Document Title	Requirements on Mode Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	69

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R25-11

Document Change History

Date	Release	Changed by	Description
2025-11-27	R25-11	AUTOSAR Release Management	• Use case based rework of BswM requirements
2024-11-27	R24-11	AUTOSAR Release Management	• Use case based rework of EcuM requirements
2023-11-23	R23-11	AUTOSAR Release Management	• Added chapter Service Discovery Control for SWCs • Editorial Changes
2022-11-24	R22-11	AUTOSAR Release Management	• Rework of PNC related ComM and NM handling • Removed "DRAFT"-tag from [SRS_ModeMgm_09249] • Editorial Changes
2021-11-25	R21-11	AUTOSAR Release Management	• Removed Draft from SRS_ModeMgm_09266 and SRS_ModeMgm_09268 • Removed SRS_ModeMgm_09252 (BswM shall be able to directly request communication modes for the available Partial Networks) • SRS_ModeMgm_09249 (PNC gateway and coordination functionality) • Editorial Changes

△

2020-11-30	R20-11	AUTOSAR Release Management	• Concept "VNSM (Vehicle Network Stage Manager)" incorporated • Concept "EthernetWakeUpOnDataLine" incorporated • Concept "Rework of PNC related ComM and NM handling" incorporated
2019-11-28	R19-11	AUTOSAR Release Management	• No content changes • changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• EcuMFixed is obsolete
2017-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Clarification of Network Management requirements • Introduced Requirements Tracing information
2015-07-31	4.2.2	AUTOSAR Release Management	• Clarified post-build configurability of some requirements
2014-10-31	4.2.1	AUTOSAR Release Management	• Moved former SWS EcuM item describing the handling of sleep modes / shutdown targets to SRS level • Removed Defensive Behavior
2014-03-31	4.1.3	AUTOSAR Release Management	• Enhanced Traceability
2013-03-31	4.1.2	AUTOSAR Release Management	• Editorial changes • Updated RS_BSWandFeature as RS_Feature
2013-03-15	4.1.1	AUTOSAR Administration	• Introduced new requirement to implement ECU degradation and EnhancedBSWAllocation • Formal update to comply with the standard doc template • Introduced links to feature documents

▽

△

2011-12-22	4.0.3	AUTOSAR Administration	• Extension of BswM in order to implement the mode management relevant parts of the Partial Networks concept. • Extension of ComM in order to implement the communication mode management relevant parts of the Partial Networks concept.
2010-09-30	3.1.5	AUTOSAR Administration	• The new module BswM (BSW Mode Manager [1]). • The EcuM-Flex (ECU State Manager [2] Flexible) which has freely configurable states. • New functionality within the EcuM-Flex to support: Multi Core, Alarm Clocks and Defensive Behaviour of BSWMs • Extension of the WdgM (Watchdog Manager [3]) by: • SRS_ModeMgm_09173 added for clarification of ECU State Manager [2] behavior • program flow monitoring • windowed watchdogs • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• BSW09088 removed due to introduction of Bus-specific State Managers below Communication Manager [4] • BSW09130 and BSW09131 removed due to direct initialization of the communication stack by the ECU State Manager [2] • BSW09170 and BSW09171 added for alive supervision during startup, shutdown, and sleep • SRS_ModeMgm_09172 added for clarification of Communication Manager [4] behavior

▽

△

			△ • SRS_ModeMgm_09173 added for clarification of ECU State Manager [2] behavior • Rephrased multiple requirements to clarify behavior and configuration classes • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• BSW09075 removed (moved to SRS General) • New requirements SRS_ModeMgm_09168 and SRS_ModeMgm_09169 created • References to OSEK OS replaced with references to AUTOSAR OS • Formal adjustments and glossary update • Legal disclaimer revised • Release Notes added • "Advice for users" revised • "Revision Information" added
2006-05-16	2.0	AUTOSAR Administration	• Initial Release

Disclaimer

Table of Contents

1 Scope of Document

The goal of this document is to define the functional and non-functional requirements for all modules of the AUTOSAR mode management:

- ECU State Manager [2] (EcuM) which manages startup and shutdown of the ECU. This includes triggering the shutdown when all requests for run are released;
- Watchdog Manager [3] (WdgM) which collects the indications of aliveness and correct execution order from the independent applications and relays them to the hardware watchdog in a suitable way;
- Communication Manager [4] (ComM) which coordinates the communication of the independent applications.
- Basic Software Mode Manager (BswM) which organizes mode handling and mode related interaction of SW-Cs and the BSW modules.

All modules are needed if more than one independent software component resides on the ECU.

The location of these modules within the whole AUTOSAR ECU SW Architecture is defined in [REF].

2 Conventions to be used

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see [5, Standardization Template].

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see [5, Standardization Template].

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 follows.

Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the adjective "LEGALLY REQUIRED", means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase, or the phrase "MUST NOT", means that the definition is an absolute prohibition of the specification due to legal issues.
- SHALL: This phrase, or the adjective "REQUIRED", means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", means 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", means 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, SHALL 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, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides).

3 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to Mode Management that are not included in the AUTOSAR Glossary [6].

Acronym:	Description:
Active wake-up	Wake-up caused by the ECU e.g. by a sensor.
Alive indication	Indication that a supervised SW-C is alive - meaning active - provided from the SW-C itself to the Watchdog Manager [3].
Application	Applications are used synonymous to a SW-C. A SW-C may span several atomic SW-Cs. AUTOSAR provides the element of a "composition" to formally group several atomic SW-Cs of an application. A composition is used to structure the description and does not affect the generated code. Note: The atomic SW-Cs in a composition can still be mapped to different ECUs. Application Modes are therefore 'local' to an application but can be 'global' to several ECUs.
Application Mode	An Application Mode is a mode that is not controlled and standardized by the Mode Managers in the BSW. The scope of an Application Mode is a limited number of SW-Cs that belongs to a logical Application. If a mode only affects one composition, it is an Application Mode. An Application Mode may be distributed across multiple ECUs or may be local to one ECU depending on the distribution of the SW-C belonging to that application. Examples: Normal Operation, Limp Home
Application Mode Manager	An Application Mode Manager is a SW-C therefore a priori not standardized. It collects environmental data and Mode Requests via application-specific (non-standardized) interfaces from other SW-Cs. It can switch Application Modes (that are not controlled and standardized by the Mode Managers in the BSW). It can also request modes from other Mode Managers, specifically from Mode Managers in the BSW.
BSW Mode	A BSW Mode is mode, which is controlled and standardized by the Mode Manager in the BSW. A BSW Mode is always local to one ECU. Examples: EcuM Modes, ComM Modes
BSW Mode Manager [1]	The BSW Mode Manager [1] is a BSW module that collects mode requests via a standardized interface and controls functionality of other BSW modules accordingly. Examples are: LIN Schedule Table Switching, Enabling/Disabling I-PDU Groups, ...
Communication mode	Related to a physical channel or to a user, it indicates if it is possible to send/receive, only receive, neither send nor receive.
Communication request	A communication request indicates a demand for communication from a given user (e.g. a runnable requiring an operational communication stack). However, it can neither be assumed that the request will be fulfilled within a certain time nor that it will be fulfilled at all. The demander itself has to make sure that communication is indeed established, by using query functions or callbacks.
ECU state	General term to indicate the states managed by the ECU State Manager [2]. They represent a structured model that extends the power modes of the ECU with states and transitions to support necessary activities of software to enter/leave these power modes.

▽

△

Acronym:	Description:
Inoperation	An artificial word to describe the ECU when it is not operational, i.e. not running. It comprises all meanings of off, sleeping, frozen, etc. Using this definition is beneficial since it has no predefined meaning.
Low-power mode	All power modes except "on" (full power).
Mode	A mode is a certain set of states of the various state machines that are running in the vehicle that are relevant to a particular entity, e.g. a SW-C, a BSW module, an application, a whole vehicle In its lifetime, an entity changes between a set of mutually exclusive modes. These changes are triggered by environmental data, e.g. signal reception, operation invocation.
Mode Declaration Group	A Mode Declaration Group is defined in the Software Component Template and implemented by the RTE [7]. A Mode Declaration Group defines a number of mutually exclusive modes. There is no hierarchy of modes within a Mode Declaration Group. Each Mode Declaration Group describes only one aspect of the whole system. All modes of all Mode Declaration Groups on an ECU describe the abstract mode of the ECU. Similarly, all modes of all Mode Declaration Groups in a system describe the abstract mode of all ECUs in the system. (The mode is abstract because it cannot be physically accessed, it just exists conceptually). There is a standardized set of Mode Declaration Groups defined in the BSW, e.g. ComM, WdgM, EcuM. Each system designer is free to extend the number of Mode Declaration Groups and define his/her own modes in there. Only one Mode Manager is allowed to switch the mode of an instance of a Mode Declaration Group. (Limitation in VFB [8])
Mode Manager	A Mode Manager is a role that can be taken by either a BSW module (and optionally additionally by an SW-C). The Mode Manager collects requests from Mode Requesters, arbitrates them and switches the mode of its Mode Declaration Group(s) accordingly. Examples of Mode Managers are: EcuM, ComM, WdgM and BswM Note: If a SW-C Mode Manager switches modes directly, the mode arbitration in the BSW Mode Manager [1] cannot resolve conflicts. Thus, it is recommended to use multi-level mode arbitration, of SW-C mode Manager and BSW Mode Manager [1]
Mode Port	A Mode Port is port that has a Sender-Receiver Interface which contains a Mode Declaration Group. Note: This is called Mode Port in SWS RTE [7] -> To distinguish it from Mode Request Ports, we should call it Mode Switch Port
Mode Request	A Mode Request is some information communicated to a Mode Manager that requests the Mode Manager to switch to a certain mode. For Mode Managers in the BSW the interface to send a Mode Request is a standardized client-server interface defined by the corresponding Mode Manager. Examples: Client-server interface ComM_UserRequest with operation RequestComMode(...). For Application Mode Managers the interface can be any client-server or sender-receiver interface defined by the Mode Manager.

▽

△

Acronym:	Description:
Mode Request Port	A Mode Request Port is a port with the special semantics of requesting a mode from a Mode Manager. It can be a Client-Server or a Sender-Receiver Port. For Mode Managers in the BSW the Mode Request Port is standardized. Note: For EcuM, ComM and WdgM it is always a Provided Client-Server Port. For BswM it is a Required Sender-Receiver Ports.
Mode Requester	A Mode Requester is an entity that requests modes from a Mode Manager.
Mode Switch Port	Favored replacement of phrase Mode Port
Mode User	A Mode User is an entity that reacts on mode changes.
OFF state	ECU state. It denotes the unpowered ECU. Depending on the hardware design, the ECU can or cannot react to passive wake-up in this state (wakeability is not required in this state).
Passive wake-up	Wakeup initiated by another ECU and propagated (e.g. by bus or wakeup-line) to the ECU currently in focus.
PNC bit vector length	Represent the length of a PNC bit vector in bytes.
PNC bit vector	Represent the Partial Network information in a NM frame.
Port Group	A Port Group is a logical group of ports that are needed to realize the same functionality in the Application. A port can be a member of multiple Port Groups. A Port Group can be used to request all associated communication resources and to inquire their state. Thereby, the Port Group also defines a mapping between Application Ports and Communication Resources. Thus, the purpose of a port group is to have an abstraction of the required communication resources of that functionality For example, an Application contains normal-operation functionality and limp-home functionality with reduced communication requirements. Then it is useful to define two Port Groups, A and B. A contains the ports that are necessary for the normal-operation functionality and B contains the ports for the limp-home functionality. Thereby, the Application can recognize the condition when the communication resources for normal operation are unavailable but are sufficient for limp home. Port Groups are defined on SW-C composition level, which means that they may be distributed on several ECUs. The requests and indications for the Port Groups shall however only affect the portion of the Port Group that is local to an ECU. If distributed control of Port Groups is needed it can be handled on "Application Mode Management" level (above the RTE [7]) using normal communication features.
Power mode	Hardware power mode of the ECU. Typically: on (full power), off, sleep, standby, etc... The last two items can take several forms depending on hardware capabilities (reduced clock, peripheral standby, etc.).
Program flow monitoring	Technique to detect errors that cause a divergence from the valid program sequence seen during valid execution of a program.
RUN state	ECU state. The ECU is fully functional, all BSW modules are initialized and application software components are able to run.
Shutdown target	The chosen low-power state (OFF, SLEEP) for the next shutdown. If the SLEEP state supports several sleep modes, the shutdown target shall indicate the chosen sleep mode.

