Document Title	Requirements on I/O Hardware Abstraction
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	75
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R20-11

Document Change History

Date	Release	Changed by	Change Description
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 related standards and norms • Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Requirement Tracing section added
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	• Formal rework of requirement tracing • Added three power state transitions. • Incorporated requirements for ECU degradation • Updated according to TPS_standardization templates

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2010-09-30	3.1.5	AUTOSAR Administration	• Uniqueness restriction of the module has been removed • New feature: 'Functional Diagnostics' interface • 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	• Initial release

Disclaimer

Table of Contents

1 Scope of this document 5

2 How to read this document 6

2.1 Conventions used 6
2.2 Requirement structure 7

3 Acronyms and abbreviations 8

3.1 Expressions - general 8
3.2 Expressions - signal attributes 8

4 Functional Overview 10

4.1 IO Hardware Abstraction 10
4.2 Overview of Attributes to qualify Signals 11

5 Requirements Tracing 12

6 Requirement Specification 14

6.1 Functional Requirements 14
6.2 Non-Functional Requirements (Qualities) 28

7 References 29

7.1 Deliverables of AUTOSAR 29

1 Scope of this document

This document defines general rules and formats for requirements specification within AUTOSAR. It shall be used as a basis for each requirements document.

The requirements are structured in the following way:

- General requirements on Basic Software Modules (other document)
- General requirements which apply to all modules of the Microcontroller and ECU Abstraction Layers (this document)
- Module specific requirements (this document)

Constraints

First scope for specification of requirements on basic software module 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 and/or questions, please refer to this unique ID rather than chapter or page numbers!

2.1 Conventions used

- The representation of requirements in AUTOSAR documents follows the table specified in [2].

- 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.)

2.2 Requirement structure

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

Functional Requirements:
- Configuration (which elements of the module need to be configurable)
- Initialization
- 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

3.1 Expressions - general

Expression	*Description*	*Example*
Class	A class represents a kind of electrical connection to the ECU. It could be for example an analogue, a discrete,…	Analogue Class, Discrete Class…
Electrical Signal	It is the electrical signal on the pin of the ECU	Physical input voltage at an ECU-Pin
ECU pin	It is an hardware electrical connection of the ECU with the rest of the electronic system	
ECU Signals	It is the **software representation** of an electrical signal. A signal has attributes and a symbolic name	Input voltage,Discrete Output, PWM Input …
ECU Signal group	It is the **software representation** of a group of electrical signals from the same Class	Only for discrete Inputs and discrete Outputs
Attributes	Characteristics that can be Software (SW) and Hardware (HW) for each kind of Signals existing in a ECU	Range, Lifetime / delay, …
Symbolic name	The symbolic name of a signal is used by the IO Hardware Abstraction module to make a link (function, pin)	

