Document Title	Requirements on Mode Management
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Identification No	069
Document Classification	Auxiliary

Document Version	1.2.0
Document Status	Final
Part of Release	3.0
Revision	0001

Document Change History

Date	Version	Changed by	Change Description
26.11.2007	1.2.0	AUTOSAR Administration	• BSW09088 removed due to introduction of Bus-specific State Managers below Communication Manager • BSW09130 and BSW09131 removed due to direct initialization of the communication stack by the ECU State Manager • BSW09170 and BSW09171 added for alive supervision during startup, shutdown, and sleep • BSW09172 added for clarification of Communication Manager behavior • BSW09173 added for clarification of ECU State Manager behavior • Rephrased multiple requirements to clarify behavior and configuration classes • Document meta information extended • Small layout adaptations made

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
31.01.2007	1.1.0	AUTOSAR Administration	• BSW09075 removed (moved to SRS General) • New requirements BSW09168 and BSW09169 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
08.05.2006	1.0.0	AUTOSAR Administration	Initial Release

Page left intentionally blank

Disclaimer

Any use of these specifications requires membership within the AUTOSAR Development Partnership or an agreement with the AUTOSAR Development Partnership. The AUTOSAR Development Partnership will not be liable for any use of these specifications.

Following the completion of the development of the AUTOSAR specifications commercial exploitation licenses will be made available to end users by way of written License Agreement only.

Advice to users of AUTOSAR Specification Documents:

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

Table of Contents

1 Scope of Document

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

1. ECU State Manager (EcuM) which manages startup and shutdown of the ECU. This includes triggering the shutdown when all requests for run are released;
2. Watchdog Manager (WdgM) which collects the alive indications from the independent applications and relays them to the hardware watchdog in a suitable way;
3. Communication Manager (ComM) which coordinates the communication of the independent applications.

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 [6].

2 Conventions to be Used

- In requirements, the following specific semantics shall be used (based on the Internet Engineering Task Force IETF).

  The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as:

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

3 Requirements Guidelines

Existing specifications shall be referenced (in form of a single requirement). Differences to these specifications are specified as additional requirements.

3.1 Requirements quality

All Requirements shall have the following properties:

- Redundancy
  Requirements shall not be repeated within one requirement or in other requirements
- Clearness
  All requirements shall allow one possibility of interpretation only. Only technical terms of the glossary may be used.
- Atomicity
  Each Requirement shall only contain one requirement. A Requirement is atomic if it cannot be split up in further requirements.
- Testability
  Requirements shall be testable by analysis, review or test.
- Traceability
  The source and status of a requirement shall be visible at all times.

3.2 Requirements identification

Each requirement has its unique identifier starting with BSW prefix. For any review annotations, remarks and/or questions please refer to this unique ID rather than chapter or page numbers!

4 Related Documentation

[1] Specification of Communication Manager
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_ComManager.pd

[2] Specification of ECU Manager
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_ECU_StateManager.pdf

[3] Specification of Watchdog Manager
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_WatchdogManager.pdf

[4] Specification of Communication Stack Types
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_ComStackTypes.pdf

[5] General Requirements on Basic Software Modules
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_General.pdf

[6] Layered Software Architecture
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_LayeredSoftwareArchitecture.pdf

[7] Specification of the Virtual Functional Bus
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_VirtualFunctionBus.pdf

[8] MetaModel
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_MetaModel

[9] Software Component Template
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SoftwareComponentTemplate

[10] Specification of System Template
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SystemTemplate

[11] Requirements on CAN
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_CAN

[12] Requirements on LIN
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_LIN

5 Terminology

5.1 Glossary

Term:	***Description:***
Active wake-up	Wake-up caused by the ECU e.g. by a sensor.
Alive indication	Indication that a supervised SWC is alive – meaning active – provided from the SWC itself to the Watchdog Manager.
Application	Software which is positioned “above” RTE, typically application software components (SWCs).
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*** (i.e. 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. 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***.
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).
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.
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.).
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***.
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***.
User	Concept for requestors of the ECU State Manager and of the Communication Manager. A user may be a runnable entity, a SWC/BSW or even a group of SWCs/BSWs, which acts as a single unit towards the ECU State Manager and the Communication Manager.

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.

5.2 Abbreviations

Abbreviation:	***Description:***
BSW	Basic Software
ComM	Communication Manager
DCM	Diagnostic Communication Manager
DEM	Diagnostic Event Manager
EcuM	ECU State Manager
FiM	Function Inhibition Manager
RE	Runnable Entity
SWC	Software Component
WdgM	Watchdog Manager

6 Requirement Specification

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

→ the ECU State Manager, controlling the startup phase of AUTOSAR BSW modules including startup of the OS;
→ the Communication Manager, in charge of the network resource management;
→ the Watchdog Manager, responsible for triggering the watchdog based on the aliveness status of application software.

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

6.1 ECU State Manager

6.1.1 Functional Overview

The ECU State Manager is a basic software module that manages the ECU states (OFF, RUN, SLEEP) and the transitions between these states (transient states: STARTUP, WAKEUP, SHUTDOWN). In detail, the ECU State Manager:

- is responsible for the initialization and de-initialization of all basic software modules, including OS and RTE;
- cooperates with the so-called Resource Managers (e.g. the Communication Manager) to shut down the ECU when needed;
- manages all wake-up events and configures the ECU for SLEEP when requested.

The ECU State Manager shall support individual preparation-activities and transitions to bring up the ECU or put it into a decreased power mode (low-power mode, e.g. sleep / standby). By a well defined usage of ECU State Manager features and capabilities, this module can then be used to perform a pre-defined strategy of power-consumption, thus allowing efficient energy management for the ECU.

Features and benefits of the ECU State Manager include the following:

- Initialization and shutdown of Basic Software Modules.
- Standardized definition of ECU main states.
- Time triggered increased inoperation[1].

Much more details about these functionalities can be found in the ECU State Manager SWS specification [2].

[1] This mechanism is intended to wake-up the ECU after a certain time-period automatically, if no other wake-up reasons have occurred, in order to transfer the ECU into another sleep mode, which may be related to further decreased capabilities and power-consumption.

