MCAL SPI Module Software Design Document

Document Version : 130
Document Owner : Texas Instruments
Document Status : Published
Last Approval Date : Mar 11, 2022

1 Revision History

Version	Date	Author	Document Status	Comments
0.1	28 Jun 2018	Sunil MS	DONE	First version
0.2	09 Oct 2018	Vibha Pant	DONE	Format change, re-order and addressed review comments
0.3	06 Oct 2021	Narni Murthy	DONE	Added Safety Diagnostic Features and changed design document format as per ASPICE
0.4	24 Jan 2022	Nikki S	IN PROGRESS	JACINTOREQ-1870
0.5	28 Feb 2022	Rakesh L	IN PROGRESS	Added Safety Diagnostic APIs
v130	04 Mar 2022	Nikki S	DONE	Review Comments Addressed

2 Terms and Abbreviations

Abbreviation /Term	Meaning / Explanation
CS	Chip Select
DIO	Digital Input Output
ECU	Electric Control Unit
DMA	Direct Memory Access
ICU	Input Capture 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

Abbreviation /Term	Meaning / Explanation
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	Run Time Environment
DET	Default Error Tracer – module to which errors are reported
DEM	Diagnostic Event Manager
SPI	Serial Peripheral Interface

3 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module SPI

- Supported AUTOSAR Release**: 4.3.1**
- Supported Configuration Variants**: Pre-Compile & Link Time & Post Build**
- Vendor ID**: SPI_VENDOR_ID (44)**
- Module ID**: SPI _MODULE_ID (83)**

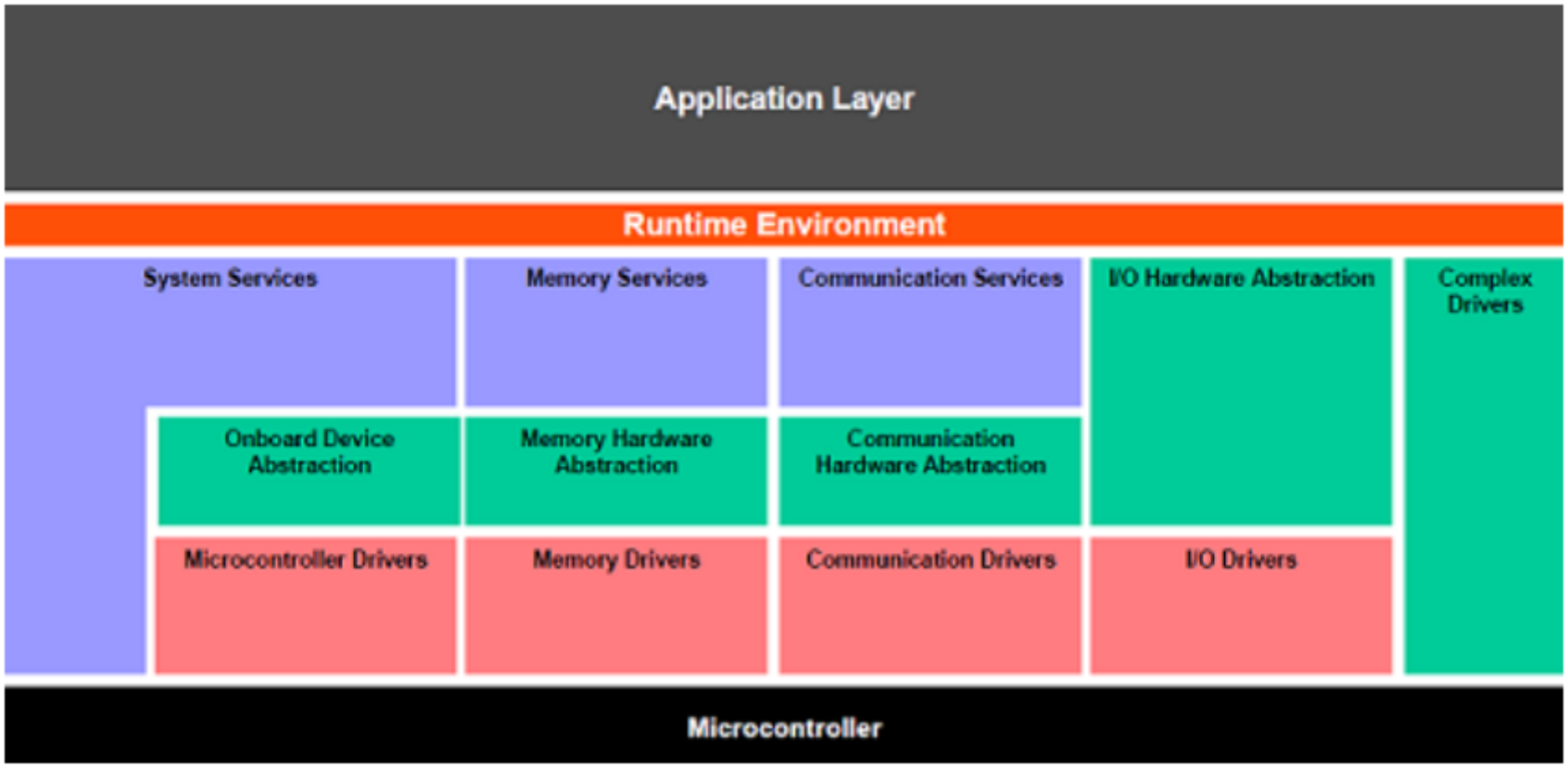
3.1 Overview

The figure below depicts the AUTOSAR layered architecture as 3 distinct layers,

- Application
- Runtime Environment (RTE) and
- Basic Software (BSW).

The BSW is further divided into 4 layers:

