| Document Title | Requirements on Operating System |
|---|---|
| **Document Owner** | AUTOSAR GbR |
| **Document Responsibility** | AUTOSAR GbR |
| **Document Version** | 2.0.2. |
| **Document Status** | Final |
| **Part of Release** | 2.1 |
| **Revision** | 0014 |

| Document Change History | | | |
|---|---|---|---|
| **Date** | **Version** | **Changed by** | **Change Description** |
| 24.01.2007 | 2.0.2. | AUTOSAR Administration | • "Advice for users" revised<br>• "Revision Information" added |
| 28.11.2006 | 2.0.1 | AUTOSAR Administration | Legal disclaimer revised |
| 29.03.2006 | 2.0.0 | AUTOSAR Administration | Minor formal changes |
| 30.06.2005 | 1.0.0 | AUTOSAR Administration | Initial Release |

Page left intentionally blank

## Disclaimer



## Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

# Table of Contents

# 1 Scope of this document

The goal of this document is to define the high-level requirements for the AUTOSAR operating system.

**Constraints**

None

# 2 How to read this document

Each requirement has its unique identifier starting with the prefix “BSW” (for “Basic Software”). For any review annotations, remarks or questions, please refer to this unique ID rather than chapter or page numbers!

## 2.1 Conventions used

In requirements, the following specific semantics are used (taken from Request for Comment RFC 2119 from the Internet Engineering Task Force IETF)

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- MUST NOT: This phrase, or the phrase „SHALL NOT“, means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective „OPTIONAL“, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, MUST be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

## 2.2 Requirement structure

Each module specific chapter contains a short functional description of the Basic Software Module. Requirements of the same kind within each chapter are grouped under the following headlines (where applicable):

Functional Requirements:
- Configuration (which elements of the module need to be configurable)
- Initialisation
- Normal Operation
- Shutdown Operation
- Fault Operation
- ...

Non-Functional Requirements:
- Timing Requirements
- Resource Usage
- Usability
- Output for other WPs (e.g. Description Templates, Tooling,...)

# 3 Acronyms and abbreviations

| *Abbreviation* | *Description* |
|---|---|
| API | Application Programming Interface |
| BSW | Basic Software Requirement |
| COM | Communications |
| ECU | Electronic Control Unit |
| HIS | Hersteller Initative Software |
| MCU | Microcontroller Unit |
| MPU | Memory Protection Unit |
| NM | Network Management |
| OIL | OSEK Implementation Language |
| OS | Operating System |
| OSEK/VDX | Offene Systeme und deren Schnittstellen für die Elektonik im Kraftfahrzeug |
| SWC | Software Component |

# 4 Requirement Specification

## 4.1 Real-Time Operating System

### 4.1.1 Functional description

The real-time operating system in an embedded automotive ECU builds the basis for the dynamic behaviour of the software. It manages the scheduling of tasks and events, the data flow between different tasks and provides features for monitoring and error handling.

However, in automotive systems the requirements on an operating system are highly domain specific. For instance, in the body, powertrain and chassis domains, the focus is on efficient scheduling of tasks and alarms, handling of shared resources and deadline monitoring. The used operating system has to be very efficient in runtime and small in memory footprint.

In multimedia and telematics applications, the feature set provided by the operating system and also the available computing resources are significantly different. Here, on top of pure task management, also complex data handling (e.g. streams, flash file systems, etc.), memory management and often even a graphical user interface are contained in the OS.

The classic domain of an automotive OS covers the core features of scheduling and synchronisation. In AUTOSAR, the additional features discussed above are outside the scope of the OS, such features are covered by the other WP4.2.2.1 work packages (e.g. SPAL). Integrating the feature sets of other OSs (e.g. QNX, VxWorks and Windows CE etc.) into a monolithic OS/communication/drivers structure is not possible under architectural constraints of AUTOSAR. Therefore, the AUTOSAR OS shall consider only the core features.

