areas

MICROSCAR.COM lets users develop their functional software by a fully bus-independent approach. All necessary tasks for transmitting messages and for cross-bus network management activities are handled by the configurable BSW modules Com, Nm, PduR and IpduM from MICROSCAR.COM.

For a gateway ECU, you do not need any additional software. The Com and PduR BSW modules from MICROSCAR.COM enable routing of signals and Tp or application messages.
3.3 Modules and Their Add-Ins

Each module of MICROSAR.COM can have extensions of Vector that go beyond the AUTOSAR standard and have additional add-ons.

> **Com - Communication**

> Services of the Com module organize transmission of messages according to their send type (cyclic, event-triggered, etc.). A key task is to implement bus-independent signals of the functional software in PDUs.

**Extensions to AUTOSAR Standard**

> Invalidity declaration of TX signals in case of RX signal timeout
> Optimizations for mainfunction run-time reduction (Rx-sided: caching of received events, Tx-sided: configuration of multiple time domains.)
> Deferred Event Caching of Rx IPDUs. This is an optimization of the Rx mainfunction run-time via an event triggered processing of Rx PDUs. The optimization allows to waive cyclic polling of all PDUs in the mainfunction Rx.

**Add-Ons**

> **GW for Com:** The Com module can be supplied with gateway functionality. Routing is possible for signals and signal groups. Routing in Com is possible via a configuration description without existence of a real signal or signal group.
> **Com Add-On HighEndFeatures:** this option activates the following functions for COM modules:
>  - description based routing: this additional routing option allows to route PDU segments (defined by their start bit and length) including class of emission management (periodic, event triggered, on changes). This creates a performant alternative to signal based routing and to cycle-retarding PDU routing. This function requires the above mentioned gateway option.

> **PduR – PDU Router**

The PDU Router (PDUR) provides the modules Com, Dcm and the complex drivers with an interface to the communication modules (interface, transport protocol and network management) of the different bus systems. This interface serves to transmit and receive data via PDUs. The PduR also implements a gateway between the communication modules of the various bus systems. The MICROSAR module CDD allows TP- and IF-PDUs to be integrated into the Com stack:

> above or below the PDU router
> above the communication interfaces

**Add-Ons**

> **GW for PDU**
>  - TP and message routing
>  - Routing via meta data in case of range routing
>  - Routing of variable adresses ("dynamic gateway")
>  - Routing of dynamic PDU lengths

> **Nm – Network Management**

The Network Management Interface (Nm) bundles inter-bus network management activities of all of the ECU’s communication channels. As NM coordinator, it synchronizes wake-up and sleep of the communication channels.

**Extensions to AUTOSAR Standard**

> Synchronous sleep and wake-up of multiple networks via different Nm coordinators
> Backup coordinator
> Support of OSEK Nm (configurable)
> Mixed operation of OSEK and AUTOSAR-Nm on one channel

> **IPduM – IPDU Manager**
The optional I-PDU Multiplexer (IpuM) module supports multiple usage of frames with different data contents, via a static configuration for the classic bus systems or alternatively via dynamic data content mapping for CAN-FD.

> **SecOC – Secured OnBoard Communication**

For details to the Secured OnBoard Communication (SecOC), please refer to the chapter "MICROSAR Security".

> **Transformer**

Allow efficient transmission of complex data structures and big PDUs over the network.

> **ComXf:** Allows efficient signal groups with many group signals. The placement is derived from the system extract.

> **SomeIpXf:** Provides a serialization strategy for various data types. LDCOM can be used here for a highly efficient transmission.

> **E2eXf:** Enables end-to-end protection for network communication that is managed using the AUTOSAR transformer concept (i.e. serialization through COMXF or SOMEIPXF).

> **vMirror – Gateway Mirroring**

Gateway Mirroring: allows mirroring internal busses to the diagnostic access which enables to read normally inaccessible messages on the bus and to identify problems. In the base version, this function allows to mirror one internal CAN or LIN channel to diagnostic CAN.

**Add-Ons**

> **ETH:** mirror multiple CAN, LIN, FR and ETH channels to diagnostic ETH

> **FR:** mirror CAN, LIN or FR channels to diagnostic CAN or FR

### 3.4 Configuration

For convenient configuration we recommend our DaVinci Configurator Pro. For more details, see the separate datasheet.

Using DaVinci Configurator Pro, you can easily link the MICROSAR.COM module PduR and the modules CanIf, LinIf, FrIf and EthIf (SoAd) to your complex drivers by configuration.

![Figure 6: Configuration of the communication modules with DaVinci Configurator Pro](image)
MICROSAR.CAN – AUTOSAR Basic Software Modules for CAN Communication

The MICROSAR.CAN package contains the BSW modules defined in the AUTOSAR architecture for CAN communication: CanIf, CanNm, CanTp, CanSm and optional modules for J1939 and Xcp.

4.1 Overview of Advantages

- Available for AUTOSAR 4.x and 3.x
- Contains numerous useful extensions
- Code and execution time are optimized based on need-specific configuration
- Inter-module configuration of all communication-specific software modules
- Fast wakeup handling at ECU startup
- CanTp: ISO 15765-2 compatibility is configurable
- OSEK Nm is available as a configurable module, compatible with CanNm
- CanNm, CanSm: Control of the communication stack (on and off) depending on the partial network state
- CAN-FD: Support of up to 64 Byte data with enhanced bandwidth. Available for many CAN-FD controllers.

4.2 Application Areas

MICROSAR.CAN is used to handle communication in CAN networks. It is also well-suited as a foundation for calibration with XCP, gateways or re-programming. You can also extend MICROSAR.CAN with the optional J1939TP package to enable operation of an AUTOSAR ECU in a J1939 net-work. The BAM and CMDT transport protocols are available for this.

4.3 Modules and Their Add-Ons

The BSW modules in MICROSAR.CAN contain functions defined in AUTOSAR 4.x. Beyond the standard itself, MICROSAR.CAN offers the following important services:
The module CanIf offers abstracted (PDU-based) access to the CAN Driver. It controls the CAN Driver (Can) as well as the transceiver driver (CanTrcv).

Extensions to AUTOSAR Standard

> Double hash search algorithm for efficient filtering the Rx messages

**CanNm – CAN Network Management**

Within a CAN network, the module CanNm is responsible for coordinated transitions between the wake up and sleep state.

Extensions to AUTOSAR Standard

> Pre-compile optimizations, e.g. for single-channel systems

**CanTp – CAN Transport Protocol**

The CanTp module conforms to ISO standard 15765-2. As the transport protocol for CAN, it is responsible for segmenting the data in the Tx direction, collecting data in the Rx direction and monitoring the data stream.

Extensions to AUTOSAR Standard

> Pre-compile optimizations, e.g. for single-channel systems
> Supports mixed addressing (11 bit CAN ID); typically for CAN/LIN gateway applications
> Optimized routing (e.g. with Burst Transmission) together with the PduR from MICROSAR COM
> ISO 15765-2 compatibility is configurable

**CanSm – CAN Station Manager**

The module CanSM is responsible for the bus-specific error handling.

Extension to AUTOSAR Standard

> Support of ECU Passive Mode

**CanTsyn – CAN Time Synchronization**

This module CanTsyn realizes the CAN specific time synchronization protocol. An access to the synchronized time base by the SWCs requires the Synchronized Time-Base Manager (StbM).

Extensions to AUTOSAR Standard

> Time Synchronization over CAN (CanTsyn) implements the time synchronization protocol for CAN. This permits clock synchronization between CAN ECUs as a part of the AUTOSAR Global Time Concept. The Synchronized Time Base Manager (StbM) BSW module from MICROSAR.SYS is available as a higher-level time coordinator.

**CanXcp – CAN XCP Module**

Xcp is a protocol for communication between a master (PC tool) and a slave (ECU). It is standardized by ASAM and is used primarily for measuring, calibrating, flashing and testing ECUs. XCP supports the bus systems CAN (CanXcp), FlexRay (FrXcp), Ethernet (EthXcp) and LIN (vLINXCP).

4.3.1 The following functionalities and Modules are also available:

**J1939Nm – Network Management for J1939**

J1939 supports adding ECUs to networks on-the-fly. The module J1939Nm is responsible for negotiating a unique ECU address (“AddressClaim”) and unlike other NM modules for handling the wake-up or going to sleep of the bus.

Add-On

> Dynamic Nm: ECUs that change their address, or communicate with ECUs that change their address

**J1939Tp – Transport Protocol for J1939**

The J1939Tp module contains the transport protocols BAM (Broadcast Announce Message) and CMDT (Connection Mode Data Transfer) of the SAE J1939 standard.
Add-Ons

- **ISOBUS**: Extended Tp (ETP) and Fast Packet Tp (FPTP). Based on ISO 11783-2 and NMEA2000
- **J1939Rm** – Request Manager for J1939
  
  The module J1939Rm implements requesting of data via Request Handling that is defined in the SAE J1939 protocol, diagnostic module J1939DCM the

4.3.2 **Cross-package Add-Ons**

There are also add-ons that affect one or more packages. These are in detail:

- **MICROSAR.IDM** – Identity Manager, for details please refer to the chapter “MICROSAR Variant Handling”
- **MICROSAR.MC** – Multi-Core, for details please refer to the chapter “MICROSAR Multi-Core”
- **MICROSAR.PBL** – Post Build Loadable, for details please refer to the chapter “MICROSAR Variant Handling”

4.4 **Configuration**

For convenient configuration, we recommend using DaVinci Configurator Pro. For more details, please refer to the separate product information.

4.5 **Other MICROSAR Products for a Complete CAN Communication Stack**

Based on the AUTOSAR architecture, a complete communication stack for CAN can be formed using MICROSAR.CAN together with the BSW modules from the separately available MICROSAR.COM, MICROSAR.MCAL and MICROSAR.EXT packages. To interface MICROSAR.CAN to the application and the hardware, you will still need the following BSW modules:

- Hardware-specific CAN driver (CanDrv) from MICROSAR.MCAL
- Hardware-specific transceiver control (CanTrcv) from MICROSAR.EXT, also for partial networking
- General communication modules (Com, Nm, PduR, IpduM) from MICROSAR.COM

Modules in MICROSAR.MCAL and MICROSAR.EXT are available for many microcontrollers and transceivers.

4.6 **Other relevant MICROSAR Products for CAN**

- Dcm and Dem from MICROSAR.DIAG
- Det, EcuM and ComM from MICROSAR.SYS
- MICROSAR XCP lets you measure and calibrate per ASAM XCP. Here, the module was specifically optimized for use together with CANoe.XCP and CANoe.AMD as well as CANape. For CAN ECUs, MICROSAR XCP contains the suitable CANXCP transport layer.
- Beyond the AUTOSAR standard, MICROSAR XCP supports generic readout of measuring objects. As a result, no addresses must be defined and updated in the a2l file. Data from any version or variant can be extracted with an a2l file independent from the MCU build. The generic readout feature requires usage of CANoe.AMD or CANape as XCP tool.
- For safety reasons, very often it is not allowed to keep the measurement and calibration interfaces active in serial production projects. The module VX1000If allows to keep the VX1000 measurement and calibration hardware driver in the BSW also in serial production, but in a deactivated mode. Through an API the VX1000 driver can be released again for check and development purposes. Delivery of the module must take place within a MICROSAR SIP to receive approval for this use case in serial production. The activation of the VX1000 driver within a serial production project at runtime however is also not allowed when using the module VX1000if.
- You will find more information on J1939 ECUs in commercial vehicles in the section "MICROSAR J1939"
4.7 The Vector toolchain for developing CAN ECUs

Figure 8: Vector offers a comprehensive range of products and services for your CAN projects.
5  MICROSAR.FR – AUTOSAR Basic Software Modules for FlexRay Communication

Vector offers you MICROSAR.FR, an AUTOSAR-conformant package for FlexRay communication. It contains the following BSW modules defined in the AUTOSAR architecture: FrIf, FrNm, FrSm and a choice between FrTp and FrIsoTp. MICROSAR.FR can be extended with Xcp as an option.

![Diagram of FlexRay modules according to AUTOSAR 4.3](image)

5.1 Overview of Advantages

- Available for AUTOSAR 4.x and 3.x
- Activates/deactivates partial networks and provides data, depending on the partial network state
- Small code size and short execution times due to optimized administration of job lists of the FlexRay interfaces
- Either the FrTp (AUTOSAR) or FrIsoTp (ISO 10681) transport protocol may be used
- Support of ECU passive mode in the FlexRay State Manager
- Early detection of synchronization losses

5.2 Application Areas

MICROSAR.FR is used to handle communication in FlexRay networks including partial networking. Furthermore, it is ideal as a foundation for calibration with XCP, gateways or flashing.

5.3 Modules and Their Add-Ons

The BSW modules in MICROSAR.FR contain functions defined in AUTOSAR 4.x, where FrIsoTp is a supplement to AUTOSAR 3.x.

- **FrIf – FlexRay Interface**

  The module FrIf offers abstracted (PDU-based) access to the FlexRay hardware. In addition, it offers support for synchronization with the global FlexRay time.
Extensions to AUTOSAR Standard

- Support of the following APIs: CancelTransmit and L-PDU reconfiguration
- Dual channel redundancy for redundant transmission of frames and PDU-specific voting function for the SWCs
- Pre-compile optimizations, e.g. for single-channel systems

> **FrNm – FlexRay Network Management**

The module FrNm is responsible for network management in FlexRay. It synchronizes the transition to the bus sleep state.

Extensions to AUTOSAR Standard

- Pre-compile optimizations, e.g. for single-channel systems

> **FrSm – FlexRay State Manager**

The module FrSM controls and monitors the wake up and startup of nodes in the FlexRay cluster.

Extensions to AUTOSAR Standard

- Support of ECU Passive Mode, immediate startup after passive wakeup, extended error handling by State Change Notification, configurable time delay for FlexRay startup at passive wakeup as well as configurable number of wakeup patterns.

> **FrTp – FlexRay Transport Protocol**

FrTp is a FlexRay transport protocol and is based on the ISO 10681-2 standard.

Extensions to AUTOSAR Standard

- Pre-compile optimizations, e.g. for single-channel systems

> **FrArTp – FlexRay AUTOSAR Transport Protocol**

FrArTp is a FlexRay transport protocol. Based on ISO 15765-2 (CanTp), it contains a frame compatibility with the CAN bus.

> **FrTSyn – Time Sync Over FlexRay**

The module FrTSyn realizes the FlexRay specific time synchronization protocol. An access to the synchronized time base by the SWCs requires the Synchronized Time Base Manager (StbM).

Extensions to AUTOSAR Standard

- Time Synchronization over FlexRay (FRTSYN) implements the time synchronization protocol for FlexRay. This permits clock synchronization between FlexRay ECUs as a part of the AUTOSAR Global Time Concept. The Synchronized Time Base Manager (STBM) BSW module from MICROSAR.SYS is available as a higher-level time coordinator.

> **FrXcp – FlexRay XCP Module**

The FrXcp module contains FlexRay-specific contents of the XCP module (Xcp).

### 5.3.1 Cross-package Add-Ons

There are also add-ons that affect one or more packages. These are in detail:

- MICROSAR.IDM – Identity Manager, for details please refer to the chapter “MICROSAR Variant Handling”
- MICROSAR.PBL – Post Build Loadable, for details please refer to the chapter “MICROSAR Variant Handling”

### 5.4 Operating System

FlexRay basic software modules can be used entirely without an operating system. However, it makes sense to use an AUTOSAR OS or a conventional OSEK-OS (e.g. Vector osCAN). Ideally suited for FlexRay applications is MICROSAR.OS from Vector.
5.5 Configuration

For convenient configuration, we recommend the DaVinci Configurator Pro. For more details, please refer to the separate product information.

5.6 Other MICROSAR Products for a complete FlexRay Communication Stack

Based on the AUTOSAR architecture, a complete communication stack for FlexRay can be made with MICROSAR.FR together with the BSW modules from the separately available MICROSAR.COM, MICROSAR.MCAL, MICROSAR.SYS and MICROSAR.EXT packages. You will also need the following BSW modules to interface MICROSAR.FR to the application and hardware:

- Hardware-specific FlexRay driver (Fr) from MICROSAR.MCAL
- Hardware-specific transceiver control (FrTcv) from MICROSAR.EXT
- General communication modules (Com, Nm, PduR, IpduM) from MICROSAR.COM

The modules in MICROSAR.MCAL and MICROSAR.EXT are available for many different microcontrollers and transceivers.

5.7 The Vector toolchain for developing FlexRay ECUs

![Vector toolchain diagram]

Figure 10: Vector offers a comprehensive portfolio of products and services for your FlexRay projects
6  MICROSRAR.LIN – AUTOSAR Basic Software Modules for LIN Communication

MICROSAR.LIN contains the BSW modules defined in the AUTOSAR architecture for LIN communication: LinIf, LinSM and LinNm. According to AUTOSAR, LinTp is a part of LinIf. The LIN transport protocol is offered as an option, because not every LIN communication stack requires a transport protocol. Xcp for the MICROSRAR.LIN Master is also available as an ASAM extension. With AUTOSAR version 4.4 LIN is now available as master and slave.

![Figure 11: MICROSRAR.LIN modules according to AUTOSAR 4.3](image)

### 6.1 Overview of Advantages

- Available for AUTOSAR 4.x and 3.x
- Contains numerous useful extensions
- Minimized scheduling jitter for multi-channel master
- Optimized routing of diagnostic requests to LIN Slaves
- Quick start of the LIN channel
- Reliable switching of schedule tables possible
- Based on Vector’s many years of experience with production software for LIN

### 6.2 Application Areas

MICROSAR.LIN handles communication tasks for a LIN Master or a LIN Slave in a LIN network. In addition, it may be used as a foundation for gateways or re-programming.

### 6.3 Modules and Their Add-Ons

The BSW modules from MICROSRAR.LIN contain functions defined in AUTOSAR 4.x.

- LinIf – LIN Interface
The module LinIf offers abstracted (PDU-based) access to the LIN hardware. It also handles schedule table processing. It is available as master or slave.

Extensions to AUTOSAR

> Configurable wakeup delay
> Separately configurable memory mapping of configuration data for LinIf and LinTp. This is especially appealing for controllers with segmented memory.
> Notification of Schedule Table End
> Configurable schedule tables to reduce maximum task run-time in multi-channel systems
> Wakeup through LIN transceiver. After an external wakeup, this function allows to omit a second (unwanted) wakeup pulse by the master.

> LinNm – LIN Network Management

The LinNm module contains a hardware-independent protocol, which coordinates the transition between normal operation and the bus sleep mode of the LIN network. With AR4.4, LinNm is only available as master. LIN slave does not use LinNm anymore.

> LinSM – LIN State Manager

The module LinSM switches between schedule tables and PDU groups in the Com module and services the LIN interface with regard to sleep and wake up. This module is available as master and slave.

Extensions to AUTOSAR

> Extended polling of LinSM sub-modes for controlled switchover of LIN Schedule Tables.
> Optimized startup behavior by automatic selection of a schedule table (configurable)

> LinTp – Vector LIN Transport Protocol

The module LinTp is responsible for segmenting data in the Tx direction, collecting data in the Rx direction and monitoring the data stream. According to the AUTOSAR specification, LinTp is part of LinIf. This module is available as master and slave.

6.3.1 Vector Modules as an Extension of AUTOSAR Standard

> vLinXcp – Vector LIN XCP Module

The vLinXcp module contains LIN-specific contents of the XCP module (Xcp).

6.3.2 Cross-package Add-Ons

There are also add-ons that affect one or more packages. These are in detail:

> MICROSAR.IDM – Identity Manager, for details please refer to the chapter “MICROSAR Variant Handling”
> MICROSAR.PBL – Post Build Loadable, for details please refer to the chapter “MICROSAR Variant Handling”

For both add-ons, Lin Slave is not supported.

6.4 Configuration

For convenient configuration, we recommend the DaVinci Configurator Pro. For more details, please refer to the separate product information.

6.5 Other MICROSAR Products for a Complete LIN Communication Stack

Based on the AUTOSAR architecture, a complete communication stack for LIN is formed by MICROSAR.LIN together with the BSW modules from the separately available MICROSAR.COM, MICROSAR.MCAL and MICROSAR.EXT packages. To interface MICROSAR.LIN to the application and hardware, you will still need the following BSW modules:

> Hardware-specific LIN driver (Lin) from MICROSAR.MCAL
> Hardware-specific transceiver control (LinTrcv) from MICROSAR.EXT
6.6 Other Relevant MICROSAR Products for LIN

- DET, ECUM and COMM from MICROSAR.SYS
- MICROSAR XCP enables measurement and calibration according to ASAM XCP. This module was specially optimized to be used together with CANoe.XCP and CANoe.AMD as well as CANape. For LIN ECUs, MICROSAR XCP contains the related transport layer vLinXcp. Since XCP-on-LIN has not been officially defined, this XCP-on-LIN implementation is a Vector extension of the ASAM standard.
- Beyond the AUTOSAR standard, MICROSAR XCP supports generic readout of measuring objects. As a result, no addresses must be defined and updated in the a2l file. Data from any version or variant can be extracted with an a2l file independent from the MCU build. The generic readout feature requires usage of CANoe.AMD or CANape as XCP tool.
- VX1000If: For safety reasons, very often it is not allowed to keep the measurement and calibration interfaces active in serial production projects. The module VX1000If allows to keep the VX1000 measurement and calibration hardware driver in the BSW also in serial production, but in a deactivated mode. Through an API the VX1000 driver can be released again for check and development purposes. Delivery of the module must take place within a MICROSAR SIP to receive approval for this use case in serial production. The activation of the VX1000 driver within a serial production project at runtime however is also not allowed when using the module VX1000If.