▽

△

Acronym:	Description:
Sleep mode	The term "mode" is related to the current availability of the ECU's capabilities. "Sleep mode" is the overall abstracted term for a variety of low-power modes that could exist at different CPU's, which have in common that they currently do not perform code-execution, but are still powered.
SLEEP state	ECU state. It is an energy saving state: no code is executed but power is still supplied, and if configured correctly, the ECU is wakeable. The SLEEP state provides a configurable set of sleep modes which typically are a trade off between power consumption and time to restart the ECU.
State/communication requestor	See User.
Supervised Entity	A supervised entity is a software entity which is included in the monitoring of the Watchdog Manager [3]. Each supervised entity has exactly one identifier. A supervised entity denotes a collection of checkpoints within a software component or basic software module. There may be zero, one or more supervised entities in a software component or basic software module.
User	Concept for requestors of the ECU State Manager [2] and of the Communication Manager [4]. A user may be a runnable entity, a SW-C/BSW or even a group of SW-Cs/BSWs, which acts as a single unit towards the ECU State Manager [2] and the Communication Manager [4].
Vehicle Mode	A Vehicle Mode is a mode that is not controlled and standardized by the Mode Managers in the BSW. The scope of a Vehicle Mode is the whole vehicle. If a mode affects multiple compositions, it is a Vehicle Mode. A Vehicle Mode is by definition distributed across the whole vehicle. Example: Transport Mode, Power Saving Mode, Ignition On, Ignition Off
Vehicle Mode Manager	A Vehicle Mode Manager is a special kind of Application Mode Manager that switches Vehicle Modes.
Wake-up event	A physical event which causes a wake-up. A CAN message or a toggling IO line can be wake-up events. Similarly, the internal SW representation, e.g. an interrupt, may also be called a wake-up event.
Wake-up reason	The wake-up event being actually the cause of the current/ last wake-up.
Wake-up source	The peripheral or ECU component which deals with wake-up events is called a wake-up source.

Table 3.1: Acronyms

Abbreviation:	Description:
BSW	Basic Software
BswM	Basic Software Mode Manager
ComM	Communication Manager [4]
DCM	Diagnostic Communication Manager [4]
DEM	Diagnostic Event Manager
EcuM	ECU State Manager [2]
FiM	Function Inhibition Manager
RE	Runnable Entity
SW-C	Software Component
WdgM	Watchdog Manager [3]

Table 3.2: Abbreviations

4 Requirements Specification

This chapter describes all requirements driving the work to define the Mode Management.

The Mode Management cluster is in charge of four Basic Software modules:

- the ECU State Manager [2], controlling the startup and shutdown phase of AUTOSAR BSW modules including startup and shutdown of the OS;
- the Communication Manager [4], in charge of the network resource management;
- the Watchdog Manager [3], responsible for triggering the watchdog based on the aliveness status and control flow status of application software.
- the Basic Software Mode Manager [1], responsible for supporting the mode handling.

In the following, requirements will be assigned to each of these modules.

4.1 ECU State Management

4.1.1 Functional Overview

The ECU State Management handles basic activities related to

- ECU startup,
- ECU shutdown,
- ECU reset (including activation of the bootloader),
- ECU sleep,
- and ECU wake-up

Figure 4.1 shows the high level state machine of the ECU State Management.

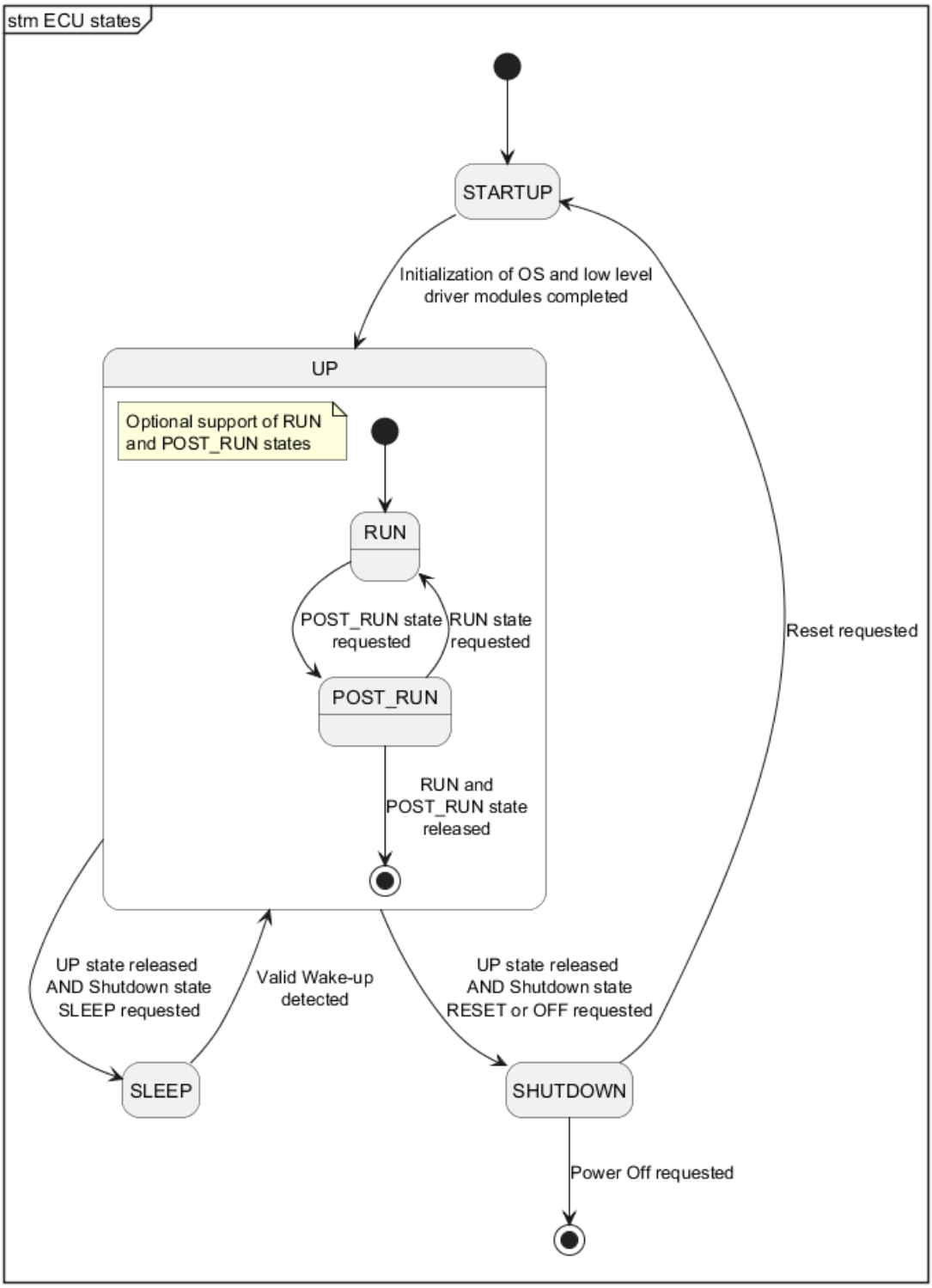

Figure 4.1: State Machine of the ECU State Management

After basic initialization of the ECU has been completed, the ECU State Management switches to the normal operation state (UP state). While in normal operation state, the ECU State Management hands over control to the BSW Mode Management (refer to chapter 4.4).

The ECU State Management will remain in that state until the BSW Mode Management indicates, that the ECU should go to sleep or should shutdown.

Applications would typically interface with the BSW Mode Management to request or release the ECU's normal operation mode, because the BSW Mode Management provides full flexibility with respect to BSW mode handling (refer to chapter 4.4).

Alternatively, applications which do not need the flexibility offered by the BSW Mode Management may interface directly with the ECU State Management to request or release the ECU's normal operation mode. For those applications the ECU State Management supports two (sub-)states "RUN" and "POST_RUN" while in the normal operation state. Specifically legacy applications make use of the standardized states RUN and POST_RUN and benefit from the simplified mode handling.

Additionally, to the above ECU state handling the ECU State Management provides the time since ECU start-up.

4.1.2 Use Cases

The following use case diagram shows the typical use cases of the ECU State Management and the actors.

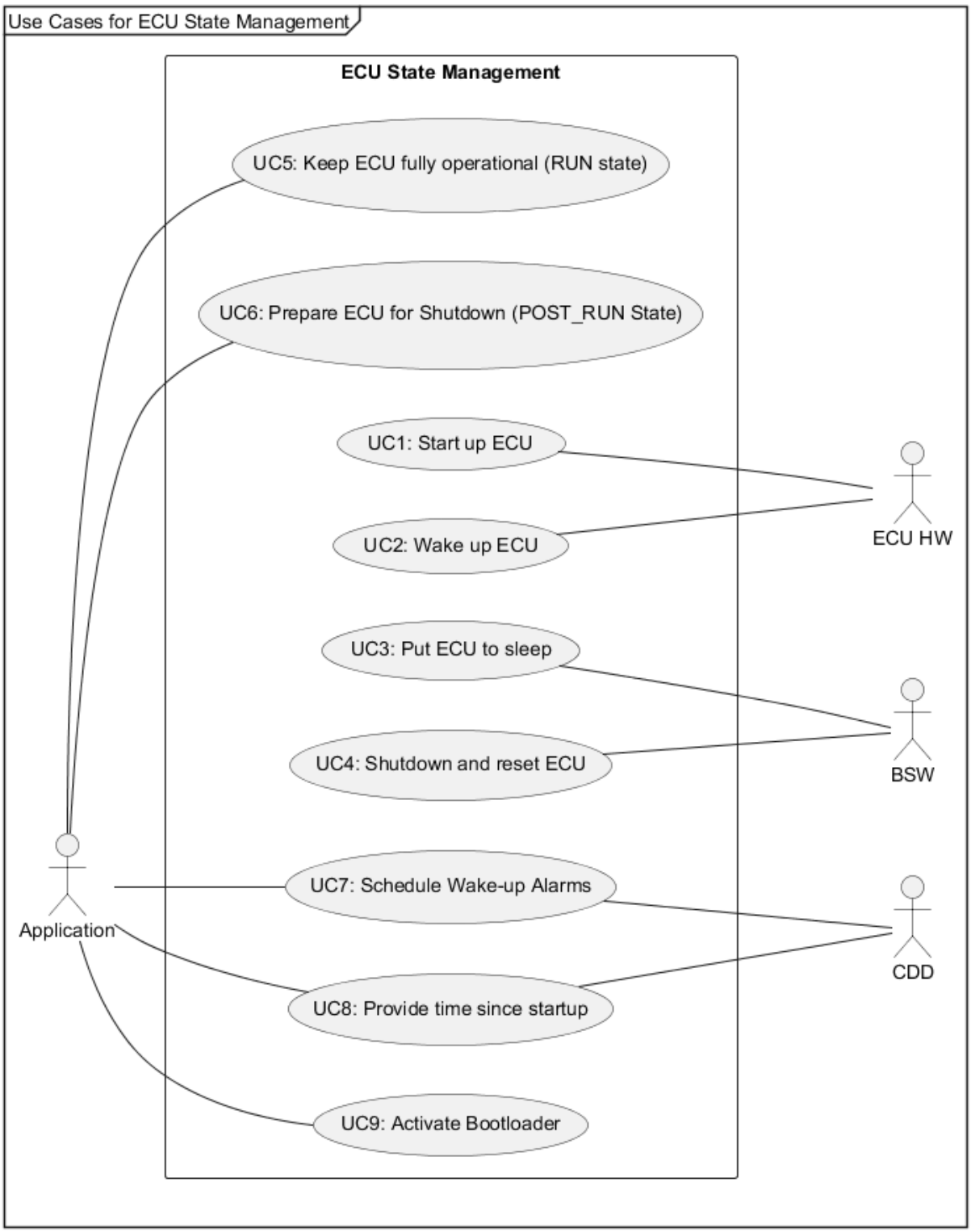

Figure 4.2: Use cases of ECU State Management

Use Case Name	Use Case Description
Start up ECU	This use case is to • initialize the ECU HW and selected BSW in a configurable order • and start the operating system • and hand-over control to BSW Mode Management for initialization of the rest of the BSW and the application.
Wake up ECU	This use case is to • validates the wake-up • and initializes selected BSW • and resumes execution of the operating system/scheduling
Put ECU to sleep	This use case suspends the operating system/scheduling and puts the ECU HW to sleep
Shutdown ECU	This use case shuts down the ECU upon request, i.e., switches off or resets the ECU
Keep ECU fully operational (RUN state)	This use case keeps the ECU alive, while application requests are pending
Prepare ECU for shutdown (POST_RUN state)	This use case prepares the ECU for shutdown, if no application request is pending to keep it alive
Activate Bootloader	This use case is to activate the bootloader upon ECU reset
Schedule Wake-up Alarms	This use case is schedule alarms and to wake up the ECU, when an alarm expires
Provide time since startup	This use case is to provide the time that has elapsed since the last ECU startup

Table 4.1: Use cases of ECU State Management

4.1.3 Functional Requirements

4.1.3.1 Use case 1: Start up ECU

[SRS_ModeMgm_00001] Initialization of Basic Software Modules ⌈

Description:	When ECU is powered up or woken up, then the ECU State Management shall • initialize the ECU HW and selected BSW in a configurable order • and start the operating system • and hand-over control to BSW Mode Management for initialization of the rest of the BSW and the application.