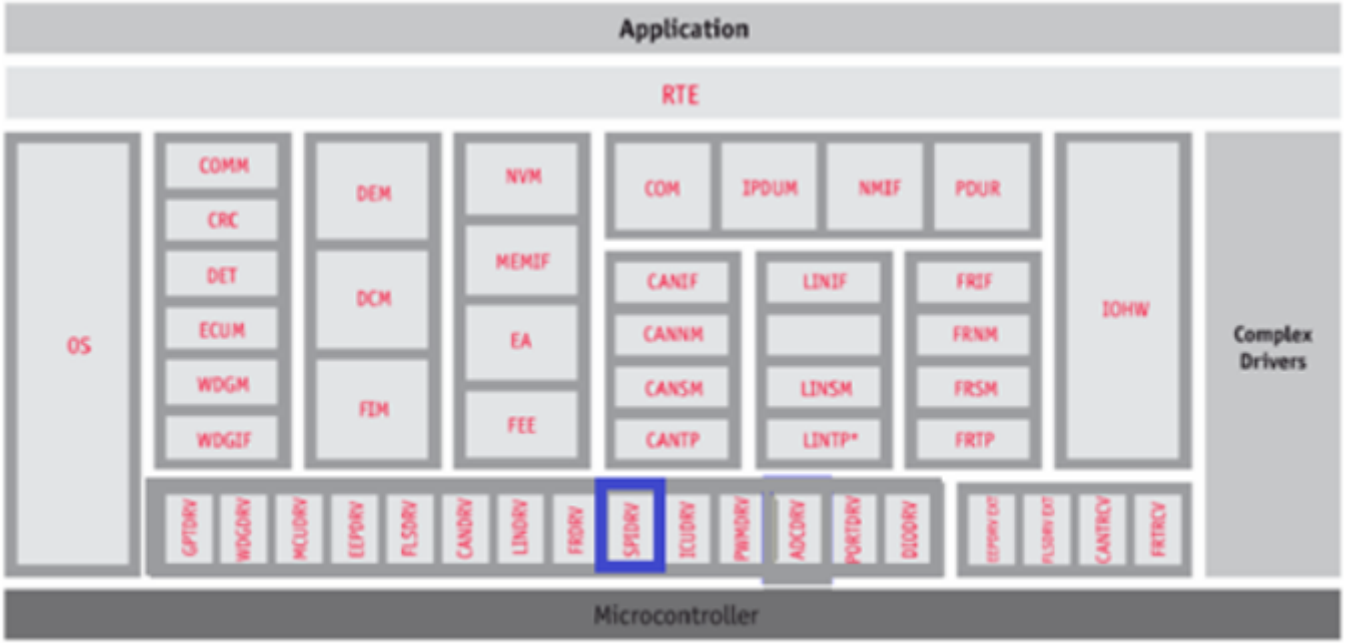
- Services
- Electronic Control Unit Abstraction
- Micro Controller Abstraction (MCAL) and
- Complex Drivers.



AUTOSAR Architecture

MCAL is the lowest abstraction layer of the Basic Software. It contains internal drivers that are software modules that interact with the Microcontroller and its internal peripherals directly.SPI driver is part of the communication Drivers module which is part of the Basic Software. SPI driver diagram below shows the position of the SPI driver in the AUTOSAR Architecture.The Spi driver provides services for basic communication with external components. These components can be used by an application. The main tasks of the Spi are:

- Handle the Spi hardware units onboard.
- Handle data transmission to the components connected via Spi.
- Take care of the settings required by external components (baud rate etc.)



AUTOSAR Architecture – SPI MCAL

3.2 Purpose and Scope

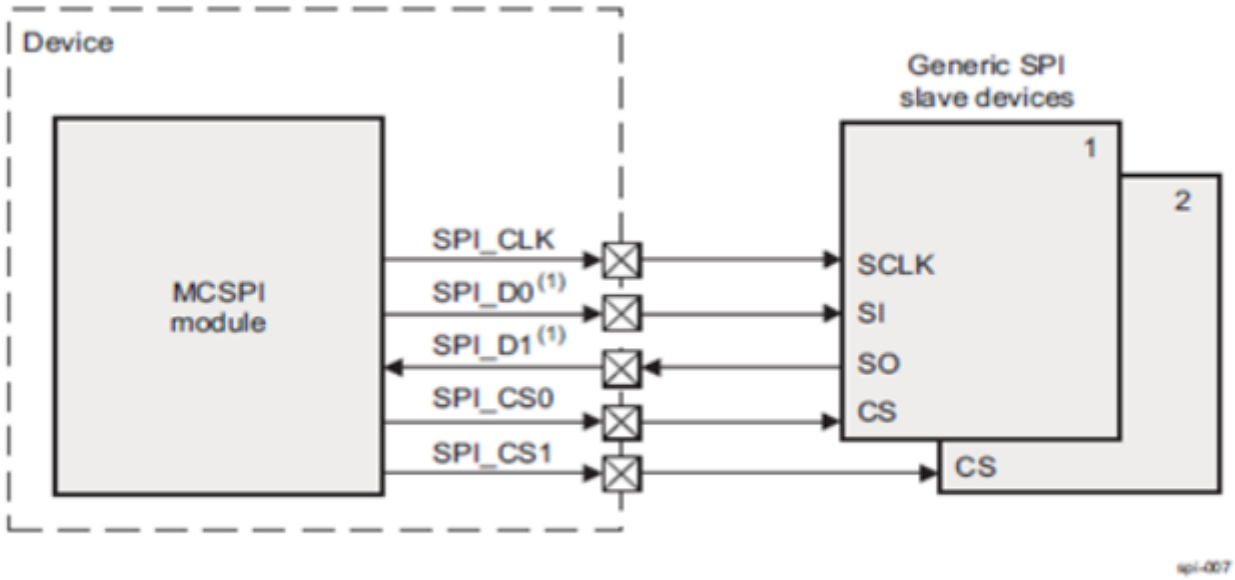
The Detailed Design document provides the design details of SPI driver and aims to provide a guide to a design that could be implemented by a software developer.

The scope of this document is to describe the software design procedure of SPI module.

3.3 Module Overview

This MCAL SPI driver supports McSPI interface. The MCSPI is a master synchronous serial bus module, and uses hardware IP "spi_10_rel.2.11.x". Below section briefly describes McSPI hardware IP features. Please note that this is for just reference purpose for details like exact no. of CS supported or no. of McSPIs on device please refers to device data manual. Diagrams are from device TRMs.

McSPI



(1) Direction depends on bits SPIDATDIR1 and SPIDATDIR0 in MCSPI_SYST

McSPI Block Diagram

The MCSPI modules include the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel.
- Wide selection of SPI word lengths, ranging from 4 to 32 bits.
- SPI configuration per channel. This means, clock definition, polarity enabling and word width can be configured individually.
- Built-in FIFO available for a single channel.
- Support for the following interrupts and status, with masking:
    - Interrupt for FIFO threshold levels. Rx empty, TX empty etc.

3.4 Requirements

The SPI Driver implements a standardized interface specified in the AUTOSAR_SWS_SPIDriver document. The SPI Driver implements a standardized interface as specified in Reference 1 - AUTOSAR 4.3.1.

3.4.1 Features Supported

- Configure Spi with
    - External devices
    - Channels
    - Jobs
    - Sequences
- Initialization and de-initialization of MCSPI hardware units and callback functions.
- Configure error detection (DEM and DET).
- Configure implementation features like:
    - Spi scalability level(s).
    - Spi channel buffers
    - Spi Interrupts

    - Spi frame transfers with 8 or 16bit clock frames
- Select simple or extended API
- Interruptible Sequences.
- All four modes of SPI transfer (mode 0 to mode 3).
- Configurable start bit enable, chip select idle time delay. Chip select maps to single channel, not leveraging the multi- channel feature which IP provides.
- Internal clock divider.
- Concurrent transfer of MCSPI devices.
- Enhanced (Synchronous/Asynchronous) SPI Handler/Driver for MCSPI channels.
- Concurrent synchronous, asynchronous transfer

Design Identifier	Description
MCAL-6422 - SWS_Spi_00041 : General behaviour PUBLISHED	SWS_Spi_00041 : General behaviour
MCAL-6412 - SWS_Spi_00034 : General behaviour PUBLISHED	SWS_Spi_00034 : General behaviour
MCAL-6690 - SWS_Spi_00255 : General behaviour PUBLISHED	SWS_Spi_00255 : General behaviour
MCAL-6487 - SWS_Spi_00053 : General behaviour PUBLISHED	SWS_Spi_00053 : General behaviour