6.1.2 Functional Requirements

6.1.2.1 Configuration

6.1.2.1.1 [BSW09120] Configuration of initialization process of Basic Software modules

Initiator:	VW (D. Gerlach, IAV)
Date:	16.12.2004
Short Description	Configuration of initialization process of Basic Software modules
Type:	--
Importance:	high
Description:	The Basic Software modules with their initialization routines and their initialization order shall be pre-compile time configurable attributes of the ECU State Manager.
Rationale:	The ECU State Manager shall have knowledge about the initialization-routines from Basic Software modules and their predecessors and successor relationships. However, it shall be possible to adapt the concrete initialization process of the ECU State Manager to the project specific initialization demands of the Basic Software modules.
Use Case:	System A requires the initialization of the watchdog driver as soon as possible to monitor the system startup whereas in system B the watchdog driver shall be initialized last, in order to avoid resets based on long lasting hardware initializations. Integration of supplier specific driver (complex driver), e.g. Display driver, into the startup sequence.
Dependencies:	Initialization of Basic Software modules.
Conflicts:	--
Supporting Material:	--

6.1.2.1.2 [BSW09147] Configuration of de-initialization process of Basic Software modules

Initiator:	PSA (P. Castelpietra)
Date:	13.07.2005
Short Description	Configuration of de-initialization process of Basic Software modules
Type:	--
Importance:	high
Description:	**SRS1206**: The Basic Software modules with their de-initialization routines and their de-initialization order shall be pre-compile time configurable attributes of the ECU State Manager.
Rationale:	The ECU State Manager shall have knowledge about the de-initialization-routines from Basic Software modules and their predecessors and successor relationships. However, it shall be possible to adapt the concrete de-initialization process of the ECU State Manager to the project specific de-initialization demands of the Basic Software modules.
Use Case:	--
Dependencies:	De-initialization of Basic Software modules.
Conflicts:	--
Supporting Material:	--

6.1.2.1.3 [BSW09122] Configuration of users of the ECU State Manager

Initiator:	VW (D. Gerlach, IAV)
Date:	16.12.2004
Short Description	Configuration of users of the ECU State Manager
Type:	--
Importance:	high
Description:	**SRS1207**:Users, indicating the operational needs of individual software components, shall be pre compile time configurable attributes of the ECU State Manager. Requests from unknown sources are ignored.
Rationale:	All users are known and configured at compile-time. This requirement supports static administration of (known) state requestors on the ECU; it supports also testing and documentation.
Use Case:	--
Dependencies:	Requesting and releasing states.
Conflicts:	--
Supporting Material:	--

6.1.2.1.4 [BSW09100] Selection of wake-up sources shall be configurable

Initiator:	WP4.2.2.1.9
Date:	13.12.2004
Short Description:	Selection of wake-up sources shall be configurable
Type:	--
Importance:	high
Description:	**SRS1208**: It shall be possible to configure in pre-compile time which wake-up sources are relevant in which sleep mode, provided that they can be actually assigned to a sleep mode (i.e. they will be enabled when the ECU enters that particular sleep mode).
Rationale:	Wake-up sources cannot be standardized in detail. Their mapping to use cases cannot be standardized neither. Therefore this flexibility is needed.
Use Case:	Depending on the ECU, the allowed wake-up source may vary: – reception of a CAN frame, – direct input(s), – either CAN frames or direct inputs, – etc. Which wake-up source has to be activated in which sleep mode, is then a configuration parameter.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.1.2.1.5 [BSW09146] Configuration of time triggered increased inoperation

Initiator:	PSA (P. Castelpietra)
Date:	08.07.2005
Short Description	Configuration of time triggered increased inoperation
Type:	--
Importance:	medium
Description:	**SRS1209**: It shall be possible to enable or disable the time triggered increased inoperation at pre compile time. This feature shall stay of course disabled if the time triggered increased inoperation is not supported by the hardware (the wake-up/sleep capability and several sleep modes should be available on the ECU; see BSW09118
Rationale:	Making this feature configurable will save resources on ECUs which do not need it.
Use Case:	--
Dependencies:	- Wake-up/sleep capability shall be available on the ECU. - Time-triggered wake-up events shall be available on the ECU. - [BSW09119] Support of several sleep modes. - [BSW09118] Time triggered increased inoperation.
Conflicts:	--
Supporting Material:	--

6.1.2.2 Normal Operation

6.1.2.2.1 [BSW09001] Standardization of state relations

Initiator:	DC
Date:	13.02.2004
Short Description:	Standardization of state relations
Type:	--
Importance:	high
Description:	**SRS1210**: The number and names of main states and the transitions between main states shall be standardized.
Rationale:	The state machine is standardized.
Use Case:	Relocatability of software components.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.1.2.2.2 [BSW09116] Requesting and releasing the RUN state

Initiator:	VW (D. Gerlach, IAV)
Date:	16.12.2004
Short Description	Requesting and releasing the RUN state
Type:	--
Importance:	high
Description:	**SRS1211**: The ECU State Manager shall provide services to request and release the RUN state.
Rationale:	The decision to hold an ECU in the RUN state is always related to the application context, which cannot be standardized. Therefore the ECU State Manager needs to get "user-requests" from application perspective to be aware of the current situation. It is recommended that independent applications (SWCs), having independent RUN state needs, shall control their "user request" by usage of the provided interface of the ECU State Manager.
Use Case:	A wake-up event (e.g. door contact) related with a preheating functionality of the catalytic exhaust system occurs on the ECU which covers and controls this feature by software. With this wake-up event, a request to enter the RUN state on this ECU shall be established at the ECU State Manager. As soon as the preheating is finished, the preheating functionality shall release its demand to stay in the RUN state. If no other application software components on this ECU are requesting to stay in the RUN state, the ECU State Manager will further proceed with the shutdown process.
Dependencies:	[BSW09115] Evaluate condition to stay in the RUN state.
Conflicts:	--
Supporting Material:	--

6.1.2.2.3 [BSW09173] Minimum duration of Run State

Initiator:	WPII-2.1.1
Date:	25/07/07
Short Description	Minimum duration of Run State
Type:	--
Importance:	high
Description:	After entering the Run state, the ECU State Manager shall maintain this state for a minimum amount of time. The minimum duration to stay in Run state shall be configurable.
Rationale:	The ECU State Manager has to stay in Run state for a period of time in order to avoid immediate shutdown and to give users the chance/time to request RUN themselves
Use Case:	--
Dependencies:	[BSW09113] Initialization of Basic Software modules [BSW09114] Starting/invoking the shutdown process
Conflicts:	--
Supporting Material:	--

6.1.2.2.4 [BSW09114] Starting/invoking the shutdown process

Initiator:	VW (D. Gerlach, IAV)
Date:	16.12.2004
Short Description	Starting/invoking the shutdown process
Type:	--
Importance:	high
Description:	After the release of the last request for RUN state OR if the BSW forces the shutdown, the ECU State Manager shall leave the RUN state and start / invoke the shutdown process.
Rationale:	The ECU State Manager is the BSW module controlling the states of the ECU. Therefore the ECU State Manager is responsible to initiate the state transitions. Defined responsibility to initiate the shutdown process.
Use Case:	--
Dependencies:	The OS shall provide an interface/service to invoke its own shutdown. [BSW09072] Force ECU shutdown [BSW09173] Minimum duration of Run State
Conflicts:	--
Supporting Material:	--