## 4.2 Core Operating System

### 4.2.1 [BSW097] Existing OSEK OS

| | |
|---|---|
| **Initiator:** | BMW |
| **Date:** | 22.01.2004 |
| **Short Description:** | Use OSEK OS |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The OS shall provide an API that is backward compatible to the API of OSEK OS. New requirements shall be integrated as an extension of functionality. |
| **Rationale:** | Guarantee migration progress |
| **Use Case:** | Existing driver software can be reused as its interface to the OS is not changed. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |

| | |
|---|---|
| **Supporting Material:** | [DOC_WP113_SAFETY] [STD_OSEK_OS] |

### 4.2.2 [BSW11001] Object Grouping

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | OS object grouping |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The OS shall provide software decomposition (Fault Containment Regions) which allows for fault isolation and fault recovery capabilities. |
| **Rationale:** | The existing specification of OSEK OS is not aware of multiple OS-applications residing on a single processor, there is therefore no facility for the containment of faults in one software component or basic software module from propagating to other software components and/or basic software modules resident on the same processor. OSEK OS has the following rules OS object manipulation:<br><br>Tasks and ISRs are the executable objects managed by the OS.<br><br>Standard resources can be manipulated by only those task/ISRs that declare this at configuration time.<br><br>Events can be set by any task or ISR. Events can only be waited on or cleared by those tasks that declare this at configuration time.<br><br>Alarms can be manipulated by any task or ISR.<br><br>Extending this general scheme to Planned Schedules (BSW098) means that Planned Schedules can be manipulated by any task or ISR.<br><br>This loose ownership of OS objects (task, ISR, alarm, event, planned schedule, resources) makes it difficult to contain certain classes of faults at runtime, for example one software component incorrectly cancelling an alarm belonging to another software component. It is therefore necessary to define the relationship between OS objects and the software components or basic software module to which they belong so that fault containment can be achieved at runtime.<br><br>The OS shall provide a higher-level abstraction to allow the user to group existing OSEK OS objects (tasks, ISRs etc.) so that objects in the group can be manipulated only by objects in the same group. Such a group is called an OS-Application.<br><br>Furthermore, defining an OS-Application allows a memory protection domain to be provided (see [BSW11005] Memory Write Access). |
| **Use Case:** | Under a failure condition the fault handling mechanism needs to stop all objects associated with a software component from executing. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

### 4.2.3 [BSW11018] Interrupt masking services

| | |
|---|---|
| **Initiator:** | SPAL |

| | |
|---|---|
| **Date:** | 08.12.2004 |
| **Short Description:** | Access to interrupt masking services before StartOS() and after ShutdownOS(). |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The OS shall provide interrupt mask functions before calling StartOS() and after a ShutdownOS() call. These functions are already defined in OSEK OS and the usage is now extended. |
| **Rationale:** | Needed by SPAL. |
| **Use Case:** | The SPAL drivers are required to manipulate the interrupt mask before, during and after normal OS operation. |
| **Dependencies:** | C initialization has to be performed before these functions can be used. |
| **Conflicts:** | -- |
| **Supporting Material:** | See meeting minutes of the joined SPAL and OS meeting December 8th2004. |

### 4.2.4 [BSW11019] Creation of Interrupt Vector Table

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 10.03.2005 |
| **Short Description:** | Creation of Interrupt Vector Table. |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The AUTOSAR OS generation tool shall create the vector table for interrupts AND shall ensure that the hardware interrupt priorities are consistent with the OS configuration. |
| **Rationale:** | Each ECU will need to have an interrupt vector table. The operating system configuration already contains details about all interrupts used by the system. The AUTOSAR OS generation tool shall be the final tool in the development process that generates the interrupt vector table. |
| **Use Case:** | Integration of other modules. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

## 4.3 Statically Defined Scheduling

In many applications it is necessary to statically define the activation of a set of tasks related to each other. This can be for guaranteeing data consistency in data-flow based designs, synchronising with a time-triggered network, guaranteeing correct run-time phasing, etc.