Design Identifier	Description
MCAL-6429 - SWS_Spi_00162 : General behaviour PUBLISHED	SWS_Spi_00162 : General behaviour
MCAL-6683 - SWS_Spi_00295 : General behaviour PUBLISHED	SWS_Spi_00295 : General behaviour
MCAL-6384 - SWS_Spi_00117 : General behaviour PUBLISHED	SWS_Spi_00117 : General behaviour
MCAL-6478 - SWS_Spi_00080 : General behaviour PUBLISHED	SWS_Spi_00080 : General behaviour
MCAL-6672 - SWS_Spi_00085 : Call-Back Functions PUBLISHED	SWS_Spi_00085 : Call-Back Functions
MCAL-6593 - SWS_Spi_00128 : General behaviour PUBLISHED	SWS_Spi_00128 : General behaviour
MCAL-6573 - SWS_Spi_00160 : General behaviour PUBLISHED	SWS_Spi_00160 : General behaviour
MCAL-6381 - SWS_Spi_00161 : General behaviour PUBLISHED	SWS_Spi_00161 : General behaviour

Design Identifier	Description
MCAL-6610 - SWS_Spi_00294 : General behaviour PUBLISHED	SWS_Spi_00294 : General behaviour
MCAL-6723 - SWS_Spi_00438 : General behaviour PUBLISHED	SWS_Spi_00438 : General behaviour
MCAL-6704 - SWS_Spi_00125 : General behaviour PUBLISHED	SWS_Spi_00125 : General behaviour
MCAL-6697 - SWS_Spi_00437 : General behaviour PUBLISHED	SWS_Spi_00437 : General behaviour
MCAL-6677 - SWS_Spi_00051 : General behaviour PUBLISHED	SWS_Spi_00051 : General behaviour
MCAL-6673 - SWS_Spi_00163 : General behaviour PUBLISHED	SWS_Spi_00163 : General behaviour
MCAL-6670 - SWS_Spi_00084 : General behaviour PUBLISHED	SWS_Spi_00084 : General behaviour
MCAL-6657 - SWS_Spi_00014 : General behaviour PUBLISHED	SWS_Spi_00014 : General behaviour

Design Identifier	Description
MCAL-6656 - SWS_Spi_00283 : General behaviour PUBLISHED	SWS_Spi_00283 : General behaviour
MCAL-6643 - SWS_Spi_00115 : General behaviour PUBLISHED	SWS_Spi_00115 : General behaviour
MCAL-6642 - SWS_Spi_00131 : General behaviour PUBLISHED	SWS_Spi_00131 : General behaviour
MCAL-6641 - SWS_Spi_00155 : General behaviour PUBLISHED	SWS_Spi_00155 : General behaviour
MCAL-6596 - SWS_Spi_00116 : General behaviour PUBLISHED	SWS_Spi_00116 : General behaviour
MCAL-6580 - SWS_Spi_00052 : General behaviour PUBLISHED	SWS_Spi_00052 : General behaviour
MCAL-6577 - SWS_Spi_00126 : General behaviour PUBLISHED	SWS_Spi_00126 : General behaviour
MCAL-6561 - SWS_Spi_00049 : General behaviour PUBLISHED	SWS_Spi_00049 : General behaviour

Design Identifier	Description
MCAL-6560 - SWS_Spi_00059 : General behaviour PUBLISHED	SWS_Spi_00059 : General behaviour
MCAL-6559 - SWS_Spi_00123 : General behaviour PUBLISHED	SWS_Spi_00123 : General behaviour
MCAL-6544 - SWS_Spi_00130 : General behaviour PUBLISHED	SWS_Spi_00130 : General behaviour
MCAL-6543 - SWS_Spi_00124 : General behaviour PUBLISHED	SWS_Spi_00124 : General behaviour
MCAL-6541 - SWS_Spi_00257 : General behaviour PUBLISHED	SWS_Spi_00257 : General behaviour
MCAL-6533 - SWS_Spi_00289 : General behaviour PUBLISHED	SWS_Spi_00289 : General behaviour
MCAL-6526 - SWS_Spi_00065 : General behaviour PUBLISHED	SWS_Spi_00065 : General behaviour
MCAL-6521 - SWS_Spi_00149 : General behaviour PUBLISHED	SWS_Spi_00149 : General behaviour

Design Identifier	Description
MCAL-6507 - SWS_Spi_00118 : General behaviour PUBLISHED	SWS_Spi_00118 : General behaviour
MCAL-6484 - SWS_Spi_00119 : General behaviour PUBLISHED	SWS_Spi_00119 : General behaviour
MCAL-6468 - SWS_Spi_00003 : General behaviour PUBLISHED	SWS_Spi_00003 : General behaviour
MCAL-6458 - SWS_Spi_00140 : General behaviour PUBLISHED	SWS_Spi_00140 : General behaviour
MCAL-6454 - SWS_Spi_00290 : General behaviour PUBLISHED	SWS_Spi_00290 : General behaviour
MCAL-6450 - SWS_Spi_00282 : General behaviour PUBLISHED	SWS_Spi_00282 : General behaviour
MCAL-6445 - SWS_Spi_00122 : General behaviour PUBLISHED	SWS_Spi_00122 : General behaviour
MCAL-6441 - SWS_Spi_00120 : General behaviour PUBLISHED	SWS_Spi_00120 : General behaviour

Design Identifier	Description
MCAL-6408 - SWS_Spi_00127 : General behaviour PUBLISHED	SWS_Spi_00127 : General behaviour
MCAL-6405 - SWS_Spi_00156 : General behaviour PUBLISHED	SWS_Spi_00156 : General behaviour
MCAL-6372 - SWS_Spi_00281 : General behaviour PUBLISHED	SWS_Spi_00281 : General behaviour

3.4.2 Features Not Supported / NON Compliance

- Supports only MSB based transfer modes(LSB is not supported).
- Data width more than 32 bits.
- In async mode of transfer only interrupt/polling based mode is supported. DMA based transfer mode is not supported.
- Supports additional configuration parameters, refer section generates global (Global Variables)
- Some SPI instances of device variants does not support master mode and are not pinned out externally. Please refer to SoC User Manual.

3.5 Assumptions

Below listed are assumed to be valid for this design/implementation, exceptions and other deviations are listed for each explicitly. Care should be taken to ensure these assumptions are addressed.

1. The functional clock to the SPI module is expected to be ON before calling any SPI service APIs. The SPI driver doesn’t perform any programming to enable the module functional clock.
2. Interrupt configuration for SPI interrupt registration and handling. SPI driver would only enable interrupt generation from SPI, a separate entity is expected to perform the required configuration.
3. SPI modules base addresses, register offsets and SOC specific defines are defined by CSL header files.

Note that assumption 1 & 2 are specified by AUTOSAR SPI specification. 3 is device specific assumption.

