Document Title	Requirements on SPI Handler/Driver
Document Category	Requirement specification (RS)
Document UID	77
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• No content changes
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes
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	• Modified requirements: [SRS_Spi_12197], [SRS_Spi_12256]
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	• Editorial changes

△

2016-11-30	4.3.0	AUTOSAR Release Management	• New chapter "Requirements tracing"
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 TPS_StandardizationTemplate [1]
2010-09-30	3.1.5	AUTOSAR Administration	• 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. • Change of the document scope (1), the functional overview (4.2/ 4.2.2 / 4.2.3) • Add of [SRS_Spi_13400] Scalable functional perimeter, [SRS_Spi_13401] Functional perimeter statically configurable • Change of following requirements [SRS_Spi_12151], [SRS_Spi_12152], [SRS_Spi_12153], [SRS_Spi_12154] and [SRS_Spi_12197], [SRS_Spi_13400], [SRS_Spi_13401] • Approvement of [SRS_Spi_12151], [SRS_Spi_13400], [SRS_Spi_13401], [SRS_Spi_12152], [SRS_Spi_12153] and [SRS_Spi_12154]

▽

△

2005-05-31	1.0	AUTOSAR Administration	• Initial release as a part of the SRS SPAL V1.0.0

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Scope of Document 6

2 Conventions to be used 7
2.1 Document Conventions . . . 7
2.2 Requirements structure . . . 8

3 Acronyms and abbreviations 9

4 Requirements Specification 10
4.1 Functional Overview . . . 10
4.1.1 SPI Handler/Driver, common functionality . . . 10
4.1.2 Asynchronous SPI functionality . . . 12
4.1.3 Synchronous SPI functionality . . . 12
4.2 Functional Requirements . . . 12
4.2.1 SPI Handler/Driver, common functionality . . . 12
4.2.1.1 General . . . 12
4.2.1.2 Configuration . . . 14
4.2.1.3 Normal Operation . . . 16
4.2.1.4 Fault Operation . . . 20
4.2.2 Asynchronous SPI functionality . . . 21
4.2.2.1 Configuration . . . 21
4.2.2.2 Normal Operation . . . 22
4.2.3 Synchronous SPI functionality . . . 25
4.2.3.1 Normal Operation . . . 25

5 References 28

A Change history of AUTOSAR traceable items 29
A.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 29
A.1.1 Added Requirements in R26-11 . . . 29
A.1.2 Changed Requirements in R26-11 . . . 29
A.1.3 Deleted Requirements in R26-11 . . . 29
A.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 29
A.2.1 Added Requirements in R25-11 . . . 29
A.2.2 Changed Requirements in R25-11 . . . 29
A.2.3 Deleted Requirements in R25-11 . . . 29

1 Scope of Document

This document specifies requirements on the monolithic SPI Handler/Driver module including:

- Multiple SPI busses handling
- Synchronous SPI transmission
- Asynchronous SPI transmission

Constraints

First scope for specification of requirements on basic software module is 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 [1, Standardization Template].

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

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as follows.

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

- MUST: This word, or the adjective “LEGALLY REQUIRED”, means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase, or the phrase “MUST NOT”, means that the definition is an absolute prohibition of the specification due to legal issues.
- SHALL: This phrase, or the adjective “REQUIRED”, means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective “RECOMMENDED”, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase “NOT RECOMMENDED”, means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective “OPTIONAL”, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.

An implementation, which does not include a particular option, SHALL be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides).

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 SPI Handler/-Driver that are not included in the AUTOSAR Glossary [2].

Abbreviation / Acronym:	Description:
CS	Chip Select
DIO	Digital Input Output
ECU	Electric Control Unit
DMA	Direct Memory Access
ICU	Input 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
MISO	Master Input Slave Output
MMU	Memory Management Unit
MOSI	Master Output Slave Input
Master	A device controlling other devices (slaves, see below)
Slave	A device being 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
SFR	Special Function Register
RTE	RunTime Environment
STD	Standard
REQ	Requirement
UNINIT	Uninitialized (= not initialized)

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

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

Term/Expression:	*Description:*
Channel	A Channel is a software exchange medium for data that are defined with the same criteria: Config. Parameters, Number of Data elements with same size and data pointers (Source & Destination) or location.
Job	A Job is composed of one or several Channels with the same Chip Select (is not released during the processing of Job). A Job is considered atomic and therefore cannot be interrupted by another Job. A Job has an assigned priority.
Sequence	A Sequence is a number of consecutive Jobs to transmit but it can be rescheduled between Jobs using a priority mechanism. A Sequence transmission is interruptible (by another Sequence transmission) or not depending on a static configuration.