6.7 The Vector toolchain for developing LIN ECUs

![Vector toolchain diagram]

Figure 12: Vector offers you a comprehensive range of products and services for your LIN projects
7 MICROSMART.ETH – AUTOSAR Basic Software Modules for Ethernet-Based Communication

The Internet Protocol and the higher-level Transport Protocols UDP and TCP are very widely used standards for high speed data exchange over Ethernet.

The MICROSMART.ETH (Ethernet) package contains the AUTOSAR BSW modules including a TCP/IP stack developed to an automotive standard for Ethernet-based communication between ECUs. AUTOSAR 4.0 is the first version to specify Ethernet as a network technology. In AUTOSAR 4.1, the specifications were substantially modified and extended. Further extensions, e.g. Ethernet switch configuration and time synchronization between ECUs are specified in AUTOSAR 4.2. The BSW modules of MICROSMART.ETH are available according to AUTOSAR 4.x and as a supplement to AUTOSAR 3.x.

7.1 Overview of Advantages of MICROSMART.ETH

- TCP/IP modules are available for AUTOSAR 4.x as well as Vector-specific extensions
- TCP/IP stack developed to an automotive standard. IETF conformity is regularly proven with 3rd party OPEN ALLIANCE TC8 test.
- No Open Source Software
- Seamless integration, e.g. of Vehicle-to-Grid communication (MICROSMART.V2G) and Audio/Video Bridging (MICROSMART.AVB), in the AUTOSAR Ethernet- and TCP/IP stack
- Simple integration of customer-specific functions/modules on all levels

7.2 Application Areas

With MICROSMART.ETH in the ECU (Server) and a conventional PC or diagnostic tester as Client, you can

- diagnose the vehicle in accordance with ISO 13400-2 (DoIP) and
- reprogram ECUs quickly and in parallel.
The larger data throughput of Ethernet shortens total software download and diagnostic times considerably. An existing gateway in the vehicle can be used to route diagnostic requests to internal vehicle networks. For instance, this gives you the ability to reprogram multiple CAN ECUs in parallel over DoIP. In combination with other MICROSAR packages, MICROSAR.ETH implements the required gateway functionality. If MICROSAR.ETH is being used in the flash bootloader (FBL), an ECU that is connected to the Ethernet network (e.g. the gateway itself) can be reprogrammed directly over DoIP.

MICROSAR XCP on Ethernet is available to you for measuring and calibrating Ethernet ECUs, which gives you the benefit of larger bandwidth. XCP routing extends a gateway by adding the ability to calibrate also CAN and FlexRay ECUs over XCP via the Ethernet (vehicle) port.

Along with the application areas of diagnostics, measurement and calibration, in which Ethernet-based communication occurs between the external infrastructure and the vehicle, MICROSAR.ETH also offers the ability to efficiently use vehicle-internal Ethernet networks. Using "scalable service oriented middleware over IP" (SOME/IP) you can transmit data service-oriented, for instance. This might involve use of the Service Discovery BSW module (SD) for the administration of services which was introduced in AUTOSAR 4.1.1. Apart from the service orientation, SOME/IP also offers a dynamic data serialization. Its' implementation is available as an RTE transformer. You will find more information on the SOME/IP transformer in the chapters on MICROSAR.RTE and MICROSAR.COM.

Of course, you can also transmit data on Ethernet in a signal- and PDU-based way.

Parts of MICROSAR.ETH also serve as the basis for Vehicle-to-Grid communication and audio/video bridging. You will find more details on these application areas in the sections on MICROSAR V2G and MICROSAR AVB.

7.3 Modules and Their Add-Ons

The following BSW modules from MICROSAR.ETH contain the functions defined in AUTOSAR 4.1. For use in an AUTOSAR 4.0 or AUTOSAR 3.x software stack, they have suitable compatible interfaces.

> **ETHIF – Ethernet Interface**

The Ethernet Interface enables hardware-independent access to the Ethernet driver (ETHDRV) and Ethernet transceiver driver (ETHTRCV). Starting with AUTOSAR 4.1, this module is also responsible for VLAN handling. Hardware-independent control of Ethernet switch drivers (ETHSWTDRV and ETHSWTDRV EXT) has been part of ETHIF since AUTOSAR 4.2.

> **ETHSM – Ethernet State Manager**

To start up or shut down communication in Ethernet clusters, the Ethernet State Manager (ETHSM) provides an abstract interface to the Communication Manager (COMM). The ETHSM accesses the Ethernet hardware over the ETHIF.

> **ETHTSYN - Time Sync Over Ethernet**

Time Synchronization over Ethernet (ETHTSYN) implements the time synchronization protocol for Ethernet, based on IEEE 802.1AS. This permits clock synchronization between Ethernet ECUs as a part of the AUTOSAR Global Time Concept. The Synchronized Time Base Manager (STBM) BSW module from MICROSAR.SYS is available as a higher-level time coordinator.

> **SOAD – Socket Adapter**

The Socket Adapter (SOAD) converts the communication over PDUs defined in AUTOSAR into socket-oriented communication. In AUTOSAR 4.0, the SOAD also contains the diagnostic functionality (DoIP) defined in ISO 13400-2. In AUTOSAR 4.1, this plug-in is made separate and is specified as an independent module (DOIP). Moreover, extensions for XCP routing are implemented in the SOAD.
The add-on “SOAD (BSD)” enables the SOAD and the overlying modules to be used in a non-AUTOSAR environment, e.g. LINUX.

> **Dolp – Diagnostic over IP**

Effective with AUTOSAR 4.1.1, the Diagnostics over IP (DOIP) module contains the diagnostic functionality by the same name according to ISO 13400-2. Up to and including AUTOSAR 4.0.3, this functionality is part of the Socket Adapter (SOAD).

> **TCPIP – Transmission Control Protocol/Internet Protocol**

This module contains all protocols for UDP- and TCP-based communication. It supports IPv4 and IPv6 and parallel operation of IPv4 and IPv6 in one ECU. It contains the following protocols:

- UDP and TCP

For some use cases, e.g. for communication with external infrastructure outside the car, additional functions for a TCP/IP stack may be necessary. The TCPIP module from MICROSAR therefore includes functions beyond the AUTOSAR standard to support this scenario.

In conjunction with Ethernet switch support in AUTOSAR 4.2, the TCPIP module was supplemented with a DHCPv4 server which assigns IP addresses based on the switch port. This DHCPv4 server is available as a TCP/IP Add-On.

Add-Ons

At least one of the Add-Ons is required for a working TCPIP:

- **TCPIP v4**: IPv4, ICMPv4, ARP and DHCPv4 (Client)
- **TCPIP v6**: IPv6, ICMPv6, NDP and DHCPv6 (Client)

**vIpSec – Vector Ethernet Security**: For details to the Vector Ethernet Security (vIpSec), please refer to the chapter “MICROSAR Security”.

**SD – Service Discovery**

Service Discovery (SD) is specified for the first time in AUTOSAR 4.1.1. An ECU informs its communication partners about the availability of its services with the protocol implemented in this module. In addition, ECUs can register to receive automatic notifications, e.g. in the event of a signal update.

**UdpNm – UDP Network Management**

You can implement synchronous going to sleep of Ethernet ECUs by network management over UDP (UDPNM).

### 7.3.1 Vector Modules as Extension to AUTOSAR Standard

- **vEtM – Vector Ethernet Testability Module**

  The Ethernet Testability Module (vEtM) implements a standardized counterpart for protocol conformity tests. The module enables an externally connected test environment to trigger defined actions, e.g. sending UDP packages or creating a TCP-connection. The vEtM module is specified in AUTOSAR 4.3 and currently available as an extension.

- **vEthFw – Vector Ethernet Firewall**

  For details to the Vector Ethernet Firewall (vEthFw), please refer to the chapter “MICROSAR Security”.

- **vTls (Client) – Vector Transport Layer Security**

  For details to the Vector Transport Layer Security vTls (Client), please refer to the chapter “MICROSAR Security”.

- **vEap – Vector ECU Authentication Protocol**

  The vEap module implements the authentication according to IEEE 802.1X.

### 7.3.2 Cross-package Add-Ons

There are also add-ons that affect one or more packages. These are in detail:

- **MICROSAR.PBL – Post Build Loadable**, for details please refer to the chapter “MICROSAR Variant Handling”
7.4 Functions

7.5 Configuration

We recommend our DaVinci Configurator Pro for a convenient configuration. You will find more details in the separate product information.

Ethernet- and TCP/IP-specific configuration parameters are saved as an extension in the "ECU Configuration Description (ECUC)" for AUTOSAR 3.x. This also applies to non-specified configuration parameters within AUTOSAR 4.x.

7.6 Other Relevant MICROSAR Products

Based on the AUTOSAR architecture, MICROSAR.ETH, together with the BSW modules from the MICROSAR.MCAL and MICROSAR.EXT that are available separately, forms a complete communications stack for Ethernet and TCP/IP.

> You will still need the following BSW modules to interface MICROSAR.ETH to the hardware:

> Hardware-specific Ethernet driver (ETHDRV from MICROSAR.MCAL)

> Hardware-specific transceiver driver (ETHTRCV from MICROSAR.EXT)

> Optional: Hardware-specific Ethernet-Switch driver (ETHSWTDRV from MICROSAR.MCAL and ETHSWTDRV EXT from MICROSAR.EXT)

> The modules in MICROSAR.MCAL and MICROSAR.EXT are available for many microcontrollers and transceivers.

> If PDUs should be passed to other software modules of the AUTOSAR stack, you will in most use cases also require the PDU Router (PDUR) module from the MICROSAR.COM package.

> The modules of the MICROSAR.SYS package can be used to control the Ethernet stack and TCP/IP stack:

> COMM: Central coordination point for starting up and shutting down the communication stack

> NM: Central coordination point for network management

> BSWM: Mode management module, used for service discovery

> STBM: Higher-level time coordinator used for time synchronization between different networks and bus systems.

> DET: Detects and evaluates errors during the development period

> DEM: You can use the DEM module from the MICROSAR.DIAG package to manage the detected system events (errors and environmental data).

> MICROSAR XCP enables measuring and calibrating according to ASAM XCP. The module was especially optimized for use together with CANoe.XCP and CANoe.AMD as well as CANape. For Ethernet ECUs, MICROSAR XCP contains the related transport layer ETHXCP.

> VX1000If: For safety reasons, very often it is not allowed to keep the measurement and calibration interfaces active in serial production projects. The module VX1000If allows to keep the VX1000 measurement and calibration hardware driver in the BSW also in serial production, but in a deactivated mode. Through an API the VX1000 driver can be released again for check and development purposes. Delivery of the module must take place within a MICROSAR SIP to receive approval for this use case in serial production. The activation of the VX1000 driver within a serial production project at runtime however is also not allowed when using the module VX1000If.

> The Vehicle-to-Grid application cases are covered by the modules of the MICROSAR V2G package. Also building upon MICROSAR.ETH are the modules from MICROSAR AVB that are available for audio/video bridging.

7.7 Other Relevant Products for Ethernet

With the CANoe option ".Ethernet" you can conveniently extend your existing CANoe installation to include the ability to analyze and simulate Ethernet-based communication.

The VN5610 network interface is recommended as the hardware interface, especially when 100BASE-T1 (formerly BroadR-Reach®) is used as the physical layer. Along with two Ethernet channels (individually configurable for 100BASE-T1 or 100BASE-TX/1000BASE-T), it also offers two High-Speed CAN channels. The VN5640 is best suitable for test and simulation purposes with even more ethernet connections. It offers 16 ethernet channels (twelve 100BASE-T1 and four 100BASE-TX/1000BASE-T) and is highly configurable.
7.8 The Vector Toolchain for Developing Ethernet ECUs

Figure 14: Vector offers you a comprehensive range of products and services for your Ethernet projects
8  MICROSA.R.CHARGE – Basic Software Modules for Communicating with External Infrastructure

The MICROSA.R.CHARGE package contains BSW modules for charging of electric and hybrid vehicles and for the communication with the infrastructure over Internet technologies such as HTTP. All of the modules from this package are not specified in AUTOSAR. However, they are integrated in the Vector AUTOSAR solution. The extensions are offered for both AUTOSAR 4.x and AUTOSAR 3.x. Required as a basis for MICROSA.R.CHARGE are modules from the MICROSA.R.ETH package.

![Figure 15: The MICROSA.R.CHARGE BSW modules according to AUTOSAR 4.3](image)

8.1 Overview of Advantages of MICROSA.R.CHARGE

- Implements all protocols needed for Smart Charging Communication (vScc)
- Supports communication over Internet mechanisms and protocols
- Seamless integration of all BSW modules into an AUTOSAR environment
- Easy to incorporate customer-specific functions by generic interfaces

8.2 Application Areas

MICROSA.R.CHARGE lets you perform intelligent charging of electric and hybrid vehicles at a suitable charging station. Supported are the standards

- ISO 15118 and
- DIN SPEC 70121
- CHAdeMO
- GB/T 27930

with their options of charging by alternating and direct current (AC and DC), wireless power transfer (WPT), automatic charging device (ACD) and bidirectional power transfer (BPT).
You can use the modules of the MICROSAR.CHARGE package to also have your ECU communicate with a server via commonly used Internet protocols.

If necessary, the communication may also be encrypted using TLS.

### 8.3 Vector Modules as Extension to AUTOSAR Standard

MICROSAR.CHARGE contains the following BSW modules:

- **vDns – Vector Domain Name System Resolver**
  - The vDns module contains a DNS Resolver. It is responsible for resolving a domain, e.g. vector.com, into a valid IP address.

- **vHttp – Vector Hypertext Transfer Protocol**
  - One application of the Hypertext Transfer Protocol is to transmit browser requests to a server. The module contains an HTTP Client.

- **vXmlSecurity – Vector XML Security**
  - This module is used to generate or validate XML signatures that will be or are attached to EXI-encoded data based on the W3C XML Security Standard. This functionality is required when using “Plug and Charge” according to ISO 15118.

- **vExi – Vector Efficient XML Interchange**
  - The vExi module is used to interpret XML documents and convert them to binary format. This makes processing and transmission of the files more efficient, which economizes communication bandwidth.

- **vScc – Vector Smart Charging Communication**
  - This module is responsible for Smart Charging Communication according to ISO 15118 and DIN SPEC 70121.

#### Add-Ons

- **AC**: AC charging
- **DC**: DC charging
- **ACD**: Automatic charging device (pantograph charging)
- **WPT**: Wireless power transfer (inductive charging)
- **BPT**: Bidirectional power transfer (energy recovery)
- **<OEM>**: Provides the application interface as specified by <OEM>

- **vCanCcGbt – Vector CAN Charging Communication according to GB/T 27930**
  - This module contains the standardized charging communication for direct current charging according to the specification GB/T 27930 for China. Support for diagnostics for this standard can be developed on request.

- **vCanCcCdm – Vector CAN Charging Communication according to CHAdEVO**
  - This module contains the standardized charging communication for direct current charging according to the specification CHAdEVO.

In addition, the following modules are available within an AUTOSAR 3.x environment. For AUTOSAR 4, they are available on request:

- **vXMLEngine**
  - The vXMLEngine module contains a parser for processing and a generator for creating valid XML 1.0 documents. It is used in the CHARGE field.

- **vJson**
  - This module contains a JSON parser. JSON is a JavaScript-based data exchange format and can be used instead of XML.
8.4 Configuration

The modules of the MICROSAR.CHARGE package are configured with DaVinci Configurator. You will find more details in the separate product information.

The specific configuration parameters are saved as an extension in the ECU Configuration Description.

8.5 Other Relevant MICROSAR Products

MICROSAR.CHARGE builds upon the MICROSAR.ETH package and requires the Ethernet stack and TCP/IP stack as a basis for communication. It comprises the following modules:

- Ethernet Interface (EthIf) for abstracting the underlying hardware
- Ethernet State Manager (EthSM) for switching Ethernet-based communication on and off
- The TCP/IP stack (Tclpl) with the related IP version (IPv4 and/or IPv6)

Also needed are an Ethernet driver (Eth (EXT)) and an Ethernet transceiver driver (EthTrcv) from MICROSAR.EXT. Special drivers and transceiver drivers for Powerline Communication (PLC) are available for Smart Charging Communication.

If the control of Smart Charging Communication is implemented via an AUTOSAR software component, we recommend the use of MICROSAR.RTE and MICROSAR.CRYPTO.

The module DET from MICROSAR.SYS is available for detecting and evaluating errors during the development period.

The module DEM from the MICROSAR.DIAG package can be used to manage detected system events (errors and environment data).

8.6 Other Relevant Products

You can conveniently extend your existing CANoe installation to include the ability to analyze and simulate Ethernet-based communication with the related CANoe Option for Smart Charging (new with CANoe 11 SP3). You can also analyze SCC traffic in CANoe. This lets you set up complex vehicle and charging station simulations based on the ISO 15118 and DIN SPEC 70121.

Vector offers a plug-in card for Powerline Communication for the VT system.

8.7 The Vector Toolchain for Developing Ethernet/CHARGE ECUs

![Diagram showing the Vector Toolchain for Developing Ethernet/CHARGE ECUs]

*Figure 16: Vector offers you a comprehensive range of products and services for your Ethernet/CHARGE projects*
9  MICROSAR AVB – Basic Software Modules for Audio/Video Communication via Ethernet

MICROSAR AVB (Audio/Video Bridging) over Ethernet enables quick and reliable transport of audio/video data. The MICROSAR AVB package contains various BSW modules which are overlaid on the Ethernet interface, e.g. from MICROSAR.ETH. The solution based on AUTOSAR 4.x supports vAvTp (Audio/Video Transport Protocol), vRtp (Transport Protocol for Real-Time Applications), vSrP (Stream Reservation Protocol), EthTSyn (Time Synchronisation over Ethernet) and on request also BMCA (Best Master Clock Algorithm). This makes it possible to implement AVB end points as well as bridge functionality.

![Diagram of MICROSAR AVB BSW modules according to AUTOSAR 4.3](image)

**Figure 17:** Figure 1: The MICROSAR AVB BSW modules according to AUTOSAR 4.3

### 9.1 Overview of Advantages of MICROSAR AVB

- The BSW modules are optimized for AUTOSAR, but they may be integrated in other environments as well.
- Trouble-free integration into the Ethernet stack MICROSAR.ETH. This enables parallel use of AVB, DoIP and Tcplp, for instance.
- Easy to incorporate customer-specific functions and modules
- Supports various Ethernet controllers, including AVB-specific hardware functionalities
- Supports VLANs for isolating and prioritizing data, e.g. for audio, video and diagnostics.

### 9.2 Application Areas

#### 9.2.1 A/V Streaming

MICROSAR vAvTp and vRtp enable the exchange of audio/video data, including their time stamps, between different end points. vAVTP is a 2-layer-protocol and is implemented according to the specification IEEE 1722/1722a. vRtp is a 3-layer-protocol based on UDP and is implemented according to the specification IETF RFC 3550.
2.2.2 Choosing the Most Precise Clock

Before a precise clock can be displayed system-wide, the device with the most precise clock (Grand Master) must be defined. Typically, this is set statically or dynamically through BMCA. This involves use of the specification IEEE 802.1AS.

2.2.3 Displaying a Synchronous System Time

The ECU with the most precise time distributes it in the network. This means that all end points and bridges are working with the same time prescribed by the Grand-Master. This makes it possible to transmit an A/V data stream and play it back time-synchronously. The protocol for distributing the time stamp is implemented in the AUTOSAR BSW module EthTSyn according to the specification IEEE 802.1AS.

To enable an exact time measurement, it is necessary to use extended hardware support to achieve a higher precision, which is implemented in the “Add-On Time Sync” of the Ethernet switch driver.

3.3 Vector Modules as Extension to AUTOSAR Standard

The following BSW modules from MICROSAR AVB contain the functions defined in the above-mentioned IEEE specifications. Also necessary are software properties which permit seamless integration in an AUTOSAR environment.

- vAvTp - Vector Audio/Video Transport Protocol
  The module vAvTp is specified in IEEE 1722. In AVB networks it is responsible for the transport of audio/video data, including the Presentation Time.

- Interface to the EthIf module for receiving and sending AVTP frames

- Distinction made between Stream and Control channels

- Display and validation of the time stamp

- Detection of the transported data stream

- vSrEp End Station - Vector Stream Registration Protocol End Station
  Data stream registration with admission control
  Ensures, that all AVB-Bridges reserve the appropriate bandwidth for each stream ID

- vSrRp Bridge - Vector Stream Registration Protocol Bridge
  Data stream registration with admission control for multiple streams from different End Stations
  Ensures, that the overall reserved bandwidth for an Ethernet AVB network does not exceed a predefined limit

- vRtp - Vector Real Time Transport Protocol
  The module vRtp is a real time capable streaming protocol with uni-/multicast support. The RealTime Control Protocol (RTCP) for negotiation and observation of Quality-of-Service-Parameter (QoS) is also included. The following profiles are supported:
  IETF RFC 6184 RTP Payload Format for H.264 Video
  IEEE 1733 Layer 3 Transport Protocol for Time-Sensitive Applications in Local Area Networks.

  Provides an end-to-end network transport function suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. It supports the following profiles:

3.4 Configuration

We recommend that you use our DaVinci Configurator Pro for a convenient configuration. You will find more details in the separate product information.

AVB-specific configuration parameters are saved as an extension in the “ECU Configuration Description”.

