Document Title	Requirements on MCU Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	195

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• No content changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• No content changes
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	• Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Removed references to HIS • Editorial changes

△

2016-11-30	4.3.0	AUTOSAR Release Management	• Added "Chapter 5 - Requirement Tracing" to trace against AUTOSAR features. • Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes
2013-03-15	4.1.1	AUTOSAR Administration	• Link Requirement with BSW Feature Document • Updating format of requirements according to FO_TPS_StandardizationTemplate [1]
2010-09-30	3.1.5	AUTOSAR Administration	• Lots requirements rephrased to make them atomic. • Debugging Concept inserted. • Insertion of a new service (Api) to read the Status after the reset.(Affected also SRS R4.0) • Insertion new configuration parameters to enable/disable PLL Apis. • Introduction of a new container to publish all the different resets that Micro Controller support. • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added
2006-11-28	2.1	AUTOSAR Administration	• Legal disclaimer revised
2006-05-16	2.0	AUTOSAR Administration	• Release as a separate document. The SRS SPAL V1.0.0 has been split into 12 independent documents for Release 2.0

▽

△

2005-05-31	1.0	AUTOSAR Administration	• 1.0.0 initial release

Disclaimer

Contents

1 Scope of Document

This document specifies requirements on the module MCU Driver.

Constraints

First scopes for specification of requirements on basic software modules are systems which are not safety relevant. For this reason safety requirements are assigned to medium priority.

2 Conventions to be used

2.1 Document Conventions

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([1]).

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([1]).

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 implemen-

tation, which does not include the option (except, of course, for the feature the option provides.)

2.2 Requirements structure

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

Functional Requirements:

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

Non-Functional Requirements:

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

3 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to MCUDriver that are not included in the AUTOSAR Glossary [2].

Abbreviation / Acronym:	Description:
CS	Chip select
DIO	Digital Input Output
ECU	Electric Control Unit
EOL	End Of Line Often used in the term 'EOL Programming' or 'EOL Configuration'
ICU	Interrupt Capture Unit
MAL	Old name of Microcontroller Abstraction Layer (replaced by MCAL because 'MAL' is a french term meaning 'bad')
MCAL	Microcontroller Abstraction Layer
MCU	Microcontroller Unit
MMU	Memory Management Unit
Master	A device controlling other devices (slaves, see below)
Slave	A device beeing completely controlled by a master device
NMI	Non maskable interrupt
OS	Operating System
PLL	Phase Locked Loop
PWM	Pulse Width Modulation
RX	Reception (in the context of bus communication)
SPAL	The name of this working group (Standard Peripheral Abstraction Layer)
SFR	Special Function Register
RTE	Runtime environment
WP	Work Package

Table 3.1: Acronyms and abbreviations used in the scope of this Document

Abbreviation:	*Description:*
STD	Standard
REQ	Requirement
UNINIT	Uninitialized (= not initialized)

As this is a document from professionals for professionals, all other terms are expected to be known.

4 Requirements Specification

This chapter describes all requirements driving the work to define the MCU Driver.

4.1 Functional Overview

The MCU Driver [**M**icro**c**ontroller **U**nit] provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required from other MCAL software modules. The initialization services allow a flexible and application related MCU initialization in addition to the start-up code (see figure below). The start-up code is very MCU specific. The provided start-up code description in this document is for guidance and implies functionality, which have to be taken into account before standardized MCU initialization is able to start.