6.1.2.2.5 [BSW09104] ECU State Manager shall take over control after OS shutdown

Initiator:	WP4.2.2.1.9
Date:	13.12.2004
Short Description:	ECU State Manager shall take over control after OS shutdown
Type:	--
Importance:	high
Description:	**SRS1213**: After OS shutdown, additional shutdown activities shall be taken like preparing hardware for the next wake-up. Finally the ECU has to be switched off or put into a sleep mode. This shall be the task of the ECU State Manager.
Rationale:	- Provide services after OS-shutdown. - Startup/shutdown symmetry.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.1.2.2.6 [BSW09113] Initialization of Basic Software modules

Initiator:	VW (D. Gerlach, IAV)
Date:	15.12.2004
Short Description	Initialization of Basic Software modules
Type:	--
Importance:	high
Description:	**SRS1214**: The ECU State Manager shall initialize the Basic Software modules. When the init process has finished, the RUN state shall be entered. Nevertheless, the ECU State Manager is not responsible for the initialization of the application software components; this responsibility lies with the RTE.
Rationale:	Organize the initialization process of the BSW modules and give SWCs the possibility to request the RUN state.
Use Case:	Defined initialization process of the basic software architecture.
Dependencies:	- Predecessor and successor relationships of the Basic Software modules during the initialization process. - [BSW09120] Configuration of initialization process of Basic Software modules. - [BSW09173] Minimum duration of Run State
Conflicts:	--
Supporting Material:	--

6.1.2.2.7 [BSW09127] De-initialization of Basic Software modules

Initiator:	WP4.2.2.1.9
Date:	03.03.2005
Short Description:	De-Initialization of Basic Software modules
Type:	--
Importance:	high
Description:	**SRS1215**: The ECU State Manager shall de-initialize Basic Software modules where appropriate during the shutdown process.
Rationale:	Coordinated ECU shutdown.
Use Case:	--
Dependencies:	[BSW09147] Configuration of de-initialization process of Basic Software modules.
Conflicts:	--
Supporting Material:	--

6.1.2.2.8 [BSW09128] Support of several shutdown targets

Initiator:	WP4.2.2.1.9
Date:	03.03.2005 / 28.06.2005 (rephrased)
Short Description:	Support of several shutdown targets
Type:	--
Importance:	high
Description:	**SRS1216**: The ECU State Manager shall offer the following targets for shutting down the ECU: Power Off Reset of ECU Sleep Modes The default target shall be defined by configuration. It can be overridden by an API-service. Several Sleep Modes may be supported (BSW09119)
Rationale:	Allow trade off between energy consumption and time to wake-up/startup the ECU.
Use Case:	--
Dependencies:	- [BSW09119] Support of several sleep modes. - [BSW09118] Time triggered increased inoperation.
Conflicts:	--
Supporting Material:	--

6.1.2.2.9 [BSW09119] Support of several sleep modes

Initiator:	VW (D. Gerlach, IAV)
Date:	16.12.2004
Short Description:	Support of several sleep modes
Type:	--
Importance:	medium
Description:	**SRS1217**: The ECU State Manager shall be able to handle a pre-defined (either pre-compile or link-time configuration) set of sleep modes.
Rationale:	Support portability of system-strategies according to increased inoperation mechanism.
Use Case:	--
Dependencies:	- The hardware has to support several sleep modes. - Interface of MCU-driver.
Conflicts:	--
Supporting Material:	--

6.1.2.2.10 [BSW09102] API for selecting the sleep mode

Initiator:	WP4.2.2.1.9
Date:	13.12.2004
Short Description:	API for selecting the sleep mode
Type:	--
Importance:	high
Description:	**SRS1218**: The ECU State Manager shall provide an API to select the sleep mode which will be chosen for next ECU shutdown. Whenever a decision to go to sleep is taken (which is independent from the use of this API; see rather BSW09072, BSW09115, BSW09165, BSW09166, BSW09114 and BSW09116), the current active sleep mode selection shall be used to choose the actual target sleep mode of the shutdown process.
Rationale:	Application needs a way to select the next shutdown target. However, the decision to start the shutdown process will be totally independent from the call of this API.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.1.2.2.11 [BSW09072] Force ECU shutdown

Initiator:	Porsche
Date:	27.09.2004
Short Description:	Force ECU shutdown
Type:	--
Importance:	high
Description:	**SRS1219**: The ECU State Manager shall provide the means (explicit API or procedure) to force starting the shutdown process BSW09114. This feature shall not be available to the application, but only accessible by BSW.
Rationale:	Controlled de-initialization of basic software.
Use Case:	Force ECU to shutdown because it is assumed that this ECU does not work properly. Putting it into defined testing condition.
Dependencies:	[BSW09114] Starting/invoking the shutdown process.
Conflicts:	--
Supporting Material:	--

6.1.2.2.12 [BSW09009] Activation of software when entering/leaving ECU states

Initiator:	DC
Date:	13.02.2004
Short Description:	Activation of software when entering/leaving ECU states
Type:	--
Importance:	high
Description:	**SRS1220**: The ECU State Manager shall provide the ability to execute external, statically-configured code at each transition between ECU states. This will be likely implemented by a callback when entering/leaving ECU states.
Rationale:	Basic Functionality.
Use Case:	Launching initialization/de-initialization routines when entering/leaving the RUN state.
Dependencies:	[BSW101] Initialization interface (WP1.1.2).
Conflicts:	--
Supporting Material:	--

6.1.2.2.13 [BSW09017] Provide ECU state information

Initiator:	Bosch
Date:	12.02.2004
Short Description:	Provide ECU state information
Type:	--
Importance:	high
Description:	**SRS1221**: The ECU State Manager shall provide an API to query the current ECU state.
Rationale:	Basic functionality. SW may behave differently depending on current ECU state.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.1.2.2.14 [BSW09138] Selection of application mode of OS

Initiator:	IAV (Gerlach), BMW (Schlaugath)
Date:	18.05.2005
Short Description:	Selection of application mode of OS
Type:	--
Importance:	high
Description:	**SRS1222**: The ECU State Manager shall control the selection of the application mode of the OS at startup (StartOS-service parameter).
Rationale:	ECU State Manager starts the OS. The “application mode” parameter has to be provided when calling the StartOS primitive (see AUTOSAR OS specification [13]). Application modes are designed to allow the operating system to come up under different modes of operations; e.g. end of line programming and normal operation.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.1.2.2.15 [BSW09136] Centralized wake-up management

Initiator:	Porsche
Date:	06.04.2005
Short Description:	Centralized wake-up management
Type:	--
Importance:	high
Description:	**SRS1224**: The ECU State Manager shall be the receiver of all wake-up events which occur on its ECU; it shall react appropriately (e.g. ECU wake-up) and propagate the information to other concerned components.
Rationale:	Unambiguous and defined behavior at wake-up events.
Use Case:	--
Dependencies:	- [BSW00375] Notification of wake-up reason (WP1.1.2). - [BSW09113] Initialization of Basic Software modules - [BSW09097] Validation of physical channel wake-up
Conflicts:	--
Supporting Material:	--