3.5 Interfaces to Related MICROSAR Products

Based on the MICROSAR architecture, MICROSAR AVB – together with the basic software modules from the separately available packages MICROSAR.MCAL, MICROSAR.EXT and MICROSAR.ETH – forms a complete communication stack for AVB in the automobile. To interface MICROSAR AVB to the hardware, you will still need the following BSW modules:
> Hardware-specific Ethernet driver (Eth from MICROSAR.MCAL)
> Hardware-specific transceiver driver (EthTrv from MICROSAR.EXT)
> Ethernet driver abstraction and control layer (EthIf, EthM from MICROSAR.ETH)

The modules in MICROSAR.MCAL and MICROSAR.EXT are already available for many microcontrollers and transceivers, and they will be also developed for new microcontrollers and transceivers as necessary.

### 9.6 Additional Relevant MICROSAR Modules for Ethernet

> Tcplp, SoAd, DoIP, Somelp from MICROSAR.ETHDCM and Dem from MICROSAR.DIAG
> Det, EcuM, ComM and Nm, from MICROSAR.SYS
> MICROXAR XCP

### 9.7 Other Relevant Products for Ethernet

The CANoe option "Ethernet" lets you conveniently extend your existing CANoe installation to include the ability to analyze and simulate Ethernet- and AVB-based communication.

The VN5610 network interface is recommended, especially when BroadR-Reach® is used as the physical layer. Along with two Ethernet channels (individually configurable for BroadR-Reach® or 100BASE-TX), it also offers two High-Speed CAN channels.

### 9.8 The Vector Toolchain for Developing Ethernet/AVB ECUs

![Figure 18: Vector offers you a comprehensive range of products and services for your Ethernet/AVB projects](image)
10 MICROSOAR.MEM – AUTOSAR Basic Software Modules for Memory Management

The MICROSOAR.MEM package contains all of the AUTOSAR modules for memory management: NvM, MemIf, Ea and Fee. They support management, checking and restoring of data from nonvolatile memories (Flash or EEPROM). The Basic Software Modules (BSW) from MICROSOAR.MEM are fast, reliable and robust.

![Microcontroller Diagram](image)

**Figure 19:** MICROSOAR.MEM modules according to AUTOSAR 4.3

10.1 Overview of Advantages

- Available for AUTOSAR 4.x and 3.x
- Exceptionally secure data transactions
- Efficient data accesses
- Efficient and robust management of nonvolatile memories
- Redundant storage of management data increases reliability of data access
- Inter-module configuration of the entire memory stack
- Platform-optimized memory stack solution from a single source

10.2 Application Areas

MICROSOAR.MEM contains AUTOSAR services for reading, writing and erasing persistent application data in flash and/or EEPROM memories. This gives the functional software hardware-independent access to memory. The application does not need to know the specific type of memory that exists on the platform or whether this memory is internal to the controller or externally connected to it.

10.3 Modules and Their Add-Ons

The BSW modules in MICROSOAR.MEM contain functions defined in AUTOSAR 4.x. In each memory stack, you will need the BSW modules NvM and MemIf from MICROSOAR.MEM. They handle block-oriented and technology-independent access to the
memory areas – without requiring prior knowledge of memory attributes. Your memory stack will need additional BSW modules, depending on the use case:

> When a flash memory is used: Flash EEPROM Emulation (Fee) e.g. from MICROSAR.MEM and a flash driver (FLs) that is suitable for your hardware included in the service of our MCAL Integration Package, or for external memory the module Ext from MICROSAR.EXT. To manage the data, the FEE module needs at least two physical flash sectors.

> When using an EEPROM: EEPROM Abstraction (Ea) from MICROSAR.MEM and an EEPROM driver (Eep), for example the module DRVEXT from MICROSAR.EXT for external memory.

It is possible to mix multiple flash and EEPROM chips in one ECU.

For special requirements, Vector offers platform-optimized solutions, e.g. for using the BSW module EA in data flashing or optimizing the FEE module for specific hardware.

> **Ea – EEPROM Abstraction**

  The Ea module offers a hardware-independent interface for accessing EEPROM data and uses an EEPROM driver (Eep) for this. In addition to reading, writing and clearing data, the Ea module also distributes write accesses to different areas of the EEPROM, so that all EEPROM cells are uniformly stressed, which increases their life-time.

  **Extension to AUTOSAR Standard**

  > Additional configurable transaction security, which is a standard feature of the Fee module

  > Redundant storage of management data for increased reliability of data access

> **Fee – Flash EEPROM Emulation**

  The Fee module offers a hardware-independent interface for accessing flash data and uses a flash driver (FLs) for this. In addition to reading, writing and clearing data, the Fee module also distributes write accesses to different areas of flash memory, so that all flash cells are uniformly stressed, which increases their life-time.

  The variant “Small sector FEE” is recommended in case a Flash Memory with a numerous amount but small sized sectors is used. It offers the two main benefits, that less management data needs to be stored, offering more space for user data and a faster evaluation of the valid data set version, which makes faster power-up as well as data storage.

  **Extension to AUTOSAR Standard**

  > High-performance administration of stored memory data

  > Common usage of Fee module by Flash Bootloader (Fbl) and application possible - also with common memory blocks.

  > An update of the ECU software can be done without adjustment of the FBL.

  > Redundant storage of management data for increased reliability of data access

  > Flexible placement of the Fee sectors in the DataFlash

  > Services for handling of undervoltage situations

  > Frequently used data is isolated from extremely important data by introducing partitions. This further increases data availability in fault situations (e.g. reset while writing or erasing data).

  > Update support for adjustment of non-volatile memory after ECU re-programming. This is done with a new configuration table (content and size).

> **MemIf – Memory Abstraction Interface**

  The module MemIf provides uniform access to the services of Ea and Fee. This lets you use multiple instances of these modules.

> **Nvm – Non-Volatile-RAM Manager**

  The NvM module manages, reads and writes data to a nonvolatile memory (Ea or Fee). At system start and at shutdown, it synchronizes the data in the RAM areas of the application. The module provides services such as saving of redundant blocks for a higher level of data protection.

  Since AR 4.0.3, the RTE also provides a simpler and more flexible interface to Nv data (NvDataInterfaces).

  **Extension to AUTOSAR Standard**
> Allocates RAM for CRC memory storage
> Dedicated interface for the Dcm diagnostic module for direct read-out and modification of data blocks
> Additional configurable transaction security, which is a standard feature of the Fee module

### 10.4 Configuration

We recommend DaVinci Configurator Pro for a convenient configuration. This contains certain functions that make work simpler, such as optimization assistance, visual representation of flash utilization, etc. See the separate product information for more details.

![Configuration of MICROSAR.MEM modules with DaVinci Configurator Pro](image)

Figure 20: Configuration of the MICROSAR.MEM modules with DaVinci Configurator Pro

### 10.5 Other MICROSAR Products relevant for a Memory Stack

Based on the AUTOSAR architecture, a complete memory stack can be formed with memory services from MICROSAR.MEM together with other platform-specific BSW modules from the separately available MICROSAR.MCAL and MICROSAR.EXT packages.

> Fls and/or Eeep from the MCAL Integration Package
> Ext from MICROSAR.EXT for external memory chips

In addition, it is easy to integrate MCAL drivers from semiconductor manufacturers into the MICROSAR memory stack.

Depending on the desired safety level, you can verify your memory data with the checksum module (Crc) from the MICROSAR.LIBS bundle.
11 MICROSA.R.SYS – System-related Basic Software Modules for AUTOSAR

The system services in the MICROSA.R.SYS basic software modules (BSW) cover an important part of your AUTOSAR ECU’s basic functionality. They are called by the functional software (via the Rte) and the remaining BSW modules. The modules of MICROSA.R.SYS offer all key functions for state handling of the ECU.

11.1 Overview of Advantages

- Available for AUTOSAR 4.x and 3.x
- The ECUM module is included as either a resource-saving Pre-Compile variant or as a flexible Post-Build solution. You will find more information on this in the “Identity Manager for AUTOSAR” section.
- Easy configuration of the initial BSWM and ECUM by assistants in DaVinci Configurator Pro
  - Create initialization sequences
  - Configure an ECU state machine (startup, shutdown, etc.)
  - Managing of the communication modes

11.2 Application Areas

System services include power and mode management, control of all communication channels and partial networks, monitoring of individual software components (SWC) of the functional software and within AUTOSAR 3.x scheduling of all BSW modules.

11.3 Modules and Their Add-Ons

The BSW modules in MICROSA.R.SYS contain functions defined in AUTOSAR 4.x:

- BswM - BSW Module Manager
The Basic Software Mode Manager manages mode change requests from the BSW modules and SWCs and executes them according to the standardized action lists. For example, the BSWM module is responsible for activating and deactivating PDU groups and NM-PDUs for diagnostics.

Extension to AUTOSAR Standard

> Support of partial networking

> **ComM – Communication Manager**

The Communication Manager monitors the state changes of the communication channels connected to the ECU and of the sub-networks configured in the ECU. It can keep the ECU awake and ready for communication as necessary. Furthermore, it coordinates access of all SWCs to the communication channels and sub-networks. Optionally, COMM supports the bus type "Internal".

Extension to AUTOSAR Standard

> Support of partial networking

> Compatible with OSEK Nm (for AUTOSAR 3.x)

> **Det - Default Error Tracer**

The Default Error Tracer collects the development errors of the SWCs and BSW modules. Optionally, Det supports the Service Ports.

> **ECUM – ECU State Manager**

The ECU State Manager is responsible for Startup, Shutdown and WakeUp. In AUTOSAR 3.x, there are other fixed defined operating states that are managed by the EcuM. In AUTOSAR 4.x, these operating states are flexibly defined by the user in the BSWM. This makes it possible to implement individual energy-saving states or different power-up behaviors.

Extension to AUTOSAR Standard

> The ECUM module is implemented according to the AUTOSAR ECUM Flex specification, so it offers a high level of configuration options that can also support complex state transitions. When ECUs are being developed that have reduced requirements for state management, the Ecum module can optionally also behave compatibly with the AUTOSAR ECUM Fixed Specification. In this case, the Ecum module offers the following functionalities:

> EcuM Run Request Protocol

> EcuM State Management

> EcuM Fixed Compatible Service SWC interfaces

> **StbM – Synchronized Time Base Management**

The Synchronized Time-Base Manager (StbM) enables precise time synchronization between different parts of the ECU software. It abstracts the bus-specific time synchronization modules for the CAN, FlexRay and Ethernet by the provision of communication services for vehicle-wide time synchronization

> **Tm – Time Services**

The Tm module is used for such tasks as measuring execution implementing active waiting. It offers a resolution from 1µs to 4.9 days.

> **WdgIf – Watchdog Interface**

The module WdgIf enables uniform access to services of the Watchdog Driver (Wdg), such as mode switching and triggering. For safety-related ECUs, the WdgIf module must be developed according to ISO 26262.

Add-Ons

> Precise supervision of defined time windows for the watchdog (even for high resolution window watchdogs)

> **WdgM – Watchdog Manager**

The module WdgM monitors the reliability and functional safety of the applications in an ECU. This includes monitoring for correct execution of SWCs and BSW modules and triggering of the watchdogs at the required time intervals. The WdgM module reacts to potential faulty behavior with multiple escalation stages. An important fact for safety-related
functions according to ISO 26262 is the monitoring of correct flow sequences of critical tasks (logical supervision). For safety-related ECUs, the WdgM must be developed according to ISO 26262.

Add-Ons

> **Prog.Flow + Deadline Monitoring**

Program flow and deadline monitoring to observe the SWCs.

Extension to AUTOSAR Standard

> Precise supervision of defined time windows for the watchdog (even for high resolution window watchdogs)

> The Watchdog Manager monitors the correct operation of the functional software with the modules WdgIf and Wdg (MICROSAR.MCAL).

11.3.1 Cross-Package Add-Ons

There are also add-ons that affect one or more packages. These are in detail:

> MICROSAR.IDM – Identity Manager, for details please refer to the chapter "MICROSAR Variant Handling"

> MICROSAR.MC – Multi-Core, for details please refer to the chapter "MICROSAR Multi-Core"

> MICROSAR.PBL – Post Build Loadable, for details please refer to the chapter "MICROSAR Variant Handling"

11.4 The Basis Software Manager (BswM)

The BswM is a central part of the mode management and is implemented according to the AUTOSAR 4 standard. However, numerous helpful features beyond the standard offer extended comfort while configuring your ECU software.

To react on mode changes in other BSW modules or to request such mode changes, the BSWM module in AUTOSAR 4 allows free configuration of arbitrage rules, logical expressions and actions.

Due to the given AUTOSAR configuration structure, this may become very complex within short time because of the fact that even simple configurations request numerous intertwining steps:

> Actions must be combined to action lists.

> These lists are linked to the true or false result of a predefined logical expression by a rule.

> The logical expression itself consists of one or more conditions which are based on the incoming modes (request ports).

Also, standard tasks such as the configuration of a state machine in the style of ECUM Fix in AUTOSAR 3, the call of the corresponding initialization functions of the BSW modules with the matching parameters or switching on and off PDU groups (I-PDU) rises to a challenge even for experienced developers.

DaVinci Configurator Pro supports you by providing intelligent and powerful assistants which are able to solve many of the above-mentioned tasks automatically and also DaVinci Configurator Pro lets you configure them by a simple click afterwards. This especially applies to

> the ECU state machine (ECU State Handling, see figure below)

> the initialization of the basic software modules (Module Initialization) and

> switching of PDU groups (Communication Control).
Though this is not a static configuration, rather all necessary parameters are considered. If parameters are changed, e.g. if a new PDU group is created, the tool will recognize this immediately and will inform the user about a necessary re-configuration.

The built-in assistants of DaVinci Configurator Pro also support free configuration tasks such as creating rules or action lists. They guide you through the configuration, offer know-how on possible and necessary parameters and they identify mistakes and offers possible corrections. However, they also accept settings which you intentionally want to set differently.

The MICROSAR BSWM provides a lot more than specified by AUTOSAR. As an example, it is also possible to switch on and off the analysis of rules at runtime. You can realize a timer whose expiration can be analyzed by the BSWM. The timer can be started and stopped by actions. If not all SWCs which you have to connect are available yet, the BSWM itself can create all necessary mode declarations and hereby allow you to perform a bottom-up configuration.

11.5 Watchdog for ISO 26262 applications

All watchdog modules are also available as SEooCs (Safety Element out of Context) for safety-relevant functions up to ISO 26262 / ASIL D. They are well suited for validation of run-time monitoring of tasks as well as flow control for the SWCs. For more details, please refer to the chapter about MICROSAR Safe.

11.6 Configuration

For convenient configuration, we recommend the DaVinci Configurator Pro. For more details, please refer to the separate Product Information.
11.7 Other Relevant MICROSAR Products

- **DIAG**: The system service for diagnostics is available separately in the MICROSAR.DIAG package.
- **OS**: MICROSAR.OS, which is available separately, may be used as the operating system.
- **LIBS**: The LIBS package contains the Cyclic Redundancy Check Library (Crc), Crypto Abstraction Library (Cal) and the E2e. It is available as a separate package for AUTOSAR 4.x (LIBS is part of MICROSAR.SYS for AUTOSAR 3.x.)
MICROSAR.DIAG – AUTOSAR-Compatible Implementation of the Diagnostic Standards UDS, OBD and J1939

MICROSAR.DIAG contains the BSW modules for implementing the diagnostic protocols UDS, OBD-II, WWH-OBD and J1939 according to AUTOSAR, making it the diagnostic software for your vehicle project. MICROSAR.DIAG handles a number of tasks:

- OEM-specific implementation of fault memory and its management
- OEM-specific implementation of the diagnostic protocol for communication between the diagnostic tester and the ECU
- Deactivation of certain functionalities due to active error entries
- Fault memory management on multiple cores
- Sending and receiving diagnostic messages as base for on-board tester

Combined with CANdelaStudio, the widely used specification tool for creating diagnostic data, you get a complete diagnostic solution from a single source.

![MICROSAR.DIAG modules according to AUTOSAR 4.3](image)

### 12.1 Overview of advantages

- OEM-independent solution for AUTOSAR 4.x and 3.x
- Customized solutions available for many automotive OEMs
- Vector’s numerous years of experience in the diagnostics field
- Supports OBD-II and WWH-OBD (Euro VI)
- Configurable with AUTOSAR Diagnostic Extract, CANdela-and ODX format
- Variants handling already included for diagnostic configuration
- Offline calibration with common calibration tools based on A2L
> Generation of optimized application code templates

### 12.2 Application Areas

Beyond the AUTOSAR standard, every OEM has its own requirements for diagnostics. That is why Vector offers MICROSAR.DIAG with OEM-specific extensions. It is well-suited for production use and is already available to many OEMs. For ECUs without a special diagnostic specification, an OEM-independent bundle of MICROSAR.DIAG is available.

MICROSAR DIAG can be used for legal requirements of today and tomorrow, such as EURO VI. Support of OBD-II (ISO 15031/ SAE J1979) and WWH-OBD (ISO 27145) is available as an option.

If your ECU requires variants in the diagnostic configuration, MICROSAR.DIAG offers a high-performance solution for this. You can define up to 31 different parameterizations and store them in the ECU, in a resource-optimized way. This avoids redundancies in the ECU software, because identical interfaces to the same data, services and DTCs are combined in the generated diagnostic code.

### 12.3 Modules and Their Add-Ons

The BSW modules in MICROSAR.DIAG contain the functions for the three BSW modules Dcm, Dem and Fim that are defined in AUTOSAR 4.x and 3.x:

> **Dem - Diagnostic Event Manager**

The Dem module contains the implementation of the respective OEM requirements concerning the fault memory of an ECU. The OEM-independent variant is available for AUTOSAR 4.x as well as 3.x. It supports the following functions as standard features:

> Management of all DTC status bits according to the UDS standard
> Definition of individual snapshots and extended data records
> Predefined extended records (e.g. OccurrenceCounter)
> Counter and time-based error de-bounce algorithms
> Suppression of low-priority errors when memory is full
> Flexible unlearning (aging) of errors
> Variants handling for diagnostic configuration
> Link time configuration
> Compressed Configuration Data to optimize code size
> Support of combined errors
> Multicore support with Dem-Satellite
> Suitable for "mixed AUTOSAR" projects
> Post-build loadable and Post-build selectable: For details, please refer to the section "MICROSAR Variant Handling".

Add-Ons

> OBD-II (ISO 15031 / SAE J1979) with support for master and primary OBD ECUs
> WWH-OBD (ISO 27145)

The Dem base functionality differs only a little between AUTOSAR 4.x and 3.x, except for the interface definitions. That is why Vector offers a migration solution that you can use to easily migrate your AUTOSAR 3.x compatible SWCs to an AUTOSAR 4.x project.

> **Dcm - Diagnostic Communication Manager**

The Dcm module implements UDS and OBD-II services in the ECU.

The OEM-independent variant of the Dcm is available for both AUTOSAR 4.x and 3.x. For a complete list of supported services, please refer to the table at the end of this chapter about MICROSAR.DIAG.
The Dcm modules for specific OEMs implement the specifications of the particular OEM. We would be glad to provide you with detailed information on this.

Extensions to AUTOSAR Standard
- Variant handling for diagnostic configurations
- Easy integration of the Vector Flash Bootloader
- Generation of an application code template for the ECU software (AUTOSAR 3.x)
- J1939Dcm: Dcm module specially designed for heavy-duty vehicles
- Add-Ons
  - Support of OBD-II (ISO15031-5)
  - WWH-OBD (ISO27145)

J1939Dcm – SAE J1939 Diagnostic Communication Manager
Designed for heavy-duty vehicles, the module J1939Dcm implements Diagnostic Messages of the SAE J1939-73 protocol, e.g. for reading out the fault memory.

12.3.1 Vector Modules as Extension to AUTOSAR Standard
- vDrm - Diagnostic Request Manager
  The MICROSAR vDrm sends diagnostic requests to the same or other ECUs and receives their answers. It provides an API for the application to send/receive UDS services. The vDrm allows managing parallel connections, identifying other ECUs in the car and provides a firewall to block potentially dangerous diagnostic requests. The module behaves like an externally connected tester and provides the basis for the implementation of an on-board tester.

- vDes - Vector Diagnostic Event Synchronizer
  The module vDes enables diagnostic monitoring over multiple MCUs. vDes implicates multi-controller Dem functionality: The diagnostic master collects qualified events communicated from the Dem on the diagnostic slaves, which receive event messages locally.

12.3.2 Cross-package Add-Ons
There are also add-ons that affect one or more packages. These are in detail:
- MICROSAR.IDM – Identity Manager, for details please refer to the chapter “MICROSAR Variant Handling”
- MICROSAR.MC – Multi-Core, for details please refer to the chapter “MICROSAR Multi-Core”
- MICROSAR.PBL – Post Build Loadable, for details please refer to the chapter “MICROSAR Variant Handling”

12.4 Configuration and Parameterization
You adapt the BSW modules from MICROSAR.DIAG to your application’s needs comfortably by configuration with DaVinci Configurator Pro. This may be done either with the help of a CANdela, AUTOSAR Diagnostic Extract or ODX file or by an “ECU Configuration Description”.

For AUTOSAR 3.x, diagnostic-specific parameterization of the Dcm is performed exclusively via a CANdela file. You can create it quickly and simply or import it from most commonly used ODX dialects with the proven “diagnostic authoring tool” CANdelaStudio.
Figure 25: Parameterization of the MICROSAR.DIAG modules is done with CANdela Studio.

Figure 26: Parameterization of MICROSAR.DIAG is done by CANdela Studio and DaVinci Configurator Pro

12.5 Scope of Delivery

In addition to the standard components, a converter for CANdela Diagnostic Descriptions is provided as well.

