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

Document Version	2.0.3
Document Status	Final
Part of Release	3.0
Revision	0001

Document Change History

Date	Version	Changed by	Change Description
31.10.2007	2.0.3	AUTOSAR Administration	• Document meta information extended • Small layout adaptations made
24.01.2007	2.0.2	AUTOSAR Administration	• “Advice for users” revised • “Revision Information” added
28.11.2006	2.0.1	AUTOSAR Administration	Legal disclaimer revised
06.04.2006	2.0.0	AUTOSAR Administration	Release as a separate document. The SRS SPAL V1.0.0 has been split into 12 independent documents for Release 2.0
02.06.2005	1.0.0	AUTOSAR Administration	1.0.0 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 this document

This document specifies requirements on the module MCU Driver.

Constraints

First scope 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 How to read this document

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

2.1 Conventions used

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

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

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

2.2 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

Acronyms and abbreviations that have a local scope are not contained in the AUTOSAR glossary. These must appear in a local glossary.

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’
HIS	Herstellerinitiative Software
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

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

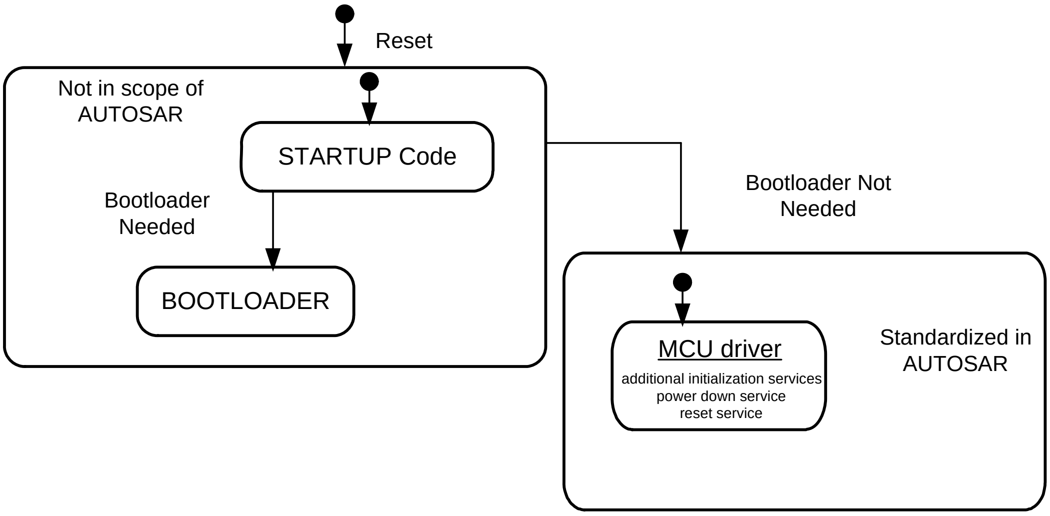
As this is a document from professionals for professionals, all other terms are expected to be known.

4 Requirement Specification

4.1 MCU Driver

4.1.1 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.



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

4.1.2 Functional Requirements

4.1.2.1 Configuration and Initialization

4.1.2.1.1 [BSW12421] Low Power Mode Configuration

Initiator:	NEC
Date:	24.11.2004
Short Description:	Low Power Mode
Type:	New
Importance:	High
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:	[BSW12268] MCU Power Management Control
Conflicts:	--
Supporting Material:	--

4.1.2.1.2 [BSW12350] Configuration of RAM segments

Initiator:	WP4.2.2.1.12
Date:	12.10.2004
Short Description:	Configuration of RAM segments
Type:	New
Importance:	High
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:	[BSW12331] RAM Initialization
Conflicts:	--
Supporting Material:	--

4.1.2.1.3 [BSW12331] RAM Initialization

Initiator:	WP4.2.2.1.12
Date:	09.11.2004
Short Description:	RAM Initialization
Type:	Changed (during joint SPAL review)
Importance:	High
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:	[BSW12057] Driver module initialization [BSW12350] Configuration of RAM segments
Conflicts:	--
Supporting Material:	--