6.1.2.2.16 [BSW09098] Storing the wake-up reasons

Initiator:	WP4.2.2.1.9
Date:	13.12.2004
Short Description:	Storing the wake-up reasons
Type:	--
Importance:	High
Description:	**SRS1225:** The ECU State Manager shall be able to store the current wake-up reason.
Rationale:	Storing the wake-up reason in one module and give every module (BSWs and SWCs) the possibility to query the wake-up reason.
Use Case:	Retrieve the last wake-up reason by means of an external diagnostic tool. Some behaviors (Application mode management may be triggered by the wake up
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.1.2.2.17 [BSW09097] Validation of physical channel wake-up

Initiator:	WP4.2.2.1.9
Date:	13.12.2004
Short Description:	Validation of physical channel wake-up
Type:	--
Importance:	high
Description:	**SRS1226**: The ECU State Manager shall start a timeout (T_wake_up_timeout greater than zero, statically configurable) after receiving a wake-up indication from the physical channel handler. If a valid message is received before the timeout expires, the timer shall be stopped and the ECU State Manager indicates a physical channel wake-up. ECU State Manager shall not indicate a wake-up if the first wake-up indication from the physical channel handler is not confirmed by a subsequent valid message within T_wake_up_timeout. If T_wake_up_timeout is set to 0 validation process is disable and wakeup is immediately confirmed
Rationale:	Avoiding wake-up because of erroneous signals (e.g. spikes). If a wake-up occurs but no succeeding activity can be detected on the physical channel, the wake-up is not considered, in order to save power.
Use Case:	--
Dependencies:	Wakeup validation is currently only need for CAN networks.
Conflicts:	--
Supporting Material:	--

6.1.2.2.18 [BSW09118] Time triggered increased inoperation

Initiator:	VW (D. Gerlach, IAV)
Date:	16.12.2004
Short Description	Time triggered increased inoperation
Type:	--
Importance:	medium
Description:	**SRS1227**: The ECU State Manager shall provide a mechanism to enter a step by step decreasing power mode (sleep modes), based on timeouts. This mechanism can only be available if the wake-up/sleep capability is available on the ECU; several sleep modes are available on the ECU. It shall be a configurable feature of the ECU State Manager (see BSW09146).
Rationale:	If the ECU provides a set of different sleep modes, it should be possible to define a time-triggered sequence to change the sleep modes without additional interaction from outside of the ECU. It is assumed that the different sleep modes represent a different level of low-power consumption. The change of sleep mode should lead into a step by step decreasing power-consumption of the ECU, while the ECU is not needed in the system.
Use Case:	If the car is left for a longer time – e.g. 3 days typically – more and more systems are shut off to save energy. The saving effect can be increased by shutting off more and more resources of the ECU itself: PLLs, clocks, sensors, etc. The aliveness of a comfort-body unit (ECU), for example, is not needed the whole time, because the user doesn't place any requests on the functionality of this unit (e.g. pressing a button) while the system is running. After a (configurable) period of time this unit could reduce its power consumption by entering a given sleep mode. Further it goes down in power-consumption by using the time triggered increased inoperation mechanism, while no user-requests are detected on this unit. Build up a system strategy with shortened startup, if the system was in usage immediately before. The longer the system is not in usage, the power-consumption is reduced by means of the time triggered increased inoperation, but as a consequence also the time for startup increases.
Dependencies:	- Wake-up/sleep capability shall be available on the ECU. - Time-triggered wake-up events shall be available on the ECU. - [BSW09119] Support of several sleep modes. - [BSW09146] Configuration of time triggered increased inoperation.
Conflicts:	--
Supporting Material:	--

6.1.2.2.19 [BSW09145] Support of wake-sleep operation

Initiator:	WP4.2.2.1.9
Date:	28.06.2005
Short Description	Support of wake-sleep operation
Type:	--
Importance:	high
Description:	**SRS1228:** The ECU State Manager shall support a sequence consisting of: o partial startup o execution of limited jobs (see use case) o quick shutdown without OS context. This behaviour will be called "wake-sleep operation" in the context of the ECU State Manager.
Rationale:	Energy saving.
Use Case:	A cyclic wake-up by a timer-event, which should only do a short check of system wake-up conditions of non wakeable-sources (standard IO-ports) or execute a cyclic alive-feedback to customer, like i.e. a blinking LED.
Dependencies:	- Reduced capabilities of basic software architecture, according to non full startup for wake-sleep operation. - The wake-up/sleep hardware capability shall be available on the ECU.
Conflicts:	--
Supporting Material:	--

6.1.2.2.20 [BSW09126] Provide an API for querying the wake-up reason

Initiator:	BMW
Date:	11.02.2004
Short Description:	Provide an API for querying the wake-up reason
Type:	--
Importance:	high
Description:	**SRS1229**: The ECU State Manager shall provide an API for querying the wake-up event causing the last ECU wake-up.
Rationale:	For diagnostics purposes. Applications shall be able to react on different wake-up.
Use Case:	At initialization, application needs to perform different treatments depending on the wake-up reason.
Dependencies:	[BSW00375] Notification of wake-up reason.
Conflicts:	--
Supporting Material:	--

6.1.2.2.21 [BSW09101] Provide an API to query the reset reason

Initiator:	WP 4.2.2.1.9
Date:	13.12.2004 (approved back on 31.01.2006)
Short Description:	Provide an API to query the reset reason
Type:	--
Importance:	high
Description:	**SRS1233**: The ECU State Manager shall provide an API to query the reason of the last reset.
Rationale:	This feature is needed for setting up different startup scenarios depending if the reset was intentional or unintentional.
Use Case:	Reset loop detection.
Dependencies:	This feature can only be provided if supported by the hardware.
Conflicts:	--
Supporting Material:	--

6.1.2.2.22 [BSW09115] Evaluate condition to stay in the RUN state

Initiator:	VW (D. Gerlach, IAV)
Date:	16.12.2004
Short Description	Evaluate condition to stay in the RUN state
Type:	--
Importance:	high
Description:	**SRS1230**: The ECU State Manager shall include a mechanism to evaluate the condition to stay in the RUN state, derived from current demands of the application on the ECU.
Rationale:	The ECU State Manager is responsible for managing the ECU states. Therefore, it is necessary to evaluate the condition to stay in the RUN state, which has to be in relationship with the needs of the application software components.
Use Case:	--
Dependencies:	[BSW09116] Requesting and releasing the RUN state.
Conflicts:	--
Supporting Material:	--