4 Requirements Specification

This chapter describes all requirements driving the work to define the SPI Handler/-Driver.

4.1 Functional Overview

4.1.1 SPI Handler/Driver, common functionality

A SPI bus is a master slave multi node bus system, the master sets a Chip Select (CS) to select a slave for data communication. The SPI (Serial Peripheral Interface) has a 4-wire synchronous serial interface. Data communication is enabled with a Chip Select wire (CS). Data is transmitted with a 3-wire interface consisting of wires for serial data input (MOSI), serial data output (MISO) and Serial ClocK (SCK).

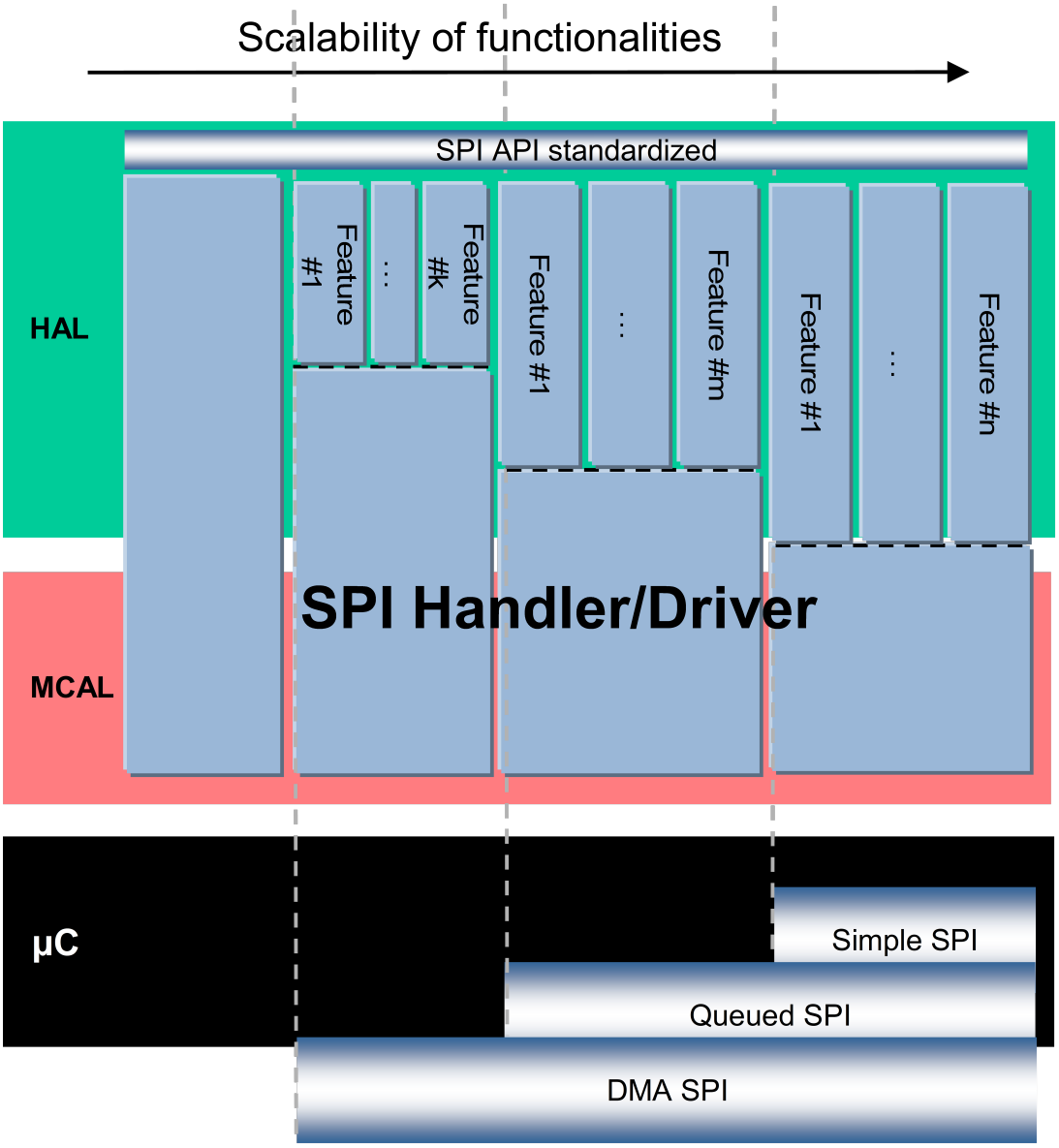


Figure 4.1

The following SPI module provides channel based read, write and transfer access to different devices on SPI busses. A SPI channel represents data elements (8 to 16 data bits). These channels could be combined in sequence which shall not be interrupted (e.g. Daisy-Chain, EEPROM). Channels have a static configuration defining baud rate, chip select,... A SPI device is generally identified by the used SPI hardware unit and the associated chip select line. The module can operate only as SPI master.

The functional perimeter of this software module will be statically configurable to fit as far as possible to the real needs of each ECU. That means for instance synchronous, asynchronous or both SPI access could be present in the ECU. Consequently, two SPI drivers could exist but just one handler interface. This chapter contains common requirements that are valid both for synchronous and asynchronous SPI drivers.

4.1.2 Asynchronous SPI functionality

This part of the monolithic SPI Handler/Driver could be so-called driver and provides asynchronous read, write and transfer access to different devices on SPI busses and callback notifications. The access to the different SPI channels is priority controlled.

4.1.3 Synchronous SPI functionality

This part of the monolithic SPI Handler/Driver could be so-called driver and provides synchronous read and write access to different devices on SPI busses.

4.2 Functional Requirements

4.2.1 SPI Handler/Driver, common functionality

4.2.1.1 General

[SRS_Spi_12093] The SPI Handler/Driver shall be able to handle multiple busses of communication ⌈

Description:	The SPI Handler/Driver shall be able to handle multiple busses of communication. Every device connected to SPI busses will be handled by channels..
Rationale:	This will abstract the upper layers from the hardware, making reference to the information and not to the hardware.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12094] The SPI Handler/Driver shall handle the chip select ⌈