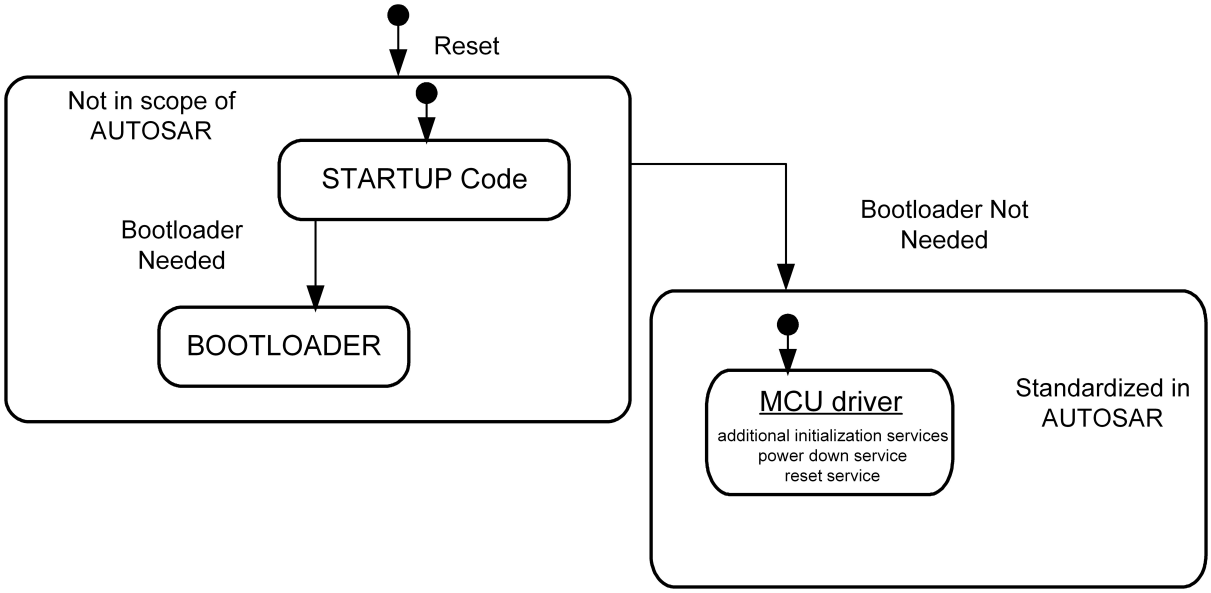

Figure 4.1

The MCU driver accesses the microcontroller hardware directly and is located in the Microcontroller Abstraction Layer (MCAL).

MCU driver Features:

- Describe set up required to configure a functionality that is not presently covered by another MCAL module e.g. global clock settings
- Set up off PLL and MCU clock distribution
- Service for RAM section initialization
- Setting of MCU dependent configuration control bits not covered by general SPAL requirements
- Activation of µC reduced power modes

- Perform a µC reset
- Get the reset reason from hardware

4.2 Functional Requirements

4.2.1 Configuration and Initialization

[SRS_Mcu_12421] Low Power Mode Configuration

Upstream requirements: RS_BRF_01184

⌈

Description:	The configuration setting of Low Power Modes is completely microcontroller specific. It shall be possible to configure the different modes supported by the hardware and required by the application.
Rationale:	Reduce power consumption of the MCU depending on application needs
Use Case:	–
Dependencies:	[SRS_Mcu_12268]
Supporting Material:	–

⌋

[SRS_Mcu_12350] The MCU Driver shall allow the static configuration of RAM segments that are initialized during start-up

Upstream requirements: RS_BRF_01096

⌈

Description:	The MCU Driver shall allow the static configuration of RAM segments that are initialized during start-up.
Rationale:	Allow to define which segments of RAM are initialized (zeroed-out) and which not.
Use Case:	Allow to save data in specific RAM segments over a reset.
Dependencies:	[SRS_Mcu_12331]
Supporting Material:	–

⌋

[SRS_Mcu_12331] The MCU Driver shall provide a service to initialize the contents of configured RAM sections

Upstream requirements: RS_BRF_01096

⌈

Description:	The MCU Driver shall provide a service to initialize the contents of configured RAM sections. RAM sections that are not configured to be initialized shall not be touched.
Rationale:	Defined RAM contents after ECU start-up.
Use Case:	It is used for flexible initialization of RAM sections with defined content. The ECU state manager can decide during ECU start-up, whether the initialization of some RAM section is required (e.g. depending on Reset reason).
Dependencies:	[SRS_Mcu_12350]
Supporting Material:	–

⌋

[SRS_Mcu_12392] The MCU driver shall provide a service to query the lock status of all PLLs in the micro controller individually