⌋

4.1.3.2 Use case 2: Wake up ECU

[SRS_ModeMgm_00005] Reason for Last Wake-up ⌈

Description:	When requested by the BSW, then the ECU State Management shall provide the reason/source of the latest validated wake-up.

⌋

[SRS_ModeMgm_00003] Validation of Wake-ups ⌈

Description:	When wake-up event occurs in ECU HW, then the ECU State Management shall validate the wake-up event.

⌋

[SRS_ModeMgm_00004] Handling of Valid Wake-ups ⌈

Description:	When a valid wake-up event has been successfully validated, then the ECU State Management shall re-start the BSW.

⌋

4.1.3.3 Use case 3: Put ECU to sleep

[SRS_ModeMgm_00002] Switch to ECU Sleep Mode ⌈

Description:	When supported by HW and the SLEEP state is entered, then the ECU State Management shall put the ECU to the selected sleep mode.

⌋

4.1.3.4 Use case 4: Shutdown and reset ECU

[SRS_ModeMgm_00006] Shutdown ECU ⌈

Description:	When the SHUTDOWN state is entered, then the ECU State Management shall shutdown the ECU. **Additional Information:** A shutdown of the ECU includes: • stop the operating system • and de-initialize selected BSW in a configurable order • and stop the ECU (e.g. MCU reset, watchdog reset).

⌋

[SRS_ModeMgm_00007] Synchronous Shutdown across Cores ⌈

Description:	When SHUTDOWN state is entered, then the ECU State Management shall shutdown the ECU synchronously across all cores.

⌋

[SRS_ModeMgm_00023] Reason for Last Shutdown ⌈

Description:	When requested by application or the BSW, then the ECU State Management shall provide the reason for the latest shutdown.

⌋

4.1.3.5 Use case 5: Keep ECU fully operational (RUN state)

[SRS_ModeMgm_00024] Configure States RUN and POST_RUN ⌈

Description:	The ECU State Management shall allow to enable or disable the states RUN and POST_RUN by configuration.

⌋

[SRS_ModeMgm_00008] Enter RUN State after Start-up ⌈

Description:	When the Start-up is finished, then the ECU State Management shall enter fully functional state (RUN state).

⌋

[SRS_ModeMgm_00010] Switch to RUN State ⌈

Description:	If • ECU is in shutdown synchronization state (POST_RUN state) • and a configurable user in the application requests to switch to fully functional state (RUN state), then the ECU State Management shall enter fully functional state (RUN state).

⌋

4.1.3.6 Use case 6: Prepare ECU for Shutdown (POST_RUN State)

[SRS_ModeMgm_00009] Switch to POST_RUN state ⌈

Description:	If • ECU is in fully functional state (RUN state) • and no configurable user in the application requests to remain in that state, then the ECU State Management shall enter synchronization state (POST_RUN state).

⌋

[SRS_ModeMgm_00011] Enter SHUTDOWN or SLEEP ⌈

Description:	If • ECU is in shutdown synchronization state (POST_RUN state) • and no user in the application requests to remain in that state any longer • and no user in the application requests to enter the fully functional state, then the ECU State Management shall shutdown the ECU (SHUTDOWN state) or enter SLEEP state.

⌋

[SRS_ModeMgm_00012] Select Target Shutdown State ⌈

Description:	The ECU State Management shall allow the BSW or the application to select target state after leaving the POST_RUN state (i.e., SHUTDOWN or SLEEP).

⌋

4.1.3.7 Use case 9: Activate Bootloader

[SRS_ModeMgm_00022] Activate Bootloader ⌈

Description:	When requested by application, then the ECU State Management shall switch to the Bootloader. **Additional information:** Activating the bootloader is only intended for application SW related to diagnostics (boot management).

⌋

4.1.3.8 Use case 7: Schedule Wake-up Alarms

[SRS_ModeMgm_00016] Start and Cancel Wake-up Alarm ⌈

Description:	When requested by users in the application, then the ECU State Management shall activate or de-activate a Wake-up Alarm.

⌋

[SRS_ModeMgm_00017] ECU Wake-up by Alarm ⌈

Description:	When the ECU is in sleep and a Wake-up Alarm expires, then the ECU State Management shall wake up the ECU.

⌋

[SRS_ModeMgm_00018] Cancel all Wake-Up Alarms ⌈

Description:	At ECU wake-up the ECU State Management shall cancel all Wake-up Alarm requests.

⌋

[SRS_ModeMgm_00019] Increment Alarm Clock ⌈

Description:	If ECU is powered, then the ECU State Management shall increment the Alarm Clock (every second).

⌋

4.1.3.9 Use case 8: Provide time since startup

[SRS_ModeMgm_00013] Increment ECU Clock ⌈

Description:	If ECU is powered, then ECU State Management shall increment the ECU Clock (every second).

⌋

[SRS_ModeMgm_00014] Set ECU Clock ⌈

Description:	ECU State Management shall allow dedicated users in the application to set the ECU Clock time. **Additional information:** Setting the ECU Clock time is only intended for test purposes during development.

⌋

[SRS_ModeMgm_00015] Provide Time Since Power-up ⌈

Description:	When requested by application, then ECU State Management shall provide current time since power-up of the ECU (with a resolution of 1 sec).

⌋

4.1.3.10 Common

[SRS_ModeMgm_00020] Execute External Code on State Transitions ⌈

Description:	The ECU State Management shall provide the ability to execute external, statically configured code at each transition between ECU states.

⌋

[SRS_ModeMgm_00021] State Changes are Global ⌈

Description:	When a state change is requested, then the ECU State Management shall switch to the same new ECU state globally across all cores.

⌋

4.2 Watchdog Manager

4.2.1 Functional Overview

The Watchdog Manager [3] is a basic software module of the standardized basic software architecture of AUTOSAR (service layer). It is intended to supervise the reliability of application execution with respect to timing constraints (temporal program flow monitoring) and with respect to the correct sequence of execution (logical program flow monitoring).

Derived from the layered architecture approach [9], a decoupling between application timing constraints and watchdog hardware timing constraints becomes possible. Based on this decoupling the Watchdog Manager [3] provides temporal program flow monitoring (alive-supervision) of several independent applications as well as logical program flow monitoring (the supervision of the correct execution order) while triggering the watchdog hardware.

The following features are provided by the Watchdog Manager [3]:

- Supervision of multiple individual applications located on the ECU, having independent timing constraints and requiring a special supervision of runtime behaviour and aliveness.
- Logical supervision of safety-related tasks and periodic functions (main functions).

- Fault-reaction mechanism for each independent supervised entity.
- Possibility to switch off supervision of individual applications, without violating the watchdog triggering (e.g. for inhibited applications).
- Triggering of internal or external, standard or window, watchdog, via a watchdog driver. The access to the internal or external watchdog will be handled by the Watchdog Interface.
- Selection of the watchdog mode (Off Mode, Slow Mode, Fast Mode) depending on the ECU state and the hardware capabilities.

Much more details about these functionalities can be found in the Watchdog Manager [3].

4.2.2 Functional Requirements

4.2.2.1 Initialization

[SRS_ModeMgm_09107] The Watchdog Manager shall provide an initialization service ⌈

Description:	The Watchdog Manager [3] shall provide an initialization service. This service allows the selection of one of the statically configured Watchdog Manager [3] modes.
Rationale:	Basic functionality.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09109] It shall be possible to prohibit the disabling of watchdog ⌈

Description:	It shall be possible to configure, by means of a pre-processor switch, whether the Watchdog Manager [3] initialization service and the Watchdog Manager [3] mode selection service allow the disabling of the watchdog or not.
Rationale:	Avoid the presence of code sequences in a safety relevant ECU that disable the watchdog.
Use Case:	Usage within safety relevant systems.
Dependencies:	–
Supporting Material:	–

⌋

4.2.2.2 Normal Operation

[SRS_ModeMgm_09112] The Watchdog Manager shall cyclically check the periodicity of the supervised entities ⌈

Description:	The Watchdog Manager [3] shall cyclically check the periodicity of the supervised entities, in order to detect aliveness of application.
Rationale:	This requirement is needed by AUTOSAR Software Components requiring a special supervision of runtime behavior and aliveness.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09221] The Watchdog Manager shall check the correct sequence of code execution in supervised entities ⌈

Description:	The Watchdog Manager [3] shall check the correct sequence of code execution in supervised entities.
Rationale:	To enable to detect if a program runs in an incorrect sequence.
Use Case:	SW-Cs and BSW modules call logical program flow monitoring in order to detect the deviation from the correct execution sequence.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09125] The Watchdog Manager shall provide a service allowing the Update temporal program flow monitoring ⌈

Description:	The Watchdog Manager [3] shall provide a service allowing supervised entities to forward an alive indication, thus updating the temporal program flow monitoring for the given entity.
Rationale:	This requirement is needed by AUTOSAR Software Components requiring a special supervision of runtime behavior and aliveness
Use Case:	–
Dependencies:	This could also be done via the service in SRS_ModeMgm_09222
Supporting Material:	–

⌋

[SRS_ModeMgm_09222] The Watchdog Manager shall provide a service allowing the Update logical program flow monitoring ⌈

Description:	The Watchdog Manager [3] shall provide a service allowing SW-Cs and BSW modules to indicate that they have reached a checkpoint in their code execution.
Rationale:	This requirement is needed by SW-Cs and BSW modules requiring a special supervision of execution sequence.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09160] The Watchdog Manager shall provide the indication of failed temporal monitoring ⌈

Description:	The Watchdog Manager [3] shall be able to notify the application when the monitoring of a supervised entity by temporal program flow monitoring fails. This information shall be forwarded through the RTE [7] in order to allow fault reaction by software. In this case, the information about which supervised entity has failed shall also be made available to the application.
Rationale:	This gives the opportunity to establish some fault-reactions by software before a watchdog-reset occurs (fault recovery/reporting to the Diagnostic Event Manager).
Use Case:	–
Dependencies:	[SRS_ModeMgm_09112]
Supporting Material:	–

⌋

[SRS_ModeMgm_09225] The Watchdog Manager shall provide the indication of failed logical monitoring ⌈

Description:	The Watchdog Manager [3] shall be able to notify the application when the monitoring of a supervised entity by logical program flow monitoring fails. This information shall be forwarded through the RTE [7] in order to allow fault reaction by software. In this case, the information about which supervised entity ID and which check point has failed shall also be made available to the application.
Rationale:	This gives the opportunity to establish some fault-reactions by software before a watchdog-reset occurs (fault recovery/reporting to the Diagnostic Event Manager).
Use Case:	–
Dependencies:	[SRS_ModeMgm_09221] Logical program flow monitoring

▽

△

Supporting Material:	–

⌋

[SRS_ModeMgm_09161] The Watchdog Manager shall reset the triggering condition in the Watchdog Driver in Case of temporal failure ⌈

Description:	The Watchdog Manager [3] shall reset via the Watchdog Interface the triggering condition in the Watchdog Driver (to the value 0) if the temporal monitoring of a given supervised entity fails continuously during a configurable amount of time (this amount of time can be set to 0 by configuration, thus excluding any software recovery possibility).
Rationale:	This means the monitoring failure is now considered unrecoverable. Then, a watchdog reset is the only solution.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09162] Indication of an upcoming watchdog reset.
Supporting Material:	–

⌋

[SRS_ModeMgm_09226] The Watchdog Manager shall reset reset the triggering condition in the Watchdog Driver in Case of logical program flow violation ⌈

Description:	The Watchdog Manager [3] shall reset via the Watchdog Interface the triggering condition in the Watchdog Driver (to the value 0) if the monitoring of the execution sequence fails.
Rationale:	This means the monitoring failure is now considered unrecoverable. Then, a watchdog reset is the only solution.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09162] Indication of an upcoming watchdog reset.
Supporting Material:	It shall not be possible to configure a number of times similar to SRS_Mode Mgm_09161.