3.6 Constraints

- A job could belong to several sequences but can't be active at the same time i.e. a job queued in a sequence cannot be queued via another sequence. This is a design limitation to reduce driver complexity.
- A channel could belong to several sequences or jobs but can't be active at the same time i.e. a channel in a job in a sequence cannot be part of another active job or sequence. This is a design limitation to reduce driver complexity.
- Non-Interruptible sequence applies only within a HW unit. If a sequence is started, another high priority job belonging to another sequence cannot interrupt the job belonging to the same hardware unit. But the job can be scheduled ahead of another pending job of the started sequence if it belongs to another HW queue. This is illustrated in below example
  Example of non-interruptible sequence across HW units:

```
SEQ1 - JOB1 (HW1, P0)

SEQ2 - JOB2 (HW2, P0), JOB3 (HW1, P0)

SEQ3 - JOB4 (HW2, P3)
```

  Consider the above sequence of calls happening back to back at time T1. The job schedule for this case will be

Time T1 - JOB1 and JOB2 (Since different HW)
Time T2 - JOB4 (could interrupts SEQ2 JOB3 if JOB1 takes more time that JOB2)
Time T3 - JOB3

- SPI Handler/Driver handles only the Master mode and supports only full-duplex mode

Design Identifier	Description
MCAL-6398 - SWS_Spi_00040 : Limitation PUBLISHED	SWS_Spi_00040 : Limitation
MCAL-6692 - SWS_Spi_00050 : Limitation PUBLISHED	SWS_Spi_00050 : Limitation

3.7 Hardware and SW platforms

Hardware Platforms

- Refer to specified SoC User Manual to check if ADC module is supported.

Software Platforms

- Bare-Metal

3.8 Dependencies

1. SPI peripherals may depend on the system clock, prescaler(s) and PLL. Thus, changes of the system clock (e.g. PLL on , PLL off) may also affect the clock settings of the SPI hardware.
2. The SPI Handler/Driver module does not take care of setting the registers which configure the clock, prescaler(s) and PLL in its init function. This has to be done by the MCU module.
3. Depending on microcontrollers, the SPI peripheral could share registers with other peripherals. In this typical case, the SPI Handler/Driver has a relationship with MCU module for initializing and de-initializing those registers.
4. If Chip Selects are done using microcontroller pins the SPI Handler/Driver has a relationship with PORT module. In this case, this specification assumes that these microcontroller pins are directly accessed by the SPI Handler/Driver module without using APIs of DIO module. Anyhow, the SPI depends on ECU hardware design and for that reason it may depend on other modules.

Design Identifier	Description
MCAL-6498 - SWS_Spi_00244 : Dependency on other modules PUBLISHED	SWS_Spi_00244 : Dependency on other modules
MCAL-6449 - SWS_Spi_00342 : Dependency on other modules PUBLISHED	SWS_Spi_00342 : Dependency on other modules
MCAL-6448 - SWS_Spi_00343 : Dependency on other modules PUBLISHED	SWS_Spi_00343 : Dependency on other modules
MCAL-6581 - SWS_Spi_00191 : Dependency on other modules PUBLISHED	SWS_Spi_00191 : Dependency on other modules

Design Identifier	Description
MCAL-6451 - SWS_Spi_00239 : Dependency on other modules PUBLISHED	SWS_Spi_00239 : Dependency on other modules
MCAL-6383 - SWS_Spi_00339 : Dependency on other modules PUBLISHED	SWS_Spi_00339 : Dependency on other modules

3.9 Stakeholders

- Developers
- Test Engineers
- Customer Integrator

3.10 References

	Specification	Comment/Link
1	AUTOSAR 4.3.1	AUTOSAR Specification for SPI Driver.
2	BSW General Requirements / Coding guidelines	Autosar and Coding guidelines for the Mcal drivers.

	Specification	Comment/Link
3	Software Product Specification (SPS)	Product Functional requirements.
4	Software Architecture	Mcal Software Architecture.

4 Design Description

Refer AUTOSAR Specification mentioned in Reference 1 - AUTOSAR 4.3.1 section 1.4 for concepts such as channel, job, sequences.

Definition:	*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.

Scalability Levels in SPI Driver:

- LEVEL 0, Simple Synchronous SPI Handler/Driver: the communication is based on synchronous handling with a FIFO policy to handle multiple accesses. Buffer usage is configurable to optimize and/or to take advantage of HW capabilities. A simple synchronous transmission means that the function calling the transmission service is blocked during the ongoing transmission until the transmission is finished.
- LEVEL 1, Basic Asynchronous SPI Handler/Driver: the communication is based on asynchronous behavior and with a Priority policy to handle multiple accesses. Buffer usage is configurable as for “Simple Asynchronous” level. An asynchronous transmission means that the user calling the transmission service is not blocked when the transmission is on-going. Furthermore, the user can be notified at the end of transmission.

- LEVEL 2, Enhanced (Synchronous/Asynchronous) SPI Handler/Driver: the communication is based on asynchronous behavior or synchronous handling, using either interrupts or polling mechanism selectable during execution time and with a Priority policy to handle multiple accesses. Buffer usage is configurable as for other levels

4.1 Priority Handling and Job Queuing Operations

Priority mechanism is implemented using a pure software function as hardware priority mechanism is not supported by the SPI module. Priority is supported at job level in a sequence. As per the AUTOSAR SPI HandlerDriver SWS jobs are scheduled in order of priority. The priority of jobs determines their scheduling even across sequences as long as a sequence is marked as interruptible.The internal implementation of job priority based scheduling is based on priority queue implemented as a doubly linked list. All jobs are queued into a work queue per hardware unit. The hardware services the next job in the work queue by de-queuing from this work queue. The work queue implementation ensures the highest priority job is de-queued always. The implementation is in Spi_Utils.c file.

Design Identifier	Description
MCAL-6721 - SWS_Spi_00157 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00157 : Spi_AsyncTransmit Functional
MCAL-6747 - SWS_Spi_00270 : Scheduling Advices PUBLISHED	SWS_Spi_00270 : Scheduling Advices
MCAL-6715 - SWS_Spi_00088 : Scheduling Advices PUBLISHED	SWS_Spi_00088 : Scheduling Advices
MCAL-6551 - SWS_Spi_00269 : Scheduling Advices PUBLISHED	SWS_Spi_00269 : Scheduling Advices

Design Identifier	Description
MCAL-6404 - SWS_Spi_00268 : Scheduling Advices PUBLISHED	SWS_Spi_00268 : Scheduling Advices