Upstream requirements: RS_BRF_01856

⌈

Description:	The MCU driver shall provide a service to query the lock status of all PLLs in the micro controller individually. This service shall return: • Locked • Un-Locked • Unsupported
Rationale:	–
Use Case:	To know the status of any PLL in the microcontroller.
Dependencies:	[SRS_Mcu_12208]
Supporting Material:	–

⌋

[SRS_Mcu_12336] The MCU Driver shall provide a service for activating the PLL clock distribution to the whole MCU

Upstream requirements: RS_BRF_01856

⌈

Description:	The MCU Driver shall provide a service for activating the PLL clock distribution to the whole MCU. This service is required if the PLL modules in the MCU provide a separate enable bit releasing the PLL clock. This service shall only be executed after the respective PLL is locked. This service shall, if supported, be provided for all the PLLs in the micro controller individually.
Rationale:	Some micro controllers have more than one PLL
Use Case:	Make the locked PLL clock active within the MCU.
Dependencies:	[SRS_Mcu_12208] [SRS_Mcu_12392]
Supporting Material:	–

⌋

[SRS_Mcu_12207] The MCU Driver shall configure the clock safety features

Upstream requirements: RS_BRF_01856

⌈

Description:	The MCU Driver shall configure the clock safety features if supported by HW like e.g.: • Loss of crystal detect enable/disable • Loss of crystal clock source (limp home mode) enable/disable • notification enable/disable on error detection
Rationale:	An example is limp mode where the loss of crystal allows a back up clock source to provide a mechanism for safe system shutdown
Use Case:	Loss of crystal recovery and orderly shutdown
Dependencies:	[SRS_Mcu_12208]
Supporting Material:	–

⌋

[SRS_Mcu_12208] The MCU Driver shall provide a service to initialize the clock system of the MCU

Upstream requirements: RS_BRF_01856

⌈

Description:	The MCU Driver shall provide a service to initialize the clock system of the MCU. This includes the initialization of the PLL factors, start PLL lock process (if selected) and other MCU specific clock options like for example clock prescalers that affect more than one driver. It is not mandatory to wait for the PLL lock.
Rationale:	–
Use Case:	Set appropriate clock speed for MCU sub systems for example after wake-up from MCU reduced power modes.
Dependencies:	[SRS_Mcu_12392]
Supporting Material:	–

⌋

4.2.2 Normal Operation

[SRS_Mcu_12000] The MCU Driver shall provide a service for querying the standardized reset reason

Upstream requirements: RS_BRF_01856

⌈

Description:	The MCU Driver shall provide a service for querying the reset reason. The following standardized reset reasons shall be distinguished (if supported by hardware): • Power On Reset (default return value) • External Hard Reset • Internal Watchdog Timer Reset • Other reset reasons
Rationale:	Different reset reasons may require different actions during the initialization phase.
Use Case:	Information for ECU state manager (e.g. decide what start-up sequence is chosen).
Dependencies:	–

▽

△

Supporting Material:	Note: The reset reasons listed above are not required to be implemented in each microcontroller device. If a microcontroller does not have the ability to distinguish between multiple reset reasons, the default value shall be "Power On Reset".

⌋

[SRS_Mcu_12215] The MCU driver shall provide a service that allows to query the raw reset status

Upstream requirements: RS_BRF_01856

⌈

Description:	The MCU driver shall provide a service that allows to query the reset reason. This service shall return the full, raw and µC specific reset information.
Rationale:	After full ECU start-up, the raw reset status can be queried and stored as reset information to the diagnostic error memory.
Use Case:	This information shall be used only to store reset information to the diagnostic error memory. Example for Reset Types: • Power On Reset • External Hard Reset • Soft Reset • Internal Watchdog Timer Reset • Debug System Reset • Reset caused by exception • ...
Dependencies:	–
Supporting Material:	If a microcontroller does not provide a reset status register, this service shall return 0 (zero).