Description:	The SPI Handler/Driver shall handle the chip select.
Rationale:	Basic functionality
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12256] The SPI Handler/Driver shall support all controller peripherals ⌈

Description:	The SPI Handler/Driver shall support all controller peripherals, which are capable of performing the SPI functionality (data in/ data out/ clock + optional chip select signal and optional IRQn signal(s)).
Rationale:	HW encapsulation
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12257] The SPI Handler/Driver shall support the communication to daisy chained HW devices ⌈

Description:	The SPI Handler/Driver shall support the communication to daisy chained HW devices. During the transfer to/from the HW devices, the CS signal shall remain asserted.
Rationale:	Due to limited controller resources (CS signals) some external HW devices can be daisy chained, using the same CS signal.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_13400] The SPI Handler/Driver shall have a scalable functionality to fit the needs of the ECU ⌈

Description:	The SPI Handler/Driver shall have a scalable functionality to fit the needs of the ECU. For example: Asynchronous, synchronous, interruptible sequences...
Rationale:	To optimize the memory and CPU resource usage.
Use Case:	If only non interruptible sequences are used do not implement any scheduling strategies based on priorities.
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.2 Configuration

[SRS_Spi_12025] The SPI Handler/Driver shall allow the static configuration of all software and hardware properties related to SPI ⌈

Description:	The SPI Handler/Driver shall allow the static configuration of all software and hardware properties related to SPI. The following list is a list of proposed properties: 1. assigned SPI HW Unit 2. assigned chip select pin (it is possible to assign no pin) 3. Chip select functionality on/off 4. Chip select pin polarity high or low 5. Chip select mode (normal mode or hold mode) 6. Baud rate 7. Timing between clock and chip select 8. data width (1 up to 32 bits) 9. transfer start LSB or MSB 10. shift clock idle low or idle high 11. data shift with leading or trailing edge 12. MCU dependent properties for the channels
Rationale:	Flexibility and Scalability
Use Case:	–
Dependencies:	[SRS_Spi_12259], [SRS_Spi_12032], [SRS_Spi_12033]
Supporting Material:	BMW Specification MCAL V1.0a, MAL40.1.1