3.2 Expressions - signal attributes

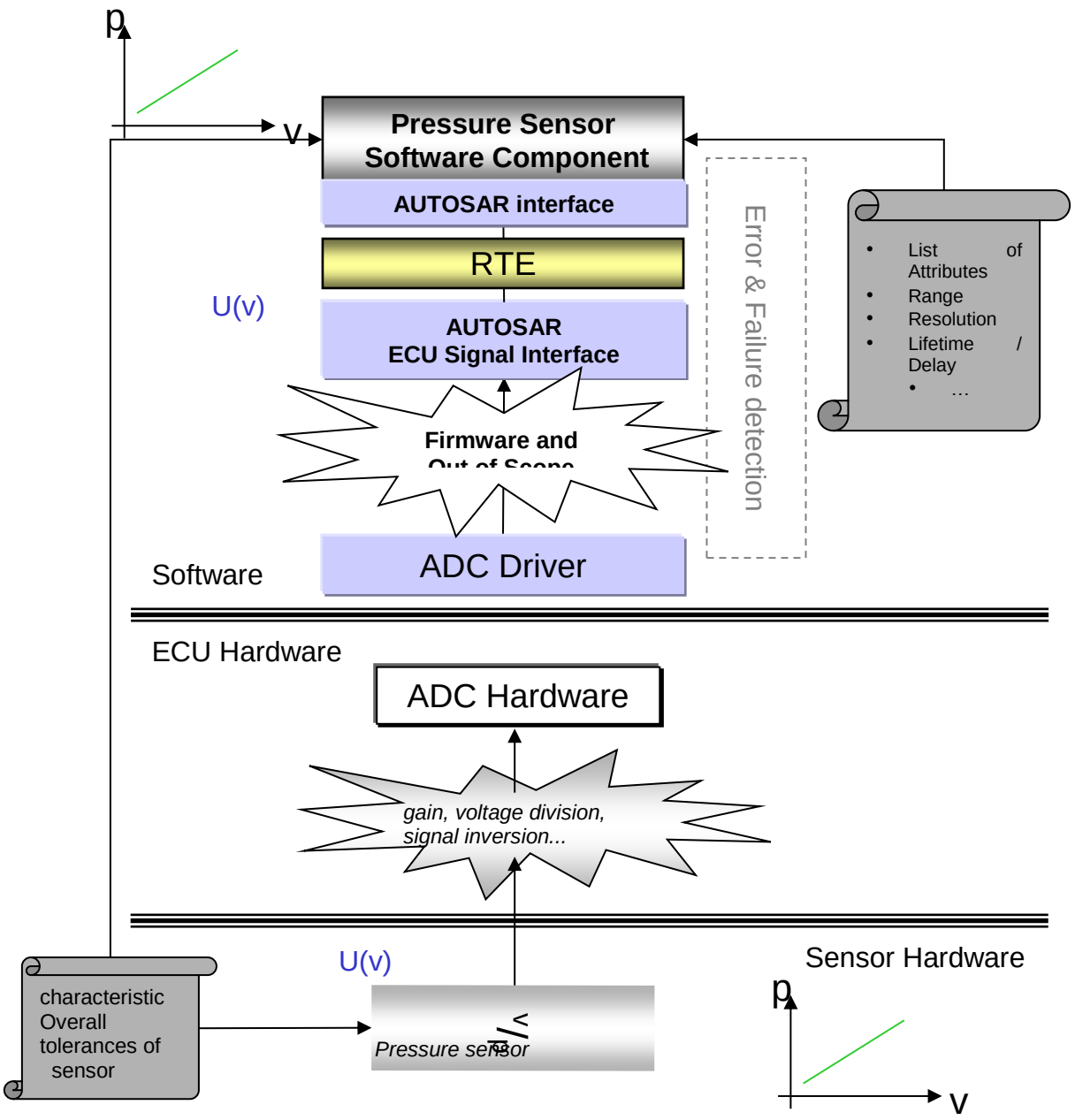
Expression	*Description*	*Example*
Data Type	Analogue: Datatype of the signal Discrete: either bool or AUTOSAR defined type (BoolType)	(VoltageType, CurrentType, ResistanceType, BoolType) Each DataType has a given size: 16 bits or 32 bits
Range	This is a functional range and not an electrical range.) For analogue signals [lowerLimit...upperLimit] (Voltage, current), [0...upperLimit] (resistance) For discrete signals [0,1] For timing signals [0…upperLimit] (period), [-100…100%] (Duty Cycle)	[-12Volts...+12Volts] (voltage)
Resolution	This attribute for many Classes is dependent on the range and the Data Type. Example: (upperLimit - lowerLimit) / ($2^{datatypelength}$ -1) For the others is known and defined.	$Voltage_{min}$ = -12 Volts $Voltage_{max}$ = 12 Volts Data Type : 16 bits Resolution => 24 / 65535
Hardware Resolution	This is the maximum possible resolution of the hardware (ADC)	ADC converter could have a 8/10/12/16 bits resolution
Hardware Accuracy	This is the accuracy of Hardware. It depends on hardware peripheral used for acquisition and/or generation	ADC converter could have an accuracy of +-3LSB
Accuracy	It depends of hardware peripheral used for acquisition and/or generation.	ADC converter could be a a 8/10/12/16 bits converter

Expression	*Description*	*Example*
Diagnosis	Diagnosis capability of the functionality	Diagnosis Not Supported (could be a static check) No valid information available Short to Power Supply Short to Ground Open Load Over Temperature Diagnosis OK
Synchronization	A signal could be synchronize with another signal or with an event like a trigger	If a discrete signal is “TRUE”, acquire an analogue signal
Access	Defines if the Signal is attached to a Get(Read) / Set(Write) feature.	
Inversion	Inversion between the physical value and the logical value. This attribute is not visible and not configurable by users of IO Hardware Abstraction.	Physical HighState → (Signal=False) Physical LowState → (Signal=True)
Lifetime	Only for **Inputs**: It is the maximum allowed age of the data (time is in microseconds). If Lifetime is 0, then the signal is directly get from the register.	Lifetime = 0 is a direct access Lifetime = 1000µs the value read is at maximum 1ms older
Delay	Only for **Outputs**: It is the maximum allowed time until an output is actually set (time is in microseconds) If Delay is 0, then the signal is set immediately	Delay = 0 is a direct access Delay = 100µs the command is set until 100µs elapse
Filtering / Debouncing	It defines if the Signal is provided as a raw value or if a filtering/debouncing method is included in the IO Hardware Abstraction module for this Signal.	Raw, Debounce 3 Samples, Wait 10ms,
Sampling Rate	Time period required to get a Signal value.	Sampling rate for a sampling windows (burst)
Report Changes	This attribute is only applicable to Discrete Inputs. It defines the capability (or not) of reporting level changes.	Enable or Disable
Pulse Test	This attribute means that the output shall be tested thanks a dedicated pulse. If this attribute is not set, diagnosis will be done while using the output.	Available or non available

4 Functional Overview

4.1 IO Hardware Abstraction

The IO Hardware Abstraction module abstracts from the signal path of the ECU hardware (Layout, Microcontroller Pins, Microcontroller external devices like IO ASIC). It provides a signal based interface to the upper software layer. It performs static abstraction and inversion (if needed) of values according to their physical representation at the inputs/outputs of the ECU hardware (compensation of static influences caused within the path between ECU IO and Microcontroller pin, e.g. voltage divider, hardware inversion).



The IO Hardware Abstraction module allows configuring each signal according to an attributes list. Interfaces are AUTOSAR Standard.