A time-triggered operating system is often proposed as a solution to this problem. However, time is simply an event so any event triggered OS, including OSEK OS, can implement a scheduler for statically scheduled real-time software in automotive electronic control units.

The requirements for schedules tables provide an OSEK OS object that can be manipulated in the same way as an OSEKtime dispatcher table.

### 4.3.1 [BSW098] Table based schedules

| | |
|---|---|
| **Initiator:** | BMW |
| **Date:** | 20.01.2004 |
| **Short Description:** | Table based schedules |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The Operating System shall provide statically configurable schedule tables based on time tables as an optional service. |
| **Rationale:** | Requirement of Standard Core users. Table based schedules are more efficient and easier to understand than tasks activated by OSEK alarm services.<br>Adding a table-based scheduling mechanism approach as an extension to OSEK OS provides users with the ability to construct an OSEKtime-like dispatcher table without needing to introduce the unnecessary restrictions of the stack-based scheduling policy or an additional OS specification. |
| **Use Case:** | Release a number of tasks synchronously with a statically defined inter-arrival time. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

### 4.3.2 [BSW099] Switchable schedules

| | |
|---|---|
| **Initiator:** | BMW |
| **Date:** | 20.01.2004 |
| **Short Description:** | Switchable schedules. |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The Operating System shall provide a mechanism which allows switching between different schedule tables. |
| **Rationale:** | For different application states (e.g. init, start-up, pre-start, normal operation, diagnosis, pre-sleep, shut down) different schedules are necessary. |
| **Use Case:** | ECU modes controlled by ECU State Manager |
| **Dependencies:** | BSW098 |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

### 4.3.3 [BSW11002] Synchronisation with global time

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Synchronisation with global time. |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The operating system shall provide the ability to synchronise the processing of schedule tables with a global system time base. It shall support immediate (hard) synchronization and gradually adapting (smooth) synchronization. |
| **Rationale:** | It is necessary for some distributed applications to be synchronised to a global (to the relevant applications) timebase. This type of feature is needed for users coming to the AUTOSAR OS from OSEKtime. |
| **Use Case:** | Users migrating from OSEKtime dispatcher tables can replicate the same functionality with schedule tables without needing to introduce an additional |

| | |
|---|---|
| | OS specification. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | [STD_OSEK_TTOS] |

## 4.4 Monitoring Facilities

A monitoring function detects an error at an appropriate stage in execution and not the instant that the error occurs. Consequently, any monitoring function is the trap of a failure at runtime rather than the prevention of a fault.

### 4.4.1 [BSW11003] Stack Monitoring

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Stack Monitoring |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The operating system shall be able to monitor stack usage and check for a stack overflow on a per executable object basis (task/ISR). |
| **Rationale:** | On some hardware it will not be possible to implement any sophisticated memory protection. Stack monitoring provides an alternative (but less secure) solution where some protection is deemed better then none. |
| **Use Case:** | If a system where an application could overflow its stack is implemented on hardware that cannot support true memory protection, stack monitoring is a useful alternative. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

## 4.5 Protection Facilities

The AUTOSAR concept requires multiply-sourced OS-Applications to co-exist on the same processor. To prevent unexpected interaction between these OS-Applications it is necessary to provide mechanisms that protect them from one another. There are two major use-cases:

1. For a safety-critical system, the development of a safety-case is made much easier if individual OS-Application safety cases can be integrated into an overall safety case. This is only feasible if it can be demonstrated that at least a fault in one OS-Application cannot propagate beyond its own boundary and cause a fault in another, unrelated, OS-Application.

2. Suppliers can only be expected to take responsibility (and some liability) for their software components and/or basic software modules if they can be assured that their software cannot be incorrectly blamed for a processor-wide failure.