12.6 Services for Diagnostic Applications

- Customer-specific extensions of MICROSAR DIAG
- Creating customer-specific diagnostic applications
- Integrating diagnostics in your ECU software

12.7 Other Relevant Vector Products

To fulfill specific ISO standards, you can combine MICROSAR.DIAG with the following MICROSAR products:

- MICROSAR.CAN (ISO 15765-3 oder ISO/DIS 14229-3)
- MICROSAR.FR (ISO/DIS 14229-4)
- MICROSAR.ETH (ISO/DIS 14229-5)
You can use CANdelaStudio to parameterize the CANdela or ODX file for configuration of MICROsAR.DIAG. For more information, please refer to the separate CANdelaStudio product information.

For heavy-duty vehicle diagnostics, you will need the J1939-specific modules from MICROsAR.CAN.

## 12.8 Supported Diagnostic Services

The module Dcm from MICROsAR.DIAG supports by default the following UDS diagnostic services:

<table>
<thead>
<tr>
<th>Diagnostic Service Name (ISO 14229-1)</th>
<th>Service ID (hex)</th>
<th>AUTOSAR 4.x: The SWC has to ...</th>
<th>AUTOSAR 3.x: The SWC has to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiagnosticSessionControl</td>
<td>10</td>
<td>(handled in DCM internally)</td>
<td>... grant service execution</td>
</tr>
<tr>
<td>ECUReset</td>
<td>11</td>
<td>(handled in DCM/BSWM)</td>
<td>(handled in DCM/BSWM)</td>
</tr>
<tr>
<td>SecurityAccess</td>
<td>27</td>
<td>... calculate seed/key for each security level</td>
<td>... calculate seed/key for each security level</td>
</tr>
<tr>
<td>CommunicationControl</td>
<td>28</td>
<td>(handled in DCM/BSWM)</td>
<td>(handled in DCM/BSWM)</td>
</tr>
<tr>
<td>TesterPresent</td>
<td>3E</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ControlDTCSetting</td>
<td>85</td>
<td>(handled in DEM module)</td>
<td>(handled in DEM module)</td>
</tr>
</tbody>
</table>

### Diagnostic and Communication Management Functional Unit

<table>
<thead>
<tr>
<th>Diagnostic Service Name</th>
<th>Service ID (hex)</th>
<th>The SWC has to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadDataByIdentifier</td>
<td>22</td>
<td>... handle data acquisition for each DataElement</td>
</tr>
<tr>
<td>ReadMemoryByAddress</td>
<td>23</td>
<td>via callout</td>
</tr>
<tr>
<td>ReadDataByPeriodic Identifier</td>
<td>2A</td>
<td>-</td>
</tr>
<tr>
<td>DynamicallyDefineData Identifier</td>
<td>2C</td>
<td>-</td>
</tr>
<tr>
<td>WriteDataByIdentifier</td>
<td>2E</td>
<td>... handle data access for each DataElement</td>
</tr>
<tr>
<td>WriteMemoryByAddress</td>
<td>3D</td>
<td>via callout</td>
</tr>
</tbody>
</table>

### Data Transmission Functional Unit

<table>
<thead>
<tr>
<th>Diagnostic Service Name</th>
<th>Service ID (hex)</th>
<th>The SWC has to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadDTCInformation</td>
<td>19</td>
<td>- (handled in DEM module)</td>
</tr>
<tr>
<td>ClearDiagnosticInformation</td>
<td>14</td>
<td>- (handled in DEM module)</td>
</tr>
</tbody>
</table>

### Stored Data Transmission Functional Unit

<table>
<thead>
<tr>
<th>Diagnostic Service Name</th>
<th>Service ID (hex)</th>
<th>The SWC has to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputOutputControlByIdentifier</td>
<td>2F</td>
<td>... control I/O for each DataElement</td>
</tr>
</tbody>
</table>

### Remote Activation Of Routine Functional Unit

<table>
<thead>
<tr>
<th>Diagnostic Service Name</th>
<th>Service ID (hex)</th>
<th>The SWC has to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoutineControl</td>
<td>31</td>
<td>... start (stop/request result) for each RoutineId</td>
</tr>
</tbody>
</table>

**Figure 27:** UDS diagnostic services for the module Dcm

The module Dcm from MICROsAR.DIAG optionally supports the following OBD II Diagnose Services:

<table>
<thead>
<tr>
<th>Diagnostic Service Name (ISO 15031-5)</th>
<th>Service ID (hex)</th>
<th>The SWC has to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Current Powertrain Diagnostic Data</td>
<td>01</td>
<td>... handle data acquisition for each PID other than the “supported ID” and DEM ones</td>
</tr>
<tr>
<td>Request Powertrain Freeze Frame Data</td>
<td>02</td>
<td>- (handled in DEM module)</td>
</tr>
<tr>
<td>Diagnostic Service Name (ISO 15031-5)</td>
<td>Service ID (hex)</td>
<td>The SWC has to …</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Request Emission-Related Diagnostic Trouble Codes</td>
<td>03</td>
<td>- (handled in DEM module)</td>
</tr>
<tr>
<td>Clear/Reset Emission-Related Diagnostic Information</td>
<td>04</td>
<td>- (handled in DEM module)</td>
</tr>
<tr>
<td>Request On-Board Monitoring Test Results for Specific Monitored Systems</td>
<td>06</td>
<td>__ handle data acquisition for each TestId of a MonitorId</td>
</tr>
<tr>
<td>Request Emission-Related Diagnostic Trouble Codes Detected During Current or Last Completed Driving Cycle</td>
<td>07</td>
<td>- (handled in DEM module)</td>
</tr>
<tr>
<td>Request Control of On-Board System, Test or Component</td>
<td>08</td>
<td>__ process each TestId</td>
</tr>
<tr>
<td>Request Vehicle Information</td>
<td>09</td>
<td>__ handle data acquisition for each InfoType ID other than the &quot;supported ID&quot; and DEM ones</td>
</tr>
<tr>
<td>Request Emission-Related Diagnostic Trouble Codes with Permanent Status</td>
<td>0A</td>
<td>- (handled in DEM module)</td>
</tr>
</tbody>
</table>

**Figure 28:** OBD2 UDS diagnostic services for the module Dcm
13 MICROSA.R.MCAL – AUTOSAR Driver for Control of Microcontroller Peripherals

The MICROSA.R.MCAL package contains the drivers for controlling the microcontroller’s peripherals. Its drivers are fully compatible with the AUTOSAR specifications of the Microcontroller Abstraction Layer (MCAL). Each MICROSA.R.MCAL driver is optimized controller-specifically.

Figure 29: MICROSA.R.MCAL modules according to AUTOSAR 4.3

13.1 Overview of advantages

> Available for AUTOSAR 4.x and 3.x
> perfect support of microcontroller peripherals
> Simplified configuration by considering cross-module parameter dependencies in the configuration tool
> Development accelerated by plausibility tests and completeness tests in the configuration tool
> Resource-saving, as functionalities can be deactivated
> Reduced hardware requirements due to optimized utilization of hardware buffers
> Gateway developments are supported by efficient auxiliary functions

13.2 Application Areas

MICROSA.R.MCAL is a turn-key solution for driving your microcontroller peripherals. When switching to different hardware, it is not necessary to make changes to the functional software. You will only need to replace MICROSA.R.MCAL to integrate the appropriate new drivers.

The MICROSA.R.MCAL drivers perfectly fit to the overall MICROSA.R bundle. Depending on the requirements of your application, you may use additional bundles (e.g. MICROSA.R.CAN, MICROSA.R.MEM, etc.) and thereby obtain a complete communications stack or memory management as an example.
13.3 Modules and Their Add-Ons

The MICROSAR.MCAL package contains the driver modules Can, Eth, Fr, Lin, and I2c, and in the version for AUTOSAR 4.x the RamTst test module. The modules conform to AUTOSAR 4.x and are available for many different available microcontrollers.

- **Can – CAN Driver**
  The Can driver abstracts access to the CAN hardware for sending and receiving messages and for switching between controller states (sleep, stop, etc.).

  **Extensions to AUTOSAR Standard**
  - Notification (Callback) on message reception and after successfully sending a message. This allows to automatically execute application-specific code.

  **Add-Ons**
  - Option “HighEnd” offers extended filter options for Multiple Basic CAN objects, RX queue to shorten the interrupt time during reception, individual polling of mailboxes to assure data consistency and reduction of interrupt load.
  - Increase the number of Full-CAN objects by combining multiple CAN controllers on one physical CAN bus (Common CAN)

- **Can (SocketCAN)**
  The SocketCan API is a hardware independent API for triggering CAN communication under Linux. The module uses an existing SocketCAN API to realize CAN communication allowing to run a MICROSAR stack under Linux.

- **Eth – Ethernet Driver**
  The Eth Driver abstracts access to the Ethernet hardware for sending and receiving data and for switching between controller states.

- **EthSwt – Ethernet Switch Driver**
  The module EthSwt provides a uniform and hardware independent interface for controlling and configuring Ethernet switches. It also coordinates the MAC learning when using multiple identical ECUs like cameras for the surround view.

- **Fr – FlexRay Driver**
  The Fr driver abstracts access to the FlexRay hardware for sending and receiving data and for switching between controller states.

  **Extension to AUTOSAR Standard**
  - Supports self-diagnostics. When the FlexRay controller detects an error, it notifies the application so that it can call up the error status
  - Optimized wakeup during operation (WUDOP)
  - Support of the following APIs: CancelTransmit and L-PdU reconfiguration
  - Pre-compile optimizations, e.g. for single-channel systems

- **Lin – LIN Driver**
  The Lin driver provides services for initiating frame transmission (header, response, sleep-mode and wake up), and for receiving responses, checking the current state and validating wake up events.

- **RamTst – Ram Test**
  The module RamTst tests internal microcontroller RAM cells. A complete test is triggered during startup and shutdown of the ECU or by a diagnostic command. During normal operation, a periodic test is performed (block-by-block or cell-by-cell).

- **Crypto(HW) – Crypto Driver**
  This is the hardware-based abstraction on 3rd party crypto drivers Crypto.

13.3.1 Vector Modules as Extension to AUTOSAR Standard

- **vl2c – Vector IPC Driver**
The vI2c driver provides services for communication with external I2C chips

Extension to AUTOSAR Standard

> The vI2c contains drivers for interfacing to external peripheral chips via the Inter-Integrated Circuit Bus (I2C)

13.3.2 Cross-package Add-Ons

There are also add-ons that affect one or more packages. These are in detail:

> MICROSAR.IDM – Identity Manager, for details please refer to the chapter “MICROSAR Variant Handling”

> MICROSAR.PBL – Post Build Loadable, for details please refer to the chapter “MICROSAR Variant Handling”

13.4 Configuration

For convenient configuration, we recommend the DaVinci Configurator Pro. For more details please refer to the separate product information.

Figure 30: Configuration with the DaVinci Configurator: Clock settings based on example of Freescale MPC560xB (Bolero)

13.5 MICROSAR MCAL Integration Package

The MICROSAR MCAL Integration Package covers several work packages which realize an unobstructed integration of a 3rd party MCAL into the embedded software and tool environment of Vector. In conjunction with the customer and the semiconductor vendor, Vector performs a set-up of the MCAL. We check its conformity - respectively its possibility to integrate it - and create supplemental packages which allow to embed the 3rd party Software into the Vector embedded software and the Vector tool chain.
The required MCAL version is requested in a project and delivery specific questionnaire, where the customer provides all necessary information prior to the delivery. In order to be able to integrate and deliver the MCAL it is mandated that the MCAL is provided to Vector well in advance of the delivery. Details can be found in the Product Information attached to your offer.

The MCAL Integration Package contains the following services:

13.5.1 Coordination with the customer

- Coaching in relation to the timeline of the MCAL vendor
- Check on the basic parameters as compiler version, AUTOSAR Version, etc.
- Early clarification of technical issues, e.g. in case of compatibility issues
- Lead contact in relation to the interaction of BSW and MCAL
13.5.2 Embedded-Integration

- Incoming goods inspection and proof of possibility to integrate it: start-up of the MCAL on a suitable evaluation board and performance of integration tests with the MICROSAR basic software.
- Compile-/ Link-Test in relation to the higher software layers
- Test of the basic functionalities of the MCAL which are relevant for the BSW (e.g. CAN communication, NV data storage, etc.)
- Create/serve embedded-interfaces between BSW and MCAL (MemMap, Compiler Config)
- Development of wrappers in case of "mixed AUTOSAR" projects. (This service is available for BSW according to AR 3.x and MCAL up to and including AR 4.0.3)

13.5.3 Tool-Integration

Depending on the characteristics of the chosen MCAL there are several possibilities to integrate it into the Vector tool chain. Our emphasis is to achieve the best and most comfortable solution for the customer. Basic conditions for embedding the MCAL components into the BSW configuration hereby are the following:

- existence of AUTOSAR conformant description files and
- the ability of the MCAL validators / configurators to work on AUTOSAR conformant configuration files.

If the MCAL supports these pre-conditions, it will be integrated into the configuration tool DaVinci Configurator Pro, which allows generating the module afterwards.

If proprietary formats are being used or if the MCAL configuration tool contains complex abstractions for an easy creation of the configuration and validation, Vector will provide adequate means to facilitate the parallel usage of different configuration files and tools.
14 MICROSTAR.EXT – AUTOSAR Drivers for Control of External Devices

MICROSTAR.EXT contains communication-related AUTOSAR transceiver drivers for CAN (CanTrcv), FlexRay (FrTrcv), LIN (LinTrcv), Ethernet (EthTrcv) and drivers for other external devices such as EEPROM, flash memory and watchdog (Ext). Functions contained in the drivers were specified by AUTOSAR in the “ECU Abstraction Layer”. They conform to AUTOSAR 4.x and have each been optimized for a specific device and are available for many commonly used devices.

Functions contained in the drivers were specified by AUTOSAR in the “ECU Abstraction Layer”. They conform to AUTOSAR 4.x and have each been optimized for a specific device and are available for many commonly used devices.

14.1 Overview of Advantages

> Optimal control of your external transceiver and memory devices
> Available for AUTOSAR 4.x and 3.x
> Support of transceivers for partial networking
> Additional support for LIN and Ethernet transceivers
> Simplified configuration by considering parameter dependencies with other modules
> Development accelerated by plausibility and completeness checks in the Configurator

14.2 Application Areas

MICROSTAR.EXT gives you a turn-key solution for driving your external peripheral devices. So, it is not necessary to modify the functional software when switching out external hardware. All you need to do is switch out the relevant drivers from MICROSTAR.EXT.

Depending on the requirements of your application, you can add other packages (e.g. MICROSTAR.CAN, MICROSTAR.MEM, etc.) for a complete communication stack or memory management per AUTOSAR specification.

Partial networking in CAN networks requires special transceivers. For many of these transceivers a suitable driver (CanTrcv) is already available with MICROSTAR.EXT.
14.3 Modules and Their Add-Ons

> **CanTrcv – CAN Transceiver Driver**
This driver is responsible for controlling the operating states of an external CAN transceiver. It contains control of wake up and sleep functions.

> **EthTrcv – Ethernet Transceiver Driver**
EthTrcv offers a uniform and hardware-independent interface for driving multiple transceivers of the same kind. Configuration of EthTrcv is transceiver-specific and considers the properties of the physical network that is used.

Extensions to AUTOSAR Standard

> Ethernet Transceiver Driver, also for Powerline Communication (PLC) and Wireless LAN (WLAN) (AUTOSAR 4. x and 3. x)

> **FrTrcv – FlexRay Transceiver Driver**
The FrTrcv driver for an external FlexRay transceiver is responsible for switching the transceiver on and off.

> **LinTrcv – LIN Transceiver Driver**
The LinTrcv module for an external LIN transceiver is responsible for monitoring and driving the wake up and sleep functions.

Extensions to AUTOSAR Standard

> LIN Transceiver Driver (also for AUTOSAR 3. x)

> **Ext – External Drivers**
On request, Vector will provide you with the implementations of drivers for externally connected devices.

> Adc (EXT)
> Eep (EXT)
> Fls (EXT)
> EthSwt (EXT)
> Eth (EXT)
> Wdg (EXT)

14.3.1 Vector Modules as Extension of the AUTOSAR Standard

> **vSbc – Vector System Basis Chip**
vSbc provides an abstraction layer for an externally connected System Basis Chip (SBC). The implementation provides a hardware independent interface that can be used by upper layers to control the SBC hardware. Depending on the SBC peripherals, vSbc includes AUTOSAR compliant upper layer modules (i.e. CanTrcv, LinTrcv and Wdg) that allow the BSW stack to access the related SBC functionality. Further SBC functionalities can be accessed by a complex driver or an IO hardware abstraction using the RAW-API that is provided by the vSbc implementation. The RAW-API allows basic (read/write) access to SBC registers. vSbc requires a suitable MCAL driver (e.g. Spi) to establish the connection to the SBC.
14.4 Configuration

For convenient configuration we recommend our DaVinci Configurator Pro. For more details, see the separate product information.

14.5 Other Relevant MICROSAR Products

The external devices are physically driven via SPI, DIO or a port. You will need the relevant drivers for this (Spi, Dio or Port) from MICROSAR.MCAL.
14.6 Additional Services

Vector can offer services for integrating the configuration of your drivers or drivers by third-party producers, such as semiconductor manufacturers, in DaVinci Configurator Pro. This lets you configure the entire ECU software with a single tool, seamlessly and quickly.
15 MICROSTAR.IO – AUTOSAR Input-Output Hardware Abstraction

The cluster IO establishes a connection between the application (e.g. the SWCs) and the MCAL modules. This gives the application or SWC access to I/O ports, for example to read sensor data or steer actors.

To fulfill this, the IO cluster contains specialized BSW modules. Additionally, Vector offers the possibility to establish a perfectly fitting IOHWAB for the ECU by using DaVinci Developer.

15.1 Overview of Advantages

- Quick implementation of user-specific code for acquiring and providing sensor and actor signals
- Generation of code examples: the user can select read/write digital signal templates with expandable C-code
- Provision of SWC descriptions with all necessary interface definitions

15.2 Vector Modules as Extension of AUTOSAR Standard

- **vDioHwAb** – Vector Digital Input Output Hardware Abstraction
  
  The Digital Input Output Hardware Abstraction (vDioHwAb) establishes a connection between the application and DIO signals from the MCAL. The MICROSTAR vDioHwAb module therefore creates a SWC interface as well as DIO specific code templates. It offers a quick and easy access to the DIO module. This way, the vDioHwAb module covers a part of the IoHwAb and can be extended by further IoHwAb modules which for example allow access to ADC or ICU channels.

15.3 Configuration

For convenient configuration, we recommend the DaVinci Configurator Pro. For further details, please refer to the separate Product Information.
15.3.1 vDioHwAb
DaVinci Configurator Pro checks for plausibility of the configuration parameters for MICROSAR.MCAL and MICROSAR.IO. The following “bottom up” approach is recommended:

> Configure the MCAL driver
> Configure MICROSAR.IO
> Generate the SWC description belonging to MICROSAR.IO

In configuring MICROSAR.IO, you define each individual signal. Any number of port prototypes may be derived from a port interface here. 1 : n allocations of port prototypes to runnable entities is possible for sender/receiver interfaces. Runnable entities in the RTE and schedulable entities in SchM are user-configurable. MICROSAR.IO provides all of the necessary Client/Server interfaces and code templates needed to give the functional software access to the I/O signals via the RTE. They let the user condition and filter the signals.

15.3.2 IoHwAb
Using DaVinci Developer, you can create an IoHwAb SWC description and implement it into the SWC design afterwards with only little effort. Beyond this, the MICROSAR.RTE or the DaVinci Developer Option .CPG both offer the possibility to directly create a code template for your individual IoHwAb implementation from this IoHwAb SWC description. The code template is enhanced by ECU specific algorithms (debouncer, signal filters, etc.) by the development engineer and connected to the MCAL API’s.

15.4 Other Relevant MICROSAR Products
If I/O signals are accessed via an external device, your I/O Stack will need an additional suitable driver from MICROSAR.EXT.

15.5 Additional Services
Vector would be glad to support you in the development of a complete and ECU specific IoHwAb layer for your ECU in the framework of project work. You benefit from Vectors detailed knowledge of the AUTOSAR specification and methodology as well as its extensive experience in integration of ECU software.
MICROSAR.RTE - The optimized Run-time Environment for Software Components per the AUTOSAR Standard

MICROSAR.RTE (Run Time Environment) is the scalable and highly optimized AUTOSAR run-time environment from Vector. The RTe is a module introduced by AUTOSAR that manages communication between the software components (SWCs). It assures consistency of the overall information flow and represents the interface between the functional software, basic software (BSW) and complex drivers (Cdd).

16.1 Overview of Advantages

- Easy to configure and scalable
- Available for AUTOSAR 4.x and 3.x
- In-depth consistency check of the configuration
- Highly optimized code with intelligent synchronization mechanisms
- Quick entry into working with AUTOSAR, e.g. based on generated code templates for the software components (SWCs)
- Well-suited for migration projects
- Simplified testing of the application by XCP access to S/R ports, InterRunnable variables and Per-Instance memory

16.2 Application Areas

When the functional software of an ECU is implemented by AUTOSAR-conformant SWCs, the user needs the RTe as a run-time environment. This modular layout of the ECU software offers the user maximum flexibility: SWCs that have been manually developed or designed by model-based tools can be re-used in multiple ECU projects. The RTe only needs to be reconfigured and regenerated for the specific ECU and, if applicable, the BSW modules. It is also possible to use a SWC in multiple instances on one ECU.

In generating the MICROSAR.RTE, the user can choose between two modes:
> Contract Phase Generation for developing individual SWCs in an early phase. In this case, the generator just creates a header file for each SWC instead of the entire Rte. This makes it possible to compile the SWCs individually, e.g. to transfer them to a development partner as object code.