4.2 Overview of Attributes to qualify Signals

Signal \ Attributes	Signal Data Type	Access	BSW-Range	Unit	BSW-Resolution	Failure Monitoring	Age (Lifetime/Delay)	Filtering / Debouncing	Sampling Rate	Report Feature	Pulse Test	Wakeup
Analogue$_{in}$	X	X	X	X	X	-	Xl	X	X	O	-	-
Analogue$_{out}$	X	X	X	X	X	O	Xd	-	-	-	O	-
Discrete$_{in}$	X	X	F	-	-	-	Xl	X	X	O	-	O
Discrete$_{Status}$	X	-	F	-	-	-	Xl	X	-	-	-	O
Discrete$_{pow}$	X	X	F	-	-	O	Xd	-	-	-	O	-
PWx Period$_{in}$	X	X	X	X	X	-	Xl	-	-	-	-	O
PWx Period$_{out}$	X	X	X	X	X	O	Xd	-	-	-	O	-
PWx Duty Cycle$_{in}$	X	X	F	F	X	-	Xl	-	-	-	-	O
PWx Duty Cycle$_{out}$	X	X	F	F	X	O	Xd	-	-	-	O	-

Table legend

- **X** means the Attribute is applicable to this Class of ECU Signal and shall be configured.
  - **Xl** means the Age Attribute applicable is the Lifetime.
  - **Xd** means the Age Attribute applicable is the Delay.

- **F** means the Attribute is applicable to this Class of ECU Signal but it is a fixed standard value.

- **O** means the Attribute is optional to this Class of ECU Signal and depends on a static configuration (disable/enable).

- **-** means the Attribute is not applicable or has no meaning to this Class of ECU Signal.

5 Requirements Tracing

Requirement	Description	Satisfied by
RS_BRF_01000	AUTOSAR architecture shall organize the BSW in a hardware independent and a hardware dependent layer	SRS_IoHwAb_12319
RS_BRF_01024	AUTOSAR shall provide naming rules for public symbols	SRS_IoHwAb_12232
RS_BRF_01048	AUTOSAR module design shall support modules to cooperate in a multitasking environment	SRS_IoHwAb_12449
RS_BRF_01056	AUTOSAR BSW modules shall provide standardized interfaces	SRS_IoHwAb_12452
RS_BRF_01080	AUTOSAR shall allow access to internal and external peripheral devices	SRS_IoHwAb_12242
RS_BRF_01184	AUTOSAR shall support different methods of degradation	SRS_IoHwAb_12453, SRS_IoHwAb_12454, SRS_IoHwAb_12455
RS_BRF_01352	AUTOSAR RTE shall offer direct read/write data access, and alternatively pre-read data before a runnable is called and post-write data after the runnable returns	SRS_IoHwAb_00002, SRS_IoHwAb_12324, SRS_IoHwAb_12338, SRS_IoHwAb_12410, SRS_IoHwAb_12411, SRS_IoHwAb_12412, SRS_IoHwAb_12413, SRS_IoHwAb_12415
RS_BRF_01384	AUTOSAR RTE shall support automatic range checks of data	SRS_IoHwAb_12409
RS_BRF_01632	AUTOSAR communication shall support data consistency of groups of signals	SRS_IoHwAb_12323
RS_BRF_01856	AUTOSAR microcontroller abstraction shall provide access to internal MCU configuration	SRS_IoHwAb_12338
RS_BRF_01952	AUTOSAR IO Hardware Abstraction shall support standardized modes for connected I/O devices	SRS_IoHwAb_12453, SRS_IoHwAb_12454, SRS_IoHwAb_12455
RS_BRF_01968	AUTOSAR IO Hardware Abstraction shall support edge triggered I/O signals	SRS_IoHwAb_12414, SRS_IoHwAb_12416, SRS_IoHwAb_12417, SRS_IoHwAb_12445
RS_BRF_02000	AUTOSAR IO Hardware Abstraction shall protect hardware against illegal operation	SRS_IoHwAb_12419, SRS_IoHwAb_13900, SRS_IoHwAb_13901, SRS_IoHwAb_13902
RS_BRF_02016	AUTOSAR shall provide mechanisms to protect the system from unauthorized modification	SRS_IoHwAb_12451
RS_BRF_02024	AUTOSAR shall provide mechanisms to protect the system from unauthorized use	SRS_IoHwAb_12248
RS_BRF_02144	AUTOSAR diagnostic shall provide standardized diagnostic services for external testers	SRS_IoHwAb_00002

RS_BRF_02160	AUTOSAR diagnostic shall allow external testers to control active functionality of the ECU	SRS_IoHwAb_12418
RS_BRF_02168	AUTOSAR diagnostics shall provide a central classification and handling of abnormal operative conditions	SRS_IoHwAb_12339, SRS_IoHwAb_13900, SRS_IoHwAb_13901, SRS_IoHwAb_13902, SRS_IoHwAb_13904
RS_BRF_02176	AUTOSAR error handling shall distinguish between defined abnormal operative conditions and unexpected exceptions from intended behavior	SRS_IoHwAb_13903
RS_BRF_02224	AUTOSAR shall support run-time hardware tests	SRS_IoHwAb_12452
RS_BRF_02272	AUTOSAR shall offer tracing of application software behavior	SRS_IoHwAb_12450