Both of these use-cases can be satisfied by the addition of protection mechanisms to OSEK OS. The following sections outline the areas of protection:

### 4.5.1 Memory Protection

#### 4.5.1.1 [BSW11005] Memory Write Access

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Memory write protection |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The operating system shall provide the ability of partitioning OS-Applications with respect to memory and prevent an OS-application from modifying the memory of other OS-Applications. |
| **Rationale:** | Where multiple OS-Applications (of different software integrity) are resident on the same processor, their memory will be globally writable by any code. This means that the data of one OS–Application could be corrupted by another unrelated OS-Application (i.e. there is fault propagation between OS-Applications). For example a task of an OS-Application may overflow its stack, causing static data of an unrelated OS-Application to be corrupted, causing it to fail.<br>To permit reasoning about adequate independence between the functions of different integrity levels, it is essential that this is prevented at runtime.<br>Note that BSW11003 is different: It only detects fault rather than preventing a memory access error from generating a fault. |
| **Use Case:** | -- |
| **Dependencies:** | Note that satisfying this requirement implies the satisfaction of the stack monitoring requirement as a stack overflow cannot occur if the stack is bounded by memory write access control.<br>The write access protection needs appropriate hardware support. |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

#### 4.5.1.2 [BSW11006] Allow data exchange

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Allow data exchange within an OS-Application |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The operating system shall allow tasks and ISRs within an OS-Application to exchange data using direct access to shared memory. |
| **Rationale:** | It is common to exchange data using shared memory for performance reasons at runtime (e.g. using global variables). However, in AUTOSAR multiple OS-Applications will share a processor and therefore any data communication that happens through shared memory breaks the memory protection scheme.<br>Therefore, it is necessary to provide OS-Applications with the ability to share data using memory which is globally accessible to tasks and ISRs within the application but which is not accessible to other OS-Applications i.e. shared memory local to scope of an OS-Application. |
| **Use Case:** | An OS-Application implements communication and uses an ISR to handle the reception of CAN frames from the vehicle network but uses a task to process the contents of the CAN frame to reduce ISR level blocking. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | [DOC_WP112_REQ] |

#### 4.5.1.3 [BSW11007] Code Sharing

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Code sharing |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The operating system shall allow OS-Applications to execute shared code. |
| **Rationale:** | If code cannot be shared then any piece of software that is common to a number of software components/basic software modules will have to be included multiple times in a software build. This has two implications:<br>a large increase in code space<br>a problem is introduced for software maintenance as a modification to logically shared code will have to be made to every instance of the code in a build of an ECU (a single change has become multiple changes) |
| **Use Case:** | Using shared libraries. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

#### 4.5.1.4 [BSW11000] Memory read access

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 21.07.2004 |
| **Short Description:** | Memory read access |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The OS may offer support to protect the memory sections of an OS-Application against read accesses by all other OS-Applications. |
| **Rationale:** | If a task/ISR can read from any memory then it may operate on incorrect data. This could result in failures at runtime. Preventing read accesses provides a way of trapping such faults as soon as they occur.<br>A secondary issue is security. While it is not anticipated that there are any security implications between OS-Applications on the same processor, read accesses does provide protection if required |
| **Use Case:** | Security: protect secret keys; Debugging support |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

### 4.5.2 Timing Protection

#### 4.5.2.1 [BSW11008] Timing Protection

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Timing Protection |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The OS shall not allow a timing fault in any OS-Application to propagate to a different application resident on the same processor. A timing fault is defined as:<br>exceeding a specified execution time |