6.1.2.2.23 [BSW09164] Shutdown synchronization support for SW-Components

Initiator:	VW (D.Gerlach, IAV)
Date:	28.11.2005
Short Description	Shutdown synchronization support for SW-Components
Type:	--
Importance:	medium
Description:	When leaving the RUN state, the ECU State Manager shall provide a synchronization point for SWCs shutdown by defining a POST-RUN state. In this state, BSW and the underlying hardware are still fully functional. This state can only be entered when all users have released their RUN request. It is expected that the users release their POST-RUN request after finalization of their shutdown preparation activities, thus allowing the subsequent de-initialization of BSW.
Rationale:	This state will be requested by SWCs needing shutdown synchronization to perform shutdown activities that are required before their access to resources is decreased (i.e. Stop of RTE; Stop of OS).
Use Case:	A digital filter algorithm that does not have the knowledge when its functionality is needed but needs to perform activities to prepare for shutdown.
Dependencies:	[BSW09001] Standardisation of state relations. [BSW09116] Requesting and releasing the RUN state. [BSW09165] Requesting and releasing the POST-RUN state. [BSW09166] Evaluate condition to stay in the POST-RUN state.
Conflicts:	--
Supporting Material:	--

6.1.2.2.24 [BSW09165] Requesting and releasing the POST-RUN state

Initiator:	DC (H. Kober)
Date:	28.11.2005
Short Description	Requesting and releasing the POST-RUN state
Type:	--
Importance:	medium
Description:	The ECU State Manager shall provide services to request and release the POST-RUN state.
Rationale:	The decision to hold an ECU in the POST-RUN state is always related to the application context, which cannot be standardized. Therefore the ECU State Manager needs to get "user-requests" from application perspective to be aware of the current situation. It is recommended that independent applications (SWCs), having independent POST-RUN state needs, shall control their "user request" by usage of the provided interface of the ECU State Manager.
Use Case:	--
Dependencies:	[BSW09164] Shutdown synchronization support for SW-Components. [BSW09166] Evaluate condition to stay in the POST-RUN state.
Conflicts:	--
Supporting Material:	--

6.1.2.2.25 [BSW09166] Evaluate condition to stay in the POST-RUN state

Initiator:	PSA (P. Castelpietra)
Date:	28.11.2005
Short Description	Evaluate condition to stay in the POST-RUN state
Type:	--
Importance:	medium
Description:	The ECU State Manager shall include a mechanism to evaluate the condition to stay in the POST-RUN state, derived from current demands of the application on the ECU.
Rationale:	The ECU State Manager is responsible for managing the ECU states. Therefore, it is necessary to evaluate the condition to stay in the POST-RUN state, which has to be in relationship with the needs of the application software components.
Use Case:	--
Dependencies:	[BSW09164] Shutdown synchronization support for SW-Components. [BSW09116] Requesting and releasing the RUN state. [BSW09165] Requesting and releasing the POST-RUN state.
Conflicts:	--
Supporting Material:	--

6.1.2.2.26 [BSW09170] Triggering Watchdog Manager during Startup / Shutdown and Sleep

Initiator:	WakeupTaskForce
Date:	11/06/07
Short Description	Triggering Watchdog Manager during Startup / Shutdown
Type:	--
Importance:	high
Description:	The ECU State Manager shall trigger Watchdog Manager during Sleep, Startup, Shutdown and Wake-Sleep Operations.
Rationale:	Only the ECU State Manager is still operational in Startup. Shutdown and Sleep State. Its activities have to be supervised by the Watchdog Manager.
Use Case:	The ECU gets stuck in Startup or Shutdown activities or while being in Sleep. A watchdog is used to supervise the ECU. The Watchdog Manager triggers the watchdog during these phases as long as the ECU State Manager is still alive. If it is stuck, the ECU will be reset by the watchdog.
Dependencies:	[BSW09171] Check aliveness of EcuM
Conflicts:	--
Supporting Material:	--

6.2 Watchdog Manager

6.2.1 Functional Overview

The Watchdog Manager 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.
Derived from the layered architecture approach, a decoupling between application timing constraints and watchdog hardware timing constraints becomes possible.

Based on this decoupling the Watchdog Manager provides alive-supervision of several independent applications while triggering the watchdog hardware.

The following features are provided by the Watchdog Manager:

- Supervision of multiple individual applications located on the ECU, having independent timing constraints and requiring a special supervision of runtime behaviour and aliveness.
- 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 SWS specification [3].

6.2.2 Functional Requirements

6.2.2.1 Configuration

6.2.2.1.1 [BSW09106] Configuration of Watchdog Manager

Initiator:	WP4.2.2.1.9
Date:	05.10.2004
Short Description:	Configuration of Watchdog Manager
Type:	--
Importance:	high
Description:	**SRS1252**: The list of entities supervised by the Watchdog Manager shall be configurable at pre-compile time.
Rationale:	Application supervising.
Use Case:	--
Dependencies:	For the RTE interface to be generated, the supervised entities have to be known.
Conflicts:	--
Supporting Material:	--

6.2.2.1.2 [BSW09158] Post build time configuration sets for the Watchdog Manager

Initiator:	PSA (P. Castelpietra)
Date:	11.10.2005
Short Description:	Post build time configuration sets for the Watchdog Manager
Type:	--
Importance:	high
Description:	The Watchdog Manager shall support several configuration sets on one ECU at post build time. These configuration sets shall include the individual timing constraints of each supervised entity. It shall also be able to switch between these different configurations sets (e.g. depending on the current scheduling table).
Rationale:	Adapting alive-supervision to the current schedule.
Use Case:	- Schedule table switch between STARTUP II and RUN states (see ECU State Manager SWS). - Allowing a limp-home mode, having a totally different schedule.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.2.2.1.3 [BSW09154] Parameter for access protection for disabling alive-supervision

Initiator:	VW (D. Gerlach, IAV)
Date:	15.09.2005
Short Description	Parameter for access protection for disabling alive-supervision
Type:	--
Importance:	high
Description:	The possibility to disable the alive-supervision for each supervised entity shall be controlled by a pre-compile configurable parameter.
Rationale:	The possibility to disable alive-supervision may vary on the application and safety critical context of each supervised entities. Therefore the option to grant access shall be a configurable attribute, which shall never change during ECU SW lifetime.
Use Case:	--
Dependencies:	[BSW09153] Access protection for disabling alive-supervision.
Conflicts:	--
Supporting Material:	--

6.2.2.2 Initialization

6.2.2.2.1 [BSW09107] Watchdog Manager initialization

Initiator:	WP4.2.2.1.9
Date:	05.10.2004
Short Description:	Watchdog Manager initialization
Type:	--
Importance:	high
Description:	**SRS1254**: The Watchdog Manager shall provide an initialization service. This service allows the selection of one of the statically configured Watchdog Manager modes.
Rationale:	Basic functionality.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.2.2.2.2 [BSW09109] Prohibit disabling of watchdog