6 Requirement Specification

6.1 Functional Requirements

6.1.1 IO Hardware Abstraction

6.1.1.1 General

6.1.1.1.1 [SRS_IoHwAb_12409] The IO Hardware Abstraction module shall provide values within one static range for each Signal

⌈

Type:	Valid
Description:	The IO Hardware Abstraction module shall provide values within one static range for each Signal. This range is independent of the basic software driver scaling factor. Examples: Analogue Signals => [lowerLimit...upperLimit] with (lowerLimit = - UpperLimit or with (lowerLimit = 0)
Rationale:	Support of a wide range with a high resolution
Use Case:	--
Dependencies:	--
Supporting Material:	--

⌋(RS_BRF_01384)

6.1.1.1.2 [SRS_IoHwAb_12410] The IO Hardware Abstraction module shall provide a service to read an input voltage with specific attributes

⌈

Type:	Valid
Description:	The IO Hardware Abstraction module shall provide a service to read an input voltage with these attributes: • DataType: VoltageType, • Range: [lowerLimit...upperLimit], lowerLimit and upperLimit can be negative ([-5Volts, -3Volts] • Resolution: VoltageType / (upperLimit - lowerLimit) • Accuracy: HW delivers • Synchronization: Yes / No • Lifetime: x = delayed of x microseconds • Filtering/Debouncing: raw, filtered (bandwith, corner frequency) • Sampling Rate: x: sample every x µs
Rationale:	Basic functionality of IO Hardware Abstraction
Use Case:	To control a component / sensor responding with voltage.
Dependencies:	--
Supporting Material:	--

⌋(RS_BRF_01352)

6.1.1.1.3 [SRS_IoHwAb_12411] The IO Hardware Abstraction module shall provide a service to read an output voltage with specific attributes

⌈

Type:	Valid

Description:	The IO Hardware Abstraction module shall provide a service to control a output voltage with these attributes:
	• DataType: VoltageType
	• Range: [lowerLimit...upperLimit], lowerLimit can be negative
	• Resolution: VoltageType / (upperLimit - lowerLimit)
	• Accuracy: HW delivers
	• Diagnosis: The IO Hardware Abstraction module is able to detect the following failures:
	  ○ Diagnosis Not Supported (could be a static check)
	  ○ No valid information available
	  ○ Short to Power Supply
	  ○ Short to Ground
	  ○ Open Load
	  ○ Over Temperature
	  ○ Diagnosis OK
	• Synchronization: Yes / No
	• Delay: x = delayed of x microseconds

Rationale:	Basic functionality
Use Case:	ECU supply Abstract the generation of an analogue signal by usage of PWM
Dependencies:	--
Supporting Material:	--

](RS_BRF_01352)

6.1.1.1.4 [SRS_IoHwAb_12413] The IO Hardware Abstraction module shall provide a service to read an input current with specific attributes

[

Type:	Valid
Description:	The IO Hardware Abstraction module shall provide a service to read an input current with these attributes: • DataType: CurrentType • Range: [lowerLimit…upperLimit], lowerLimit can be negative • Resolution: CurrentType / (upperLimit - lowerLimit) • Accuracy: HW delivers • Synchronization: Yes / No • Lifetime: x = delayed of x microseconds • Filtering/Debouncing: raw, filtered (bandwith, corner frequency) • Sampling Rate: x: sample every x µs
Rationale:	Basic functionality of IO Hardware Abstraction
Use Case:	To control a component / sensor driven by current.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01352)

6.1.1.1.5 [SRS_IoHwAb_12415] The IO Hardware Abstraction module shall provide a service to measure a connected resistance using specific attributes

[

Type:	Valid
Description:	The IO Hardware Abstraction module shall provide a service to measure a connected resistance using these attributes: • DataType: ResistanceType • Range: [lowerLimit…upperLimit], lowerLimit is null or positive • Resolution: ResistanceType / (upperLimit - lowerLimit) • Accuracy: HW delivers • Synchronization: Yes / No • Lifetime: x = delayed of x microseconds • Filtering/Debouncing: raw, filtered (bandwith, corner frequency) • Sampling Rate: x: sample every x µs
Rationale:	Basic functionality of IO Hardware Abstraction
Use Case:	Measurement of temperature sensor
Dependencies:	--
Supporting Material:	--

](RS_BRF_01352)

6.1.1.1.6 [SRS_IoHwAb_12412] The IO Hardware Abstraction module shall provide a service to get/read a discrete input with specific attributes

[

Type:	Valid
Description:	The IO Hardware Abstraction module shall provide a service to get/read a discrete input with these attributes: • DataType: boolean • Range: 0 or 1 • Resolution: Logical State • Accuracy: 1 bit • Synchronization: Yes / No • Lifetime: x = delayed of x microseconds • Filtering/Debouncing: raw, filtered (bandwith, corner frequency) • Sampling Rate: x: sample every x µs • Change Reporting: Enable or Disable
Rationale:	Basic functionality of IO Hardware Abstraction
Use Case:	Get/Read a logical value (0 / 1) from a ECU pin
Dependencies:	--
Supporting Material:	--

](RS_BRF_01352)

6.1.1.1.7 [SRS_IoHwAb_12324] The IO Hardware Abstraction module shall offer a service to Get / Read simultaneously several discrete inputs