> Rte generation for the entire ECU software. The code generated in this mode is highly efficient and requires only little memory space. It is optimized for the entire ECU configuration and makes few demands on system resources due to short execution times and minimal interrupt disable times. One way this is achieved is by using intelligent synchronization mechanisms that are tuned to the properties of the hardware used.

### 16.3 Modules and Their Add-Ons

#### Rte – Runtime Environment

MICROSAR.RTE is compatible with AUTOSAR 4.x and 3.x. Specifically, MICROSAR.RTE contains the following functionality:

> Sender/Receiver and Client/Server communication

> Mode management

> InterRunnable variables as well as Exclusive Areas.

> Access to Nv block software components via sender/receiver ports

> Trigger for runnables

> Online and offline calibration of SWCs are supported as well as measurement of S/R ports, InterRunnable variables and Per-Instance memory using the XCP protocol.

> Multiple instancing of SWCs and per-instance memory.

> Schedule Manager/BSW Scheduler (SchM): Since AUTOSAR 4.0, the Rte has assumed the functionality of the SchM module. This was previously contained in MICROSAR.SYS. For more information on the SchM module, please refer to the chapter about MICROSAR.SYS.

> Support of the transformer interfaces for COMXF, SOMEIPXF and E2EXF. For details please refer to the chapter MICROSAR.COM.

> External Client/Server communication (Inter-ECU) through the optional SOMEIPXF.

### Extensions to AUTOSAR Standard

Beyond the standard, MICROSAR.RTE for AUTOSAR 3 includes the following important services:

> Generation of code templates for SWCs based on the “SWC Description”. These templates contain all APIs of the Rte.

> Use of memory protection mechanisms as specified in the AUTOSAR operating system. This support is especially optimized when MICROSAR.OS is used.

> Configuration of initialization runnables for the AUTOSAR concept of “Mode-Dependent Runnables”.

> Generation of an HTML report showing Rte properties. It contains information such as the calculated Rte resource load (RAM + constants).

> Generation of an A2L file for simple linkage to existing calibration and diagnostic standards.

### Add-Ons

> MPU support for memory protection

> Multi-core support

> Post-build selectable: within the MICROSAR product family, this feature is available in the module "Identity Manager". For details please refer to the section “MICROSAR Variant Handling”.

#### BRE - Basic Runtime Environment

The Basic Runtime Environment (BRE) allows the integration of a non-AUTOSAR based application on the proven MICROSAR basic software. Using BRE a formal Software Component (SWC) Design as it is defined by AUTOSAR is not required. As the BRE allows using the existing application architecture usage of the BRE is the first step when migrating ECU projects towards an AUTOSAR architecture.
Using BRE the application accesses the BSW functionality directly without the need to define AUTOSAR software components (SWCs). The Rte as middleware between SWCs and BSW is not used and replaced by application code.

To simplify the integration of your application with the MICROSAR BSW, the BRE provides basic functionality usually implemented by the Rte:

- Generation of type definitions for service layer BSW modules
- Call of BSW main functions based on a configurable task mapping
- Managing of exclusive areas for BSW modules

As the BSW uses AUTOSAR conformant interfaces, the usage of the BRE requires that the application takes over the interface function normally provided by the Rte towards the BSW.

If the BRE provides too little functionality the MICROSAR RTE is the right choice. Experienced MICROSAR coaches can support you with your migration efforts towards an AUTOSAR compatible application and BSW architecture.

- **SchM – BSW Scheduler**
  
The Schedule Manager/BSW Scheduler coordinates execution of the BSW modules. In configuring you define tasks and the cycle times of the BSW modules. You also define the exclusive areas settings for each module centrally. For AUTOSAR 3.x, SchM is part of MICROSAR.SYS. In AUTOSAR 4.x, the MICROSAR.RTE assumes the functionality of the SchM.

16.4 Configuration

For configuration of the Rte, you need either DaVinci Configurator Pro with Option .RTE or DaVinci Developer. For more details, please refer to separate product information.

16.5 Scope of Delivery

In addition to the standard product components, you also get sample programs with make files.

16.6 Other Relevant MICROSAR Products

The Rte requires the presence of an operating system such as MICROSAR.OS or osCAN.
17 MICROSAR AMD – AUTOSAR Monitoring and Debugging

MICROSAR AMD contains many useful functions that simplify the development and testing of ECUs significantly. Core functions of AMD are reporting of errors and events from the application and the MICROSAR BSW as well as providing information on momentary CPU load and software execution times.

Figure 38: MICROSAR AMD modules according to AUTOSAR 4.3

17.1 Overview of Advantages

> Simplified testing of ECUs by acquiring internal ECU data – already implemented for AUTOSAR 3
> Easy measurement of startup and go-to-sleep behavior
> Determining CPU load and execution times of the application and the basic software
> Automatic generation of required A2L file
> Reporting of BSW error states and program flow trace messages
> Use of the widely-used XCP protocol standardized by ASAM

17.2 Application Areas

The MICROSAR AMD package efficiently supports you in testing your AUTOSAR ECU software. The basic software modules from MICROSAR AMD have access to all important internal variables, states and error messages of your MICROSAR basic software.

The XCP protocol (Universal Calibration Protocol) – familiar from the measurement and calibration field – is best suited for transferring ECU-internal parameters. Therefore, Vector decided to develop MICROSAR AMD based on XCP.
17.3 Functions

The functionality of the Dbg (Debugging) and Dlt (Diagnostic Log and Trace) modules is specified in AUTOSAR 4.x. Nonetheless, it is already available from Vector for AUTOSAR 3. In addition to the AUTOSAR standard, MICROSAR AMD offers the ability to determine CPU load and any execution times.

For display and analysis, the values read out with MICROSAR AMD are transmitted to an XCP Master via the XCP Slave.

You can use one of the following Vector tools as the XCP Master on the PC side:

- CANape – together with MICROSAR modules Dlt und Dbg
- CANoe.AMD from Version 8.1 – together with MICROSAR modules Dlt, Dbg and vRtm.

With these tools and a description of the measurement objects in the form of an ASAM A2L file, you can select internal ECU data and analyze its flows. CANape was extended for AMD with the “Digital Window” and the “Status” signal display. CANoe.AMD was extended with the special “State Tracker” window. This lets you display separate states and binary signals in a well-organized layout in one window.

17.3.1 Acquiring internal variables from AUTOSAR BSW modules

In the case of ECU tests, you can use the Dbg BSW module from MICROSAR AMD to determine which external and internal actions caused a state change of the BSW modules. The DBG module sends the needed internal variables of the MICROSAR BSW modules to the Master via XCP.
17.3.2 Acquiring internal variables from the MICROSAR.RTE

When using a MICROSAR.RTE as a run-time environment in your ECU software, it is possible to monitor the data flow between SWCs over XCP. After activating the “Measurement Support” option in the RTE configuration, the following RTE-internal objects are acquired:

- Inter-runnable variables
- Sender/Receiver Ports
- Mode Ports
- Per-Instance Memory

This is how you monitor the intra- and inter-ECU communication and measure or stimulate not connected sender/receiver ports.

17.3.3 Acquiring internal error messages and warnings

During run-time, the Dlt BSW module acquires and transmits, via Xcp, all occurring error messages and warnings that the BSW reports to the Det module. In addition, the Dlt module can actively forward to the XCP Master notifications that have been made to the error memory (Dem).

To monitor the program flow and status of the application, the Dlt module can report on any desired trace messages in plain text. This is done by providing a "WriteLine"-like API which can transmit any desired text at run-time. As an alternative, predefined text blocks optimized for execution time can be transmitted.

17.3.4 Determining CPU load and execution time

MICROSAR AMD and CANoe.AMD let you determine the execution time of any selected code sections from the application software or from the BSW modules. The measurement results are presented in HTML and CSV reports, which are generated with the CANoe Test Feature Set.

17.3.5 Generating the A2L file for the XCP Master

Based on the configuration data from the ECU configuration file (ECUC), you generate the A2L description file for the XCP Master with the DaVinci tools. If symbolic names exist for values of measurable objects, they are also stored for later visualization. It is also possible to incorporate other application-specific A2L fragments into the Master A2L file. Before starting the measurement, the ASAP2 Updater is used to update the A2L file with the actual addresses of the variables. The ASAP2 Updater is supplied as part of CANape and CANoe.AMD.
17.4 Configuration
For user-friendly configuration, we recommend DaVinci Configurator Pro. You will find more details on this in the separate product information.

17.5 Scope of Delivery
MICROSAR AMD consists of the following components:
- Software modules as source code
- A2L Generator (for Windows XP/Windows 7)
- BSW Module Description files
- Documentation

17.6 Other relevant products
Prerequisites for using MICROSAR AMD are the following Vector software modules:
- the simultaneous use of MICROSAR XCP for CAN, FlexRay or Ethernet, or
- VX1000 systems from Vector. These are advisable for maximum data throughput with minimal effects on execution time. You will find details on this at our product website: www.vector.com/vx. For safety reasons, very often it is not allowed to keep the measurement and calibration interfaces active in serial production projects. The module VX1000If allows keeping the VX1000 measurement and calibration hardware driver in the BSW also in serial production, but in a deactivated mode. Through an API the VX1000 driver can be released again for check and development purposes. Delivery of the module must take place within a MICROSAR SIP to receive approval for this use case in serial production. The activation of the VX1000 driver within a serial production project at runtime however is also not allowed when using the module VX1000If.
18 MICROsAR Solutions

Now the descriptions of the already introduced solutions will follow. These cross-packet solutions can contain very different modules from very different clusters. These solutions are (in alphabetical order):

- MICROsAR Gateway
- MICROsAR.HSM
- MICROsAR.IPC
- MICROsAR J1939
- MICROsAR Multi-core
- MICROsAR.OTA
- MICROsAR POSIX
- MICROsAR Security
- MICROsAR Safe
- MICROsAR Variant Handling
- MICROsAR vVIRTUAL Target
MICROSAR Safe – Functional Safety According to ISO 26262 up to ASIL D for ECU Software

The ISO 26262 safety standard defines criteria by which safety-related ECUs shall be developed in the automotive field. MICROSAR Safe from Vector gives you a solution for implementing safety-related functionality up to the highest safety level (ASIL D) in an AUTOSAR project.

AUTOSAR basic software developed according to ISO 26262 can help to reduce the number of partitions in the system and hereby can contribute to a higher performance. Many of our MICROSAR BSW modules are developed with the methods of ISO 262162/ASIL D and subsequently are able to co-exist with safety relevant SWCs without partitioning. Beyond this, further safety requirements can be implemented into the BSW if necessary.

### Figure 42: MICROSAR Safe Module according to AUTOSAR 4.2
Additionally to the above indicated, further modules are available as ASIL software, such as OESK NM or OEM specific software components.

### 19.1 Overview of Advantages

- Certified solution for all Automotive Safety Integrity Levels (ASIL A to ASIL D)
- ASIL and QM software sections are executed on a single microcontroller (Mixed ASIL)
- Reduces qualification costs
- Runtime optimized implementation of context management
- Monitoring of flow control and deadlines
- Validation of external communications
- Less partition switches and subsequently less runtime through safe basic software
- Suitable for Multi-core projects
19.2 Application areas

The MICROSAR Safe modules are “Safety Elements out of Context” (SEooCs), which are developed according to ISO 26262 / ASIL D.

MICROSAR Safe enables “freedom from interference” in executing safe software parts with different ASIL and non-safe software parts (QM software) on the same ECU (Mixed-ASIL systems). MICROSAR Safe is the result of many years of experience in the field of functional safety.

19.3 Functions

For projects according to the AUTOSAR 4 standard, MICROSAR Safe contains safe basic software with its core products SafeOs, SafeE2E and SafeWatchdog. They comply with the AUTOSAR specification and are compatible with all of the remaining MICROSAR modules which optionally can be provided in the context of SafeBSW as well. Beyond this, MICROSAR Safe also provides a SafeRte.

The combination of SafeOs, SafeE2E and SafeWdg is the minimum expansion stage which is necessary for mixed-ASIL systems (ECUs which contain both ASIL functions and functions without safety relevance). An extension of this mandatory package is possible according to your specific requirements - up to a full BSW stack according to ASIL D.

![MICROSAR Safe with SafeRte and SafeBSW according to AUTOSAR 4.2](image)

The core products of MICROSAR Safe contain the following functionalities:

19.3.1 SafeOS – the safe AUTOSAR operating system

The MICROSAR operating system’s option SafeOs assures memory protection for ECUs with safety-relevant applications on suitable microprocessors, e.g. with a Memory Protection Unit (MPU). MICROSAR.OS (SC3/SC4) is used to separate the various software partitions from one another. This prevents unauthorized writing by SWCs to the memory of other SWCs that could corrupt the data. In addition, it is assured that the context is switched correctly during a task switch or interrupt.

By offering scheduling with related timing protection and application termination functionality developed according to ISO 26262, SafeOs improves the support of safe systems with high availability requirements.

SafeOs contains additional functions that support partitioning by the MPU e.g. protected access to privileged registers, MPU test functions and protected data exchange across partition boundaries.
19.3.2 SafeE2E - Safe intra-ECU and inter-ECU communication

Data that is exchanged between safety-relevant applications on different ECUs must be checked for correct transmission. A checksum protects the data contents and the correct sequence of data is monitored by a message counter. If one of these checks fails, the application is informed, and it can react accordingly. This detects errors such as masquerading, failure, reversed data, etc.

The transformers ComXf, SomelpXf and E2eXf according to AUTOSAR 4.2 are available within SafeCom to support a safe communication. As an alternative Vector offers a protection wrapper within SafeE2e which provides an AUTOSAR-conformant and signal-based interface. This enables convenient use of the E2E verification in the application above the RTE.

19.3.3 SafeWdg - Flow control of safety-relevant software components

The WdgM module from SafeWdg monitors the correct timing and execution behavior of safety-related functions. This includes the “Program Flow Monitoring”, which is specifically defined in AUTOSAR 4.x for safety-relevant applications.

The SafeWdg package implements the interface to an internal or external hardware watchdog with the help of the modules WdgIf, Wdg resp. Wdg (EXT) and -on demand- can support external system basis chips (SBC) as well.

19.3.4 SafeRTE – safe ECU-internal communication

MICROSAR.RTE supports partitioning of memory sections via the operating system’s option “SafeOs”. According to ISO 26262, a qualification of the RTE is necessary to establish a safe communication between applications within an ECU. The static analysis tool “RTE Analyzer” is available as option for this purpose.

19.3.5 Validation of the hardware

The modules of MICROSAR Safe assume a safe hardware within the customer project. If this requirement is not fulfilled by the hardware, you may also accomplish this by suitable software test functions. This requires that you consider the project specific safety goals.

SafeOs already offers auxiliary functions in reference to the MPU. Other functions for validating the RAM, Flash, MPU, IO, etc. can be provided by Vector in the context of project work.

19.4 Configuration

We recommend that you use the DaVinci Configurator Pro tool to configure the BSW modules of MICROSAR Safe.

19.5 Scope of Delivery

MICROSAR Safe is developed according to the AUTOSAR 4 standard. In addition to the BSW modules every delivery of MICROSAR Safe also comes with the required safety manuals for the modules developed according to ASIL. The delivered software package altogether is one single SEooC for which Vector will provide a corresponding project specific Safety Case document.

MICROSAR Safe is also available for AUTOSAR 3 projects, enabling you to realize Mixed-ASIL projects according to this standard. The aspect “freedom from interference” in this case is supported by the functionalities MPU support including safe context switch, runtime measurement and safe external communication. MICROSAR Safe for AUTOSAR 3 contains SafeOs, SafeE2e and SafeWdg, plus SafeCrc.
19.6 Implementation

Safety-relevant functions are supported by the above mentioned core products from MICROSAR Safe. They conform to the AUTOSAR specification, are developed according to ISO 26262/ ASIL D and in detail contain the following features:

<table>
<thead>
<tr>
<th>package</th>
<th>content</th>
</tr>
</thead>
</table>
| SafeOs  | > Memory protection and safe context switching (option for MICROSAR.OS)  
          | > Timing Protection (Option for MICROSAR.OS) |
| SafeE2e | > Protection Wrapper: End to End Protection Wrapper  
          | > ComXf, SomelpXf, E2eXf  
          | > E2e (End to End-Library)  
          | > Crc |
| SafeWDG | > WdgM: Watchdog Manager  
          | > WdgIf: Watchdog Interface  
          | > Wdg: Driver for internal Watchdog devices  
          | > Wdg (EXT): Driver for external Watchdog devices  
          | > vSbc: Driver for System Basis Chip devices (with Watchdog and Transceiver functionality) |
MICROSAR Security – Access Security for AUTOSAR ECU’s

Along with the growing volumes of security-related and personal information in the automobile, it is also becoming increasingly more important to protect against manipulation and theft. Security mechanisms are used to protect the integrity, authentication and confidentiality of information. In this area, Vector offers you the components specified in AUTOSAR 4.2 and AUTOSAR 4.3. The following figure shows the MICROSAR Security modules based on AUTOSAR 4.3.

![MICROSAR Security Module based on AUTOSAR 4.3.](image)

**Figure 44:** MICROSAR Security Module based on AUTOSAR 4.3.

### 20.1 Overview of the advantages of MICROSAR Security

- Standard-conformant implementation of security functions from a single source
- Established cryptographic algorithms
- Protection against unauthorized modification of critical data
- Protection against unauthorized reading of data
- Protection against replay attacks
- Authentication of communication Tx points

### 20.2 Modules and their Add-ons

#### 20.2.1 Crypto Service Manager (CSM)

- Access to cryptographic services
- Configuration of cryptographic services and algorithms used to execute the services
- Configuration for synchronous or asynchronous execution
- Configuration of secure counters
- Configuration of operations on cryptographic keys
20.2.2 Crypto Interface (CRYIF)

The Crypto Interface (CryIf) module makes it possible to use CSM hardware-based and software-based crypto solutions. The necessary allocation scheme is managed by the Crypto Interface. The CRYIF module provides a uniform interface for different crypto solutions, such as the following:

- Software-based implementation of algorithms which are provided by the Crypto (SW) module
- Hardware-based crypto functions, which are implemented by a Secure Hardware Extension (SHE) or a Hardware Security Module (HSM) via the Crypto (HW) module.

20.2.3 Crypto (SW)

The Crypto (SW) module provides implementations for cryptographic algorithms and functions in software which are supplied via the CSM. All computations are executed in software, and no special hardware is required to execute cryptographic operations.

20.2.4 Key Manager (KeyM)

The Key Manager provides capabilities for parsing and verifying certificates based on configurable rules. It uses CSM interfaces for storing certificates and cryptographic operations.

20.2.5 Vector Security Modules (vSecMod)

The OEM-specific vSecMod includes the Freshness value manager (vFVM) and the Key Manager (vKeyM) with the following functionalities:

- vFVM: Provides a freshness value to SecOC component to prevent replay attacks. This module is used by (and therefore requires) the Secure Onboard Communication (SecOC)
- vKeyM: Handles key exchange and key updates

For more information, please see the OEM-specific amendments in the program related product information attached to your offer.

20.2.6 Crypto (HW)

The Crypto (HW) module acts as the driver for accessing security algorithms and functions, which are provided via a Hardware Trust Anchor (HTA). Different HTA types are available such as Secure Hardware Extensions (SHE) and Hardware Security Modules (HSM). Vector offers the following options for the Crypto (HW) module according to the hardware platform and the derivative used:

- Integration of a Crypto (HW) developed by Vector
- Integration of a 3rd party Crypto (HW) developed at the semiconductor manufacturer

As a special implementation of the Crypto (HW), Vector can optionally provide the crypto module Vector Hardware Security Module (vHsm). The vHsm from Vector supplies the necessary algorithms for the CSM here (find more in the next chapter MICRO SAR.HSM).

20.2.7 Secured OnBoard Communication (SecOC)

The SecOC module, also called Authenticated Messaging, is used to authenticate the communication between two ECU’s. This validation prevents a third party from intervening or pretending to be the correct communication partner. This prevents manipulative interventions. The SecOC interacts with the PDU router. This interaction can be controlled by the application. The module offers the following functions:

- Transmission of authenticated and integrity-protected I-PDU’s.
- Authentication with a Message Authentication Code (MAC). The actual generation and validation of the Message Authentication Code is performed by the CSM.
Prevention of replay attacks. A counter, the “Freshness Value”, is used here. It is generated by an independent component, the Freshness Value Manager (FVM). Different approaches are supported for generating the Freshness Values.

20.2.8 Vector Ethernet Firewall (vEthFw)

The Ethernet Firewall (vEthFw) provides the implementation of a firewall for Ethernet communication. Its main task is to block undesirable incoming or outgoing data traffic to enhance the security of the overall network.

The Ethernet Interface (ETHIF) module permits additional callouts to the vEthFw module to enable the firewall to check the current Ethernet frame and to reject it if it does not conform to the configured set of rules. The vEthFw module enables the specification of filter rules for different types of data traffic (IPv4/IPv6, AVB and RAW Ethernet) and for different layers (UDP, TCP and RAW Ethernet).

20.2.9 Vector Internet Security (vIpSec)

The add-on vIpSec allows to establish an IPsec communication according to IETF RfC 4301. The functionality is restricted to transport mode and the usage of Authentication Header only according to RfC 4302. The Authentication Header adds data integrity and data authentication to the payload but does not allow confidentiality.

20.2.10 Vector Transport Layer Security (vTls (Client))

This module contains a Transport Layer Security Client. TCP-based communication is encrypted with vTls. You can select the encryption algorithm to be used.

20.2.11 Vector Hardware Security Module (vHsm)

Find more about vHsm in the next chapter MICROSAR.HSM.