⌋

[SRS_Spi_12179] The SPI Handler/Driver shall allow linking consecutive SPI channels by static configuration ⌈

Description:	The SPI Handler/Driver shall allow linking consecutive SPI channels by static configuration.
Rationale:	Allow to form streams of SPI communication.
Use Case:	As a clarifying example: To communicate with an external SPI EEPROM someone uses channels 30 to 35 in such a way that: • Channel 30 is the action command • Channel 31 is the high address • Channel 32 is the low address • Channel 34 is the first byte of the data • Channel 35 is the second byte of the data
Dependencies:	[SRS_Spi_12181]
Supporting Material:	[SRS_Spi_12093]

⌋

[SRS_Spi_12026] The SPI Handler/Driver shall allow the static configuration of the desired number of SPI channels ⌈

Description:	The SPI Handler/Driver shall allow the static configuration of the desired number of SPI channels (max. 255)..
Rationale:	The SPI-Master normally controls more than one device.
Use Case:	–
Dependencies:	–
Supporting Material:	BMW Specification MCAL V1.0a, MAL40.1.2

⌋

[SRS_Spi_12197] The transmission data width of each SPI channel shall be configurable ⌈

Description:	The SPI Handler/Driver shall offer the possibility of configuring the transmission data width for each SPI channel in the range of 1 to 64 bits (not only 8, 16 or 32 bits).
Rationale:	There is HW IP available with 64 bits. (Allowed HW configuration to be configurable)
Use Case:	ADC result register is 10 bit, port extension is 8 bit. Whole transfer to one device done without releasing the CS.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_13401] The SPI Handler/Driver functionalities shall be statically configurable ⌈

Description:	The SPI Handler/Driver functionalities shall be statically configurable to include only those needed by the ECU.
Rationale:	To optimize the memory and CPU resource usage.
Use Case:	If only synchronous SPI access is required, do not include asynchronous SPI access.
Dependencies:	[SRS_Spi_13400]
Supporting Material:	–

⌋

4.2.1.3 Normal Operation

[SRS_Spi_12258] Data shall be accessible from each device individually ⌈

Description:	The SPI Handler/Driver shall support access to transferred data (read /write) related to a certain HW device independent of the HW configuration
Rationale:	To ensure HW device abstraction, the transferred data shall be individually accessible by the corresponding HW device driver, independent of the HW configuration.
Use Case:	In case of daisy chained HW devices (different HW devices using same CS signal), the data related to each of the HW devices must be accessible individually.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12259] Different timing and HW parameters shall be supported ⌈

Description:	The SPI Handler/Driver shall support the configuration of the following parameters for each HW device: • Baud rate • Chip select pin polarity high or low • Timing between clock and chip select • shift clock idle low or idle high • data shift with leading or trailing edge
Rationale:	Each connected HW device has different timing requirements
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12260] Different priorities of sequences shall be supported ⌈

Description:	The SPI Handler/Driver shall support static assignment of a priority to each sequence
Rationale:	Allow prioritization of asynchronous communication requests.
Use Case:	Saving of crash data to external EEPROM should not be delayed due to other SPI communication. Already requested other SPI communication shall be delayed until crash data is saved.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12180] The SPI Driver shall access the SPI bus only for the channel ⌈

Description:	If an SPI access request for a single (not linked) SPI channel is performed, the SPI Handler/Driver shall access the SPI bus only for this channel.
Rationale:	This is nearly trivial, but helps understanding the following requirement [SRS_Spi_12181].
Use Case:	Simple transmission of one SPI channel.
Dependencies:	–
Supporting Material:	[SRS_Spi_12093]

⌋

[SRS_Spi_12181] If an SPI access request for a linked channel is performed, the SPI Handler/Driver shall use this SPI channel and all the linked channels ⌈