[

Type:	Valid
Description:	The AUTOSAR IO Hardware Abstraction module shall offer a service to Get / Read simultaneously several discrete inputs. The number of inputs shall be configurable. This is limited to a physical portt.
Rationale:	All inputs belong to the same functionality or the same enhanced onboard chip.
Use Case:	For motor control but also for runtime optimization
Dependencies:	--

Supporting Material:	--

](RS_BRF_01352)

6.1.1.1.8 [SRS_IoHwAb_12450] The IO Hardware Abstraction module shall provide a mechanism to report to the Signal Client, when a discrete input changes

[

Type:	Valid
Description:	The IO Hardware Abstraction module shall provide a mechanism to report to the Signal Client, when a discrete input changes. This functionality is only available in case of Change Reporting attribute is enabled for this Signal.
Rationale:	To ensure a real time behavior
Use Case:	To detect a sensor activity and react in time for example about wiping and brake pedal.
Dependencies:	--
Supporting Material:	--

](RS_BRF_02272)

6.1.1.1.9 [SRS_IoHwAb_12419] The IO Hardware Abstraction module shall provide a service to monitor hardware failure and to set a status

[

Type:	Valid

Description:	The IO Hardware Abstraction module shall provide a service to monitor hardware failure and to set a status:
	• DataType: StatusType
	• Range:
	  ○ No valid information available
	  ○ Short to Power Supply
	  ○ Short to Ground
	  ○ Open Load
	  ○ Over Temperature
	  ○ Diagnosis OK
	• Synchronization: Yes / No
	• Lifetime: x = delayed of x microseconds
	• Filtering/Debouncing: raw, filtered (bandwith, corner frequency)

Rationale:	Basic functionality
Use Case:	Know the actual relay/lamp output state
Dependencies:	--
Supporting Material:	--

](RS_BRF_02000)

6.1.1.1.10 [SRS_IoHwAb_12418] The IO Hardware Abstraction module shall provide a service to control a discrete powered output with specific attributes

[

Type:	Valid

Description:	The IO Hardware Abstraction module shall provide a service to control a discrete powered output with these attributes:
	• DataType: boolean
	• Range: 0 or 1
	• Resolution: Logical State
	• Accuracy: 1 bit
	• Diagnosis: The IO Hardware Abstraction module is able to detect the following failures:
	  ○ Diagnosis Not Supported (could be a static check)
	  ○ No valid information available
	  ○ Short to Power Supply
	  ○ Short to Ground
	  ○ Open Load
	  ○ Over Temperature
	  ○ Diagnosis OK
	• Synchronization: Yes / No
	• Delay: x = delayed of x microseconds
	
	A simple Output (without powered) is a subset of Powered Outputs with Diagnosis attribute always as “Diagnosis Not Supported (could be a static check)”

Rationale:	Basic functionality
Use Case:	Relay control, lamp control
Dependencies:	--
Supporting Material:	--

](RS_BRF_02160)

6.1.1.1.11 [SRS_IoHwAb_12323] The IO Hardware Abstraction module shall offer a service to update simultaneously several discrete outputs

[

Type:	Valid
Description:	The AUTOSAR IO Hardware Abstraction module shall offer a service to update simultaneously several discrete outputs. The number of outputs shall be configurable. This is limited to a physical port.
Rationale:	All outputs belong to the same functionality or the same enhanced onboard chip.
Use Case:	For synchronization and for runtime optimization
Dependencies:	--
Supporting Material:	--

](RS_BRF_01632)

6.1.1.1.12 [SRS_IoHwAb_12417] The IO Hardware Abstraction module shall provide a service to measure the period time between two falling or rising edges on a Signal

[

Type:	Valid
Description:	The IO Hardware Abstraction module shall provide a service to measure the period time between two falling or rising edges on a Signal using these attributes: • DataType: PeriodType

	• Range: [lowerLimit…upperLimit], lowerLimit is null or positive • Resolution: PeriodType / (upperLimit - lowerLimit) • Accuracy: HW delivers • Lifetime: x = delayed of x microseconds
Rationale:	Basic functionality of IO Hardware Abstraction
Use Case:	Measure the period of a PWM sensor
Dependencies:	--
Supporting Material:	--

](RS_BRF_01968)

6.1.1.1.13 [SRS_IoHwAb_12416] The IO Hardware Abstraction module shall provide a service to control the period time between two falling or rising edges on a Signal using

[

Type:	Valid

Description:	The IO Hardware Abstraction module shall provide a service to control the period time between two falling or rising edges on a Signal using these attributes:
	• DataType: PeriodType
	• Range: [lowerLimit…upperLimit], lowerLimit is null or positive
	• Resolution: PeriodType / (upperLimit - lowerLimit)
	• Accuracy: HW delivers
	• Diagnosis:
	  ○ Diagnosis Not Supported (could be a static check)
	  ○ No valid information available
	  ○ Short to Power Supply
	  ○ Short to Ground
	  ○ Open Load
	  ○ Over Temperature
	  ○ Diagnosis OK
	• Synchronization: Yes / No
	• Delay: x = delayed of x microseconds

Rationale:	Basic functionality of IO Hardware Abstraction
Use Case:	Control the period of a PWM output signal
Dependencies:	--
Supporting Material:	--

](RS_BRF_01968)