20.3 Configuration

We recommend that you use the configuration tool DaVinci Configurator Pro, which makes the task of configuring the modules of MICROSAR Security easy. You will find details on this in separate product documentation.
21 MICROSER.HSM – Vectors Solution for Hardware Security Modules

The MICROSER.HSM cluster includes the module vHsm, which is a packet for a set of sub-modules. This vHsm is Vectors solution for Hardware Security Modules (HSM) of different semiconductor vendors. The solution allows for easy and extensive configuration according to the use case and its requirements. It provides cryptographical services running on a separate and secure core of the microcontroller and makes use of available hardware accelerators.

The layer illustration below gives an overview of modules that are related to the vHsm solution. The vHsm additionally needs the crypto driver Crypto(vHsm). For a detailed view on the architecture of vHsm refer to section Modules and their Add-Ons.

The vHsm includes various cryptographic functions and features, which are described in detail in the following sections. The package is a standalone package for a HSM including scheduling and crypto job processing. In combination with the Crypto(vHsm) driver, it provides all necessary parts for interfacing with an AUTOSAR application stack.

21.1 The Benefits of MICROSER.HSM

- vHsm ensures that crypto material is never passed from the secure core on to the application core
- vHsm provides a modular architecture which can be configured by the user to adapt the vHsm to match performance and foot print requirements
- vHsm comes with a large library of crypto algorithms which can be executed on the HSM
- vHsm is provided as source code together with comprehensive configuration tool and debugging capabilities

21.2 Application Areas

The cluster MICROSER.HSM allows to use hardware acceleration as well as isolation features for cryptographic operations. The vHsm is meant to be used on a microcontroller with integrated HSM including a secure core. Hence, vHsm is running
separately and isolated from the AUTOSAR stack on the secure core with independent memory (flash and RAM), while the AUTOSAR stack is running on the application core(s).

The communication of the vHsm with the AUTOSAR Stack is completely abstracted and managed by the crypto driver Crypto(vHsm). Crypto(vHsm) is AUTOSAR 4.3 compliant and part of the MICROSAR BSW.

Due to the integration with an AUTOSAR Stack, development errors can be reported via a DET proxy on the HSM to the DET of the AUTOSAR Stack. Furthermore, errors can be written into secure data flash and read by the application.

Use cases of the vHsm include efficient cryptographic algorithm processing and acceleration as well as key management for e.g. secure boot and Secure OnBoard Communication (SecOC).

21.3 Functions

vHsm comes with an extensive set of features and crypto algorithms that can be executed on the HSM whether in software on the secure core or hardware accelerated.

21.3.1 Key Storage

- Secure storage inside the HSM (for symmetric/asymmetric keys, certificates and any other security relevant data e.g. mileage, freshness values)
- Key installation of symmetric keys according SHE 1.1
- Free choice of key storage (flash or RAM)
- Redundant and reset safe storage
- Keys are pre-loaded / cached on startup to avoid loading with every use
- Keys can be locked until secure boot has finished
- Keys can be configured as write once
- Keys can be persisted immediately or delayed being able to persist multiple keys at once (optimizes speed and reduces amount of flash accesses)

21.3.2 Secure Boot Support

- Support of Secure Boot and Authenticated Boot
- Can be split up in several jobs/phases
- One job is started automatically, e.g. for bootloader verification
- Bootloader can start one or more further jobs (Sequential or parallel secure boot)
- Secure Boot sanctions in case of failures are configurable (System reset, logging only, no unlock of keys)

21.4 Modules and Their Add-Ons

The vHsm software stack is not specified in the AUTOSAR standard, even though it makes use of some AUTOSAR compliant modules. Hence, vHsm consists of several dedicated modules as well as Vector standard software.

This section gives an overview of the internal architecture of MICROSAR.HSM and its functional scheme. Furthermore, available Add-Ons are described.

21.4.1 Modules Contained in vHsm

- vHsm – Vector Hardware Security Module Core
  
  vHsm is the key component of the vHsm solution. It offers:
  
  - Reception of incoming crypto jobs from the AUTOSAR crypto driver Crypto(vHsm)
  - Evaluation of incoming crypto jobs and dispatching to the according sub-modules and drivers (supported job types are processing of crypto algorithms and key management)
  - Management of reads and writes to a non-volatile memory
vHsm_Hal – vHsm Hardware Abstraction Layer
vHsm_Hal provides the hardware abstraction of the HSM in the microcontroller and provides hardware accelerated algorithms (e.g. CMAC, AES, TRNG)

vHsm_Core – vHsm Software Authentication
vHsm_Core provides functionalities for secure boot and secure software download

vHsm_Custom – vHsm Custom Extension
vHsm_Custom provides a template for implementing custom extensions

Furthermore, the vHsm stack makes use of the following Vector standard software modules:

- CryIf, Crc, Crypto(Sw), Det, Fee, MemIf and FlsDrv

21.4.2 Add-Ons
For the vHsm add-ons and customizations are optionally available. Especially the crypto driver vHsm_Custom is available to integrate special user functionality to meet certain use case requirements. This AUTOSAR compliant crypto driver template needs to be configured and filled by the user. On request additional features can also be integrated by Vector.

Add-Ons

vHsm Asymmetric Crypto
- Provision of asymmetric crypto algorithms
- RSA and ECC signature generation and verification
- Secure storage and update of asymmetric keys

vHsm Certificate Handling
- Enables parsing and validation of certificates based on configurable rules in HSM

vHsm Updater
- Provides the possibility to update the vHsm (an update of the data flash of the hsm is not intended)
- Comes as a separate module which needs to be places as separate binary in the HSM code flash
- Allows for protected vHsm updates (encrypted and signed)

21.4.3 Integration Scenarios
For the integration and usage of MICROSAR.HSM there are two major scenarios. MICROSAR.HSM is usable with MICROSAR from Vector, and it is also compatible with any 3rd party AUTOSAR stacks that are compliant to AUTOSAR 4.3. Figure 2 depicts the interaction of the vHsm with an AUTOSAR stack and a flash bootloader.
21.4.3.1 vHsm and Vector MICROsAR
In this constellation, vHsm is used in combination with MICROsAR from Vector. This single-source solution provides optimized interaction on embedded code and tooling level, as well as pre-integration of vHsm and MICROsAR.

In this case, the delivery consists of the following items:

- vHsm software integration package (SIP) and source code
- MICROsAR SIP including crypto driver Crypto(vHSM)
- DaVinci Configurator Pro for vHsm and MICROsAR configuration

21.4.3.2 vHsm and 3rd Party AUTOSAR
In this constellation, vHsm is used in combination with a 3rd party AUTOSAR 4.3 compliant stack.

Note: In case of using vHsm with a 3rd party AUTOSAR stack, special care about the consistency of the vHSM and AUTOSAR configurations must be taken by the user. No tool support can be provided by Vector in such a case.

In this case, the delivery consists of the following items:

- vHsm software integration package (SIP) and source code
- Crypto driver Crypto(vHSM) (AUTOSAR 4.3 compliant) for the Application core(s)
- DaVinci Configurator Pro for vHsm configuration

21.4.3.3 vHSM and Flash Bootloader
In this constellation, vHSM is used in combination with the Flash Bootloader (FBL) from Vector. This provides several advantages:

- Seamless integration with Vector Flash Bootloader for e.g. Secure Boot use cases
- More detailed testing and pre-integration possibilities for vHsm and FBL at Vector
- Simplified tool support
  - DaVinci Configurator is used for vHsm as well as for Flash Bootloader

In this case, the delivery consists of the following items:
MICROSAR

> vHsm software integration package (SIP) and source code
> Flash Bootloader SIP
> DaVinci Configurator Pro for vHsm and FBL configuration

21.5 Configuration

For convenient configuration we recommend our DaVinci Configurator Pro. For more details, see the separate datasheet. Using DaVinci Configurator Pro, you can easily ensure consistency with your MICROSAR configuration.

![Figure 47: Configuration of MICROSAR.HSM (vHsm) in DaVinciConfigurator Pro](image)
22 MICROSAR Gateway – Basic Software for Gateway ECUs

Gateway ECUs are central node points in the often-heterogeneous network architecture of a vehicle. In this role, a gateway interconnects the various networks to enable data transfer between distributed ECUs also across different network types. The performance indicators of a gateway range between the contradictory priorities of high data throughput and low resource load (especially RAM and CPU utilization) and low latency times in transmission.

Vector offers optimized modules for gateways, which enable routing to different protocol levels and between different bus systems (CAN, LIN, FlexRay, Ethernet). Building upon these basic functions, the MICROSAR Gateway offers a number of special functions, such as mirroring of sub-networks and a modular plug-in concept for extending existing functionality. The MICROSAR Gateway offers the necessary flexibility to cover specific application cases, because of its three-stage concept of basic functionality, special functions and extendibility.

To enable optimal gateway performance, certain aspects of the MICROSAR Gateway follow an independent architecture. They meet the requirements of AUTOSAR Architecture Conformity Class 2 (ICC2), so AUTOSAR software components can be incorporated.

Figure 48: Multi-Layer Gateway in MICROSAR

22.1 Overview of the advantages of the MICROSAR Gateway:

> Routing on different protocol levels (PDU, high-level TP, PDU section, signal)
> Routing between different network types (CAN, LIN, FlexRay, Ethernet)
> Flexible configuration of buffer concepts (Rx FIFO queue, Tx FIFO queue, priority-based queue)
> Flexible configuration of processing concepts (interrupt, task)
> Post-build approach to flexible post-loading of routing relationships
> Mirroring of sub-networks to diagnostic access (CAN and Ethernet)
> Extended support for NM coordinators
> Toolchain for automatic processing of input data (.dbc, .ldf, .fibex, .arxml and proprietary formats)
> Defined interfaces for extending the toolchain (scripting, extensions) and the basic software (Cdd support)
> Optional project-related support of gateway projects by Vector in close cooperation with Tier1

22.2 Application Areas

MICROSAR Gateway offers users the basic software they need to route data on the optimal protocol level between CAN, LIN, FlexRay and Ethernet networks. This flexible approach enables use of the MICROSAR Gateway in developing:

> Local gateways (such as in the door area)
> Domain controllers (such as Body Controller)
> Central gateways (such as central diagnostic access or for connectivity applications).

### 22.3 Functions

> Routing on different protocol levels
  > PDU routing (on interface level):
  >   > Efficient event-based routing on message and PDU levels
  >   > Support of various buffer strategies (last-is-best, FIFO, priority-based)
  >   > Routing of “contained PDUs” (SoAd and IpduM)
  >   > Supported networks: Ethernet, CAN, LIN, FlexRay
  > High-Level TP routing
  >   > Routing between different transport protocols
  >   > Segmented transmission with time-based flow control and bandwidth management
  >   > Special buffer handling: “Routing-on-the-fly”, “request queuing”, “buffer sharing”
  >   > Supported networks: Ethernet, CAN, LIN, FlexRay
  > PDU section routing (description-based routing)
  >   > Efficient routing of data sections in PDUs, defined by start bit, target bit and length
  >   > Controlled Tx behavior (periodic sending, sending on data change, reaction to timeouts, minimum Tx interval, etc.)
  >   > Supported networks: network-independent routing
  > Signal-based routing
  >   > Routing of decompressed data signals
  >   > Controlled Tx behavior (periodic sending, sending on data change, reaction to timeouts, minimum Tx interval, etc.)
  >   > Option to perform conversions (Endianess, signal length, signal contents)
  >   > Supported networks: network-independent routing

![Figure 49: Routing on different protocol levels](image)

> Extendable basic software (plug-in architecture)
> Efficient integration of your own software into the data flow (implemented via Complex Device Driver (Cdd) in accordance with AUTOSAR)
> Configuration of Cdds in the configuration tool
> Generation of templates for integrating the Cdd into the basic software
> Cdds can be integrated above the interfaces, beneath the PDU router and above the PDU router
> Callouts for integration of your own software into the control flow (often include option of influencing the data as well)

> Gateway optimizations
> Deferred event caching: efficient processing of Rx events on COM level
> Timing domains: efficient handling of Tx events on COM level
> Description-based routing: efficient routing of data sections

> Dynamic routing
> Learning diagnostic routing: routing of diagnostic requests according to the learned ECU position (e.g. by receipt of a diagnostic response)
> Routing according to CAN ID, e.g. routing of parts of the CAN ID by use of the AUTOSAR Meta-Data feature

> Mirroring
> Mirroring of an internal CAN or LIN channel to diagnostic CAN
> Mirroring of multiple internal CAN, LIN networks and a FlexRay (A+B channel) network on diagnostic Ethernet

> NM coordination
> Post-build: update of gateway configuration without having to reprogram the entire ECU. For more information on post-build, please refer to the separate chapter “MICROSAR Variants Handling.”

22.4 Configuration

Gateway configurations are often very extensive. This makes it all the more important to be able to automatically parameterize the basic software according to the input data provided by the OEM. The DaVinci Configurator Pro configuration tool from Vector supports automatic parameterization by various methods:

> Configuration via an AUTOSAR System Extract (.arxml): The data format defined by AUTOSAR permits explicit specification of routing information. This information is read in by DaVinci Configurator Pro and is transformed into a software configuration.
> Configuration via extended network files: The proprietary network files (.dbc / .ldf / .fibex) generally do not offer any ability to specify routing information natively. However, DaVinci Configurator Pro interprets various supplemental attributes and can evaluate rules (e.g. an existing identical naming of signals), in order to derive routing information.
> Configuration via proprietary data (e.g. .xls): The Vector toolchain also offers the option of defining routing information via a separate additional file. This file makes it possible to define relationships between messages and signals from the network files in a Vector-specified XML format (Vector System Description Extension, VSDE), thereby specifying routing relationships. A VSDE file is especially well-suited for integrating proprietary routing configurations into the toolchain. This requires implementation of a conversion routine, which transfers the proprietary file into a VSDE file.
> Scripting: The methods described above may also be combined with a scripting solution. Using DaVinci Configurator Pro Option WF (Workflow), you can create scripts specifically for your project, which you can use to supplement routing information in programming.

The software configuration generated by these methods is not restricted to a one-time process. If the network description in the input files should change, the software configuration can be automatically updated.
In addition to automatic derivation of the configuration, all parameters can also be processed manually in DaVinci Configurator Pro. For more details on this, please refer to the separate product information.

22.5 Project Support

Requirements placed on gateway ECUs are especially frequently both OEM-specific and application-specific, and therefore a standard product generally cannot satisfy them fully. Nonetheless, Vector offers a two-stage approach for efficiently addressing these requirements in your project:

> First, MICROSAR Gateway supports many extension options in the form of a plug-in architecture that lets you integrate software you have developed yourself or procured from third-party suppliers.

> As an option that builds upon these interfaces, we can also offer the service of working together with you to handle project requirements. This enables much more flexible implementation of customer requirements than by a product approach which is characterized by release cycles: Within a joint project, we are then glad to also implement the product extensions needed to satisfy customer requirements.
23 MICROSAR Multi-Core – The AUTOSAR Solution for Multi-Core Processors

The introduction of multi-core processors changes the design of the AUTOSAR software. Individual AUTOSAR applications can now be distributed to different processor cores and can then be operated simultaneously. Crucial to success is an optimal distribution with minimal synchronization losses, e.g. caused by wait times. MICROSAR Multi-Core assists you comprehensively in this effort.

23.1 Overview of MICROSAR Multi-Core Advantages

- MICROSAR Multi-core is fully integrated into the overall MICROSAR package and is mainly realized by MICROSAR.OS and MICROSAR.RTE. This has the following advantages:
  - A single OS configuration for all processor cores
  - Shared memory and efficient spinlocks for the RTE to shorten execution time and memory usage
  - MICROSAR Multi-core is closely interlocked with the Vector MICROSAR Safe solution: All multi-core functions fit seamlessly into the MICROSAR safety concept
  - MICROSAR Multi-core implements AUTOSAR Basic Software satellites on slave processor cores to assure time-optimized handling of service functions
  - Mixed ASIL Partitioning & BSW Split enables to shift part of the runtime of the basic software to another core in order to relieve the core on which the basic software runs. It also supports partitioning of communication clusters in mixed ASIL systems (e.g. CAN and Ethernet stacks in different partitions).
  - Easy to configure
  - MICROSAR Multi-core is available for individual MICROSAR modules.
  - With the TA Tool Suite from Vector you can efficiently optimize your application regarding CPU load balancing between the cores, response time of effect chains, reduction of inter-core communication and more.

23.2 Application Areas

MICROSAR Multi-core is used for the operation of existing processor cores in applications in which:

- the Basic Software runs on one of the existing cores and
- any desired distribution of the application software to any of the available cores is possible.

The goal of this approach is to distribute the application to multiple processor cores with the greatest degree of CPU utilization. MICROSAR Multi-core offers the following functionalities to achieve this:

- It offers runtime-efficient mechanisms for all processor cores to utilize BSW services
- It implements BSW satellites on slave processor cores as necessary
- It uses standard RTE mechanisms to share the services of the Basic Software, which optimizes execution time and minimizes memory usage

This multi-core approach also works with the MICROSAR Safe solution.

23.3 Functions

23.3.1 RTE

In the multi-core field, shared memory is advantageous for the MICROSAR.RTE, because it minimizes the execution time and the memory usage. Spinlocks are used as they are necessary to assure data consistency. Wait times for the processor cores then only occur if they access the same memory range at the same time.

23.3.2 Operating System

Only a single OS configuration is required for all participating processor cores. One aspect that goes beyond the AUTOSAR standard is that MICROSAR Multi-core offers faster spinlock APIs which allow to keep the data in generally accessible memory areas.
23.3.3 ECU Manager

The ECU Manager runs on the main processor core. Coordination between the cores is handled by a satellite on each of the other cores. This means that just a single configuration is required, even for the EcuM. It is valid for all processor cores.

23.3.4 Watchdog Manager

The Watchdog Manager also runs on the main processor core. On all other cores, there is a satellite which processes the application’s “checkpoint reached” function and synchronizes its status with the master. This assures that service functions of the WdgM are processed to optimize execution time.

23.3.5 Diagnostic Event Manager

The Diagnostic Event Manager implements the Master-Satellite pattern to allow both:

- Runtime efficient multi-core architectures due to core-local APIs for reading and writing of diagnostic event status
- Mixed-ASIL systems in which the Dem entities are being executed in the same ASIL-level as the application SWC

23.3.6 Mixed ASIL Partitioning & BSW Split

Mixed ASIL Partitioning & BSW Split enables to shift part of the runtime of the basic software to another core in order to relieve the core on which the basic software runs. This relief is achieved by outsourcing one or more bus systems. Specifically, the main functions and interrupts of the outsourced busses are moved to the other core. However, this does not accelerate the basic software, nor does it increase the efficiency of the stack. Latencies and overhead are even created by communication across the core boundaries. A measurable positive effect can only be achieved if those bus systems are moved, which contributes significantly to the load on this core. Then this feature can mean the difference between an overloaded and a stable running system.

23.4 Configuration

23.4.1 Configuration of the Os

During the configuration it is statically defined, on which processor cores the OS should run. OS applications are configured and each is individually assigned to one specific core. Via an allocation of OS objects like e.g. tasks, interrupts and alarms to OS applications, their core membership is finally determined.

23.4.2 Configuration of the RTE

The RTE must know, which software is running on which processor core. Only then the Rte is able to:

- assure consistency in the data exchange
- activate runnables across processor cores if necessary.

In the configuration process, this involves the mapping of the individual runnables to OS tasks and subsequently allocating these tasks to an OS application. Each OS application is then mapped back to a specific processor core.
We recommend the use of DaVinci Configurator Pro to make it easy to configure your ECU. You will find more details on this process on our website [https://vector.com/vi_davinci_configurator_pro_de.html](https://vector.com/vi_davinci_configurator_pro_de.html).

23.5 Scope of Delivery

With this product you get all of the individual modules of the MICROSAR package that are necessary for multi-core projects. Some modules are especially modified to deal with multi-core in a more efficient way.

- Master-Satellite-Modules: WdgM / WdgI, Dem, EcuM, Xcp, Rtm, Det
- Application Proxies: NvM, Com
24 MICROsAR Variant Handling – Solutions for Flexible Configurations in AUTOSAR

In traditional task distribution for ECU development, the automotive OEM defines the communication and diagnostic interfaces. The supplier implements and builds the ECU according to these requirements. If its parameters need to be changed after delivery of the ECU, the supplier must develop a new version of the ECU. In the case of minor changes, this development chain costs unnecessary time and money. The development of ECUs with a relatively volatile BSW configuration in particular can benefit from the added flexibility of **Post-Build Loadable**.

Additionally, in the automobile there is an increase in the number of ECUs that can be installed in different variants; they are known as multiple ECUs. Your advantage is the use of a single hardware for the different application areas. This reduces hardware costs and effort in inventory management. On the other hand, it results in increased effort in software development and the management of software variants for production and service. For this use case, a suitable **Post-build selectable** solution is available with the optional MICROsAR Identity Manager.

24.1 Post-Build Loadable – Post-Build Modification of BSW Properties

24.1.1 Overview of Advantages

- Many BSW parameters can be modified directly by the automotive OEM
- Flexible and spontaneous modification of the BSW in the framework of development and testing
- Post-Build Updates without the use of compilers (compiler licenses)
- Central resource management makes it possible to flexibly extend the BSW configuration by adding other elements (e.g. messages)
- Special extensions in DaVinci Configurator Pro ensure trouble-free Post-Build configuration
- Maximum of safety by consistency checks in the MICROsAR BSW and DaVinci Configurator Pro
- No change needed to the application architecture: Use of the extensively proven MICROsAR BSW according to AUTOSAR
- Post-Build Loadable is available as option for numerous BSW modules and functions

24.1.2 Application Areas

Post-Build Loadable permits modification of certain BSW properties from the areas of diagnostics and communication after the ECU has been built. Along with modifying parameters such as the CAN ID, send type and default values, new objects can also be introduced into the ECU at post-build time. For example, gateway routing tables can be extended by adding new messages or signals.