⌋

[SRS_Mcu_12277] The MCU driver shall provide a reset trigger function

Upstream requirements: RS_BRF_01856

⌈

Description:	The MCU driver shall provide a reset trigger function using the features of the microcontroller hardware. As the MCU's provides different kind of reset variant, the configuration of the reset trigger shall be defined in the configuration structure of the MCU driver. If the microcontroller does not support methods for triggering a reset by software, this function shall not be used. In this case the upper layer is responsible to use other application specific methods for example to switch an I/O pin to trigger external reset circuit.
Rationale:	Force a controlled initialization of the microcontroller hardware if the software detects particular unknown system states.
Use Case:	Could be triggered in case of fatal errors, e.g. • if non-recoverable µC traps occur (e.g. bus error trap), • if SW statemachines suddenly encounter undefined states (e.g. due to bit errors in RAM) and no other reaction is possible
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mcu_13701] The MCU Driver shall provide a service for querying the RAM status

Upstream requirements: RS_BRF_00129

⌈

Description:	The MCU Driver shall provide a service for querying the RAM status. The following standardized RAM states shall be distinguished (if supported by hardware): • RAM state invalid [default] • RAM state valid If the hardware does not support this feature, this function shall be disabled.
Rationale:	Different RAM states may require different actions during the Initialization phase.
Use Case:	ECU State Manager can use this information to reload the Ram content after a reset.
Dependencies:	–

▽

△

Supporting Material:	Note: The RAM states listed above are not required to be implemented in each microcontroller device. If a microcontroller does not have the ability to distinguish between multiple RAM states, the default value shall be "RAM invalid"

⌋

4.2.3 Fault operation

[SRS_Mcu_12394] The MCU driver shall provide a notification of failure of the clock source

Upstream requirements: RS_BRF_01064, RS_BRF_02168

⌈

Description:	The MCU driver shall provide a notification in order to report failure conditions for the clock source when a failure has occurred with the clock generation in the MCU. The notification shall be provided if MCU is able to detect these kind of failures. As the Diagnostic Event Manager will interface to this function, the notification shall not be called during startup phase.
Rationale:	Once a fault is detected a choice of recovery may be desired.
Use Case:	Loss of crystal recovery and orderly shutdown
Dependencies:	–
Supporting Material:	–

⌋

4.2.4 Shutdown operation

[SRS_Mcu_12268] The MCU driver shall provide a service to activate MCU power saving modes of the µC

Upstream requirements: RS_BRF_01184

⌈

Description:	The MCU driver shall provide a service to activate MCU power saving modes of the µC.
Rationale:	–

▽

△

Use Case:	The upper layer intends to enter ECU reduced power mode. The upper layer is calling the MCU driver to activate the appropriate MCU setting.
Dependencies:	[SRS_Mcu_12421]
Supporting Material:	Note: The MCU modes Low Power Modes are not required to be implemented in each microcontroller device.

⌋

4.3 Remarks

This chapter [MCU driver] has many types of functionality gathered together to solve a problem of correct initialization following recovery from an MCU power saving mode or a reset.

A configuration tool is very important for the correct implementation of this functionality and may need to be part of the final solution.

5 Requirements Tracing

The following table references the features specified in [3] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[RS_BRF_00129]	AUTOSAR shall support data corruption detection and protection	[SRS_Mcu_13701]
[RS_BRF_01064]	AUTOSAR BSW shall provide callback functions in order to access upper layer modules	[SRS_Mcu_12394]
[RS_BRF_01096]	AUTOSAR shall support start-up and shutdown of ECUs	[SRS_Mcu_12331] [SRS_Mcu_12350]
[RS_BRF_01184]	AUTOSAR shall support different methods of degradation	[SRS_Mcu_12268] [SRS_Mcu_12421]
[RS_BRF_01856]	AUTOSAR microcontroller abstraction shall provide access to internal MCU configuration	[SRS_Mcu_12000] [SRS_Mcu_12207] [SRS_Mcu_12208] [SRS_Mcu_12215] [SRS_Mcu_12277] [SRS_Mcu_12336] [SRS_Mcu_12392]
[RS_BRF_02168]	AUTOSAR diagnostics shall provide a central classification and handling of abnormal operative conditions	[SRS_Mcu_12394]

Table 5.1: Requirements Tracing

6 References

[1] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Requirements on AUTOSAR Features
AUTOSAR_CP_RS_Features