6.1.1.1.14 [SRS_IoHwAb_12414] The IO Hardware Abstraction module shall provide a service to control the ratio between the active level and the inactive level of a periodic output signal

[

Type:	Valid
Description:	The IO Hardware Abstraction module shall provide a service to control the ratio between the active level and the inactive level of a periodic output Signal using these attributes: • DataType: DutyCycleType • Range: [-100%…+100%] • Resolution: to be defined • Accuracy: HW delivers

	• Synchronization: Yes / No • Delay: x = delayed of x microseconds
Rationale:	Basic functionality of IO Hardware Abstraction
Use Case:	A negative range is justified to allow for instance a motor direction control on the level of the hardware (negative duty cycle = to the left, positive duty cycle = to the right)
Dependencies:	--
Supporting Material:	--

](RS_BRF_01968)

6.1.1.1.15 [SRS_IoHwAb_12445] The IO Hardware Abstraction module shall provide a service to measure the ratio between the active level and the inactive level of a periodic input signal

[

Type:	Valid
Description:	The IO Hardware Abstraction module shall provide a service to measure the ratio between the active level and the inactive level of a periodic input Signal using these attributes: • DataType: DutyCycleType • Range: [-100%…+100%] • Resolution: 100 / M • Accuracy: HW delivers • Lifetime: x = delayed of x microseconds
Rationale:	Basic functionality of IO Hardware Abstraction
Use Case:	ICU requirement
Dependencies:	--
Supporting Material:	--

](RS_BRF_01968)

6.1.1.1.16 [SRS_IoHwAb_12338] The IO Hardware Abstraction shall provide synchronous signal access functions

[

Type:	Valid
Description:	The IO Hardware Abstraction shall provide synchronous signal access functions (signal access) even if the sampling is asynchronous.
Rationale:	Abstraction of different mechanisms.
Use Case:	Access to the buffer of a cyclic ADC conversion.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01352, RS_BRF_01856)

6.1.1.1.17 [SRS_IoHwAb_00002] The I/O Hardware Abstraction shall provide an interface to the DCM that allows to control and read the configured signals

[

Type:	Valid

Description:	The I/O Hardware Abstraction shall provide an interface to the DCM that allows to control and read the configured signals. The interface shall implement the following functionalities:
	
	• control the signals by setting the following states:
	  ○ IOHWAB_CONTROLTOECU: Unlock the signal
	  ○ IOHWAB_RESETTODEFAULT: Lock the signal and set it to a configured default value
	  ○ IOHWAB_FREEZE: Lock the signal to the current value
	  ○ IOHWAB_ADJUSTMENT: Lock the signal and adjust it to a value given by the DCM module
	• read the signals (in any signal state set by the 'control-functionality')
	
	Locking a signal means, that the certain signal is software-locked towards the SW-C, i.e. the SW-C's requests have no effect on the hardware in the locked state.
	Nevertheless, the DCM shall have full access to the hardware. In case C/S-communication is used for input signals, it might be necessary to have a IoHwAb-internal buffer, whose value can be adjusted by the DCM.

Rationale:	Diagnosis of I/O signals via DCM
Use Case:	System Diagnosis
Dependencies:	--
Supporting Material:	--

](RS_BRF_01352,RS_BRF_02144)

6.1.1.2 Configuration

6.1.1.2.1 [SRS_IoHwAb_12232] A Symbolic Name for each Signal shall be unique

[

Type:	Valid
Description:	The IO Hardware Abstraction shall allow to configure statically an unique Symbolic Name for each Signal.
Rationale:	No change in the upper layers when changing the hardware allocation.
Use Case:	Flexible ECU-Design
Dependencies:	--
Supporting Material:	--

](RS_BRF_01024)

6.1.1.2.2 [SRS_IoHwAb_12319] The IO Hardware Abstraction module shall be independent of physical level

[

Type:	Valid
Description:	The AUTOSAR IO Hardware Abstraction module shall be independent of physical level (i.e. : output command active at low state or high state) and provide only logical level to the upper layer for digital IO.
Rationale:	Independency between users and hardware design

Use Case:	For instance, doors could be OPEN / CLOSE and theses states are independent of real hardware input status (0v, 5v, 12v)
Dependencies:	--
Supporting Material:	--

](RS_BRF_01000)

6.1.1.2.3 [SRS_IoHwAb_12449] The IO Hardware Abstraction shall be able to handle more than one electrical signal simultaneously

[

Type:	Valid
Description:	The IO Hardware Abstraction shall be able to handle more than one electrical signal simultaneously. The definition of a group is done during the configuration step. Signals belonging to a group have always the same type.
Rationale:	Control without time delay a group of signal
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01048)

6.1.1.3 Normal Operation

6.1.1.3.1 [SRS_IoHwAb_12242] The IO Hardware Abstraction shall hide any communication over ECU internal onboard peripherals to access Signals