For the adaptation of BSW parameters at a post-build time point, only the MICROsAR delivery is needed. Neither the application nor compiler / compiler licenses are needed for a post-build update. Also, there is no need for an adaptation of the application layer. Modifications to the BSW can be made by the automotive OEM – directly and in an uncomplicated way.

24.1.3 Familiar Configuration Process

The configuration process at the post-build time point is performed with the standard configuration tool Da Vinci Configurator Pro. This makes a powerful tool available for smooth synchronization with many different databases (DBC, System Description, ODX, CDD, etc.).

The MICROsAR Post-Build Loadable tool chain generates a HEX file of the updated BSW configuration in a single step, which can then be loaded into the ECU using standard flash tools.
24.1.4 Scope of Delivery "Post-Build Loadable"

- License for enabling the option in the configuration tool
- Tools for creating a Build Loadable Update
- Documentation

24.2 MICROsAR Identity Manager - The Post-Build Selectable Solution for Multiple ECUs

24.2.1 Overview of Advantages

- Efficient handling of ECU variants
- Less administration effort
- Reduced inventory costs
- Resource optimized implementation of the BSW configuration
- Available as option for numerous BSW modules and functions

24.2.2 Application Areas

The Identity Manger alternatively can take over two functions:

Example "Door ECU": **ECUs that perform nearly identical tasks**, differing only in their Rx and Tx PDUs or in their diagnostics and address in the network, can be implemented in a vehicle as one part with just one part number. In this case, a single ECU is developed and produced, which during start-up knows which one it is and where it is installed, and therefore also which functions it needs to fulfill. The buffer for the Rx and Tx PDUs can be completely overlaid if their layout is identical. The application accesses signals and data elements independent of its identity. No differentiation is necessary in the code.

Example "carry-over ECU": **Functionally similar ECUs in multiple model series** can be developed and built as one part. These ECUs contain the software for all of the car lines in which they are used and support each model series with a communication description that may be completely different. In this case, the ECU Extracts upon which the configuration of the AUTOSAR ECU software is based, may differ very significantly for the different variants – for example they may exhibit a completely different signal layout or support a different number of buses.
24.2.3 Familiar Configuration Process

The configuration process of an ECU project that supports multiple variants is largely identical to a classic project without variants. Nonetheless, a separate set of system descriptions (Extract of System Description or legacy formats such as DBC, diagnostics descriptions (CDD, ODX)) is required for each variant.

24.2.4 Variant Selection

If an ECU supports multiple configuration variants, the variant to be used is selected in ECU initialization. The source of the variant information is user-definable, and it is implemented by the application. Such information might be connector coding or memory coding in flash.

24.2.5 Saving Resources

An ECU that contains multiple variants reduces the number of physical ECUs that need to be developed, produced and kept in stock. As there are multiple variants stored in the ECU now, the need for resources in the ECU will increase.

In code generation for the MICROSAR BSW, a number of optimization options especially focus on the most efficient memory use possible in RAM and ROM.

24.2.6 Scope of Delivery "Post-Build Selectable"

- License for enabling in the configuration tool
- BSW modules with Identity Manager functionality
- Documentation
25 MICRO SAR J1939 – AUTOSAR Basic Software Modules Specially Designed for Heavy-Duty Vehicles

In this section, we introduce the BSW modules defined in the AUTOSAR architecture for communication in J1939 networks: The network manager J1939Nm, the request manager J1939Rm, the transport protocol J1939Tp and the diagnostic module J1939Dcm. These modules are part of MICRO SAR.CAN and MICRO SAR.DIAG.

Figure 55: The MICRO SAR J1939 modules according to AUTOSAR 4.3

25.1 Overview of Advantages of J1939-Specific MICRO SAR Modules

- Available for AUTOSAR 4.x.

  - J1939Nm: Address arbitration and communication control according to SAE J1939-81. Available as option is the support for fully-dynamic addressing and address monitoring for self-configuring ECUs.
  - J1939Nm can be used together with CANNM to combine address arbitration and Sleep/Wakeup.
  - J1939Rm: Direct interface to the J1939Nm, J1939Dcm, Com and Rte modules as well as Cdds. Full access to request with timeout monitoring and acknowledgment (SAE J1939-21).
  - J1939Tp:
    - Full implementation of the J1939 transport protocol according to SAE J1939-81 with the variants BAM and CMDT (RTS/CTS), extended by support for the extended transport protocol according to ISO 11783-3 (ETP) and the FastPacket transport protocol according to NMEA2000.
    - Automatic selection of the right transport protocol (direct, BAM, CMDT, ETP) based on message size and destination address, reception of messages via any of the transport protocols.
  - J1939Tp (BAM and CMDT) also for AUTOSAR 3.x.
  - J1939Dcm: Diagnostic module for heavy-duty vehicle diagnostics according to SAE J1939-73. Fully integrated solution with joint use of the saved diagnostic data of the Dem over J1939 and UDS diagnostics.
  - Routing of messages based on their PGN, independent of source and destination addresses, including for consecutive messages.
> Access to message addresses in Cdds.
> Code- and run-time-optimized by need-specific configuration.
> Inter-module configuration of all communication-specific software modules.

### 25.2 Application Areas

The application area of the J1939 modules is in handling communication in heavy-duty vehicles over CAN networks with the special features defined in the SAE J1939 standard. They are implemented in J1939-specific BSW modules and are supported by extensions in neighboring modules. In addition, MICROSAR.CAN can also be used to implement ISOBUS ECUs (according to ISO 11783) in agricultural vehicles and implements. For that purpose, J1939Nm and CanIf were extended by functionalities for fully dynamic address arbitration and address tracking, and the ETP and FastPacket transport protocols were also implemented in the J1939Tp. Also, maritime use cases according to NMEA2000 can be supported by FastPacket and fully dynamic address arbitration.

### 25.3 Modules and Their Add-Ons

The BSW modules for J1939 contain the functions defined in AUTOSAR 4.1, specifically:

> **J1939Nm – SAE J1939 Network Management**
  > Address arbitration per SAE J1939-81 for J1939 networks with unchangeable addresses
  > Extensions to AUTOSAR Standard
  > Extension that adds fully-dynamic addressing, automatic change of the own address in case of conflict and automatic adaptation of all of the addresses used to the current changes

> **J1939Rm – SAE J1939 Request Management**
  > Supports request/acknowledgment protocol per SAE J1939-21

> **J1939Tp – SAE J1939 Transport Layer**
  > J1939 transport layer with support for broadcast (BAM) and directed communication (CMDT or RTS/CTS) according to SAE J1939-21
  > Extension to AUTOSAR Standard
  > Extension that adds the ETP (per ISO11783-3 and -7) and FastPacket (per NMEA200) transport protocols

> **J1939Dcm – SAE J1939 Diagnostic Communication Manager**
  > Support of the most important diagnostic messages of SAE J1939-73, parallel operation with DCM

#### 25.3.1 Paket-übergreifende Add-Ons

Es gibt auch Add-Ons, die sich auf ein oder mehrere Pakete auswirken. Im Detail sind das:

> MICROSGAR.PBL – Post-Build Loadable, Details hierzu finden Sie im Kapitel "MICROSAR Variantenhandling "
> MICROSGAR.IDM – Identity Manager, Details hierzu finden Sie im Kapitel "MICROSAR Variantenhandling "

### 25.4 Configuration

We recommend our DaVinci Configurator Pro for convenient configuration. You will find more details in the separate product information.

### 25.5 Other Relevant MICROSAR Products for J1939

> Dem from MICROSGAR.DIAG
> Det, BswM, EcuM and ComM from MICROSGAR.SYS
> MICROSGAR XCP enables measuring and calibrating according to ASAM XCP. The module was optimized, especially with regard to its use in conjunction with CANoe.XCP, CANoe.AMD and CANape. MICROSGAR XCP contains the related transport layer CanXcp for CAN ECUs.
Beyond the AUTOSAR standard, MICROSAR XCP supports generic readout of measuring objects. As a result, no addresses must be defined and updated in the a2l file. Data from any version or variant can be extracted with an a2l file independent from the MCU build. The generic readout feature requires usage of CANoe.AMD or CANape as XCP tool.

For safety reasons, very often it is not allowed to keep the measurement and calibration interfaces active in serial production projects. The module VX1000If allows keeping the VX1000 measurement and calibration hardware driver in the BSW also in serial production, but in a deactivated mode. Through an API the VX1000 driver can be released again for check and development purposes. Delivery of the module must take place within a MICROSAR SIP to receive approval for this use case in serial production. The activation of the VX1000 driver within a serial production project at runtime however is also not allowed when using the module VX1000If.
MICROSAR vVIRTUALtarget – Virtual Integration with vVIRTUALtarget

The process for developing an ECU typically is begun by creating the individual software modules (SWC). However, because of the continual trend towards shorter project times, there is generally insufficient time to sequentially create all of the necessary SWCs first and then test their interaction with one another and finally test their interplay with the basic software on the target hardware.

Numerous projects today are characterized by the need to start test runs in a very early project phase - sometimes in a phase in which the later target hardware has not yet been fully defined.

vVIRTUALtarget offers the right solution for this situation. It provides a runtime environment in which ECU software can be executed without requiring access to a real ECU. This environment enables decoupling of test execution from the real hardware and from the existence of the basic software. This, in turn, leads to substantial time advantages. The same configuration can be used for both the target hardware and the environment of vVIRTUALtarget:

After development of the SWCs and their system integration, the ECU software and its parameterization are at an advanced stage. Along with developing the application, another focus is on configuring the basic software. Here, vVIRTUALtarget enables testing that is independent of the target platform.
26.1 Overview of vVIRTUALtarget Advantages

- Early integration and testing of AUTOSAR 4 software modules
- Development of the ECU software independent of the availability of the real target hardware
- Dual target option for parallel integration on virtual and real hardware
- Greater depth of testing by additional testing in the virtual environment
- Microsoft Visual Studio is used as a convenient development and debugging environment
- Use of vVIRTUALtarget basic does not change workflow in ECU development
- Improved simulation of interruptions by interrupts and task preemption
- Simulation of multi-core systems

26.2 Application Areas

MICROSAR vVIRTUALtarget can be used in versatile use cases, such as:

- Acceleration of ECU integration
- Enhancement of available testing options by integration and system tests in a virtual environment
- Evaluation platform for demonstration purposes or prototyping

26.3 Functions

vVIRTUALtarget basic is intended for ECU integration and enables testing of the entire stack (application, SWCs, RTE and BSW) in a virtual environment. VTT offers the following functions:

- Early testing of the entire stack, even if the real ECU is still unavailable
- Use of the same MICROSAR configuration on the target hardware and in the virtual environment
- More convenient testing of the ECU software in the virtual environment
- Support of AUTOSAR-conformant 3rd party modules

Figure 58: Range of uses for vVIRTUALtarget (Version vVIRTUALtarget pro currently in development)
26.3.1 vVIRTUALtarget in conjunction with CANoe

Based on the current MICROSAR configuration, the tool vVIRTUALtarget basic generates a Visual Studio project in which a “System under Test” Windows DLL (SUT DLL) can be generated automatically from the BSW modules and the application. Currently, Microsoft Visual Studio 2013 (Express & Professional) and 2017 (Community & Professional) are supported.

![DualTarget Configuration](image)

**Figure 59:** The dual target principle

CANoe is used as runtime environment. The created DLL is referenced within the ECU node in the CANoe configuration. Debugging the virtual ECU is possible via the Visual Studio project. All functions of CANoe can be used for stimulation via the buses and I/O interfaces. Especially a simultaneous operation with real ECUs is supported.

A CANoe license is needed to stimulate and measure communication and I/O interfaces.

26.3.2 vVIRTUALtarget in conjunction with MICROSAR.MCAL and MICROSAR.OS

The BSW modules MICROSAR.MCAL and MICROSAR.OS were ported for the use of vVIRTUALtarget. The MCAL interfaces and behavior conform to AUTOSAR 4 specifications here. The MICROSAR.OS that is available for vVIRTUALtarget implements the SC2 “Scalability Class” with multi-core extensions.

The rest of the BSW modules do not require that any modifications be executed in the vVIRTUALtarget environment. The BSW modules are the same as those used in the real ECU.

26.4 Workflow

The workflow for configuring the BSW modules in the vVIRTUALtarget area does not differ from the approach taken with the real hardware platform. The only exception is that the MCAL modules specific to the vVIRTUALtarget and the vVIRTUALtarget OS may need special settings, if they cannot be automatically derived by the DaVinci Configurator Pro configuration tool.
26.5 Configuration

For convenient configuration, we recommend our DaVinci Configurator Pro. You will find more details in the separate product information. The generation of vVIRTUALtarget initialization codes and a Visual Studio solution requires vVIRTUALtarget basic license.

26.6 Product Components

The modules of vVIRTUALtarget are supplied as a Software Integration Package (SIP). The customer may choose to have the SIP supplied just for the vVIRTUALtarget platform (just vVIRTUALtarget MCAL and OS), or a dual-target SIP can be supplied together with drivers for a customer-defined real hardware platform (real MCAL and OS, as well as vVIRTUALtarget MCAL and OS).

26.7 System Requirements

Our product vVIRTUALtarget is a 64-bit application and therefore requires a 64-bit Microsoft Windows.

26.8 Further relevant MICROSAR Products for VTT

The "AUTOSAR Evaluation Bundle VTT" is available to get to know virtual integration and for testing its functionalities. For further information, please see chapter "AUTOSAR Evaluation Bundle".
27 MICROSAR POSIX – Connecting AUTOSAR Classic to POSIX Operating Systems

Driver assistance systems and infotainment are leading to increasingly more intensive intermeshing of established static ECU solutions with new dynamic services. That is why POSIX-based systems are being used more frequently in combination with classic AUTOSAR technology in ECU projects. The newly defined “AUTOSAR Adaptive Platform” standard underscores this trend. Therefore, MICROSAR offers solutions for using classic AUTOSAR functions under a POSIX operating system as well, or for exchanging data between a POSIX domain and an AUTOSAR domain.

![Figure 61: Typical use cases in POSIX based ECUs](image)

27.1 Overview of Advantages

- MICROSAR basic software is operated as a process in a POSIX-based system
- Supports multi-controller architectures
- MICROSAR basic software can be integrated with POSIX-based third-party operating systems

27.2 Use Areas

- AUTOSAR Classic basic software modules and legacy components operated as process under POSIX
- Diagnostics in multi-controller and multi-core architectures
- Inter Process Communication (IPC) for data exchange between different operating systems
- Flash Bootloader for POSIX operating systems
- ECUs designed to the new “AUTOSAR Adaptive Platform” standard for which Vector offers a comprehensive solution. Please refer to the separate product information sheet for details.

27.2.1 AUTOSAR Classic Basic Software Modules and Legacy Components as Process under POSIX

POSIX operating systems usually provide dedicated drivers for the typical bus systems in vehicle manufacturing, Ethernet and CAN. However, they lack the higher protocol layers for interfacing to an in-vehicle electrical/electronic system. One example of a frequently occurring application is the diagnostic interface over DoIP. Here it is advantageous to adopt existing implementations if possible.
For this use case, the necessary interface modules are available for the relevant MICROSAR modules, so that they can be connected to a POSIX system. All other modules remain unchanged, as do the related configuration tools. Consequently, this reduces the effort needed to migrate or work in this use case to a minimum. At the same time, the use of released components also assures bus compatibility of the POSIX ECU.

27.2.1.1 Runtime Environment

Handling of a POSIX environment with AUTOSAR basic software is uncomplicated, provided that an AUTOSAR OS is available as a runtime environment. For basic software modules which only require a call of their “main function,” however, it is sufficient to accordingly set up a pThread of the POSIX OS. On the other hand, some components such as the RTE are more tightly coupled to the AUTOSAR operating system, which is challenging to replace.

For this purpose, MICROSAR offers the “Guest OS” operating system variant, which can be integrated as a process under Linux, which gives the basic software modules an AUTOSAR runtime environment. All mechanisms are supported, from alarms to schedule tables. However, real-time behavior is limited, because the operating system can only act within the time slot provided by Linux.

When Green Hills INTEGRITY is used, it is not necessary to use MICROSAR Guest OS, because Green Hills already offers an interface to the AUTOSAR OS.

27.2.1.2 Driver Interfaces

For Ethernet, Linux offers native drivers and a TCP/IP stack. They are sufficient for pure data transmission and a diagnostic connection (DoIP). Here, the MICROSAR protocol stack relies on BSD sockets. If quality of service qualification (QoS) or time synchronization (TSyn) is also needed, it is necessary to implement a low-level connection nearer to the hardware components of the kernel.

Also available is a SocketCAN driver for CAN communications over Linux.

Under Green Hills INTEGRITY, MICROSAR drivers are used which are adapted to the interfaces of the operating system kernel.

27.2.2 Distributed Diagnostics

MICROSAR offers OEM-specific diagnostic modules and workflows for all automotive OEMs. With the MICROSAR POSIX solution, these modules can also be used under POSIX operating systems.
In ECUs with a POSIX-based application, an architecture consisting of two controllers is often used. Both supply data and services for diagnostics, but the tester only sees the ECU as a whole. Usually, the POSIX controller provides the diagnostic interface to the tester and routes the diagnostic messages to the second controller or core. This second controller contains a classic AUTOSAR system including the Diagnostic Master. Both systems are expanded by adding the Diagnostic Event Synchronizer (vDes) module. The vDes collects the local diagnostic events and routes them to the DEM Master. Only qualified diagnostic events are routed to reduce inter-controller communication to the minimal necessary traffic.

![Figure 63: Distributed diagnostics in a POSIX system](image)

### 27.2.3 Inter Process Communication (IPC) for Data Exchange between Different Operating systems

In an ECU with two controllers, cores or a more complex SoC (System-on-a-Chip), on which different operating systems are running, there is no internal system mechanism for transferring data. To avoid additional costs for the use of bus systems like CAN or Ethernet, MICROsAR offers an IPC solution here. It transfers arriving data over a serial interface like SPI or UART via the CAN protocol stack. In multi-core architectures, transmission works via shared memory.

The advantage of taking the approach of imitating a CAN transmission is that existing configuration tools and methods can be used. This simplifies the routing of existing PDUs enormously.

Prerequisites to the IPC solution are an AUTOSAR runtime environment and a CAN protocol stack. These modules are supplemented by the IPC-specific interface and the necessary hardware drivers.

### 27.2.4 Flash Bootloader for POSIX operating systems

The proven Vector Flash Bootloader for AUTOSAR-conformant ECUs is also available with suitable extensions for POSIX-compatible operating systems like Linux. The communication with the flash tool is realized via Ethernet per ISO 13400-2 (DoIP) in accordance with OEM-specific download specifications.

The Flash Bootloader is based on the Linux runtime environment, and besides offering an address-based software download, it also offers a file-based software download. This makes the process of updating software – including individual software sections – very efficient during development, in production and in vehicle service. Supplemental security options which are available effectively protect ECUs with sensitive vehicle data.

The Vector Flash Bootloader is available for all major automotive OEMs and embedded Linux systems in the automotive industry. If necessary, the basic software can be used for other POSIX-compatible operating systems.

For more details on the Vector Flash Bootloader, please refer to the separate product information sheet.

### 27.3 Configuration

For convenient configuration, we recommend our DaVinci Configurator Pro. For more details, please refer to the separate product information sheet.
28 MICROSAR.OTA – Basic Software Modules for Software Downloads

The MICROSAR.OTA bundle contains BSW modules for processing, saving and activating software updates for your vehicle. OTA stands for “Over the Air”. With our flexibly configurable modules, various strategies can be implemented for updating the software on an ECU. The software download, and subsequent update can be validated with the assistance of modules from MICROSAR.CRYPTO (for a description see MICROSAR Security).

---

**Figure 64: The MICROSAR.OTA Module**

### 28.1 Overview of the Benefits of MICROSAR.OTA

- **Available for AUTOSAR 4.x**
- **Efficient and configurable management of update-capable software areas in non-volatile memory**
- **Efficient data accesses**
- **Optimized interplay of MICROSAR.MEM and MICROSAR.OTA with parallel usage of the ECU’s internal memory**
- **Efficient switchover between the old and new software by exploiting hardware support (if available)**
- **User can select whether internal or external flash should be supported**
- **Integrated solution with the Vector Flash Bootloader (CANfbl) solution, which updates the software at the next restart**

### 28.2 Application Areas

MICROSAR.OTA contains services for initializing, writing and activating software updates for flash memories. The offered services do not act directly on the physical memory addresses of the available (internal and/or external) memory devices, but instead on software regions which abstract physical conditions and simplify the management of software updates considerably.

OEM solutions for OTA updates generally differ in their packet formats and the protocols used to control the updates. MICROSAR.OTA offers OEM extensions which implement OEM-specific requirements.
28.3 Vector Modules as Extension of the AUTOSAR Standard

The BSW modules of MICROSAR.OTA are not specified in AUTOSAR.

- **vOtaDi – Vector OTA Download**
  
  vOtaDi contains all hardware-independent parts of the software download. Primarily, these are services which are provided for initializing, writing (transferring) and activating software updates. The following functionalities have been implemented to support these services:
  
  - Access control during simultaneously use of flash memory by multiple users (MICROSAR.MEM and MICROSAR.OTA)
  - Buffered data handling and routing to the memory driver
  - Data compression and data validation
  - State management
  - Configurable memory abstraction of physical addresses to virtual addresses or software regions

  For OEM-specific solutions, vOtaDi offers additional functions.