4.2 Interrupt Service Routines

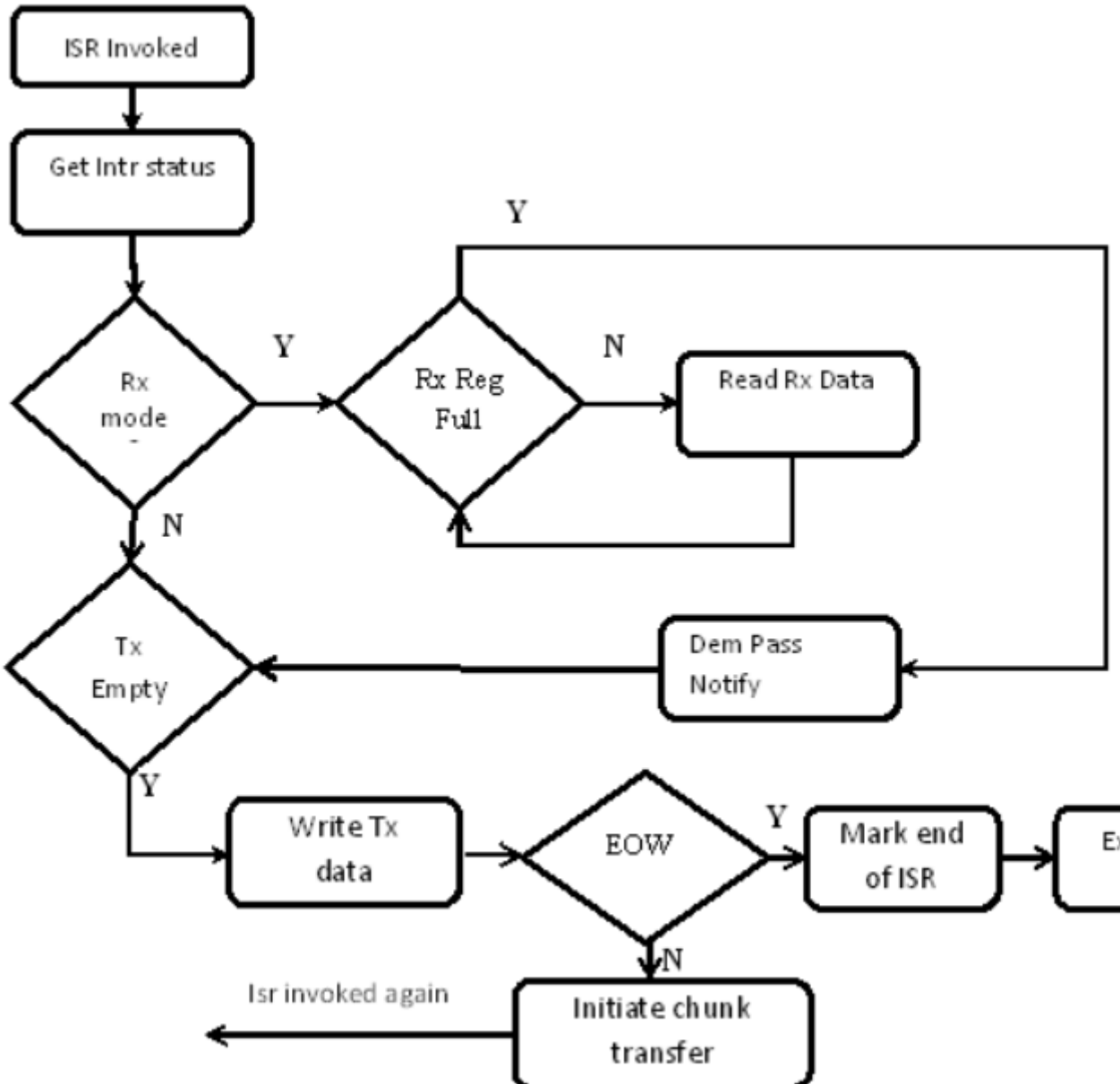
For each of the configured hardware units (MCSPI), one interrupt service routine has to be mapped. The Integrator has to map the interrupt service routines to the interrupt sources of the respective HW unit interrupt. The supported ISRs are part of the Spi_Irq.h file. Spi_Irq.h contains ISR for each of MCSPI hardware units. These should be used for the registration.

Handling MCSPI FIFO Interrupt: MCSPI Hardware Behavior: The hardware doesn't generate TX empty interrupt for the last chunk of data write when the write is not equal to the FIFO depth. This means that the EOW (End of Word) interrupt cannot be used for data transfer (TX) completion.

To handle scenario when the actual transfer size is not a multiple of FIFO size the following steps shall be performed.

- The transfer request needs to be split into two –
  - one with multiple of FIFO trigger level and
  - another with just with the reminder words
- For the last chunk transfer, the FIFO level is reconfigured to be equal to the chunk size in the ISR context. This will generate the EOW interrupt

From timing point of view, the only change with this two stage transfer is that, there will be a slightly different delay in between these two transfers compared to the full un-interrupted transfer. This is due to the ISR time and also we are waiting for the first transfer to fully complete (FIFO is fully drained). This delay is similar to the delay between channels in a job. So there is no real impact on performance.


ISR Invoked
Get Intr status
Rx mode
Y
Rx Reg Full
N
Read Rx Data
Y
N
Tx Empty
Dem Pass Notify
Y
Write Tx data
EOW
Y
Mark end of ISR
Exit
N
Isr invoked again
Initiate chunk transfer

4.3 Dynamic Behavior

The SPI driver at any time will be in one of the following states. The state transition will depend on the APIs invoked by the application

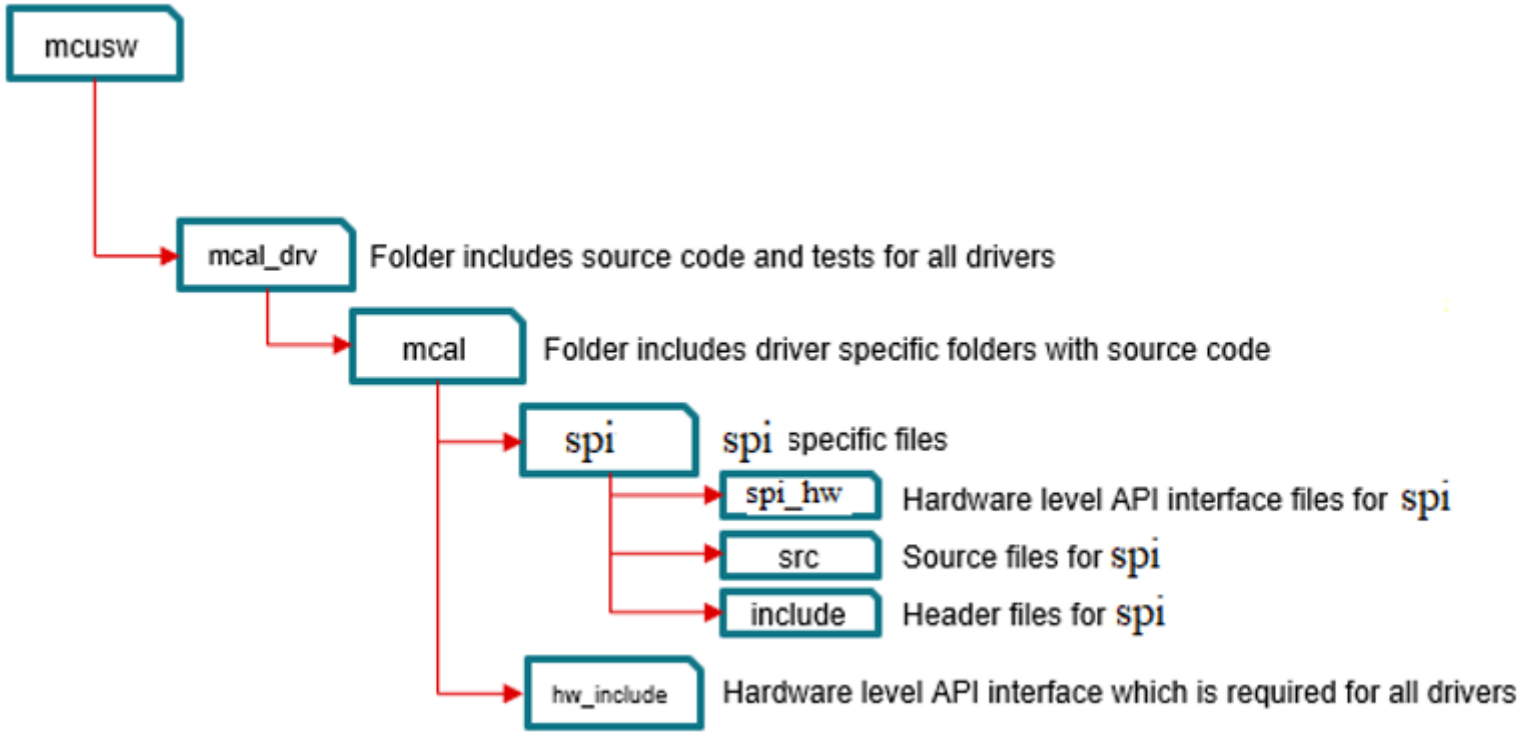
- SPI_UNINIT: The SPI Handler/Driver is not initialized or not usable. This is the state before starting driver initialization or after the driver is de-initialized.
- SPI_IDLE: The SPI Handler/Driver is not currently transmitting any Job. This is the state before the transmission is started or after the transmission is completed.
- SPI_BUSY: This is the state after a transmission has started i.e. transmission for the sequence is ongoing.