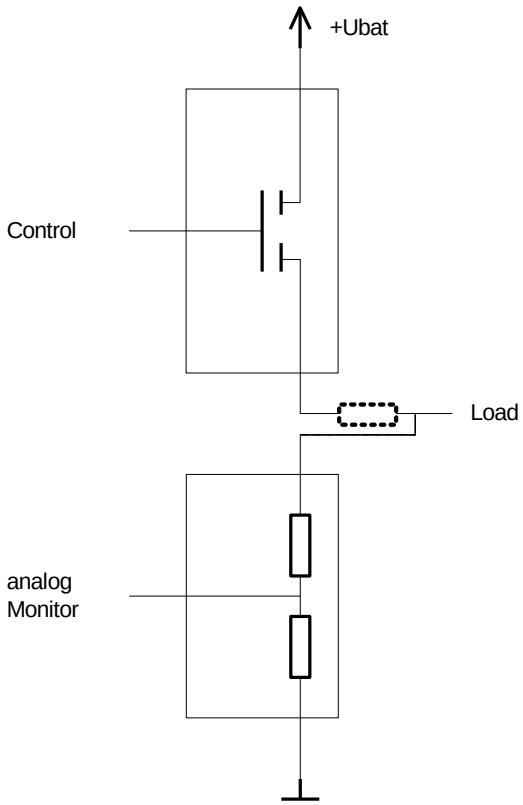
[

Type:	Valid
Description:	The IO Hardware Abstraction shall hide any communication over ECU internal onboard peripherals to access Signals. The Signal routing on the ECU is abstracted by this interface.
Rationale:	It shall be no difference for the upper layers, if a port is connected directly to the Microcontroller or an onboard ASIC.
Use Case:	--
Dependencies:	--
Supporting Material:	--

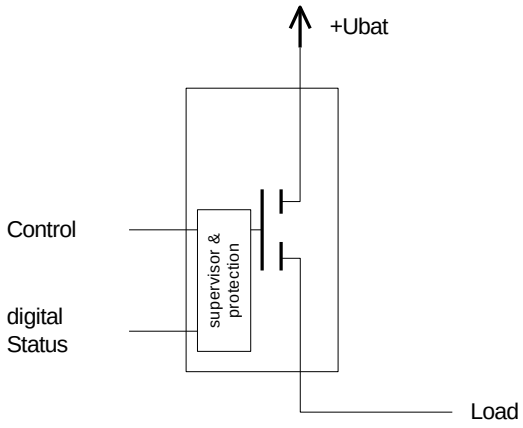
](RS_BRF_01080)

6.1.1.4 Diagnostic Functions

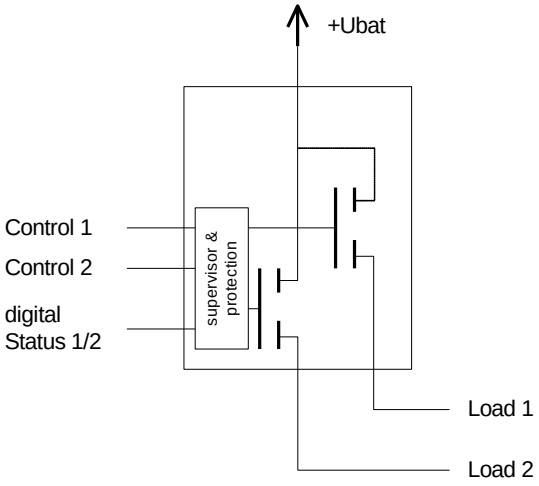
The following schematics show output driver stages and their diagnostic capability. They shall help to understand the following requirements.



Concept scheme for diagnostic of digital output by analogue monitor line



Concept scheme for diagnostic of digital output by digital monitor line of a single driver stage



Concept scheme for diagnostic of digital output by digital monitor line of a n-channel driver stage.

6.1.1.4.1 [SRS_IoHwAb_13900] The IO Hardware Abstraction shall detect a failure short circuit to the ground, according to the hardware capabilities

⌈

Type:	Valid
Description:	The IO Hardware Abstraction shall detect a failure short circuit to the ground, according to the hardware capabilities
Rationale:	Basic functionality
Use Case:	Analog monitoring of a discrete load signal, Fault detection for diagnosis and/or software driven driver stage protection
Dependencies:	--
Supporting Material:	Specific Hardware Design : existing diagnostic monitoring.

⌋(RS_BRF_02000,RS_BRF_02168)

6.1.1.4.2 [SRS_IoHwAb_13901] The IO Hardware Abstraction shall detect a failure short circuit to +UBat, according to the hardware capabilities

⌈

Type:	Valid
Description:	The IO Hardware Abstraction shall detect a failure short circuit to +UBat, according to the hardware capabilities
Rationale:	Basic functionality
Use Case:	Analog monitoring of a discrete load signal, Fault detection for diagnosis and / or error reaction
Dependencies:	--
Supporting Material:	Specific Hardware Design : existing diagnostic monitoring.

⌋(RS_BRF_02000,RS_BRF_02168)

6.1.1.4.3 [SRS_IoHwAb_13902] The IO Hardware Abstraction shall detect an open circuit, according to the hardware capabilities

⌈

Type:	Valid
Description:	The IO Hardware Abstraction shall detect an open circuit, according to the hardware capabilities
Rationale:	Basic functionality
Use Case:	Analog monitoring of a discrete load signal, Fault detection for diagnosis and / or error reaction
Dependencies:	--
Supporting Material:	Specific Hardware Design : existing diagnostic monitoring.