⌋

[SRS_ModeMgm_09169] The Watchdog Manager shall be able to immediately reset the MCU ⌈

Description:	The Watchdog Manager [3] shall be able to immediately reset the MCU when a temporal or logical program flow monitoring failure is detected, without waiting for the hardware watchdog to expire. This feature shall be configurable. If this feature is enabled, no notification will be sent to the application (see SRS_ModeMgm_09162).

▽

△

Rationale:	When the Watchdog Manager [3] has detected an unrecoverable fault it will simply reset the triggering condition of the watchdog(s). A reset will only occur when the first watchdog expires. In some use cases it is necessary to reset the ECU as soon as the unrecoverable fault has been detected. In these cases the WdgM shall perform a reset as soon as possible. This shall be a configurable feature.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09162] Indication of an upcoming watchdog reset.
Supporting Material:	–

⌋

[SRS_ModeMgm_09162] The Watchdog Manager shall be able to notify the software of an upcoming watchdog reset ⌈

Description:	The Watchdog Manager [3] shall be able to notify the software (BSW and SW-C) when the decision to reset the triggering condition has been taken due to a temporal or logical program flow monitoring failure. This information shall be forwarded through the RTE [7] in order to allow software preparing to the imminent reset.
Rationale:	Some applications need to perform some activity before a reset.
Use Case:	Storing of persistent data in the NV-RAM.
Dependencies:	[SRS_ModeMgm_09163] Delay before provoking a watchdog reset.
Supporting Material:	–

⌋

[SRS_ModeMgm_09163] It shall be possible to configure a delay before provoking a watchdog reset ⌈

Description:	It shall be possible to configure a delay between the moment the decision has been taken to reset the triggering condition due to a temporal or logical program flow monitoring failure and the moment the Watchdog Manager [3] actually resets the triggering condition.
Rationale:	Giving the time to the whole software to prepare properly to the upcoming reset.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09162] Indication of an upcoming watchdog reset
Supporting Material:	–

⌋

[SRS_ModeMgm_09143] The Watchdog Manager shall set the triggering condition during inactive monitoring ⌈

Description:	The Watchdog Manager [3] set the triggering condition in the Watchdog Driver also when no monitoring is active/necessary.
Rationale:	Prevent unintended watchdog resets.
Use Case:	There are no supervised entities or for all supervised entities monitoring is switched off.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09231] The Watchdog Manager shall periodically set the triggering condition in the Watchdog Driver as long as the monitoring has not failed ⌈

Description:	The Watchdog Manager [3] shall periodically set the triggering condition in the Watchdog Driver (via the Watchdog Interface) as long as the monitoring of the supervised entities has not failed, thus preventing the hardware watchdog from expiring.
Rationale:	Basic functionality.
Use Case:	The Watchdog Manager [3] sets the triggering condition to a configured value. The Watchdog Driver triggers the hardware watchdog cyclically and decrements the triggering condition. If the triggering condition reaches zero, the Watchdog Driver stops triggering the hardware watchdog. If the Watchdog Manager [3] periodically sets the triggering condition, the hardware watchdog will never expire. Even if the Watchdog Manager [3] fails, it will also fail to set the triggering condition, and thus cause a watchdog reset.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09110] The watchdog Manager shall provide a service interface, to select a mode of the Watchdog Manager ⌈

Description:	The watchdog Manager shall provide a service interface, to select a mode of the Watchdog Manager [3]. Each mode corresponds to a watchdog driver mode.
Rationale:	The watchdog triggering will be provided by three modes of the watchdog driver (off, slow, fast), which will be related to the length of the time-period before the watchdog expires (no expiration, short time-period and long time-period). The decision about which mode is necessary could not be done internally by the Watchdog Manager [3]. Therefore, the Watchdog Manager [3] provides an interface to perform the watchdog driver mode selection.

▽

△

Use Case:	Appropriate modes of the watchdog are typically related to the current state of the ECU. The ECU State Manager [2] could forward a mode-selection from slow to fast, when initialization phase is finished.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09028] The Watchdog Manager shall support multiple watchdog instances ⌈

Description:	The Watchdog Manager [3] shall support multiple watchdog instances.
Rationale:	There are ECUs including both an internal and an external watchdog for monitoring the system.
Use Case:	Due to the usage of the same clock, the internal watchdog doesn't recognize the "hang-up" of a system. To achieve a higher robustness an external watchdog is used too.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09233] The Watchdog Manager shall support independent triggering condition values for each watchdog instance ⌈

Description:	The Watchdog Manager [3] shall support independent triggering condition values for each watchdog instance under its control.
Rationale:	If multiple watchdog instances exist on one platform (e.g. internal and external watchdog) it's very likely, that these watchdogs will come up with different timing constraints. Thus, the values of the triggering condition need to be adapted to the different cycles in the Watchdog Drivers.
Use Case:	HW-platform with internal and external watchdog
Dependencies:	[SRS_ModeMgm_09028] Support multiple watchdog instances
Supporting Material:	–

⌋

[SRS_ModeMgm_09232] The Watchdog Manager shall provide a service to cause a watchdog reset ⌈

Description:	The Watchdog Manager [3] shall provide a service that can be used to cause a watchdog reset independent of the monitoring states of supervised entities.
Rationale:	DCM needs to cause a watchdog reset when it switches between application and boot loader mode.

▽

△

Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

4.2.2.3 Configuration

[SRS_ModeMgm_09106] The list of entities supervised by the Watchdog Manager shall be configurable at pre-compile time ⌈

Description:	The list of entities supervised by the Watchdog Manager [3] shall be configurable at pre-compile time. The usage of additional configuration classes is not restricted
Rationale:	Application supervising.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09220] It shall be possible to configure all the transition relations ⌈

Description:	It shall be possible to configure the transition relations for each supervised entity and global transitions between supervised entities.
Rationale:	There are two possibilities to establish logical supervision of the execution sequence: • One global transition relation for the whole ECU • One transition relation for each supervised entity. Both of the possibilities shall be supported.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09158] The Watchdog Manager shall support Post build time and mode dependent selectable configuration sets for the Watchdog Manager ⌈

Description:	The Watchdog Manager [3] shall support several mode dependent selectable configuration sets on one ECU. These configuration sets shall be post build time configurable and shall include the individual timing constraints of each supervised entity. It shall also be possible to switch between these different configurations sets at runtime (e.g. depending on the current scheduling table).
Rationale:	Adapting temporal monitoring to the current schedule.
Use Case:	- Schedule table switch between STARTUP II and RUN states (see ECU State Manager [2] SWS). - Allowing a limp-home mode, having a totally different schedule.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09223] The Watchdog Manager shall support Post build time and mode dependent selectable configuration of transition relations ⌈

Description:	The Watchdog Manager [3] shall support several mode dependent selectable configuration sets on one ECU. These configuration sets shall be post build time configurable and shall include the transition relations of each supervised entity. It shall also be possible to switch between these different configurations sets at runtime (e.g. depending on the current scheduling table).
Rationale:	Adapting logical program flow monitoring to the current schedule.
Use Case:	• Schedule table switch between STARTUP II and RUN states (see ECU State Manager [2] SWS). • Allowing a limp-home mode, having a totally different schedule.
Dependencies:	–
Supporting Material:	–

⌋

4.2.3 Fault Operation

[SRS_ModeMgm_09159] The Watchdog Manager shall report failure of temporal or program flow monitoring to DEM ⌈

Description:	The Watchdog Manager [3] shall report to DEM the failure of temporal or program flow monitoring (When a failure is detected). The report to DEM shall be a configurable option of the Watchdog Manager [3].

▽

△

Rationale:	Error tracing, Diagnostics.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

,

4.3 Communication Manager

4.3.1 Functional Overview

The Communication Manager [4] collects and coordinates the communication access requests from communication requestors (users, see glossary).

The purpose of the Communication Manager [4] is:

- Simplifying the usage of the communication protocol stack for the user. This includes the starting and stopping physical channel communication and a simplified network management handling.
- Temporarily disabling sending of messages to prevent the ECU from (actively) waking up the physical channel(s).
- Controlling of more than one physical channels of an ECU by implementing a state machine for every physical channel.
- Requesting the appropriate communication state to the BusState Manager
- Simplifying the resource management by allocating all resources necessary for the requested communication mode.

In order to do so, the Communication Manager [4] defines "communication modes", indicating if a given physical channel is available for the application and how (send/ receive; only receive, neither send nor receive).

Much more details about these functionalities can be found in the Communication Manager [4].

4.3.2 Functional Requirements

4.3.2.1 Normal Operation

[SRS_ModeMgm_09078] The Communication Manager shall coordinate multiple communication requests ⌈

Description:	The Communication Manager [4] shall coordinate multiple communication requests for multiple physical channels independently. The rule for coordination is that the highest requested communication mode is the target state of the physical channel (see SRS_ModeMgm_09083 for a brief description of the communication modes).
Rationale:	Main functionality of Communication Manager [4]. A SW-C cannot know with which other SW-C it will share an ECU in a particular configuration. Coordination of communication requests has to be provided by BSW.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09246] The communication manager shall arbitrate and coordinate requests from users on physical channel and users on PNCs ⌈

Description:	The communication manager shall arbitrate and coordinate requests from users on physical channel and users on PNCs.
Rationale:	PNCs are virtual channel running on physical channels. More than one PNC can be located on a single physical channel and the communication mode of the channel has an effect on the PNCs running on it.
Use Case:	A channel is used directly by a single use and indirectly by one or more PNC users. The channel remains in Full Comm as long as the channel user remains in Full Comm, even in the case all the PNC user have released their communication.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09247] For each configured PNC an independent state machine shall be instantiated ⌈

Description:	For each configured PNC an independent state machine shall be instantiated, in order to keep track of the activation state of each PNC.
Rationale:	PNC must be able to be activated and deactivated independently of each other, in the same way physical channels are independent of each other.

▽

△

Use Case:	On a specific physical channel, three PNCs are mapped and each can have a different activation status at any given moment.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09248] it shall be possible to distinguish between internal and external PNC activation requests ⌈

Description:	When a PNC is in full communication mode, there shall be substates indicating whether the activation request is originating from the ECU local applciation or from an external PNC activator.
Rationale:	Decision whether to release a PNC or not depends on the origin of the activation request and therefore it must be taken track of it.
Use Case:	A PNC is requested from the ECU local application, as long as this is true, the PNC shall not be deactivated. In case it is required from outside, it shall only remain active as long as it is constantly requested.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_00049] The Communication Manager shall initiate the wake-up and keep awake physical channels ⌈

Description:	The Communication Manager [4] controls the Network Management modules assigned to the physical channels according to the target communication modes of these physical channels. As soon as a user requests communication, the Communication Manager [4] shall initiate the physical channel wake-up by switching the affected physical channel's NM to the AWAKE state (requesting communication for this physical channel).
Rationale:	If the Network Management has set the communication system into sleep mode the Communication Manager [4] shall wake it up again if at least one ECU/Application requires physical channel communication. Basic functionality, responsibility to initiate physical channel wake-up.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09087]
Supporting Material:	–

⌋

[SRS_ModeMgm_09080] Each physical channel shall be controlled by an independent communication mode ⌈

Description:	Each physical channel shall be controlled by an independent communication mode.
Rationale:	Possibility to have partial networks at physical channel level. Unnecessary physical channels shall not be activated.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09081] The Communication Manager shall provide an API allowing collecting communication requests ⌈

Description:	The Communication Manager [4] shall provide an API allowing collecting communication requests.
Rationale:	Means of interaction necessary
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09083] The Communication Manager shall support two communication modes for each physical channel ⌈

Description:	The Communication Manager [4] shall support Two communication modes for each physical channel. Theses modes are: • full communication (send & receive operations) • no communication (neither send nor receive operations) The modes have an implicit order with full communication being the "highest" mode and no communication the "lowest" mode.
Rationale:	Full communication mode should be self explanatory, needed for normal operation of distributed functions.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09084] The Communication Manager shall provide an API which allows application to query the current communication mode ⌈

Description:	The Communication Manager [4] shall provide an API which allows application to query the current communication mode.
Rationale:	SW-Cs shall have the possibility to query the real mode because it is unknown if and when a requested mode is reached. The querying of the requested mode is added for completeness (see SRS_ModeMgm_09149).
Use Case:	–
Dependencies:	[SRS_ModeMgm_09149]
Supporting Material:	–

⌋

[SRS_ModeMgm_09172] It shall be possible to evaluate the current communication mode ⌈

Description:	If more than one channel is linked to one user request and the modes of the channels are different, the user shall get always the lowest mode indicated.
Rationale:	One user may request more than one communication Channel
Use Case:	–
Dependencies:	[SRS_ModeMgm_09084] ,[SRS_ModeMgm_09149]
Supporting Material:	–

⌋

[SRS_ModeMgm_09149] The Communication Manager shall provide an API for querying the requested communication mode ⌈