Initiator:	WP4.2.2.1.9
Date:	21.10.2004
Short Description:	Prohibit disabling of watchdog
Type:	--
Importance:	high
Description:	**SRS1255**: It shall be possible to configure, by means of a pre-processor switch, whether the Watchdog Manager initialization service and the Watchdog Manager 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:	--
Conflicts:	--
Supporting Material:	--

6.2.2.3 Normal Operation

6.2.2.3.1 [BSW09112] Check aliveness of application

Initiator:	WP4.2.2.1.9
Date:	05.10.2004
Short Description:	Check aliveness of application
Type:	
Importance:	high
Description:	**SRS1256**: The Watchdog Manager 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:	--
Conflicts:	--
Supporting Material:	--

6.2.2.3.2 [BSW09171] Check aliveness of EcuM

Initiator:	WPII-2.1.1
Date:	19/06/07
Short Description:	Check aliveness of EcuM
Type:	
Importance:	High
Description:	During startup, shutdown and sleep, Watchdog Manager shall supervise the activity of the ECU State Manager
Rationale:	The Watchdog Manager is the central instance that deals with watchdogs in all phases.
Use Case:	The Watchdog Manager triggers the watchdog during Startup, Sleep and Shutdown as long as the ECU State Manager is still alive. If it is stuck, the ECU will be reset by the watchdog.
Dependencies:	[BSW09170] Triggering Watchdog Manager during Startup / Shutdown and sleep
Conflicts:	--
Supporting Material:	--

6.2.2.3.3 [BSW09125] Update alive-supervision

Initiator:	WP4.2.2.1.9
Date:	05.10.2004
Short Description:	Update alive-supervision
Type:	--
Importance:	High
Description:	**SRS1257**: The Watchdog Manager shall provide a service allowing supervised entities to forward an alive indication, thus updating the alive-supervision 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:	--
Conflicts:	--
Supporting Material:	--

6.2.2.3.4 [BSW09142] Enabling / disabling alive-supervision

Initiator:	Hitachi (Kentaro Yoshimura)
Date:	17.04.2005
Short Description	Enabling / disabling alive-supervision
Type:	--
Importance:	high
Description:	**SRS1258**: The Watchdog Manager shall provide API to enable / disable alive-supervision of individual supervised entities.
Rationale:	Supervised entities can be temporarily disabled. For example, the FiM can change the running status of runnable entities from “enable” to “disable”. The Watchdog Manager shall only check the aliveness of “enabled” entities.
Use Case:	A SWC will change its behavior based on the ECU state, by changing the set of running REs of the SWC.
Dependencies:	[BSW09154] Parameter for access protection for disabling alive-supervision
Conflicts:	--
Supporting Material:	--

6.2.2.3.5 [BSW09160] Indication of failed alive-supervision

Initiator:	WP4.2.2.1.9
Date:	16.11.2005
Short Description	Indication of failed alive-supervision
Type:	--
Importance:	high
Description:	The Watchdog Manager shall be able to notify the application when the alive-supervision of a supervised entity fails. This information shall be forwarded through the RTE 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:	Alive-supervision of the watchdog-manager. [BSW09112] Check aliveness of application.
Conflicts:	--
Supporting Material:	--

6.2.2.3.6 [BSW09161] Condition to stop triggering the Watchdog Driver

Initiator:	PSA (P. Castelpietra)
Date:	03.11.2005
Short Description	Condition to stop triggering the Watchdog Driver
Type:	--
Importance:	high
Description:	The Watchdog Manager shall stop triggering the watchdog driver if the alive-supervision 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 alive-supervision failure is now considered unrecoverable. Then, a watchdog reset is the only solution.
Use Case:	--
Dependencies:	[BSW09162] Indication of an upcoming watchdog reset.
Conflicts:	--
Supporting Material:	--

6.2.2.3.7 [BSW09169] Immediate reset of the Watchdog Manager

Initiator:	WP4.2.1.1.9
Date:	01.08.2006
Short Description	Immediate reset of the Watchdog Manager
Type:	--
Importance:	high
Description:	The Watchdog Manger shall be able to immediately reset the MCU when an alive-supervision 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 BSW09162).
Rationale:	When the Watchdog Manager has detected an unrecoverable fault it will simply stop triggering 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:	[BSW09162] Indication of an upcoming watchdog reset.
Conflicts:	--
Supporting Material:	--

6.2.2.3.8 [BSW09162] Indication of an upcoming watchdog reset

Initiator:	PSA (P. Castelpietra)
Date:	03.11.2005
Short Description	Indication of an upcoming watchdog reset
Type:	--
Importance:	high
Description:	The Watchdog Manager shall be able to notify the software (BSW and SWC) when the decision to stop triggering the watchdog driver has been taken due to an alive-supervision failure (see BSW09161). This information shall be forwarded through the RTE 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:	[BSW09161] Condition to stop triggering the Watchdog Driver. [BSW09163] Delay before provoking a watchdog reset.
Conflicts:	--
Supporting Material:	--

6.2.2.3.9 [BSW09163] Delay before provoking a watchdog reset

Initiator:	PSA (P. Castelpietra)
Date:	03.11.2005
Short Description	Delay before provoking a watchdog reset
Type:	--
Importance:	high
Description:	It shall be possible to configure a delay between the moment the decision to stop triggering the watchdog driver has been taken (due to an alive-supervision failure, see BSW09161) and the moment the Watchdog Manager will actually stop triggering the watchdog driver. This delay will be used by the software in order to prepare itself to the upcoming reset (see BSW09162).
Rationale:	Giving the time to the whole software to prepare properly to the upcoming reset.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.2.2.3.10 [BSW09143] Behaviour of the WdgM during inactive alive-supervision

Initiator:	VW (D. Gerlach, IAV)
Date:	24.05.2005
Short Description:	Behaviour of the WdgM during inactive alive-supervision
Type:	--
Importance:	high
Description:	**SRS1260:** The Watchdog Manager shall trigger the watchdog itself if no supervision is active/necessary.
Rationale:	Prevent unintended watchdog-resets.
Use Case:	There are no supervised entities or all supervised entities are switched off from supervision.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.2.2.3.11 [BSW09111] Watchdog driver triggering

Initiator:	WP4.2.2.1.9
Date:	05.10.2004
Short Description:	Watchdog driver triggering
Type:	--
Importance:	high
Description:	**SRS1261:** The Watchdog Manager shall trigger the watchdog driver (via the Watchdog Interface) as long as the timing constraints / schedule periods of the entities under its control are met, thus preventing the HW watchdog from expiring.
Rationale:	Basic functionality.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.2.2.3.12 [BSW09110] Watchdog Manager mode selection service