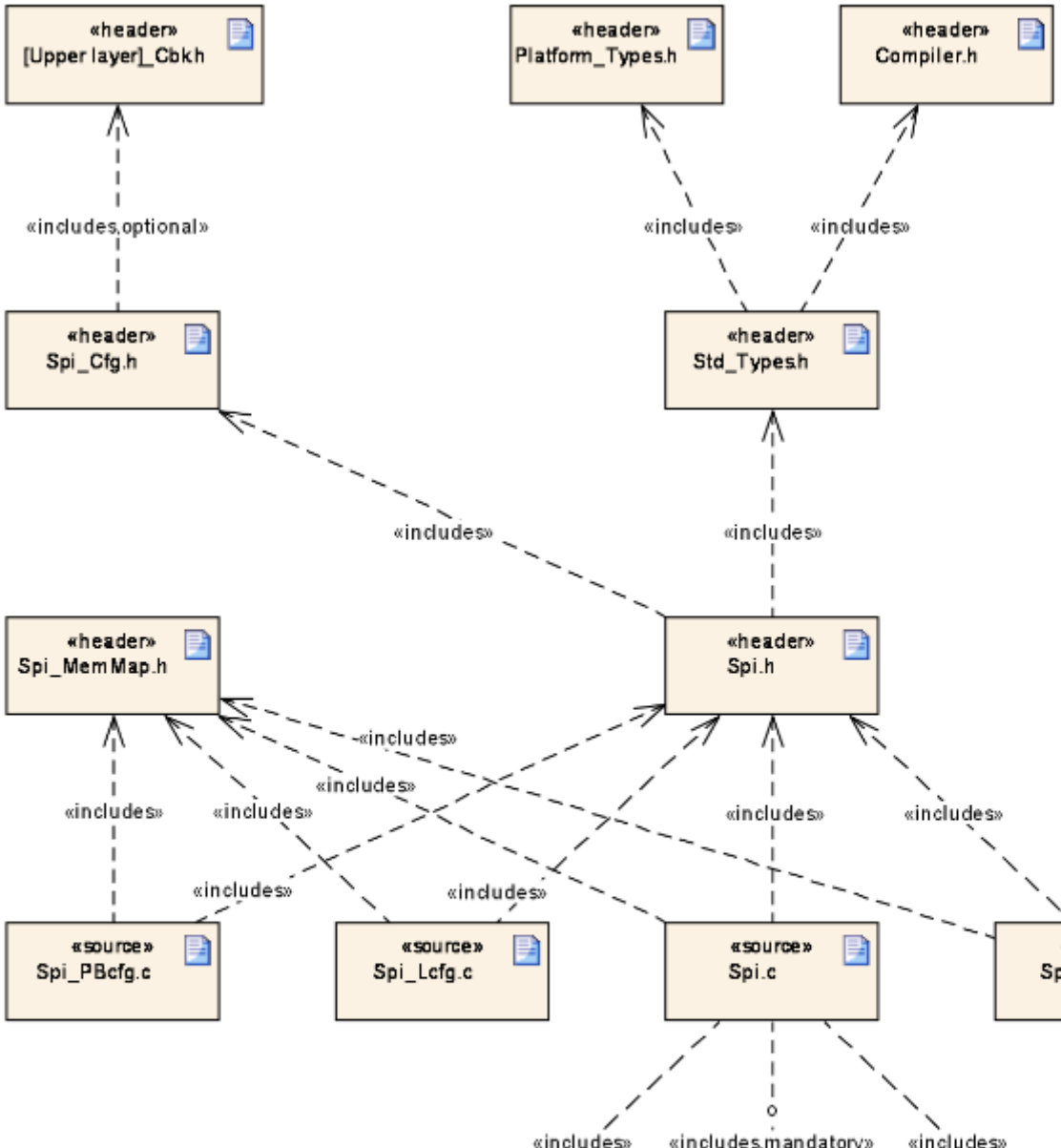
4.4 Directory Structure

The below diagram shows the overall files structure for the SPI driver. . The Spi.c and Spi.h are the 2 files that contain the SPI driver's APIs.