Description:	If an SPI access request for a linked channel is performed, the SPI Handler/Driver shall use this SPI channel and all consecutive channels of the same link for SPI bus access.
Rationale:	Support different communication stream lengths. In an SPI communication using linked channels, the starting channel of an action could be any of the channels that form the stream.
Use Case:	Channels 30 to 38 are linked. If an SPI access request selects channel 35 as starting channel for the access, only channels 35, 36, 37 and 38 will be used for that SPI access.
Dependencies:	[SRS_Spi_12179]
Supporting Material:	[SRS_Spi_12093]

⌋

[SRS_Spi_12032] For an SPI channel assigned to an SPI HW Unit the chip select mode "normal" shall be available ⌈

Description:	For an SPI channel assigned to an SPI HW Unit the chip select mode "normal" shall be available: Selection of the assigned chip select pin before the transfer starts and deselection after the transfer has been finished. The SPI HW unit is released.
Rationale:	Normal SPI transfer.
Use Case:	–
Dependencies:	–
Supporting Material:	BMW Specification MCAL V1.0a, MAL40.6.0

⌋

[SRS_Spi_12033] For an SPI channel assigned to an SPI HW Unit the chip select mode "hold" shall be available ⌈

Description:	For an SPI channel assigned to an SPI HW Unit the chip select mode "hold" shall be available: Selection of the assigned chip select pin before the transfer starts. If the transfer has been finished, the chip select is kept active. The SPI HW is kept allocated.
Rationale:	Some SPI slave devices require to be kept selected during data processing or between some coherent data transmissions.
Use Case:	–
Dependencies:	–
Supporting Material:	BMW Specification MCAL V1.0a, MAL40.6.0

⌋

[SRS_Spi_12198] The SPI Handler/Driver shall provide the functionality of transferring one short data sequence with variable data content ⌈

Description:	The SPI Handler/Driver shall provide the functionality of transferring one short data sequence with variable data content. "Variable" means data contents change between two transmissions not during a transmit. "Short data sequence" means e.g. 10 words.
Rationale:	Base requirement for data transfer
Use Case:	Transfer data to a simple SPI slave device
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12253] The SPI Handler/Driver shall provide the functionality of transferring one short data sequence with constant data content ⌈

Description:	The SPI Handler/Driver shall provide the functionality of transferring one short data sequence with constant data content. "Short data sequence" means about 10 words.
Rationale:	–
Use Case:	Send commands and addresses, receive results
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12199] The SPI Handler/Driver shall provide the functionality of transferring any data to any devices in one transfer sequence ⌈

Description:	The SPI Handler/Driver shall provide the functionality of transferring any data to any devices like [SRS_Spi_12198] and [SRS_Spi_12253] in one transfer sequence.
Rationale:	The amount of data sent shall not be limited by HW implementation. Static definition of communication sequences.
Use Case:	Transfer data to multiple devices connected to the same SPI bus. One single trigger (e.g. periodic 10 ms) can start the communication to multiple HW devices.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12200] Reading large data sequences from one slave device using dummy send data shall be possible ⌈

Description:	The SPI Handler/Driver shall provide the functionality of transferring large (up to the magnitude of 100 words) data sequences with only one constant data to send.
Rationale:	–
Use Case:	Read multiple result registers from a complex SPI device sending only a dummy data
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12261] Reading large data sequences from one slave device using variable send data shall be possible ⌈

Description:	The SPI Handler/Driver shall provide the functionality of transferring large (up to the magnitude of 100 words) data sequences with variable data to send to one device.
Rationale:	–
Use Case:	Read multiple result registers from a complex SPI device transferring addresses of the registers
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12201] Reading large data sequences from multiple slave devices using dummy send data shall be possible ⌈

Description:	The SPI Handler/Driver shall provide the functionality of transferring large (up to the magnitude of 100 words) data sequences with constant data to send to multiple SPI slave devices.
Rationale:	–
Use Case:	Read multiple result registers from multiple complex SPI devices
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12262] Reading large data sequences from multiple slave devices using variable send data shall be possible ⌈

Description:	The SPI Handler/Driver shall provide the functionality of transferring large (up to the magnitude of 100 words) data sequences with variable data to send to multiple slave devices.
Rationale:	–
Use Case:	Read multiple result registers from multiple complex SPI devices
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12202] The SPI Handler/Driver shall support data streams to a HW device with variable number of data ⌈

Description:	The SPI Handler/Driver shall support data streams to a HW device (CS signal) with variable number of data.
Rationale:	–
Use Case:	Some external EEPROM devices support Burst modes and are capable of transfering data streams from 1 to 32 data bytes.
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.4 Fault Operation

As the behaviour of the SPI bus is synchronous, no timeout detection is supported by the SPI Handler/Driver itself.