⌋(RS_BRF_02000,RS_BRF_02168)

6.1.1.4.4 [SRS_IoHwAb_13903] The IO Hardware Abstraction shall detect a failure over load

⌈

Type:	Valid
Description:	The IO Hardware Abstraction shall detect a failure over load. This detection is done if controlled Output Signal is activated.
Rationale:	Basic functionality
Use Case:	Analog monitoring of a discrete load signal, Fault detection for diagnosis and / or error reaction
Dependencies:	--
Supporting Material:	Specific Hardware Design : Diagnostic monitoring.

⌋(RS_BRF_02176)

6.1.1.4.5 [SRS_IoHwAb_13904] The IO Hardware Abstraction shall detect an over temperature

⌈

Type:	Valid
Description:	The IO Hardware Abstraction shall detect an over temperature. This detection is done if controlled Output Signal is activated.
Rationale:	Basic functionality
Use Case:	--
Dependencies:	--
Supporting Material:	Specific Hardware Design : Diagnostic monitoring.

⌋(RS_BRF_02168)

6.1.1.5 Fault operation

6.1.1.5.1 [SRS_IoHwAb_12248] The IO Hardware Abstraction module shall keep the ECU hardware safe

⌈

Type:	Valid
Description:	The IO Hardware Abstraction module shall keep the ECU hardware safe. The IO Hardware Abstraction shall be able to cut off an output signal when a failure is detected on this output. This will be done in order to protect the hardware.
Rationale:	Protection against ECU deterioration.
Use Case:	Short circuit to the ground, short circuit to the supply, over temperature, overload. Deactivation of the output after three orders.
Dependencies:	--
Supporting Material:	--

](RS_BRF_02024)

6.1.1.5.2 [SRS_IoHwAb_12451] The IO Hardware Abstraction module shall not decide on its own to switch an output on again that has been switched off for hardware protection reasons

[

Type:	Valid
Description:	The IO Hardware Abstraction module shall not decide on its own to switch an output on again that has been switched off for hardware protection reasons. Such a strategy to recover a failure shall be defined in a Software Component.
Rationale:	Strategies are included in SW-C
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_02016)

6.1.1.5.3 [SRS_IoHwAb_12452] The IO Hardware Abstraction module shall offer an interface to trigger an output after an output has been cut off

[

Type:	Valid
Description:	The IO Hardware Abstraction module shall offer an interface to trigger an output after an output has been cut off.
Rationale:	Detect electrical defaults
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01056,RS_BRF_02224)

6.1.1.6 Power state transitions

6.1.1.6.1 [SRS_IoHwAb_12453] Encapsulation of power state preparation shall be available

⌈

Type:	Valid
Description:	In case power control on the peripherals is implemented, IoHwAbstraction shall define an API for each application Power Mode, in which all perpheral power state transitions relevant to the given application power mode are made effective.
Rationale:	It is necessary to create an encapsulation which allows to map an application power mode, eventually valid for different ECUs on the vehicle system, to the power state of each peripheral on each ECU. In this way the setting phase for a given power mode can be atomically executed, so that all HW peripheral transition to the validpower state synchronously.
Use Case:	--
Dependencies:	--
Supporting Material:	Concept Proposal 596: ECU Degradation

⌋ (RS_BRF_01952, RS_BRF_01184)

6.1.1.6.2 [SRS_IoHwAb_12454] Encapsulation of power state setting shall be available

⌈

Type:	Valid
Description:	In case power control on the peripherals is implemented, IoHwAbstraction shall define an API for each application Power Mode, in which all peripheral power state transitions relevant to the given application power mode are to be prepared (power state preparation).
Rationale:	It is necessary to create an encapsulation which allows to map an application power mode, eventually valid for different ECUs on the vehicle system, to the power state of each peripheral on each ECU. In this way the preparation phase for a given power mode can be atomically executed.
Use Case:	--
Dependencies:	--
Supporting Material:	Concept Proposal 596: ECU Degradation

⌋(RS_BRF_01952, RS_BRF_01184)

6.1.1.6.3 [SRS_IoHwAb_12455] Callback notification for each power state preparation completion shall be provided

⌈

Type:	Valid
Description:	In case power control on the peripherals is implemented, IoHwAbstraction shall define a callback for each power state of each peripheral needed to implement a given Application Power Mode.

Rationale:	It is necessary to decouple the preparation and the setting phase of a power transition. In this way the IoHwAbs or CDD can avoid polling the state of all peripherals to check when the power state can be set.
Use Case:	--
Dependencies:	--
Supporting Material:	Concept Proposal 596: ECU Degradation

](RS_BRF_01952, RS_BRF_01184)

6.2 Non-Functional Requirements (Qualities)

6.2.1 [SRS_IoHwAb_12339] The IO Hardware Abstraction shall guarantee a given worst case delay time for each signal

[

Type:	Valid
Description:	The IO Hardware Abstraction shall guarantee a given worst case delay time for each signal. The time can be used for the evaluation of the time constraints.
Rationale:	Fulfill the request reaction times.
Use Case:	A lot of time constraints e.g. reaction after a pushed button
Dependencies:	--
Supporting Material:	--

](RS_BRF_02168)

7 References

7.1 Deliverables of AUTOSAR

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] Software Standardization Template
AUTOSAR_TPS_StandardizationTemplate.pdf