Description:	The Communication Manager [4] shall provide an API which allows application to query the requested (target) communication mode.
Rationale:	SW-Cs shall have the possibility to query the real mode because it is unknown if and when a requested mode is reached (see SRS_ModeMgm_09084). The querying of the requested mode is added for completeness.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09084] API for querying the current communication mode
Supporting Material:	–

⌋

[SRS_ModeMgm_09085] The Communication Manager shall provide an indication of communication mode changes ⌈

Description:	The Communication Manager [4] shall provide an indication in order to notify application and DCM when the communication mode of the user has changed.
Rationale:	State changes shall be communicated to affected entities, constant polling would hinder performance.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09168] The Communication Manager shall support users that are connected to no physical channel ⌈

Description:	The Communication Manager [4] shall support users that are connected to no physical channel. If one or more SW-Cs request communication and are connected to the pseudo-channel for local communication, the channel state shall be Full communication and Run shall be requested from EcuM.
Rationale:	This requirement is necessary to support the AUTOSAR architecture. A SW-C description has to be independent of the mapping of SW-Cs to ECUs. Consequently, the service ports to the ComM have to be defined independent of the mapping of the SW-Cs connections being ECU internal or external.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09090] User-to-channel relationship.
Supporting Material:	–

⌋

[SRS_ModeMgm_09071] It shall be possible to limit communication modes independently for each physical channel ⌈

Description:	It shall be possible to limit communication modes independently for each physical channel. An API shall be provided to set the highest available mode. Limiting communication modes takes effect immediately: • If the current mode of a physical channel exceeds the selected limit, the Communication Manager [4] shall take action to set down the communication mode to such limit, as soon as possible. This mode reduction will propagate to all users linked to the affected physical channel. • Requests for modes exceeding the selected limit are not satisfied until the limitation is revoked. Communication mode ordering is described in SRS_ModeMgm_09083. Passive wake-up shall always be possible, disregarding of the limitation.

▽

△

Rationale:	This feature is mainly to be used under error conditions, in order to force communication capabilities limitations.
Use Case:	• Prevention of physical channel wake-up. • Forcing into no communication mode. • Force ECU to sleep because it is assumed that this ECU keeps without reason, the physical channel awake or floods it with junk messages.
Dependencies:	[SRS_ModeMgm_09083],[SRS_ModeMgm_09157]
Supporting Material:	–

⌋

[SRS_ModeMgm_09157] It shall be possible to revoke a communication mode limitation, independently for each physical channel ⌈

Description:	It shall be possible to revoke a communication mode limitation, independently for each physical channel. An API shall be provided.
Rationale:	This feature is necessary to cancel the effects of the mode limitation service described in SRS_ModeMgm_09071.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09083],[SRS_ModeMgm_09071]
Supporting Material:	–

⌋

[SRS_ModeMgm_09087] The Minimum duration of communication request after wakeup shall be configurable ⌈

Description:	The Communication Manager [4] shall maintain the state Full com of a communication channel after communication request The minimum duration of the FullCom shall be configurable at pre build Time for each channel
Rationale:	Safeguard against possible irregularities caused by short time span with no communication request e.g. due to delayed functions.
Use Case:	• Passive wake-up: keeping the communication stack awake until the application is able to forward the first user request. • Active wake-up: keeping the communication stack awake for a configured time after application has NoCom requested, to prevent toggling between COMM_NO_COMMUNICATION and COMM_FULL_COMMUNICATION.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09089] The Communication Manager shall be able to prevent waking up physical channels ⌈

Description:	The Communication Manager [4] shall be able to prevent the host ECU from waking up physical channels. An API to set the wake-up prevention shall be provided.
Rationale:	Prevent constant wake-up from an ECU which is triggered by an error.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09141]
Supporting Material:	–

⌋

[SRS_ModeMgm_09155] The Communication Manager shall provide a counter for inhibited communication requests ⌈

Description:	The Communication Manager [4] shall provide a counter for all rejected "Full Communication" mode requests, due to communication mode limitations. The counter shall be stored in non-volatile memory.
Rationale:	The counter is used for detecting latent software problems relating to unmotivated communication bus wake-up.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09071],[SRS_ModeMgm_09089]
Supporting Material:	–

⌋

[SRS_ModeMgm_09156] It shall be provided an API to retrieve the number of inhibited "Full Communication" mode requests ⌈

Description:	It shall be possible to read out (and reset) the number of "Full Communication" mode requests that were inhibited due to communication mode limitations. This value shall be accessible via a Communication Manager [4] API.
Rationale:	–
Use Case:	It shall be possible to read out and reset the current status of the counter by a diagnostic service.
Dependencies:	[SRS_ModeMgm_09155] Counting of inhibited communication requests.
Supporting Material:	–

⌋

[SRS_ModeMgm_09249] PNC gateway and coordination functionality ⌈

Description:	Communication Manager [4] shall be aware of the distribution of each PNC over different ComM channels (where each channel represents a particular connection to a bus (e.g. CAN, FlexRay) or network (e.g. switched Ethernet network)) and shall take care to forward the activation requests from one channel to the other. The Communication Manager of a PNC gateway can either act per PNC as top-level PNC coordinator or as intermediate PNC coordinator. The top-level PNC coordinator is responsible for the coordinated shutdown of all PNC members, because it is the only node which has the full overview of the states of all PNCs. If the top-level PNC coordinator detects, that a PNC request has been released, then a NM message is transmitted where this PNC is indicated as released. The NM message is forwarded as usual NM message by the intermediate PNC coordinate to the PNC leaf nodes. If the optional feature "synchronized PNC shutdown" is enabled the top-level PNC coordinator transmits a NM message as PN shutdown message. The intermediate PNC coordinator is responsible to forward a PN shutdown message (request for a synchronized PNC shutdown) per channel has fast as possible.
Rationale:	A PNC can span over different buses which are not necessarily connected to each involved ECU. AUTOSAR supports Partial Network topologies, where different PNCs are coordinated by different ECUs in the role of a top-level PNC coordinator. Please note, a PNC can have only one top-level coordinator across a Partial Network. This has to be ensured by a proper Partial Network communication design.
Use Case:	ECU A is connected to network 1 and 2, ECU B is connected to network 2 and 3, ECU C is connected to network 3; as PNC 1 involves SWCs on all these ECUs, the communication manager on each ECU must know where and how to forward the activation requests. ECU A is acting as top-level PNC coordinator for PNC1. If all ECUs have released PNC1 and optional feature "synchronized PNC shutdown" is enabled, then ECU A transmit a PN shutdown message on network 1 and 2 to start a synchronized PNC shutdown of PNC1. ECU B is acting as intermediate PNC coordinator and forwards the request for a synchronized PNC shutdown of PNC1 to network 3
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09250] PNC activation requests shall be exchanged with the Network Management via a PNC bit vector ⌈

Description:	Communication Manager [4] shall provide callback functions get external PNC activation requests as PNC bit vector and call functions of the Network Management in order to forward PNC activation requests as PNC bit vector according to the specified bit codes. The content of the PNC bit vector is provided/required by the Network Management.

▽

△

Rationale:	Internal activation requests are encoded as PNC bit vector and exchanged between Communication Management and Network Management via dedicated APIs. External activation requests are received by the Network Management as encoded PNC bit vector and forwarded as PNC bit vector to Communication Manager. The Communication Management and Network Management decode the PNC bit vector for internal processing.
Use Case:	ECU A is connected to CAN A and CAN B, ECU B is connected to CAN B and Flexray A, ECU C is connected to Flexray A; as PNC 1 involves SWCs on all these ECUs, the communication manager on each ECU must know where and how to forward the activation requests.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09278] The Communication Manager shall support synchronous and asynchronous request upon a indicated wakeup ⌈

Description:	ComM shall support to request its configured ComMChannels and / or PNC in a synchronous or asynchronous manner upon an indicated wakeup.
Rationale:	In dependency of the expected behavior upon a indicated wakeup, ComM may need to request all its configured ComMChannels and / or PNCs (synchronous wakeup) or only the given ComChannel or PNC (asynchronous wakeup).
Use Case:	In some cases the network communication design may expect to start all configured PNCs and the affected TX paths to provide any data as soon as possible on the network. In some cases the network communication design may expect to start the affected channels and postpone the startup of the PNCs until a NM message is received which contain PNC request. Upon the reception, the affected PNCs are started.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09279] The Communication Manager shall support a coordinated release of PNCs ⌈

Description:	ComM shall support to release its configured PNCs in a coordinated manner.
Rationale:	A released PNC should stay for a configurable amount of time in active state to avoid toggling between requested and released state. This should support the robustness of PNC handling across the PN topology.
Use Case:	Reliable and predictable release process of a PNC
Dependencies:	–

▽

△

Supporting Material:	–

⌋

[SRS_ModeMgm_09251] PNC communication state shall be forwarded to the BswM ⌈

Description:	BswM shall be notified of each change in the communication state of each configured PNC, meaning with this communication mode and relative substate.
Rationale:	This is needed in order to let BswM operate the necessary actions to allow an according operation mode change in the other BSM modules involved.
Use Case:	PNC A is deactivated, I-PDU group A, which contains the I-PDUs corresponding to this PNC, is disabled by a corresponding action in the BswM configuration.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09256] PNC Gateway Functionality shall consider systems with more than one gateways connected to the same network ⌈

Description:	If there are multiple PNC Gateways within the system, then mirroring of cluster-requests need to be limited to avoid keeping awake each other (forwarding of cluster-requests is unaffected). For this, the following distinction shall be possible: Channel Connector of PNC Gateway is of type "active", then PNC requests are forwarded and mirrored on the channel Channel Connector of PNC Gateway is of type "passive", then PNC requests are only forwarded, but not mirrored on the channel Channel Connector of PNC Gateway is of type "none", then PNC requests are not forwarded and not mirrored back on the channel Constraint: If there are multiple PNC Gateways within the system, there shall always be one Gateway on top of the PNC Gateways hierarchical topology.
Rationale:	–
Use Case:	The concept shall be designed to avoid every deadlock situation where the system does not go to sleep (for instance by mirroring back PNC requests).
Dependencies:	
Supporting Material:	–

⌋

[SRS_ModeMgm_09257] ComM shall forward PNC-Clusters also to busses that are currently not awake ⌈

Description:	When PNCs need to be forwarded to busses that are currently not awake the ComM shall wakeup the according busses and forward the PNCs.
Rationale:	–
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09258] Optional Dynamic Extension of PNC Gateway ⌈

Description:	The PNC Gateway shall be extended to support a dynamic PNC Learning mechanism so that new PNC routing rules can be learned during run-time. This feature shall be configurable and only available if enabled. For dynamic Extension of PNC Gateway, it shall be possible to have PN-Clusters available that are not mapped to any network
Rationale:	Dynamic entries to PNC-to-channel-mapping are needed when cluster members (ECUs) might change, for instance by install/uninstall/relocate software components on adaptive platforms.
Use Case:	Dynamic PNC Mapping
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09259] ComM API shall provide interfaces to access PNC Mapping (optional) ⌈

Description:	The API of the ComM shall provide interfaces for the configurable PNC Mapping stored in NVRAM: • to read out the current entries (includes "statically" entries) * • to update / add entries (optional) * • to reset all entries ("statically" entries will be unaffected) * for testing reasons only
Rationale:	Debug/Test and reset learning routine to default-values
Use Case:	Dynamic PNC Mapping
Dependencies:	[SRS_ModeMgm_09258]
Supporting Material:	–

⌋

[SRS_ModeMgm_09260] ComM API shall provide an interface to start PNC Learning mechanism for PNC Mapping (optional) ⌈

Description:	The API of the ComM shall offer an interface to start the learning process, so that it can be triggered by any specific event.
Rationale:	–
Use Case:	Dynamic PNC Mapping
Dependencies:	[SRS_ModeMgm_09258], [SRS_ModeMgm_09265]
Supporting Material:	–

⌋

[SRS_ModeMgm_09261] ComM shall forward the information for Partial Networking Learning (optional) ⌈

Description:	Gateways only: When the bit with value for partial networking learning is received by a PNC-Gateway on an actively coordinated channel, it shall forward this bit to all other coordinated channels. When the bit with value for partial networking learning is received by a PNC-Gateway on an passively coordinated channel, it shall forward this bit to all other actively coordinated channels. Hint: Partial network learning bit must be sent to all nodes in the network but it must not be mirrored back (this must also consider possible circles in the network topologies)."
Rationale:	–
Use Case:	Dynamic PNC Mapping
Dependencies:	[SRS_ModeMgm_09258], [SRS_ModeMgm_09262], [SRS_ModeMgm_09265]
Supporting Material:	–