Driver Implemented by

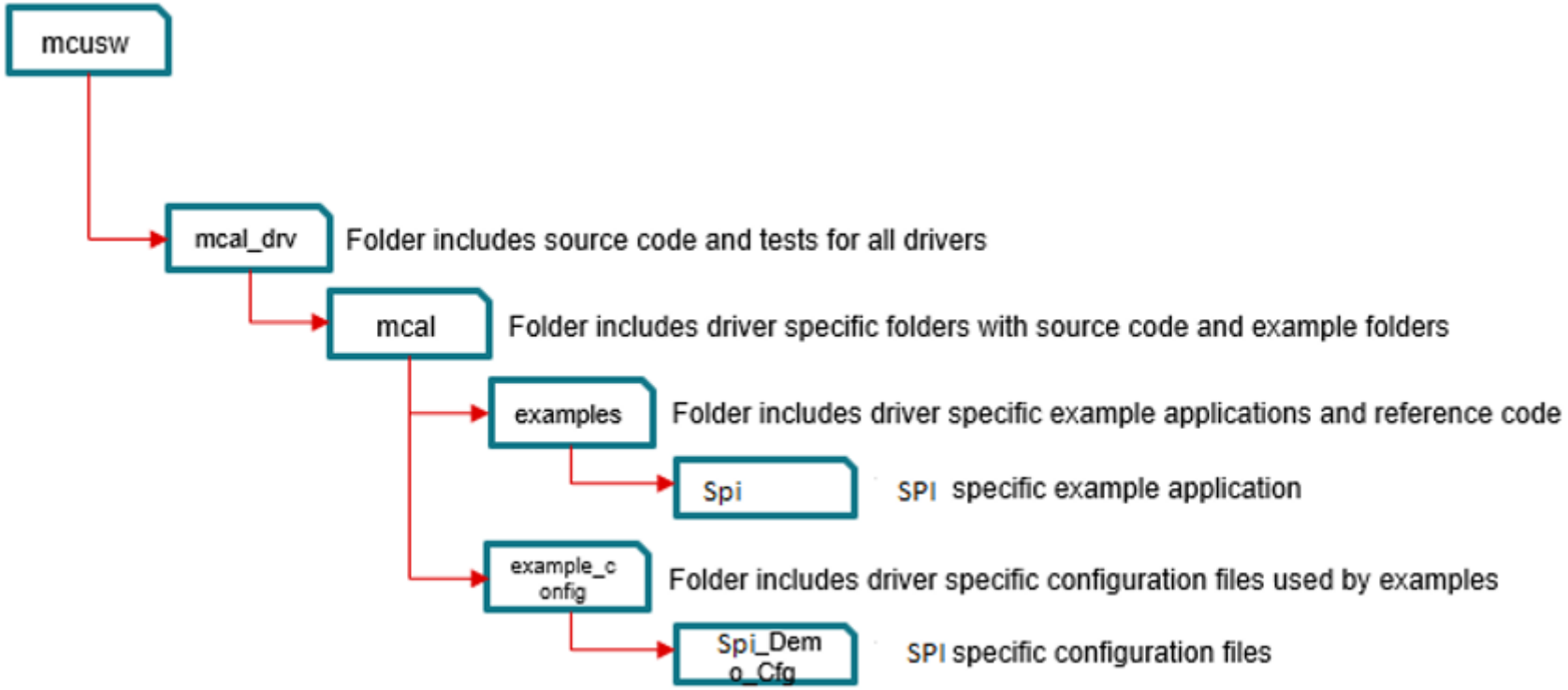
- Spi.h : Shall implement the interface provided by the SPI driver
- Spi_Irq.h Contains ISR function declaration.
- Spi_Dbg.h Contains debug variables.
- Spi.c, Spi_Irq.c, Spi_Mcspi.c, : Shall implement the driver functionality
- Spi_Priv.c, Spi_Priv.h, Spi_Utils.c,Spi_Utils.h : shall have the internal functions and data structures


mcusw
mcal_drv
Folder includes source code and tests for all drivers
mcal
Folder includes driver specific folders with source code
spi
spi specific files
spi_hw
Hardware level API interface files for spi
src
Source files for spi
include
Header files for spi
hw_include
Hardware level API interface which is required for all drivers


«header»
[Upper layer]_Cbk.h
«header»
Platform_Types.h
«header»
Compiler.h
«includes,optional»
«includes»
«includes»
«header»
Spi_Cfg.h
«header»
Std_Types.h
«includes»
«includes»
«header»
Spi_MemMap.h
«header»
Spi.h
«includes»
«includes»
«includes»
«includes»
«includes»
«includes»
«includes»
«source»
Spi_PBcfg.c
«source»
Spi_Lcfg.c
«source»
Spi.c
«source»
Spi_Irq.c
«includes»
«includes.mandatory»
«includes»

Example Application

- Spi_Cfg.h and Spi_Lcfg.c: Shall implement the generated configuration for link-time variant
- Spi_PBcfg.h : Shall implement the generated configuration for post-build variant
- McSpiApp.c: Shall implement the example application that demonstrates the use of the driver


mcusw
mcal_drv
Folder includes source code and tests for all drivers
mcal
Folder includes driver specific folders with source code and example folders
examples
Folder includes driver specific example applications and reference code
Spi
SPI specific example application
example_config
Folder includes driver specific configuration files used by examples
Spi_Demo_Cfg
SPI specific configuration files

Design Identifier	Description
MCAL-6685 - SWS_Spi_00159 : File structure PUBLISHED	SWS_Spi_00159 : File structure

4.5 Configurator

The AUTOSAR SPI Driver Specification details mandatory parameters that shall be configurable via the configurator. Please refer section 10 1. Configurator is common for all the device variants because HW IP between variants doesn't change. Error Checks are common for variants of devices and we can do this in xdm or generator code.

Design Identifier	Description
MCAL-6583 - ECUC_Spi_00240 : SpiDemEventParameterRefs PUBLISHED	ECUC_Spi_00240 : SpiDemEventParameterRefs
MCAL-6557 - ECUC_Spi_00241 : SPI_E_HARDWARE_ERROR PUBLISHED	ECUC_Spi_00241 : SPI_E_HARDWARE_ERROR
MCAL-6377 - ECUC_Spi_00091 : Container Name SpiDriver PUBLISHED	ECUC_Spi_00091 : Container Name SpiDriver
MCAL-6439 - ECUC_Spi_00197 : SpiMaxChannel PUBLISHED	ECUC_Spi_00197 : SpiMaxChannel

Design Identifier	Description
MCAL-6470 - ECUC_Spi_00198 : SpiMaxJob PUBLISHED	ECUC_Spi_00198 : SpiMaxJob
MCAL-6663 - ECUC_Spi_00199 : SpiMaxSequence PUBLISHED	ECUC_Spi_00199 : SpiMaxSequence
MCAL-6455 - ECUC_Spi_00104 : Container Name SpiChannel PUBLISHED	ECUC_Spi_00104 : Container Name SpiChannel
MCAL-6375 - ECUC_Spi_00200 : SpiChannelId PUBLISHED	ECUC_Spi_00200 : SpiChannelId
MCAL-6500 - ECUC_Spi_00201 : SpiChannelType PUBLISHED	ECUC_Spi_00202 : SpiDataWidth
MCAL-6602 - ECUC_Spi_00203 : SpiDefaultData PUBLISHED	ECUC_Spi_00203 : SpiDefaultData
MCAL-6494 - ECUC_Spi_00204 : SpiEbMaxLength PUBLISHED	ECUC_Spi_00204 : SpiEbMaxLength
MCAL-6693 - ECUC_Spi_00205 : SpiIbNBuffers PUBLISHED	ECUC_Spi_00205 : SpiIbNBuffers