Initiator:	WP4.2.2.1.9
Date:	21.10.2004
Short Description:	Watchdog Manager mode selection service
Type:	--
Importance:	high
Description:	**SRS1262:** The watchdog Manager shall provide a service interface, to select a mode of the Watchdog Manager. 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. Therefore, the Watchdog Manager 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 could forward a mode-selection from slow to fast, when initialization phase is finished.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.2.2.3.13 [BSW09028] Support multiple watchdog instances

Initiator:	Siemens VDO Automotive (D. Sauerbier)
Date:	09.08.2004
Short Description:	Support multiple watchdog instances
Type:	--
Importance:	high
Description:	**SRS1263:** The Watchdog Manager 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:	[BSW09167] Independent timing-constraints for watchdog-instances
Conflicts:	--
Supporting Material:	--

6.2.2.3.14 [BSW09153] Access protection for disabling alive-supervision

Initiator:	VW (D. Gerlach, IAV)
Date:	15.09.2005
Short Description:	Access protection for disabling alive-supervision
Type:	--
Importance:	high
Description:	The disabling of alive-supervision for certain supervised entities shall not be executed, if the disabling is prohibited for the corresponding supervised entity.
Rationale:	It may be of interest for safety critical applications, that supervised entities could never be disabled from alive-supervision, once the supervision was enabled. The Watchdog Manager shall protect unintended or not allowed disabling of alive-supervision for such entities.
Use Case:	An application may have detected errors or other conditions that typically shall lead into a limitation of comfort functions (e.g. low battery voltage). As a possible reaction, this application inhibits (e.g. by usage of FiM mechanisms) certain applications and therefore their alive-supervision may need to be disabled, too. The service to disable the alive-supervision will be used with a parameter in order to select the corresponding supervised entity. The selection is out of responsibility of the Watchdog Manager, therefore a mechanism inside the Watchdog Manager is needed to protect prohibited accesses.
Dependencies:	[BSW09142] Enabling / disabling alive-supervision. [BSW09154] Parameter for access protection for disabling alive-supervision.
Conflicts:	--
Supporting Material:	--

6.2.2.3.15 [BSW09167] Independent timing-constraints for watchdog-instances

Initiator:	VW (D. Gerlach)
Date:	15.12.2005
Short Description	Independent timing-constraints for watchdog-instances
Type:	--
Importance:	medium
Description:	The Watchdog Manager shall support independent timing-constraints for each watchdog-instance that needs to be triggered.
Rationale:	If multiple watchdog-instances exist on one platform (i.e. internal and external watchdog) it's very likely, that these watchdogs will come up with independent timing-constraints, which needs to be fulfilled by individual triggering. To support independent time periods for triggering each watchdog instance offers full flexibility to adapt the watchdog manager on any platform.
Use Case:	HW-platform with internal and external watchdog
Dependencies:	[BSW09028] Support multiple watchdog instances
Conflicts:	--
Supporting Material:	--

6.2.3 Fault Operation

6.2.3.1.1 [BSW09159] Reporting failure of the alive-supervision to DEM

Initiator:	WP4.2.2.1.9
Date:	18.10.2005
Short Description:	Reporting failure of the alive-supervision to DEM
Type:	--
Importance:	high
Description:	When it stops triggering the watchdog, the Watchdog Manager shall report to DEM the failure of the alive-supervision. The report to DEM shall be a configurable option of the Watchdog Manager.
Rationale:	Error tracing, Diagnostics.
Use Case:	--
Dependencies:	The current API of the DEM does not allow reporting the information about which supervised entity has failed.
Conflicts:	--
Supporting Material:	--

6.3 Communication Manager

6.3.1 Functional Overview

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

The purpose of the Communication Manager 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 Manger
- Simplifying the resource management by allocating all resources necessary for the requested communication mode.

In order to do so, the Communication Manager 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 SWS specification [1].

6.3.2 Functional Requirements

6.3.2.1 Configuration

6.3.2.1.1 [BSW09090] User-to-channel relationship

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	User-to-channel relationship
Type:	--
Importance:	high
Description:	**SRS1277**: Relationship between users and physical channels (which user communicates on which physical channel) are configurable at pre compile time 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).
Rationale:	Necessary for physical channel independency; communication shall be only activated if communication is needed on this physical channel.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.3.2.1.2 [BSW09133] Assigning physical channels to the Communication Manager

Initiator:	WP4.2.2.1.9 (D. Gerlach, IAV)
Date:	23.02.2005
Short Description	Assigning physical channels to the Communication Manager
Type:	--
Importance:	high
Description:	**SRS1278**: Physical channels, which should be handled by the Communication Manager, shall be assigned to the module by pre compile time configuration.
Rationale:	The ECU specific implementations of the Communication Manager shall have knowledge about how many and which physical channels have to be treated by the Communication Manager on the dedicated ECU. This knowledge is required to enhance the independency of the physical channel management within the Communication Manager. Additionally, it gives flexibility to tailor required data-resources for the specific implementations.
Use Case:	The Communication Manager shall cope with individual communication resources of different ECUs i.e.: ECU A : two FlexRay busses, one CAN bus ECU B : one CAN bus, one LIN bus
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.3.2.1.3 [BSW09132] Assigning Network Management to physical channels

Initiator:	WP4.2.2.1.9 (D. Gerlach, IAV)
Date:	23.02.2005
Short Description	Assigning Network Management to physical channels
Type:	--
Importance:	high
Description:	**SRS1279**: 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.
Rationale:	The Communication Manager 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 Manger 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:	Assigning physical channels to the Communication Manager. This issue is only relevant for assigned physical channels.
Conflicts:	--
Supporting Material:	--

6.3.2.1.4 [BSW09141] Configuration of physical channel wake-up prevention

Initiator:	WP4.2.2.1.9
Date:	18.05.2005
Short Description:	Configuration of physical channel wake-up prevention
Type:	--
Importance:	high
Description:	**SRS1280**: The Communication Manager shall be able to prevent wake-up at the level of physical channels (BSW09089). This feature shall be configurable at pre compile Time or Post build Time.
Rationale:	This feature will not appear in all ECUs.
Use Case:	--
Dependencies:	[BSW09089] Preventing waking up physical channels.
Conflicts:	--
Supporting Material:	--

6.3.2.2 Normal Operation

6.3.2.2.1 [BSW09078] Coordinating communication requests

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	Coordinating communication requests
Type:	--
Importance:	high
Description:	**SRS1281**: The Communication Manager 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 BSW09083 for a brief description of the communication modes).
Rationale:	Main functionality of Communication Manager. A SWC cannot know with which other SWC it will share an ECU in a particular configuration. Coordination of communication requests has to be provided by BSW.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.3.2.2.2 [BSW049] Initiating wake-up and keeping awake physical channels

Initiator:	DC
Date:	09.01.2004
Short Description:	Initiating wake-up and keeping awake physical channels
Type:	--
Importance:	high
Description:	**SRS1282** :The Communication Manager 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 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 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:	[BSW09087] Minimum duration of communication request after wakeup .
Conflicts:	--
Supporting Material:	--