⌋

[SRS_ModeMgm_09262] ComM shall set all its assigned PNCs when partial networking learning is requested (optional) ⌈

Description:	When the bits for partial networking learning and Repeat Message Request bit are received with value set, ComM shall request internally the PNCs which it wants to communicate to the gateway and transmit them in its NM PDUs on all buses with NM until the bit is cleared. Within this response message the bit for partial networking learning shall be set, but the Repeat Message Request bit shall be cleared. Hint: Due to the activation of all its PNCs it is ensured that the PNC-membership will be provided on the bus by every ECU. Hint: Gateways can also be at the same time normal nodes providing PNC information and should therefore merge their received information with their provider information when sending NM messages.
Rationale:	–

▽

△

Use Case:	Dynamic PNC Mapping
Dependencies:	[SRS_ModeMgm_09258], [SRS_ModeMgm_09261]
Supporting Material:	Partial Network learning differs between: • request NM PDU, where the bit for partial networking learning and Repeat Message Request bit, both are set • response NM PDU, where only the bit for partial networking learning is set (but Repeat Message Request bit stays unset)

⌋

[SRS_ModeMgm_09263] ComM API shall provide an interface to set PNC-membership on Host-ECU (optional) ⌈

Description:	The API of the ComM shall offer an interface to set the PNC-membership of the Host-ECU in order to answer during PN learning phase.
Rationale:	–
Use Case:	Dynamic PNC Mapping: When the PN learning is requested, the ECU / Gateway shall be aware of all PNCs, of which itself is a member of. This might change dynamically, for example in the case of a Software Component being the only member of a specific PNC but might be de-/activated by diagnosis.
Dependencies:	[SRS_ModeMgm_09258], [SRS_ModeMgm_09262]
Supporting Material:	–

⌋

[SRS_ModeMgm_09265] ComM shall send the information for Partial Networking Learning (optional) ⌈

Description:	When the start of PN learning procedure is requested ComM shall request to send the bit for partial networking learning via the <Bus>Nms for as long as it remains in NmRepeatMessageState.
Rationale:	–
Use Case:	Dynamic PNC Mapping
Dependencies:	[SRS_ModeMgm_09258]
Supporting Material:	–

⌋

[SRS_ModeMgm_09266] ComM shall support communication channels that act as communication slaves with wake-up capability ⌈

Description:	Communication channels which act as communication slaves with wake-up capability are in relation to a communication master. The communication channels which act as communication slaves with wake-up support are able to wake-up the communication master.

▽

△

Rationale:	A communication channel that acts as an communication slave with wake-up capability could be requested locally by a user
Use Case:	LIN slaves
Dependencies:	–
Supporting Material:	ISO 17987-2:2016 Road vehicles - Local Interconnect Network (LIN) - Part 2: Transport protocol and network layer services ISO 17987-3:2016 Road vehicles - Local Interconnect Network (LIN) - Part 3: Protocol specification

⌋

[SRS_ModeMgm_09267] ComM shall support communication channels which act as communication slaves without wake-up capability ⌈

Description:	Communication channels which act as communication slaves without wake-up capability in relation to a corresponding communication master. The communication channels which act as communication slaves without wake-up support are not able to wake-up the corresponding communication master. The communication slave will just follow and react on the wake-up request of the corresponding communication master.
Rationale:	A communication channel that acts as a communication slave without wake-up capability could only be requested remotely (passive wake-up) by its corresponding communication master.
Use Case:	Ethernet communication channels which use Ethernet hardware that support of OPEN ALLIANCE Sleep/Wake-up Specification Version 2.0 (Rel Feb 21, 2017) and do NOT using network management. Only single ECU which do NOT maintain an Ethernet switch could have a communication channels which act as communication
Dependencies:	–
Supporting Material:	OPEN ALLIANCE Sleep/Wake-up Specification Version 2.0 (Rel Feb 21, 2017)

⌋

[SRS_ModeMgm_09268] ComM shall support the possibility to forward the information if the communication request is active or passive to it's lower layer layer ⌈

Description:	It is necessary to distinguish between active and passive communication requests to the lower layers for communication channels that uses hardware which have the possibility to send wake-up requests on the network.
Rationale:	Active communication requests will end up with a wake-up request on the network, while passive communication requests will end up without wake-up request on the network
Use Case:	Ethernet communication channels which comply to the OPEN ALLIANCE Sleep/Wake-up Specification Version 2.0 (Rel Feb 21, 2017)
Dependencies:	–

▽

△

Supporting Material:	OPEN ALLIANCE Sleep/Wake-up Specification Version 2.0 (Rel Feb 21, 2017)

⌋

[SRS_ModeMgm_09269] The Communication Manager shall support synchronized PNC shutdown ⌈

Description:	The ComM of a ECU in the role of a top-level PNC coordinator shall forward PNCs which are detected as released and forward the request to Nm per Com MChannel. The ComM of a ECU in the role of an intermediate PNC coordinator shall be able to react on received PN shutdown messages and forward the PNC bit vector to active coordinated ComMChannels
Rationale:	An ECU in the role of a top-level PNC coordinator has to transmit a PN shutdown message. A receiving ECU in the role of an intermediate PNC coordinator shall react immediately and forward the received PNC bit vector of a PN shutdown message as fast as possible on the affected ComMChannels.
Use Case:	Synchronized PNC shutdown of PNCs across the whole PN topology
Dependencies:	[RS_Nm_02517]
Supporting Material:	–

⌋

4.3.2.2 Configuration

[SRS_ModeMgm_09090] Relationship between users and physical channels shall be configurable at pre compile time ⌈

Description:	Relationship between users and physical channels (which user communicates on which physical channel) are configurable at pre compile time. The usage of additional configuration classes is not restricted.
Rationale:	Necessary for physical channel independency; communication shall be only activated if communication is needed on this physical channel.
Use Case:	–
Dependencies:	–
Supporting Material:	Note: A possible solution is a 2D matrix with users as rows and physical channels as columns; an entry in this matrix links the user (row) with the corresponding physical channel (column).

⌋

[SRS_ModeMgm_09133] It shall be possible to assign physical channels to the Communication Manager ⌈

Description:	Physical channels, which should be handled by the Communication Manager [4], shall be assigned to the module by pre compile time configuration. The usage of additional configuration classes is not restricted.
Rationale:	The ECU specific implementations of the Communication Manager [4] shall have knowledge about how many and which physical channels have to be treated by the Communication Manager [4] on the dedicated ECU. This knowledge is required to enhance the independency of the physical channel management within the Communication Manager [4]. Additionally, it gives flexibility to tailor required data-resources for the specific implementations.
Use Case:	The Communication Manager [4] shall cope with individual communication resources of different ECUs e.g.: ECU A : two FlexRay busses, one CAN bus ECU B : one CAN bus, one LIN bus
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09132] It shall be possible to assign Network Management to physical channels ⌈

Description:	The usage of Network Management shall be assigned to the physical channels by pre compile time configuration. This option shall be provided independently for all physical channels, which are assigned to the Communication Manager [4]. The usage of additional configuration classes is not restricted.
Rationale:	The Communication Manager [4] needs the knowledge of the Network Management usage on each physical channel in order to provide each channel with the suitable synchronization-mechanism to the Network Management (handling of additional indications and acknowledges from/to Network Management). For physical channels without Network Management, the Communication Manager [4] has to be aware of the simplified handling (no indication or acknowledges needed to/from Network Management).
Use Case:	An ECU with two physical channels (A, B): A is related to powertrain domains, B is related to comfort-components. - physical channel A does not use Network Management - physical channel B must use Network Management
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09141] The Communication Manager shall be able to configure the physical channel wake-up prevention ⌈

Description:	The Communication Manager [4] shall be able to prevent wake-up at the level of physical channels (SRS_ModeMgm_09089). This feature shall be configurable at pre compile time or post build time. The usage of additional configuration classes is not restricted.
Rationale:	This feature will not appear in all ECUs.
Use Case:	–
Dependencies:	[SRS_ModeMgm_09089] Preventing waking up physical channels.
Supporting Material:	–

⌋

[SRS_ModeMgm_09243] The Communication Manager shall be able to handle the Partial Networks on Flexray, CAN and Ethernet ⌈

Description:	The Communication Manager [4] shall be able to handle Partial Networks spanning on CAN, Flexray and Ethernet physical channels.
Rationale:	Partial Network implementation is needed at least on these bus types.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09244] The number of supported PNCs shall be configurable strictly at pre-compile time ⌈

Description:	The number of supported PNCs shall be configurable strictly at pre-compile time.
Rationale:	Letting complete freedom in configuring the number of PNCs also at Post-Build time would increase complexity in comparison to the advantages.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09245] Enabling or disabling the Partial Network Cluster management in ComM shall be post-build selectable. ⌈

Description:	The management of the PNC states shall be post-build activatable.
Rationale:	It has to be possible to disable or enable the management without having to recompile the code.

▽

△

Use Case:	Activation and deactivation via diagnostic command
Dependencies:	–
Supporting Material:	–

⌋

[SRS_ModeMgm_09207] ComM shall allow for additional bus specific state managers ⌈

Description:	ComM shall allow for additional bus specific state managers
Rationale:	Allow writing of CDDs as defined in the AUTOSAR architecture
Use Case:	Use cases for CDDs need not to be given. However, to state some current problems: • CDDs accessing MCAL (e.g. PWM, but call back routines of MCAL only call IOHWA, not a CDD) • CDDs accessing PduR (e.g. Debugging; but PduR only interfaces to Com or Dcm), CDDs accessing CanIf (e.g. OSEK NM or XCP, but there exists a parameter to only select PduR, CanTp or CanNm in CanIf) • validI/O busses besides SPI like USB etc.
Dependencies:	–
Supporting Material:	Covered feature request: • RS_BRF_00225 (Enabling CDDs in the BSW architecture)

⌋

4.4 Basic Software Mode Manager

4.4.1 Functional Overview

The VFB [8] defines and the RTE [7] implements the concept of Mode Declaration Groups.

There can be multiple Mode Declaration Groups that are independent of each other. E.g. in Figure 1 the Mode Declaration Group is "AMM_KeySwitch" and the Modes in this group are "Off", "Accessory", "Ignition".

A Mode Declaration Group can not contain parallel Modes. Parallel Modes must be implemented by separate Mode Declaration Groups. E.g. diagnostic modes are independent of key switch modes, i.e., they go into a validMode Declaration Group "AMM_Diagnostics".

There is no hierarchy of Modes within a Mode Group. E.g. "Diagnostics_On" and "Diagnostics_Off" cannot have sub-modes.

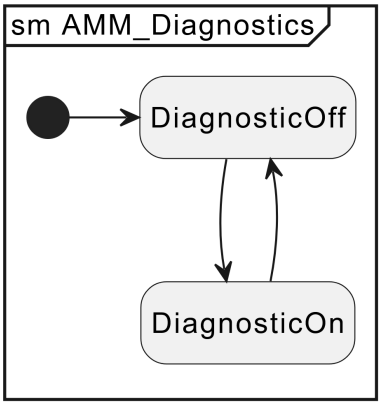

Figure 4.3: Mode Declaration Groups(1)

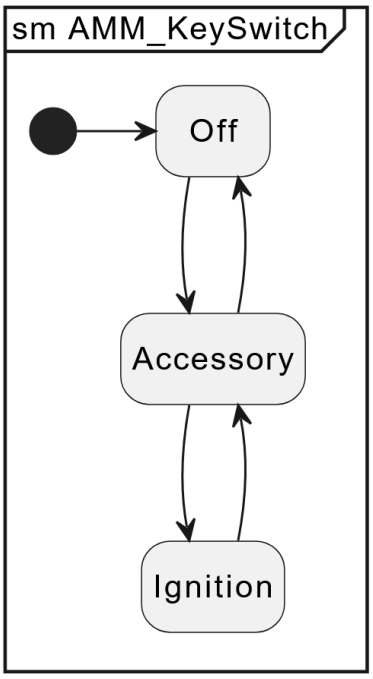

Figure 4.4: Mode Declaration Groups(2)

4.4.1.1 Interfaces between Mode -Requester, -Manager and -User

As defined above there are three roles that an SW-C or a BSW module can take: Mode Requester, Mode Manager, and Mode User.