4.1.2.1.4 [BSW12392] Provide lock status of PLL

Initiator:	WP4.2.2.1.12
Date:	15.03.2005
Short Description:	Provide lock status of PLL
Type:	Changed
Importance:	High
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:	[BSW12208] Initialization of the MCU clock
Conflicts:	--
Supporting Material:	--

4.1.2.1.5 [BSW12336] Activate PLL Clock distribution

Initiator:	WP4.2.2.1.12
Date:	15.03.2005
Short Description:	Activate PLL Clock distribution
Type:	Changed (during joint SPAL review)
Importance:	High
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:	[BSW12208] Initialization of MCU Clock; [BSW12392] Provide lock status of PLL
Conflicts:	--
Supporting Material:	--

4.1.2.1.6 [BSW12207] Configuration of clock safety features

Initiator:	Freescale
Date:	17.08.2004
Short Description:	Configuration of clock safety features

Type:	Modified (during joint SPAL review)
Importance:	Medium
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:	[BSW12208] Initialization of MCU Clock [BSW12393] Start-up code
Conflicts:	--
Supporting Material:	--

4.1.2.1.7 [BSW12208] Initialization of MCU Clock

Initiator:	WP4.2.2.1.12
Date:	09.11.2004
Short Description:	Initialization of MCU Clock
Type:	Changed (during joint SPAL review – previous name of the requirement was initialization of global clock sources)
Importance:	High
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:	[BSW12392] Provide lock status of PLL
Conflicts:	--
Supporting Material:	--

4.1.2.2 Normal Operation

4.1.2.2.1 [BSW12000] Provide standardized reset reason

Initiator:	WP4.2.2.1.12
Date:	14.09.2004
Short Description:	Provide standardized reset reason
Type:	New
Importance:	High
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:	--
Conflicts:	--
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”.

4.1.2.2.2 [BSW12215] Provide raw reset status

Initiator:	BOSCH
Date:	27.06.2004
Short Description:	Provide raw reset status
Type:	Changed (during review on 14.09.2004)
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	If a microcontroller does not provide a reset status register, this service shall return 0 (zero).

4.1.2.2.3 [BSW12277] Reset trigger function

Initiator:	BOSCH
Date:	25.11.2004
Short Description:	Reset trigger function
Type:	Changed (use case changed)
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	--

4.1.2.3 Fault operation

4.1.2.3.1 [BSW12394] Fault condition handling of clock safety features

Initiator:	Freescale
Date:	23.10.2004
Short Description:	Fault condition handling of clock safety features
Type:	Changed (13.04.2005: not to be called during start-up phase)
Importance:	Medium
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.
Rationale:	Once a fault is detected a choice of recovery may be desired.
Use Case:	Loss of crystal recovery and orderly shutdown
Dependencies:	As the Diagnostic Event Manager will interface to this function, this notification shall not be called during the start-up phase.
Conflicts:	--
Supporting Material:	--

4.1.2.4 Shutdown operation

4.1.2.4.1 [BSW12268] MCU Power Management Control

Initiator:	Infineon
Date:	24.11.2004
Short Description:	MCU Power Management Control
Type:	Changed (modified description and excluded mode description)
Importance:	High
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:	[BSW12421] Low Power Mode Configuration
Conflicts:	--
Supporting Material:	Note: The MCU modes Low Power Modes are not required to be implemented in each microcontroller device.

4.1.3 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 References

5.1 Deliverables of AUTOSAR

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

[AUTOSAR_GLOSSARY] Glossary,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_Glossary.pdf

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

[SRS_BSW_SPAL] General Requirements on SPAL,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_SPAL_General.pdf

5.2 Related standards and norms

[STD_HIS_IO_DRIVER] HIS API IO Driver, V2.1.3, April 29th, 2004,
https:/svn2.autosar.org/repos2/22_Releases
API_IODriver_2_1_3.pdf