| | |
|---|---|
| | exceeding a specified arrival rate |
| **Rationale:** | When these parameters are specified for every task/ISR in the system it is possible to determine whether or not each task/ISR always meets its deadline.<br><br>Timing correctness on an ECU running any fixed-priority pre-emptive OS, including OSEK OS, can only be guaranteed using schedulability analysis. This uses information about the tasks and interrupts (how often they run, how long they run for, which resources they access, how long they hold them for) and then calculates that the system will meet its real-time performance deadlines.<br><br>The scope of timing protection is to ensure that an AUTOSAR system that has been shown to meet its deadlines does not violate the model used for analysis at runtime due to failures in the functional behaviour of applications (or their constituent parts).<br><br>Strict enforcement of the assumptions of the real-time performance analysis means two important things:<br><br>A timing fault is detected early, and hence can be picked up earlier in the software life cycle. For example, a faulty software component from a supplier can be rejected prior to full integration test. The costs of remedying a fault are therefore reduced.<br>A timing fault is not propagated. By detecting the fault as it occurs the effects of the fault are confined to the OS-Application where the fault occurred. Thus the problems of real-time failures induced in the wrong sub-system (or even the wrong ECU in a network) are eliminated. |
| **Use Case:** | An object in one OS-Application executing for too long, causes an object in another OS-Application to miss its deadline as a result. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

### 4.5.3 Service Protection

The OS must preserve both its own integrity and the integrity of the OS-Applications that it schedules at runtime.

#### 4.5.3.1 [BSW11009] Protection of the OS

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Protection of the OS |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The operating system shall prevent the corruption of the OS by any call of a system service. |
| **Rationale:** | If it was possible to place the OS into an unknown state, or corrupt OS data structures at runtime then this would damage every OS-Application resident on the same processor. This means that either:<br>every OS service call must have defined behaviour in all cases; or<br>the OS must not allow service calls to be made from contexts would potentially result in the OS being placed into an undefined state.<br>This increases the integrity of the OS itself. |

| | |
|---|---|
| **Use Case:** | Avoid undefined behaviour from e.g. calling services from wrong context. |
| **Dependencies:** | In case the current specification of OSEK OS allows configurations which do not protect the OS the AUTOSAR configuration has to make sure that these configurations can not be selected. |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

#### 4.5.3.2 [BSW11010] Protection of OS-Applications

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Protection of OS-Applications |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The operating system shall prevent an OS-Application modifying OS objects that are not owned by that OS-Application. |
| **Rationale:** | An OS-Application could manipulate objects in another OS-Application that cause it to behave outside the scope of its design at runtime. Protecting the integrity of OS-Applications means that one application cannot manipulate the objects owned by another OS-Application, causing potential failure in another OS-Application through OS service calls, unless expressly permitted. This increases the ability to trace faults arising from OS-Application coupling by restricting the possible sources of the fault. |
| **Use Case:** | Cancelling an alarm that activates a task in another OS-Application |
| **Dependencies:** | In the case where the current specification of OSEK OS allows configurations which do not protect OS-Applications, the AUTOSAR configuration has to make sure that these configurations can not be selected. |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

### 4.5.4 [BSW11011] Protecting the OS managed hardware

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Protecting the OS managed hardware |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The OS shall protect itself against OS-Applications attempting to modify control registers directly which are managed by the OS. |
| **Rationale:** | The OS must be protected against OS-Applications attempting (directly or indirectly) to circumvent the protection mechanisms. Typically this means that OS-Applications should be prevented from accessing the MCU status registers and memory protection registers that might be in use. |
| **Use Case:** | OS uses the processor status word for managing interrupts and the register is written by a rogue OS-Application at runtime, corrupting the internal data structures of the OS. |
| **Dependencies:** | The target hardware must support privileged/non-privileged modes and a MPU for this protection to be possible. This feature will therefore not be available on those targets that do not provide sufficient hardware support. |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