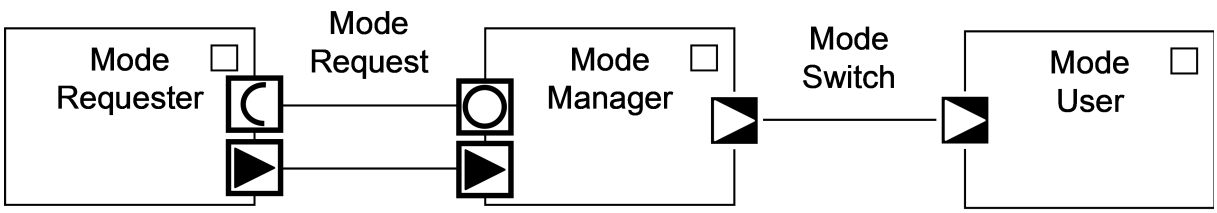

Figure 4.5: Mode Management Roles and Interfaces

The Mode Requester requests a Mode from a Mode Manager by sending some data via a port with a Mode Request interface. As shown in Figure 2, this port can be a Client or a Sender port, in case of a Mode Manager in the BSW even a C function call. This interface is not standardized.

The Mode Manager receives the incoming information via its Server or Receiver ports or C functions with Mode Request Interfaces, arbitrates the requests and decides upon a resulting mode. It uses a local Sender port with a Mode Switch Interface [1] to switch a Mode Declaration Group into the resulting mode.

To react upon mode changes, Mode Users have a local Receiver port where it receives Mode Switch Notifications. It can either read and evaluate the information directly or via its Software Component Description instruct the RTE [7] to start and stop some of its runnables.

Note that in all cases there may be multiple ports with Mode Request and Mode Switch Interfaces attached to the corresponding roles. E.g., one Mode Requester may request Modes from multiple Mode Managers (This requires the use of a sender-receiver-interface). One Mode Manager may receive requests from multiple Mode Requesters. One Mode Manager may switch multiple Mode Declaration Groups each with multiple Mode Users. And one Mode User may receive Mode Switch Notifications from multiple Mode Managers, where one mode machine instance is switched by only one mode manager.

4.4.1.2 Relation of Modes

Every system contains modes at different levels of granularity. As shown in Figure 3, there are Vehicle Modes and several Applications with modes and ECUs with local BSW Modes.

[1] A Mode Switch Interface is a special kind of Sender-Receiver Interface that is tagged as a local Service and contains a Mode Declaration Group Prototype as a data element.

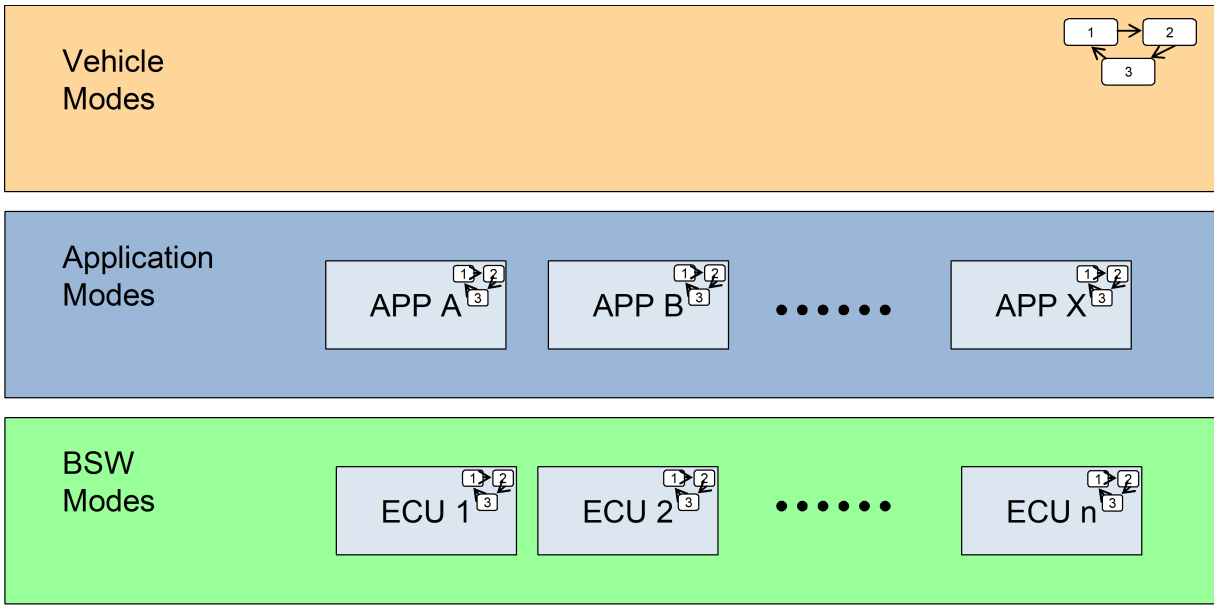

Figure 4.6: Levels of Modes

But all these modes are not really independent: In reality, Modes at all levels influence each other.

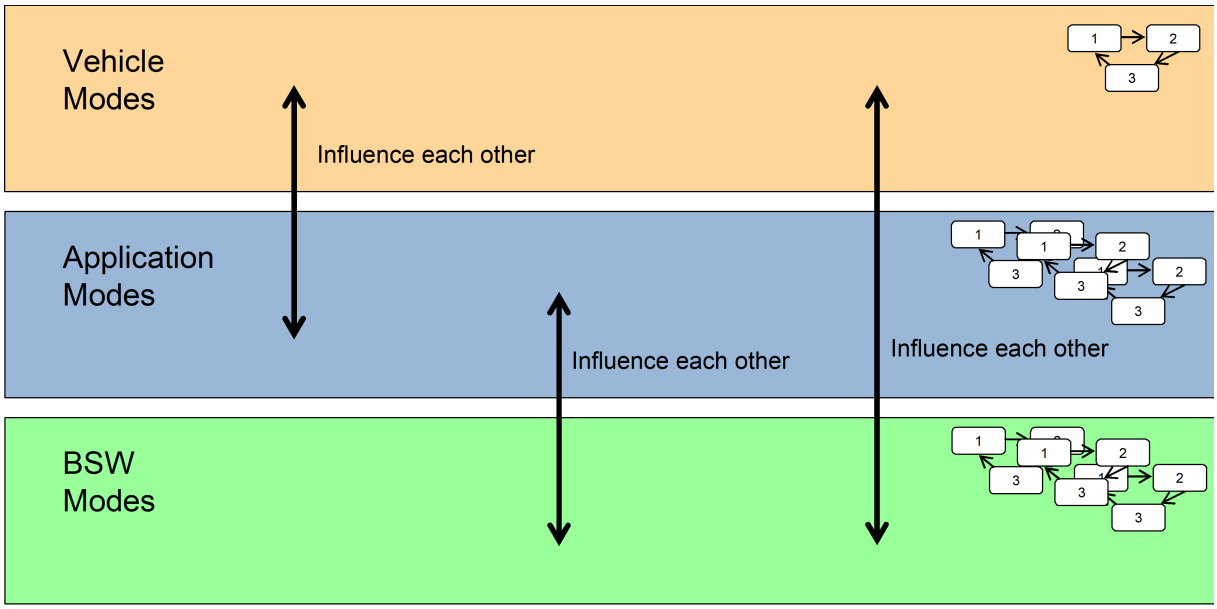

Figure 4.7: Influences between Mode Levels

Figure 4.7 shows the following relationships:

- Depending on Vehicle Modes, Applications may be active or inactive and thus be in different Application Modes.
- Vice versa, the operational state of certain Applications may cause Vehicle Mode changes.

- Depending on Vehicle and Application Modes, the BSW modes may change, e.g. the communication needs of an Application may cause a change in the BSW Mode of a communication network.
- Vice versa, BSW Modes may influence the Modes of Applications and even the whole vehicle, e.g. when a communication network is unavailable, Applications that depend on it may change into a Limp-Home Mode.

There are also cross-dependencies within all levels of Modes through mode relations. These get even more complicated because Applications can be distributed over several ECUs.

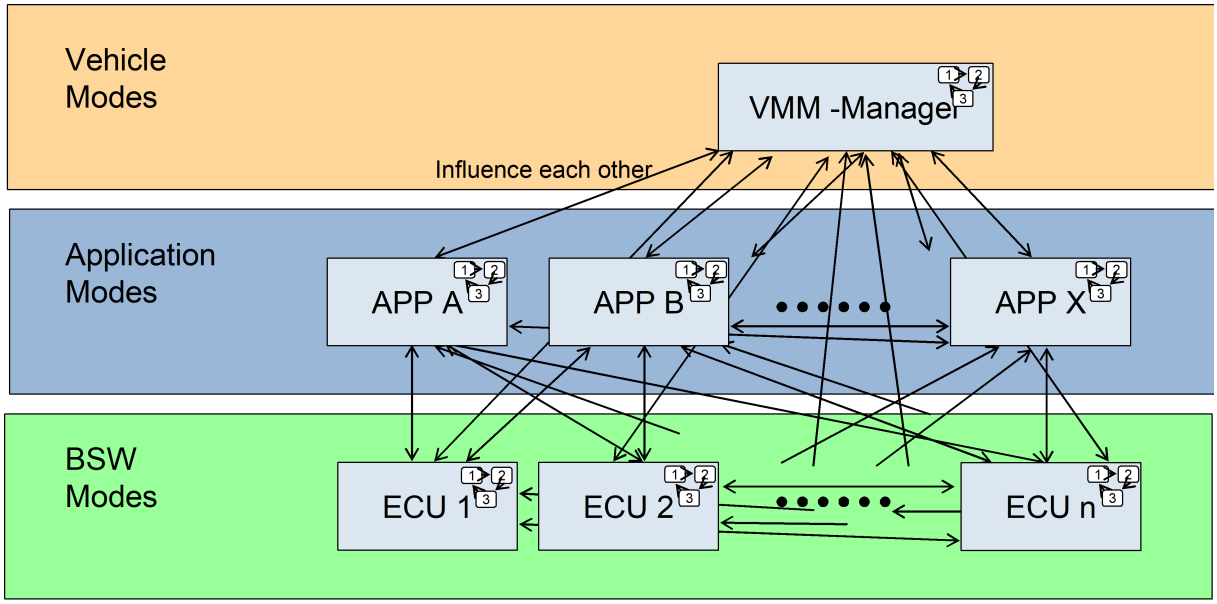

Figure 4.8: Influences between Modes

Figure 4.8 gives an impression of the resulting network of complex dependencies.

4.4.2 Use Cases

The following use case diagram shows the actors and their use cases.

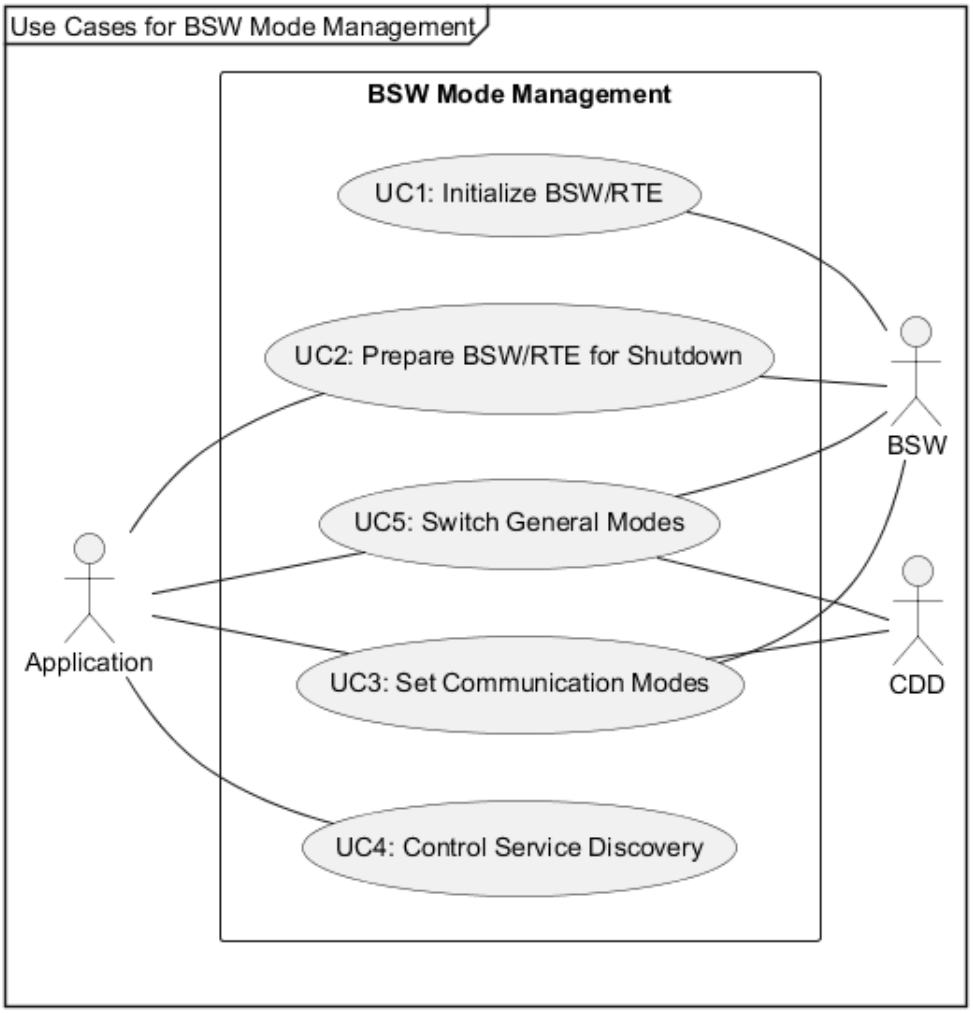

Figure 4.9: Use cases for BSW Mode Management

Use Case Name	Use Case Description
Switch General Modes	This use case allows the application or the BSW to arbitrate and switch to pre-defined (i.e., standardized) and user-defined modes in a generic way
Initialize BSW	This use case is triggered by the ECU State Management to initialize BSW beyond drivers and OS (which are initialized in scope of ECU state management)
Prepare BSW for Shutdown	This use case prepares BSW for shutdown, before control is handed over to ECU state management which actually shuts down the OS and the ECU
Set Communication Mode	This use case allows application and BSW to enable or disable network communication - or change the communication according to the current mode
Control Service Discovery	This use case allows the application to trigger a Service Discovery at any point in time - in case default behavior ("auto offer / auto subscribe") is not sufficient

4.4.3 Functional Requirements

4.4.3.1 Use Case: Switch General Modes

[SRS_ModeMgm_02000] Mode Arbitration on Request ⌈

Description:	When the BSW Mode Management receives a mode request from a mode requester, then the BSW Mode Management shall arbitrate the new mode.