4.2.2 Asynchronous SPI functionality

For the asynchronous SPI Driver also the general SPI Handler/Driver requirements apply.

4.2.2.1 Configuration

[SRS_Spi_12024] The SPI Handler/Driver shall allow the static configuration of the following options ⌈

Description:	The SPI Handler/Driver shall allow the static configuration of the following options: • Buffer / FIFO usage • MCU dependent properties for SPI HW unit
Rationale:	Flexibility and Scalability
Use Case:	Configuration of DMA buffering for SPI
Dependencies:	–
Supporting Material:	BMW Specification MCAL V1.0a, MAL40.1.0

⌋

[SRS_Spi_12150] The SPI Handler/Driver shall allow the static configuration of all software and hardware properties related to asynchronous SPI aspects ⌈

Description:	The SPI Handler/Driver shall allow the static configuration of all software and hardware properties related to asynchronous SPI aspects. The following list is a list of proposed properties: • Priority: 4 levels • Transmission end notification function
Rationale:	Flexibility and Scalability
Use Case:	1: If a window watchdog and an EEPROM are connected to the SPI interface, the triggering of the watchdog shall have a higher priority than reading/writing streams from the external EEPROM. Other HW configurations may require different behavior and priorities.
Dependencies:	[SRS_Spi_12025]
Supporting Material:	BMW Specification MCAL V1.0a, MAL40.1.1

⌋

4.2.2.2 Normal Operation

[SRS_Spi_12108] The SPI Handler/Driver shall call the statically configured notification function ⌈

Description:	The SPI Handler/Driver shall call the statically configured notification function associated to : • A single SPI channel when its transmission has been performed, • Linked SPI channels when their transmissions have been performed.
Rationale:	Real time behavior, flexibility.
Use Case:	–
Dependencies:	[SRS_Spi_12180], [SRS_Spi_12181]
Supporting Material:	–

⌋

[SRS_Spi_12099] The SPI Handler/Driver shall provide an asynchronous read functionality ⌈

Description:	The SPI Handler/Driver shall provide an asynchronous read functionality. This functionality shall read a data block with the passed length from the selected SPI device giving the following parameters to the driver: • Channel • Address of data buffer where received data is written to • Length of the data This action shall be buffered and done when the driver is ready again. The caller shall be informed about the end of the transaction with a notification as configured.
Rationale:	To allow reading buffered data without blocking SPI transmissions.
Use Case:	–
Dependencies:	–
Supporting Material:	BMW Specification MCAL V1.0a, MAL40.2.0

⌋

[SRS_Spi_12101] The SPI Handler/Driver shall provide an asynchronous write functionality ⌈

Description:	The SPI Handler/Driver shall provide an asynchronous write functionality. This functionality shall write a data block with the passed length to the selected SPI device giving the following parameters to the driver: • Channel • Source address • Length of the data The caller shall be informed about the end of the transaction with a notification as configured. The application should be able to read asynchronously the requested information.
Rationale:	This action will be buffered and done when the driver is ready again.
Use Case:	–
Dependencies:	–
Supporting Material:	BMW Specification MCAL V1.0a, MAL40.4.0

⌋

[SRS_Spi_12103] The SPI Handler/Driver shall provide an asynchronous read-write functionality ⌈

Description:	The SPI Handler/Driver shall provide an asynchronous read-write functionality. This functionality shall write a data block with the passed length to the selected SPI device and simultaneously read a data block of the same length from the selected SPI device, giving the following parameters to the driver: • Channel • Source address • Write address • Length of the data The application should be able to read asynchronously the requested information.
Rationale:	This action will be buffered and done when the driver is ready again.
Use Case:	Write-read functionality for SPI devices with simultaneous feedback, e.g. control outputs using a SPI ASIC. These devices use to give you a feedback of the status of the outputs.
Dependencies:	–
Supporting Material:	BMW Specification MCAL V1.0a