### 4.5.5 [BSW11012] Scalable Protection

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Scalable Protection |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The OS shall provide scalability for its protection features. |
| **Rationale:** | Take full advantage of the processor's hardware features: The key protection features may not be available on all hardware (e.g. some types of memory protection are not possible when the processor has no MPU), but this should not prevent users for using the other protection features that can be supported.<br>Customize to specific user's needs: Protection may only be necessary around some applications (ones where we cannot be sure of their run-time behaviour) and protection can be applied selectively based on assessment of the risk of failure. |
| **Use Case:** | Implementing an AUTOSAR compliant OS on a microcontroller without hardware memory protection.<br>Where an ECU is engineered using a process that can statically guarantee that no protection violations will occur at runtime it does not need to dedicate resources to check for violations. For example, if worst-case execution times are statically analysed then timing protection is not needed at runtime. Furthermore, because the analysis shows that the conditions that trigger execution of the code will never occur, the code is “dead code” and should be removed because of the potential safety risk it brings. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

### 4.5.6 Protection Errors

The OS must be able to identify when an error that violates the protection schemes has occurred and must also provide facilities through which action can be taken to correct the fault. However, it is not the task of the OS to define the error handling scheme.

#### 4.5.6.1 [BSW11013] Error Notification

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Protection Error Notification |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The OS shall be capable of notifying the occurrence of a protection error at runtime.<br>A protection error is any memory access violation, timing fault, unauthorised call to OS service or software trap (for example division by zero, illegal instruction). |
| **Rationale:** | If protection errors are notified at runtime this provides scope to potentially correct or handle the error according to a predefined fault handling strategy. |
| **Use Case:** | The application needs to provide some kind of runtime fault tolerance that needs to take action on the type and/or number of errors that occur to |

| | |
|---|---|
| | improve availability at runtime. |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

#### 4.5.6.2 [BSW11014] Protection Error Handling

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 07.07.2004 |
| **Short Description:** | Protection Error Handling (fault recovery) |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | In case of a protection error, the OS shall provide an action for recovery on OS-, OS-Application and task/ISR-level. The user shall be able to select the action. |
| **Rationale:** | The action taken on the occurrence of an error is a function of the failure modes of the system as a whole. For example, in some cases it will be appropriate to simply terminate the faulty task, in others this may pose more of a risk to safety than allowing it to continue to execute.<br>Therefore, the decision which action is appropriate is up to the application. |
| **Use Case:** | -- |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

## 4.6 Timer Services

Timer Services provide software timers for use in application and basic software.

The core of a timing mechanism is already provided by the counters and alarms in OSEK OS. Introducing an almost identical mechanism, in the form of Timer Services, is therefore unnecessary.

However, to provide general purpose software timing a few supplementary features need to be added to AUTOSAR OS. These are described in BSW11020 and BSW11021.

### 4.6.1 Functional Requirements

#### 4.6.1.1 [BSW11020] Standard interface for ticking counters

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 25.04.2005 |
| **Short Description:** | Standard interface for ticking counters |
| **Type:** | New. Added after discussion with WP1.1.2 about timer services. |
| **Importance:** | High |
| **Description:** | The OS shall provide a standard interface to tick a software counter. |
| **Rationale:** | OSEK OS does not define the interface between counters and alarms. This creates a problem when porting applications between different vendors' implementations. Defining this interface in AUTOSAR OS removes this |

| | |
|---|---|
| | portability problem. |
| **Use Case:** | -- |
| **Dependencies:** | -- |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

#### 4.6.1.2 [BSW11021] Support for cascading counters

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 25.04.2005 |
| **Short Description:** | Support for cascading counters |
| **Type:** | New. Added after discussion with WP1.1.2 about timer services. |
| **Importance:** | High |
| **Description:** | The OS shall provide a mechanism to cascade multiple software counters from a single hardware counter. |
| **Rationale:** | If counters with different resolutions are required it may not be possible (e.g. because of limited hardware timers) or desirable (e.g. because of interrupt interference) to use multiple hardware timer sources. In many cases a lower resolution software counter can be driven from a higher resolution counter by ticking the lower resolution counter from the higher resolution counter. |
| **Use Case:** | Drive a 1ms software counter by a 1ms timer interrupt and a 100ms counter from the 1ms counter. |
| **Dependencies:** | This requirement implies that an implementation must support more than one counter (otherwise cascading would not be possible). Specification of the lower limit on the number of counters that must be supported by an implementation is provided in the AUTOSAR OS SWS. |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