⌋

[SRS_ModeMgm_02001] Configuration of Mode Requesters ⌈

Description:	The BSW Mode Management shall allow to register mode requester in/from application, BSW or CDD.

⌋

[SRS_ModeMgm_02002] Configuration of Mode Arbitration Rules ⌈

Description:	The BSW Mode Management shall allow to configure the rules for mode arbitration.
Rationale:	Each each project requires very specific rules for mode changes. Configuration allows to tailor the generic arbitration mechanism of the BSW Mode Management to the needs of a given project.

⌋

[SRS_ModeMgm_02003] Performing Mode Specific Actions ⌈

Description:	When the BSW Mode Management has finished mode arbitration, then the BSW Mode Management shall • execute a configurable list of actions (standardized or user-defined ones) • and then switch to the arbitrated mode

⌋

[SRS_ModeMgm_02004] Mode Notification to Users ⌈

Description:	When the BSW Mode Management switches to a new mode, then the BSW Mode Management shall notify all registered mode users in application, BSW or CDD about the mode switch.

⌋

4.4.3.2 Use Case: Initialize BSW

[SRS_ModeMgm_02005] Initialize BSW ⌈

Description:	When requested by BSW, then the BSW Mode Management shall initialize BSW modules and RTE. **Additional Information:** • Typically ECU State Management initializes drivers, OS and BSW Mode Management before BSW Mode Management takes over • RTE is initialized on each core

⌋

4.4.3.3 Use Case: Prepare BSW for Shutdown

[SRS_ModeMgm_02006] Prepare BSW for Shutdown ⌈

Description:	When requested by BSW or application, then the Basic SW Mode Management shall stop the RTE and prepare BSW modules for shutdown. **Additional Information:** • BSW Mode Management only prepares for shutdown • ECU State Management actually shuts off ECU or puts it to sleep

⌋

4.4.3.4 Use Case: Set Communication Modes

[SRS_ModeMgm_02007] Set Communication Modes ⌈

Description:	When requested by reqistered mode users in application, CDD or BSW, then the Basic SW Mode Management shall request • to enable/disable communication • or to change the communication mode on supported networks. **Additional Information:** Communication on a network may also be partially startup or shutdown.

⌋

4.4.3.5 Use Case: Control Service Discovery

[SRS_ModeMgm_02008] Control Service Discovery ⌈

Description:	When requested by application, then the Basic SW Mode Management shall allow • a server/provider of a service – to offer/ stop offering a service – and to query the service subscribers • and a client/subscriber of a service – to subscribe to a service – and to query the subscription status.

⌋

5 References

[1] Specification of Basic Software Mode Manager
AUTOSAR_CP_SWS_BSWModeManager

[2] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUStateManager

[3] Specification of Watchdog Manager
AUTOSAR_CP_SWS_WatchdogManager

[4] Specification of Communication Manager
AUTOSAR_CP_SWS_COMManager

[5] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[6] Glossary
AUTOSAR_FO_TR_Glossary

[7] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[8] Virtual Functional Bus
AUTOSAR_CP_TR_VFB

[9] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

A History of Requirements

Please note that the lists in this chapter also include requirements that have been removed from the specification in a later version. These requirements do not appear as hyperlinks in the document.

A.1 Requirement History of this Document According to AUTOSAR Release R25-11

A.1.1 Added Requirements in R25-11

[SRS_ModeMgm_02000] [SRS_ModeMgm_02001] [SRS_ModeMgm_02002] [SRS_ModeMgm_02003] [SRS_ModeMgm_02004] [SRS_ModeMgm_02005] [SRS_ModeMgm_02006] [SRS_ModeMgm_02007] [SRS_ModeMgm_02008]

A.1.2 Changed Requirements in R25-11

none

A.1.3 Deleted Requirements in R25-11

[SRS_ModeMgm_09174] [SRS_ModeMgm_09175] [SRS_ModeMgm_09176] [SRS_ModeMgm_09177] [SRS_ModeMgm_09178] [SRS_ModeMgm_09179] [SRS_ModeMgm_09180] [SRS_ModeMgm_09182] [SRS_ModeMgm_09183] [SRS_ModeMgm_09184] [SRS_ModeMgm_09228] [SRS_ModeMgm_09229] [SRS_ModeMgm_09230] [SRS_ModeMgm_09240] [SRS_ModeMgm_09241] [SRS_ModeMgm_09253] [SRS_ModeMgm_09255] [SRS_ModeMgm_09281]

A.2 Requirement History of this Document According to AUTOSAR Release R24-11

A.2.1 Added Requirements in R24-11

[SRS_ModeMgm_00001] [SRS_ModeMgm_00002] [SRS_ModeMgm_00003] [SRS_ModeMgm_00004] [SRS_ModeMgm_00005] [SRS_ModeMgm_00006] [SRS_ModeMgm_00007] [SRS_ModeMgm_00008] [SRS_ModeMgm_00009] [SRS_ModeMgm_00010] [SRS_ModeMgm_00011] [SRS_ModeMgm_00012] [SRS_ModeMgm_00013] [SRS_ModeMgm_00014] [SRS_ModeMgm_00015] [SRS_ModeMgm_00016] [SRS_ModeMgm_00017] [SRS_ModeMgm_00018] [SRS_ModeMgm_00019] [SRS_ModeMgm_00020] [SRS_ModeMgm_00021] [SRS_ModeMgm_00022] [SRS_ModeMgm_00023] [SRS_ModeMgm_00024]

A.2.2 Changed Requirements in R24-11

[SRS_ModeMgm_09256]

A.2.3 Deleted Requirements in R24-11

[SRS_ModeMgm_09001] [SRS_ModeMgm_09009] [SRS_ModeMgm_09017] [SRS_ModeMgm_09072] [SRS_ModeMgm_09097] [SRS_ModeMgm_09098] [SRS_ModeMgm_09100] [SRS_ModeMgm_09101] [SRS_ModeMgm_09102] [SRS_ModeMgm_09104] [SRS_ModeMgm_09113] [SRS_ModeMgm_09114] [SRS_ModeMgm_09115] [SRS_ModeMgm_09116] [SRS_ModeMgm_09118] [SRS_ModeMgm_09119] [SRS_ModeMgm_09120] [SRS_ModeMgm_09122] [SRS_ModeMgm_09126] [SRS_ModeMgm_09127] [SRS_ModeMgm_09128] [SRS_ModeMgm_09136] [SRS_ModeMgm_09145] [SRS_ModeMgm_09146] [SRS_ModeMgm_09147] [SRS_ModeMgm_09164] [SRS_ModeMgm_09165] [SRS_ModeMgm_09166] [SRS_ModeMgm_09173] [SRS_ModeMgm_09185] [SRS_ModeMgm_09186] [SRS_ModeMgm_09187] [SRS_ModeMgm_09188] [SRS_ModeMgm_09189] [SRS_ModeMgm_09190] [SRS_ModeMgm_09194] [SRS_ModeMgm_09199] [SRS_ModeMgm_09234] [SRS_ModeMgm_09235] [SRS_ModeMgm_09236] [SRS_ModeMgm_09237] [SRS_ModeMgm_09238] [SRS_ModeMgm_09239] [SRS_ModeMgm_09254] [SRS_ModeMgm_09264] [SRS_ModeMgm_09270] [SRS_ModeMgm_09271] [SRS_ModeMgm_09272] [SRS_ModeMgm_09274] [SRS_ModeMgm_09275] [SRS_ModeMgm_09276] [SRS_ModeMgm_09277]

A.3 Requirement History of this Document According to AUTOSAR Release R23-11

A.3.1 Added Requirements in R23-11

[SRS_ModeMgm_09281]

A.3.2 Changed Requirements in R23-11

none

A.3.3 Deleted Requirements in R23-11

none

A.4 Requirement History of this Document According to AUTOSAR Release R22-11

A.4.1 Added Requirements in R22-11

none

A.4.2 Changed Requirements in R22-11

[SRS_ModeMgm_00049] [SRS_ModeMgm_09009] [SRS_ModeMgm_09017] [SRS_ModeMgm_09028] [SRS_ModeMgm_09078] [SRS_ModeMgm_09081] [SRS_ModeMgm_09083] [SRS_ModeMgm_09084] [SRS_ModeMgm_09085] [SRS_ModeMgm_09089] [SRS_ModeMgm_09097] [SRS_ModeMgm_09098] [SRS_ModeMgm_09101] [SRS_ModeMgm_09102] [SRS_ModeMgm_09104] [SRS_ModeMgm_09106] [SRS_ModeMgm_09107] [SRS_ModeMgm_09110] [SRS_ModeMgm_09112] [SRS_ModeMgm_09114] [SRS_ModeMgm_09115] [SRS_ModeMgm_09118] [SRS_ModeMgm_09119] [SRS_ModeMgm_09120] [SRS_ModeMgm_09122] [SRS_ModeMgm_09125] [SRS_ModeMgm_09127] [SRS_ModeMgm_09128] [SRS_ModeMgm_09133] [SRS_ModeMgm_09136] [SRS_ModeMgm_09141] [SRS_ModeMgm_09143] [SRS_ModeMgm_09149] [SRS_ModeMgm_09155] [SRS_ModeMgm_09158] [SRS_ModeMgm_09159] [SRS_ModeMgm_09160] [SRS_ModeMgm_09161] [SRS_ModeMgm_09162] [SRS_ModeMgm_09165] [SRS_ModeMgm_09166] [SRS_ModeMgm_09168] [SRS_ModeMgm_09169] [SRS_ModeMgm_09172] [SRS_ModeMgm_09174] [SRS_ModeMgm_09179] [SRS_ModeMgm_09180] [SRS_ModeMgm_09182] [SRS_ModeMgm_09185] [SRS_ModeMgm_09186] [SRS_ModeMgm_09187] [SRS_ModeMgm_09188] [SRS_ModeMgm_09189] [SRS_ModeMgm_09190] [SRS_ModeMgm_09194] [SRS_ModeMgm_09199] [SRS_ModeMgm_09221] [SRS_ModeMgm_09222] [SRS_ModeMgm_09223] [SRS_ModeMgm_09225] [SRS_ModeMgm_09226] [SRS_ModeMgm_09228] [SRS_ModeMgm_09231] [SRS_ModeMgm_09232] [SRS_ModeMgm_09233] [SRS_ModeMgm_09235] [SRS_ModeMgm_09237] [SRS_ModeMgm_09238] [SRS_ModeMgm_09239] [SRS_ModeMgm_09243] [SRS_ModeMgm_09249] [SRS_ModeMgm_09254] [SRS_ModeMgm_09269] [SRS_ModeMgm_09270] [SRS_ModeMgm_09271] [SRS_ModeMgm_09272] [SRS_ModeMgm_09274] [SRS_ModeMgm_09275] [SRS_ModeMgm_09276] [SRS_ModeMgm_09277]

A.4.3 Deleted Requirements in R22-11

[SRS_ModeMgm_09280]