28.4 Configuration

For convenient configuration, we recommend our DaVinci Configurator Pro. It contains functions that make your work easier such as optimization tools, visual display of flash utilization, etc. For more details, please refer to product information of DaVinci Configurator Pro.

28.5 Other Relevant MICROSAR Products for Your Memory Stack

MICROSAR OTA requires other modules from the following bundles:

- **MICROSAR MCAL:vMem**
  
  When the internal flash memory is used, an OTA flash driver (vMem) is needed, which is suitable for your hardware. vMem provides access to memory devices such as flash. In contrast to the AUTOSAR Fls, vMem allows access to data- and programm flash, depending on the device and driver capabilities. vMem is intended to be used with MICROSAR upper layer modules that support the streamlined vMem API.

- **MICROSAR.EXT:vMem Ext**
  
  When an external flash memory is used, an external flash driver (vMem Ext) from MICROSAR.EXT is needed, which is suitable for your hardware.

- **MICROSAR.CRYPTO**
  
  For other functionalities such as checksum computation and cryptographic algorithms, the MICROSAR.LIBS and MICROSAR.CRYPTO bundles are relevant.

- **Vector Flash Bootloader CANfbI**
  
  The Vector Flash Bootloader with the OTA Manager add-on is needed to make the switch between old and new software levels.

When just the internal flash memory is used, it should generally be considered that the flash memory offers memory partitioning, in which reading can be performed in one partition simultaneous to writing in another partition (Read-While-Write). This makes it possible to write software updates to the flash memory in background while the application continues to run on the ECU without any undesirable effects.


29 MICROSTAR.IPC – AUTOSAR Basic Software Modules for Interprocessor Communications

In complex ECU projects that combine several microcontrollers and/or several operating systems, the question of how to communicate between these different systems usually arises.

If field buses typical in the automotive area such as CAN or Ethernet are available and have the necessary bandwidth, communication can be performed over them.

If these conditions are not fulfilled, alternative transmission media are possible. They are generally project-specific and range from shared memory (RAM) over SPI to UART and other serial interfaces.

Besides the question of how to implement pure transmission of the data over a suitable medium, there is also the question of how the data can be exchanged between the communications stack and the application software in the relevant communication end nodes.

The MICROSTAR.IPC cluster was developed precisely for such cases. It offers all software modules needed for communication between two or more processors or cores of a microcontroller. MICROSTAR.IPC is designed flexibly and can support nearly any transmission media.

In addition, MICROSTAR.IPC can be optimally integrated into typical systems such as AUTOSAR (Adaptive and Classic) and POSIX. MICROSTAR.IPC offers both the relevant interface components and tool support for description of the communication on the signal PDU and service levels.

29.1 The Benefits of MICROSTAR.IPC

- Communication end node for different systems
  - AUTOSAR (Classic and Adaptive)
  - Linux
  - Other POSIX operating systems
  - Support of different sorts of communication

Vector Standard Software

V2X

ICP

xipc

xipcAd Post

xipcMem/
xipcMemI

xipcPam

Complex Driver

Different variants available
29.2 Application Areas

MICROSAR.IPC enables communication between the processes of two operating system instances. They may run on different microcontrollers or cores of a System on Chip (SoC). This enables applications such as the following:

- Providing data from internal vehicle bus systems on non-AUTOSAR microcontrollers:
  - Controller A: AUTOSAR communication with low latency time and short start time
  - Controller B: Network Attached Device (NAD) for interfacing to a backbone or cloud
- Mixed AUTOSAR ECUs with AUTOSAR Classic and Adaptive applications
- Diagnostics and calibration of multi-microcontroller ECUs in which just one of the controllers can be accessed via the internal vehicle bus systems.

29.3 Signal-Based Communication

The data is prepared by the basic software and is provided in its logical representation in the application or it is received by the application. Typical examples are the signals that the AUTOSAR Interaction Layer provides. In this case, the basic software handles the conversion between logical representation and representation on the transmission medium.
29.4 Service Oriented Communication

Events and methods are transformed and serialized according to the SOME/IP standard. The transformation is performed specific to the end node, either by the basic software or the application.

29.5 Transmission of Raw Data

The data is interpreted and conditioned directly by the application. The basic software and MICROSR.IPC only transmit non-transparent data blocks. An interpretation or preprocessing, such as the conversion between logical representation and representation on the transmission medium, is performed by the application.

29.6 System Service Communication

Serialized calls of system services are transmitted via vlpc. They are decoded and executed at the receiving end node. Typical examples of this are functions related to vehicle diagnostics, but they also include ECU calibration via XCP.

i. PDU Based Routing

For systems whose field buses have fixed assignments to different microcontrollers, PDUs must often be routed between the field buses. MICROSR.IPC handles the transport of PDUs between the source and target end nodes. The PDUs are received by the basic software directly, or they are fed into the basic software at the receiver side.
29.7 Signal-based routing

This use case is comparable to PDU-based routing. However, in signal-based routing the individual signals are packed in PDUs and routed. This makes it possible to customize which signals are transmitted, i.e. to transmit only those which are relevant to the receiver. Furthermore, it is possible to adjust the transmission time point and the sending frequency of the signals. For example, a signal received over a field bus can be routed to the vlpc target node at a reduced cycle time.

29.8 Functions

MICROSAR.IPC offers the following basic functions:

- Provides multiple virtual channels (PDUs)
- Simultaneous operation of multiple physical channels
- Multiplexing of multiple virtual channels on one physical channel
- Prioritization of the virtual channels
- Segmenting of the data for
  - Improved fairness in transmission on multiplex channels
  - Support of transmission media with limited packet size
- Modular architecture

29.9 Modules and Their Add-Ons

Extensions to the AUTOSAR Standard

All interprocessor communication modules are not included in the AUTOSAR standard and represent extensions of Vector.

- vlpc – Vector Inter Processor Communication
  vlpc is the key component of the IPC solution. It offers:
  - Configurable segmenting of the data of virtual channels
  - Data prioritization of the virtual channels
  - Multiplexer for multiple virtual channels on one physical channel

- vlpcIf – Vector Inter Processor Communication Interface
  vlpcIf is the translation module between the static interfaces of vlpc and the driver interfaces that are operating system and bus system dependent.
vlpcIvf offers interfaces for shared memory (vlpcMem) and SPI (vlpcSpi). Both are available in three variants, CP – Classic Platform, Linux and PikeOs.

> vlpc – Vector Inter Processor Communication

vlpcDrv is the operating system dependent driver for the bus system being used. It may be a standard driver which is provided by the basic software or the operating system, or it may be an in-house development for the specific use case.

vlpc offers drivers for shared memory (vlpcMem) and SPI (vlpcSpi). Both are available in three variants, CP – Classic Platform, Linux and PikeOs.

> vlpcSpi – Vector Inter Processor Communication (SPI)

available for Linux

> vlpcSpilf – Vector Inter Processor Communication (SPI)

available for CP – Classic Platform and Linux

> vDma – Vector Inter Processor Communication (SPI)

available for CP – Classic Platform and Linux

> vlpcAd POSIX – Vector Inter Processor Communication Adaptor POSIX

Translation module between the vlpc interfaces and the network sockets of POSIX as well as AUTOSAR Adaptive systems.

29.10 Configuration

For convenient configuration, we recommend our DaVinci Configurator Pro. You will find more details in the product information sheet on the DaVinci Configurator Pro.

Service-oriented communication is configured via the AUTOSAR system descriptions. They can be generated with our system modeling tool PREEvision, for instance.

29.11 Other Relevant MICROSAR Products for IPC

On end nodes with AUTOSAR basic software, MICROSAR.COM optimally supplements with its COM and SOMEIPXF MICROSAR.IPC modules. The COM module enables signal-based communication via MICROSAR.IPC. The COM module takes over the transformation between logical representation and the representation of the transmission medium.

Similarly, SOMEIPXF enables service-oriented communication. SOMEIPXF provides events and methods over defined application software interfaces, and it handles serialization and de-serialization according to the SOME/IP protocol.
MICROSAR.SIP and MICROSAR.EIP – A Quick Start to Your AUTOSAR Project

The Software Integration Package (SIP) and the Extended Integration Package (EIP) from Vector give you a decisive advantage in developing your ECU software: we test your software package before delivery, and you can put the entire package into operation within just a few days.

30.1 Software Integration Package (SIP)

MICROSAR.SIP is a standard delivery item, and it focuses on the broadest possible range of use for your stack. It optimizes the usability of our delivery to you, even when constraints are altered slightly. Our portfolio in brief:

- Check the OEM-specific aspects of the BSW, including SWCs and the associated tool chain (e.g. support of OEM data formats for communication and diagnostics) and assure conformance to OEM specifications.
- Check the project-specific combination of BSW components and features for a defined ECU use case.
- Check the package under abstract constraints: both the configuration and the initial database can still be changed over the course of your project. This even occurs very frequently in practice. Our goal is to implement your MICROSAR package so that it covers a wide range, i.e. to enable implementation of as many additional variants to the initial configuration and database as possible.
- Check the package under project-specific constraints: (microcontroller derivative, compiler/linker version and compiler/linker options). Here we look at the specific constraints of your project as closely as possible, so that the product can be integrated smoothly for you. For example, we select an evaluation board with a suitable device from the microcontroller product line you have selected, and test the software package on it. The goal here is to ensure that our delivery to you can be run on as many devices as possible from the preselected processor line.
- Offer management specific to delivery of the ordered MICROSAR modules in the context of our configuration management. The option for after-sale redeliveries is made sure for a time period of 10 years.
- Actively issue reporting at regular intervals.

30.1.1 Application Area of the SIP

MICROSAR.SIP is a fixed component of every MICROSAR delivery, regardless of whether it is a prototype, beta, update or production delivery. The performance range is adapted to the purpose of the SIP and the context in which it will be used.

Via a questionnaire, we catalog your requirements in as much detail as possible in advance of delivery. Afterwards we custom-build your Software Integration Package as individually as possible on this basis.

SIP Maintenance gives you active issue tracking beginning with the initial product delivery; it informs you of known errors and potential work-arounds. In addition, SIP Maintenance includes one SIP update delivery per calendar year, and Extended Maintenance even gives you two SIP update deliveries per year.

By offering the update SIP, we actively help you to introduce a new SIP and/or a new database into your project and to test it.
Optional extensions for the SIP

The Extension "Start Application" is an included service to the MICROSAR SIP if it is technically possible for the customer project. The SIP delivery in this case includes a Start Application based on the ECU specific input data for communication (e.g. ECU Extract) and diagnostics (e.g. ECUC). This Start Application demonstrates basic SIP functionality based on the ordered modules. Some examples are listed below:

- **Communication**: The Start Application included SWCs demonstrate the reception and transmission of one signal on runnable level. The BSW is configured in a way that the system gets initialized and communicates on all communication busses defined in the input data.

- **Diagnostics**: The Start Application included SWCs will be extended by a sample implementation of an RDBI/WDBI service on runnable level. Moreover it offers the possibility to trigger a DTC.

- **Memory**: The Start Application included SWCs will be extended by a sample implementation to store data to an NV-Block.

Prerequisite for the execution of these tasks is that the required MICROSAR modules are part of the delivery. Please see the technical MICROSAR PI for further details.

Beyond this, the following additional services are available with delivery of the basic SIP product:

**Automotive OEM-independent extensions** give you an easy way to start by having our integration team implement key tasks even before your MICROSAR Stack is delivered to you:

- Extension "Customer Hardware":
  - Puts the ECU into an operationally ready state including CPU clock, PLL and internal watchdog
  - Configures the network transceiver
  - Executes the general delivery test at Vector on real customer hardware

**Automotive OEM-dependent extensions** require close interaction between you and our team; they handle extended ECU dependencies such as:

- **OEM application**: incorporates modules specially developed by the automotive OEM that supplement the AUTOSAR basic software
- **OEM test**: executes special tests that incorporate OEM modules and generates all test reports
To find out which options are available for your automotive OEM, please refer to the OEM-specific product information sheet that is automatically provided to you with our quotation.

### 30.2 Extended Integration Package (EIP)

Building upon the MICROSAR.SIP, in conjunction with extended SIP extensions (see above), MICROSAR.EIP assists in follow-up activities after an initial delivery. It offers crucial support in achieving a quick and comprehensive startup. The goal here is to pass the first bench test at the OEM.

Vector employees handle preparations that are normally conducted on-site at the customer. We perform this service at a fixed price and in the framework of task planning that is detailed in an agreement with you. We can:

- Place a start application in your ECU with a specification-conformant configuration and using the databases (communication and diagnostics) that are relevant to the project
- Perform other project management jobs such as coordination via regular consultation meetings and project reporting
- Create release planning tailored to your needs
- Perform on-site startup together with you at the end of the EIP service package
- Execute test cases related to the BSW that are required by the OEM

The results of these prepared activities are then also part of the delivery:

- Your basic software package including configured start application
- Release notes
- Write-up of related test reports

---

**Figure 67:** The Extended Integration Package and its interaction with the SIP options
30.2.1 Application Area of the EIP

MICROSAR.EIP represents an extended service with the goal of submitting a sample; it is made available for the first bench test at the OEM very soon after delivery. Therefore, the EIP is an option that may be of special interest when you are performing your first project in the AUTOSAR field or when there are special project conditions in the context of an initial delivery:

- You are working on your first project in the AUTOSAR field and want extended support
- You are unfamiliar with a certain OEM, and so you want to take advantage of Vector’s experience
- You need to add supplemental services (e.g. OEM-specific components or 3rd-party modules)
- If test verification is necessary under precisely your constraints (ECU, configuration)

MICROSAR.EIP is offered for select OEMs. If you are interested, our service team would be glad to explain this offered service and whether, and to what extent, it might support the OEM relevant to your project.

In the EIP service, we also determine your requirements in as much detail as possible by questionnaire. These requirements then serve as a foundation for all other activities:

First, as in the case of a SIP, we put together your basic software. Then an employee of our service team configures and tests the overall package under precisely your production conditions and starts up your package at your business site.

The service package is rounded out by documentation of the installation process, generation of test reports and subsequent support by Vector.

30.3 Configuration

We recommend the DaVinci Configurator Pro product for convenient configuration of your MICRO dar delivery. For more details please see the separate product information.

30.4 Other relevant MICRO SAR Products for SIP and EIP

Our services are supplemented by a broad line-up of support options that assist you in successful project startup, project migration and project review. For details, please refer to the separate "Services" product information document that is available online: [http://vector.com/pi_embeddedservices_en](http://vector.com/pi_embeddedservices_en).
31 AUTOSAR Evaluation Bundle – The Complete Package for the Evaluation of AUTOSAR Basic Software and Tools

The AUTOSAR Evaluation Bundle is a comprehensive package of OEM independent AUTOSAR basic software (MICROSAR) and the tools DaVinci Developer and DaVinci Configurator Pro. This package lets you develop your first ECU software with AUTOSAR-conformant software architecture. You get an in-depth look into the AUTOSAR world – from the design and configuration process to implementation of the actual basic software. You can obtain OEM-specific BSW modules, e.g. for diagnostics, in our MICROSAR Prototype SIP (see end of this chapter).

31.1 Overview of advantages

- Tools and basic software in production quality according to AUTOSAR 4.x or 3.x to evaluate the Vector solution for AUTOSAR
- Enables realistic evaluation of execution time and memory requirements for your ECU project
- Available for many different microcontrollers
- Quick way to train in AUTOSAR with a detailed sample project
- Support of both AUTOSAR-conformant files and conventional description files

31.2 Application Areas

The AUTOSAR Evaluation Bundle supports both the automotive OEM – in evaluating AUTOSAR processes and methods – and suppliers in creating a first AUTOSAR-conformant ECU software. Since the tools and basic software are at a level of production maturity, you can reliably use the Vector solution to evaluate AUTOSAR with regard to:

- Efficiency of the basic software
- Integration of the tools in your development environment
- Potential uses of AUTOSAR concepts in your application area

The AUTOSAR Evaluation Bundle also provides an optimal foundation for initial developments of AUTOSAR-conformant software components (SWCs) for service providers who focus on the application level.

31.3 Functions

The AUTOSAR Evaluation Bundle contains the tools and embedded software from Vector for creating a complete set of AUTOSAR ECU software, which consists of software components (SWCs), Run-time Environment (RTE) and basic software (BSW). The DaVinci tools are tailored to AUTOSAR and simplify your work in designing complex AUTOSAR applications. As input for configuring the MICROSAR software, you would use an “ECU Extract of System Description” (AUTOSAR XML) or as an alternative a conventional network description file (DBC, FIBEX, LDF).

- The DaVinci Developer tool gives you an easy way to generate AUTOSAR-conformant ECU applications. Using the graphic editor, you can describe your AUTOSAR software components quickly and clearly and define their interfaces. The SWCs serve as a basis for the RTE configuration process, which you also perform with DaVinci Developer.

- The DaVinci Configurator Pro tool is used to configure the basic software modules and the RTE. You can use the convenient and intuitive user interface to adapt parameter values for your ECU project.

- The CANdelaStudio tool from Vector is also available. It lets you define diagnostic data for your networks and ECUs. You can export this data via standard formats, and use the data in automatic configuration of the MICROSAR diagnostic basic software.
The AUTOSAR Evaluation Bundle is available for AUTOSAR 4.x and 3.x. The included MICROSAR basic software modules efficiently and flexibly implement all functions of the related AUTOSAR Releases. They also contain many extensions that go beyond the standard.
31.4 Included BSW Packages

The following table gives you an overview of the individual MICROSAR bundles contained in the AUTOSAR Evaluation bundle. For a complete description of the individual bundles, please refer to the separate chapters in this document.

<table>
<thead>
<tr>
<th>In EVAL bundle</th>
<th>Available options</th>
</tr>
</thead>
</table>
| MICROSAR.OS    | Implementation of the “Scalability Class” SC1 is standard  
                 > SC2-SC4 are available as options, provided that they are supported by the processor  
                 > Multi-core is available as option |
| MICROSAR.COM   |                   |
| MICROSAR.CAN   | Modules from the product segment as chosen by the customer  
                 > XCP for measuring and calibrating over CAN/LIN/FR/ETH |
| MICROSAR.LIN   |                   |
| MICROSAR.FR    |                   |
| MICROSAR.ETH   |                   |
| MICROSAR.MEM   | Ea module for internal EEPROMs  
                 > Fee module for internal flash memory  
                 > Driver for driving external memory chips in MICROSAR.EXT |
| MICROSAR.SYS   | Csm (Crypto Service Manager) module for AUTOSAR 4.x  
                 > StbM (Synchronized Time-Base Manager) module for AUTOSAR 4.x  
                 > Drivers for driving external memory chips (see MICROSAR.EXT) |
| MICROSAR.DIAG  |                   |
| MICROSAR.MCAL  | I2c (driver for interfacing to external peripheral chips via the Inter-Integrated Circuit Bus “I2C”)  
                 > FlsTst, RamTst, CorTst |
| MICROSAR.IO    |                   |
| MICROSAR.RTE   |                   |

In addition, the following options can be ordered with the evaluation bundle:

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICROSAR.EXT</td>
<td>Modules for driving external chips</td>
</tr>
<tr>
<td>MICROSAR: VTT</td>
<td>Solution for virtual integration with vVIRTUALtarget</td>
</tr>
<tr>
<td>MICROSAR Safe</td>
<td>Complete solution for safety-related functional software according to ISO 26262</td>
</tr>
</tbody>
</table>

31.5 Special Functions

DaVinci Developer has an import/export interface for AUTOSAR XML files. This interface lets you exchange design and configuration data. For example, you might use it to integrate AUTOSAR software components in an ECU you have developed in a model-based approach using tools like MATLAB® Simulink®.

All MICROSAR products conform to
- “Implementation Conformance Class” ICC3 and
- “Configuration Conformance Class” CCC 2.

31.6 Additional Included Features

- Sample application in source code, and a detailed guide on its use
- AUTOSAR Training at Vector

31.7 Further Options

The AUTOSAR Evaluation Bundles CAN, LIN, IP and FlexRay may be used with one another in any combination.
Upon request, Vector can support you with extensive MICROSAR Coaching at initial startup and during integration of the MICROSAR basic software in your application. Vector can also provide coaching at your business site.

### 31.8 Available Hardware Platforms

The AUTOSAR Evaluation Bundle is available for the most commonly used 16-bit and 32-bit hardware platforms. Due to the hardware dependency of the MICROSAR.MCAL modules and the MICROSAR.OS, binding statements cannot be made without specific processor device numbers. The Vector Sales Team would be glad to provide this information.

Within an Evaluation you can save noteworthy resources by using the VC hardware from Vector. Additional modules within the MICROSAR basic software enable a quick and easy setup of your ECU project. The hardware of the series VC121-12 and VC36B-12 are supported by the following contents in the MICROSAR BSW:

- VCx SW LIB
- MCAL VCx
- vFlash (with matching template for the integrated Flash Bootloader on the VC121-12 and VC36B-12 hardware)

For details on the VC hardware please refer to the separate product information in our marketing portal at [http://vector.com/vi_universal_controller_vc_en.html](http://vector.com/vi_universal_controller_vc_en.html). Further variants of the VC hardware are in development.

**Note:** The integration and test of the Evaluation Bundle is done on an evaluation board. Vector will use Vector test databases for communication and diagnostics.

### 31.9 MICROSAR Prototype SIP

If you need software for the prototype phase of a specific OEM project that goes beyond pure evaluation purposes, we recommend using our MICROSAR Prototype SIP (Software Integration Package). Please contact us for further information.
32 Additional Information

For further information on our products and our configuration tool DaVinci Configurator Pro, please see our internet site: http://vector.com/vi_embedded_software_en.html.