## 4.7 Time Triggered Operating System

### 4.7.1 Functional Overview

A time-triggered operating system implements a scheduler for statically scheduled real-time software in automotive electronic control units.
Further on the operating system offers all basic services for real-time applications i.e. interrupt handling, dispatching, system time and clock synchronisation, local message handling, and error detection mechanisms.
All services are hidden behind a well-defined API. The application interfaces to the OS and the communication layer only via this API.
For a particular application the operating system can be configured such that it only comprises the services required for this application. Thus the resource requirements of the operating system are as small as possible.

### 4.7.2 Functional Requirements

#### 4.7.2.1 [BSW11016] Scalability of the OS

| | |
|---|---|
| **Initiator:** | WP4.2.2.1.11 |
| **Date:** | 06.09.2004 |
| **Short Description:** | The OS implementation shall offer scalability configurable by a generation tool |
| **Type:** | New |
| **Importance:** | High |
| **Description:** | The OS implementation shall provide the following configurations :<br>Class1 : OSEK OS + Planned Schedules<br>Class2 : Class1 + Timing Protection<br>Class3 : Class1 + Memory Protection<br>Class4 : Class1+ Class2 + Class3 |
| **Rationale:** | Hardware support is required for Classes 3 and 4. Mandating this functionality would prevent AUTOSAR OS from being implemented on many commonly used microcontrollers.<br>Implementations may choose different strategies for implementation, with a corresponding increase in performance, if some features are not required. |
| **Use Case:** | -- |
| **Dependencies:** | BSW11012 |
| **Conflicts:** | -- |
| **Supporting Material:** | -- |

# 5 References

## 5.1 Deliverables of AUTOSAR

**[AUTOSAR_GLOSSARY]** Glossary,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_Glossary.pdf

**[DOC_LAYERED_ARCH]** Layered Software Architecture,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

**[DOC_VFB]** Specification of the Virtual Function Bus,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_VirtualFunctionBus.pdf

**[DOC_WP112_REQ]** General Requirements on Basic Software Modules,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_General.pdf

## 5.2 Related standards and norms

### 5.2.1 OSEK

**[STD_OSEK_OS]** OSEK/VDX Operating System, Version 2.2.2,
http://www.osek-vdx.org/mirror/
os223.pdf

**[STD_OSEK_OIL]** OSEK / VDX Implementation Language (OIL) V2.5,
OSEK Implementation Language,
http://www.osek-vdx.org/mirror/
oil25.pdf

**[STD_OSEK_TTOS]** OSEK/VDX Time-Triggered Operating System, Version 1.0,
July 24, 2001
http://www.osek-vdx.org/mirror/
ttos10.pdf

**[STD_OSEK_ORTI]** OSEK/VDX ORTI (OSEK RunTime Interface) Part A Version 2.1.1, Part B Version 2.1
http://www.osek-vdx.org/mirror/
ORTI-A-211.pdf

### 5.2.2 HIS

**[STD_HIS_PROTECTED_OS_REQ]** Requirements for Protected Applications under OSEK, Version 1, 25.09.2002
http://www.automotive-his.de/download/
HIS Protected OS.pdf

**[STD_HIS_PROTECTED_OS]** Requirements for Protected Applications under OSEK, Version 1.0, July 27, 2003.
http://www.automotive-his.de/download/
HIS_ProtectedOSEK10.pdf

### 5.2.3 Company Reports, Academic Work, etc

**[REP_DC_PROTECTED_OS]** Extensions of OSEK OS for Protected Applications, OSEK Support Project, DC058_02, Daimler-Chrysler AG