⌋

[SRS_Spi_12037] The SPI Handler/Driver shall allow a priority controlled allocation of the HW SPI unit ⌈

Description:	The SPI Handler/Driver shall allow a priority controlled allocation of the HW SPI unit: The SPI channels can have different priorities. After release of the SPI HW unit, the requesting SPI channel with the highest priority gets the transfer right.
Rationale:	Efficient allocation algorithm with deterministic job execution.
Use Case:	–
Dependencies:	–
Supporting Material:	BMW Specification MCAL V1.0a, MAL40.7.0

⌋

[SRS_Spi_12104] The SPI Handler/Driver shall provide a synchronous functionality which returns any transfer status ⌈

Description:	The SPI Handler/Driver shall provide a synchronous functionality which returns any transfer status..
Rationale:	Check whether the SPI transmission is done.
Use Case:	To know if data transfer is done but also to know if ECU could go to sleep.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12170] The SPI Handler/Driver shall not provide the ability to prevent a channel data overwrite ⌈

Description:	The SPI Handler/Driver does not provide the ability to prevent a channel data overwrite. Because of this, it is the user's responsibility to take care of data consistency by waiting for the completion of a SPI channel's transmission before writing new data to the same SPI channel. This has to be described as a constraint in the software specification of the SPI Handler/Driver..
Rationale:	–

▽

△

Use Case:	The user shall follow the following sequence: 1. Call 'write SPI channel' 2. Call 'start SPI channel transfer' 3. Wait for end of transmission 4. Call 'write SPI channel' The following sequence may cause overwriting of data: 1. Call 'write SPI channel' 2. Call 'start SPI channel transfer' 3. Call 'write SPI channel' (may overwrite the data if the SPI is not fast enough)
Dependencies:	–
Supporting Material:	–

⌋

4.2.3 Synchronous SPI functionality

For the synchronous SPI Driver also the general SPI Handler/Driver requirements apply.

4.2.3.1 Normal Operation

[SRS_Spi_12152] The SPI Handler/Driver shall provide a synchronous read functionality ⌈

Description:	The SPI Handler/Driver shall provide a synchronous read functionality. This functionality shall allow the reading of a data block with the passed length from the selected SPI device giving the following parameters to the driver: • Channel • Address of data buffer where received data is written to • Length of the data This action shall be done synchronously with the call of function.
Rationale:	–
Use Case:	Read data from an I/O Shift register on board device.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12153] The SPI Handler/Driver shall provide a synchronous write functionality ⌈

Description:	The SPI Handler/Driver shall provide a synchronous write functionality. This functionality shall allow the writing of a data block with the passed length to the selected SPI device giving the following parameters to the driver: • Channel • Source address • Length of the data This action shall be done synchronously with the call of function.
Rationale:	–
Use Case:	Write data to an external EEPROM device.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12154] The SPI Handler/Driver shall provide a synchronous write-read functionality ⌈

Description:	The SPI Handler/Driver shall provide a synchronous write-read functionality. This functionality shall allow the writing of a data block with the passed length to the selected SPI device, and simultaneously the reading of a data block with the same length from the selected SPI device, giving the following parameters to the driver: • Channel • Source address • Destination address • Length of the data This action shall be done synchronously.
Rationale:	–
Use Case:	Write-read functionality for SPI devices with simultaneous feedback (SMART devices for power stages).
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Spi_12151] The SPI Handler/Driver shall perform jobs in the order requested by the caller ⌈

Description:	The SPI Handler/Driver shall perform jobs in the order requested by the caller. During the processing of an SPI bus transmission all other requests to the same SPI bus shall be discarded.
Rationale:	To support a pre-emptive multi tasking system.

▽

△

Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

5 References

[1] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[2] Glossary
AUTOSAR_FO_TR_Glossary

A Change history of AUTOSAR traceable items

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

A.1 Traceable item history of this document according to AUTOSAR Release R26-11

A.1.1 Added Requirements in R26-11

none

A.1.2 Changed Requirements in R26-11

none

A.1.3 Deleted Requirements in R26-11

none

A.2 Traceable item history of this document according to AUTOSAR Release R25-11

A.2.1 Added Requirements in R25-11

none

A.2.2 Changed Requirements in R25-11

none

A.2.3 Deleted Requirements in R25-11

none