6.3.2.2.3 [BSW09080] Physical channel independency

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	Physical channel independency
Type:	--
Importance:	high
Description:	**SRS1283:** The physical channels may have independent communication modes.
Rationale:	Possibility to have partial networks at physical channel level. Unnecessary physical channels shall not be activated.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.3.2.2.4 [BSW09081] API for requesting communication

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	API for requesting communication
Type:	--
Importance:	high
Description:	**SRS1284:** The Communication Manager shall provide an API allowing collecting communication requests.
Rationale:	Means of interaction necessary
Use Case:	--

Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.3.2.2.5 [BSW09083] Support of different communication modes

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	Support of different communication modes
Type:	--
Importance:	high
Description:	**SRS1285:** The Communication Manager 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:	--
Conflicts:	--
Supporting Material:	--

6.3.2.2.6 [BSW09084] API for querying the current communication mode

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	API for querying the current communication mode
Type:	--
Importance:	high
Description:	**SRS1286:** The Communication Manager shall provide an API which allows application to query the current communication mode.
Rationale:	SWCs 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 BSW09149).
Use Case:	--
Dependencies:	[BSW09149] API for querying the requested communication mode.
Conflicts:	--
Supporting Material:	--

6.3.2.2.7 [BSW09172] Evaluation of current communication mode

Initiator:	WPII-2.1.1
Date:	09/07/07
Short Description:	Evaluation of current communication mode
Type:	--
Importance:	High
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:	[BSW09084] API for querying the current communication mode [BSW09149] API for querying the requested communication mode.
Conflicts:	--
Supporting Material:	--

6.3.2.2.8 [BSW09149] API for querying the requested communication mode

Initiator:	PSA (P. Castelpietra)
Date:	28.07.2005
Short Description:	API for querying the requested communication mode
Type:	--
Importance:	high
Description:	**SRS1287:** The Communication Manager shall provide an API which allows application to query the requested (target) communication mode.
Rationale:	SWCs shall have the possibility to query the real mode because it is unknown if and when a requested mode is reached (see BSW09084). The querying of the requested mode is added for completeness.
Use Case:	--
Dependencies:	[BSW09084] API for querying the current communication mode
Conflicts:	--
Supporting Material:	--

6.3.2.2.9 [BSW09085] Indication of communication mode changes

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	Indication of communication mode changes
Type:	--
Importance:	high
Description:	**SRS1288:** The Communication Manager 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:	--
Conflicts:	--
Supporting Material:	--

6.3.2.2.10 [BSW09168] Pseudo-channel for local communication

Initiator:	WP4.2.1.1
Date:	31.07.2006
Short Description:	Pseudo-channel for local communication
Type:	--
Importance:	high
Description:	The Communication Manager shall support users that are connected to no physical channel. If one or more SWCs 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:	[BSW09090] User-to-channel relationship.
Conflicts:	--
Supporting Material:	--

6.3.2.2.11 BSW09071]Limit Communication Manager modes

Initiator:	DC
Date:	27.09.2004
Short Description:	Limit Communication Manager modes
Type:	--
Importance:	high
Description:	**SRS1289**: 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 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 BSW09083. 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:	[BSW09083] Support of different communication modes. [BSW09157] Revoke Communication Manager mode limitation.
Conflicts:	--
Supporting Material:	--

6.3.2.2.12 [BSW09157] Revoke Communication Manager mode limitation

Initiator:	PSA (P. Castelpietra)
Date:	04.10.2005
Short Description:	Revoke Communication Manager mode limitation
Type:	--
Importance:	high
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 BSW09071.
Use Case:	--
Dependencies:	[BSW09083] Support of different communication modes. [BSW09071] Limit Communication Manager modes.
Conflicts:	--
Supporting Material:	--

6.3.2.2.13 [BSW09087] Minimum duration of communication request after wakeup

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	Proxy communication request after wake-up
Type:	--
Importance:	high
Description:	**SRS1290:** The Communication Manager shall maintained 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 while waiting for NM messages from other ECUs.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

6.3.2.2.14 [BSW09089] Preventing waking up physical channels

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	Preventing waking up physical channels
Type:	--
Importance:	high
Description:	**SRS1292:** The Communication Manager 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:	[BSW09141] Configuration of physical channel wake-up prevention.
Conflicts:	--
Supporting Material:	--

6.3.2.2.15 [BSW09155] Counting of inhibited communication requests

Initiator:	DC, PSA (P. Castelpietra)
Date:	21.09.2005
Short Description	Counting of inhibited communication requests
Type:	--
Importance:	high
Description:	The Communication Manager 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:	[BSW09071] Limit Communication Manager modes. [BSW09089] Preventing waking up physical channels.
Conflicts:	--
Supporting Material:	--

6.3.2.2.16 [BSW09156] API to retrieve the number of inhibited "Full Communication" mode requests

Initiator:	DC, PSA (P. Castelpietra)
Date:	21.09.2005
Short Description	API to retrieve the number of inhibited "Full Communication" mode requests
Type:	--
Importance:	high
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 API.
Rationale:	--
Use Case:	It shall be possible to read out and reset the current status of the counter by a diagnostic service.
Dependencies:	[BSW09155] Counting of inhibited communication requests.
Conflicts:	--
Supporting Material:	--

6.3.3 Non-Functional Requirements (Qualities)

6.3.3.1.1 [BSW09079] Transparent relationship between software components and physical channels

Initiator:	BMW (Schlaugath)
Date:	01.12.2004
Short Description:	Transparent relationship between software components and physical channels
Type:	--
Importance:	high
Description:	**SRS1295:** SWCs shall be defined independently from the physical channels which are available on the ECUs. Thus, they shall not have any knowledge about these physical channels We call this a “transparent relationship” between application software components and physical channels.
Rationale:	Communication philosophy of AUTOSAR. Possibility to (almost) freely move SWCs on different ECUs connected to different physical channels.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

7 References

[1] Specification of Communication Manager
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_ComManager.pdf

[2] Specification of ECU State Manager
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_EcuStateManager.pdf

[3] Specification of Watchdog Manager
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_WatchdogManager.pdf

[4] Specification of Communication Stack Types r
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_ComStackTypes.pdf

[5] General Requirements on Basic Software Modules
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_General.pdf

[6] Layered Software Architecture
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_LayeredSoftwareArchitecture.pdf

[7] Specification of the Virtual Functional Bus
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_VirtualFunctionBus.pdf

[8] MetaModel
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_Metamodel.eap

[9] Software Component Template
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SoftwareComponentTemplate.pdf

[10] Specification of System Template
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SoftwareComponentTemplate.pdf

[11] Requirements on CAN
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_CAN.pdf

[12] Requirements on LIN
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_LIN.pdf

[13] Specification of Operating System
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_OS.pdf