Design Identifier	Description
MCAL-6746 - ECUC_Spi_00206 : SpiTransferStart PUBLISHED	ECUC_Spi_00206 : SpiTransferStart
MCAL-6388 - ECUC_Spi_00207 : SpiExternalDevice PUBLISHED	ECUC_Spi_00207 : SpiExternalDevice
MCAL-6612 - ECUC_Spi_00209 : SpiCsIdentifier PUBLISHED	ECUC_Spi_00209 : SpiCsIdentifier
MCAL-6432 - ECUC_Spi_00210 : SpiCsPolarity PUBLISHED	ECUC_Spi_00210 : SpiCsPolarity
MCAL-6546 - ECUC_Spi_00239 : SpiCsSelection PUBLISHED	ECUC_Spi_00239 : SpiCsSelection
MCAL-6624 - ECUC_Spi_00211 : SpiDataShiftEdge PUBLISHED	ECUC_Spi_00211 : SpiDataShiftEdge
MCAL-6549 - ECUC_Spi_00212 : SpiEnableCs PUBLISHED	ECUC_Spi_00212 : SpiEnableCs
MCAL-6515 - ECUC_Spi_00213 : SpiShiftClockIdleLevel PUBLISHED	ECUC_Spi_00213 : SpiShiftClockIdleLevel

Design Identifier	Description
MCAL-6516 - ECUC_Spi_00105 : Container Name SpiJob PUBLISHED	ECUC_Spi_00105 : Container Name SpiJob
MCAL-6701 - ECUC_Spi_00238 : SpiHwUnitSynchronous PUBLISHED	ECUC_Spi_00238 : SpiHwUnitSynchronous
MCAL-6709 - ECUC_Spi_00218 : SpiJobEndNotification PUBLISHED	ECUC_Spi_00218 : SpiJobEndNotification
MCAL-6480 - ECUC_Spi_00219 : SpiJobId PUBLISHED	ECUC_Spi_00219 : SpiJobId
MCAL-6569 - ECUC_Spi_00220 : SpiJobPriority PUBLISHED	ECUC_Spi_00220 : SpiJobPriority
MCAL-6427 - ECUC_Spi_00216 : SpiDeviceAssignment PUBLISHED	ECUC_Spi_00216 : SpiDeviceAssignment
MCAL-6705 - ECUC_Spi_00233 : Container Name SpiChannelList PUBLISHED	ECUC_Spi_00233 : Container Name SpiChannelList
MCAL-6674 - ECUC_Spi_00234 : SpiChannelIndex PUBLISHED	ECUC_Spi_00234 : SpiChannelIndex

Design Identifier	Description
MCAL-6575 - ECUC_Spi_00215 : SpiChannelAssignment PUBLISHED	ECUC_Spi_00215 : SpiChannelAssignment
MCAL-6647 - ECUC_Spi_00106 : Container Name SpiSequence PUBLISHED	ECUC_Spi_00106 : Container Name SpiSequence
MCAL-6511 - ECUC_Spi_00222 : SpiInterruptibleSequence PUBLISHED	ECUC_Spi_00222 : SpiInterruptibleSequence
MCAL-6401 - ECUC_Spi_00223 : SpiSeqEndNotification PUBLISHED	ECUC_Spi_00223 : SpiSeqEndNotification
MCAL-6518 - ECUC_Spi_00224 : SpiSequenceId PUBLISHED	ECUC_Spi_00224 : SpiSequenceId
MCAL-6380 - ECUC_Spi_00221 : SpiJobAssignment PUBLISHED	ECUC_Spi_00221 : SpiJobAssignment
MCAL-6403 - ECUC_Spi_00225 : Container Name SpiGeneral PUBLISHED	ECUC_Spi_00225 : Container Name SpiGeneral
MCAL-6749 - ECUC_Spi_00226 : SpiCancelApi PUBLISHED	ECUC_Spi_00226 : SpiCancelApi

Design Identifier	Description
MCAL-6668 - ECUC_Spi_00227 : SpiChannelBuffersAllowed PUBLISHED	ECUC_Spi_00227 : SpiChannelBuffersAllowed
MCAL-6438 - ECUC_Spi_00228 : SpiDevErrorDetect PUBLISHED	ECUC_Spi_00228 : SpiDevErrorDetect
MCAL-6741 - ECUC_Spi_00229 : SpiHwStatusApi PUBLISHED	ECUC_Spi_00229 : SpiHwStatusApi
MCAL-6651 - ECUC_Spi_00230 : SpiInterruptibleSeqAllowed PUBLISHED	ECUC_Spi_00230 : SpiInterruptibleSeqAllowed
MCAL-6391 - ECUC_Spi_00231 : SpiLevelDelivered PUBLISHED	ECUC_Spi_00231 : SpiLevelDelivered
MCAL-6555 - ECUC_Spi_00242 : SpiMainFunctionPeriod PUBLISHED	ECUC_Spi_00242 : SpiMainFunctionPeriod
MCAL-6402 - ECUC_Spi_00237 : SpiSupportConcurrentSyncTransmit PUBLISHED	ECUC_Spi_00237 : SpiSupportConcurrentSyncTransmit
MCAL-6730 - ECUC_Spi_00243 : SpiUserCallbackHeaderFile PUBLISHED	ECUC_Spi_00243 : SpiUserCallbackHeaderFile

Design Identifier	Description
MCAL-6479 - ECUC_Spi_00232 : SpiVersionInfoApi PUBLISHED	ECUC_Spi_00232 : SpiVersionInfoApi
MCAL-6671 - ECUC_Spi_00235 : Container Name SpiPublishedInformation PUBLISHED	ECUC_Spi_00235 : Container Name SpiPublishedInformation
MCAL-6424 - ECUC_Spi_00236 : SpiMaxHwUnit PUBLISHED	ECUC_Spi_00236 : SpiMaxHwUnit
MCAL-6743 - ECUC_Spi_00202 : SpiDataWidth PUBLISHED	ECUC_Spi_00202 : SpiDataWidth
MCAL-6550 - ECUC_Spi_00217 : SpiHwUnit PUBLISHED	ECUC_Spi_00217 : SpiHwUnit
MCAL-6681 - ECUC : SpiDeviceVariant PUBLISHED	ECUC : SpiDeviceVariant

4.5.1 NON Standard configurable parameters

The design's specific configurable parameters are as follows:

Parameter	Usage comment
SpiMaxHwUnit	Maximum number of HW unit.
SpiCsMode	This selcts which CS mode the device should enter. SPI_SINGLE: the CS will return to default level after one transmission unit has been sent. SPI_CONTINUOUS: the CS will remain active during the whole transmission.
SpiCsIdleTime	CS idle time (Timing between clock and chip select) if single mode is chosen.
SpiExtDeviceClockDivider	Clock divider. This is used to derive the required baudrate from the functional clock. This value should be 1 less than the actual divider value. So a value of 0 means the divider is 1
SpiExtenalDeviceConfigMCSPI	MCSPI HW specific external device config. Should be populated only if hwUnitId is of type MCSPI
SpiExtDeviceMCSPITxRxMode	TX and RX mode .RX only mode doesn't make sense in master mode because to receive data the master has to generate clock, which means it should transmit. Hence this mode is not supported. The user can alternatively set the TX buffer pointer to NULL and set the default TX value (defaultTxData) to make TX data line at the desired level.
SpiExtDeviceMCSPIStartBitEnable	Start bit D/CX added before SPI transfer. Polarity is defined by StartBitLevel
SpiExtDeviceMCSPIStartBitLevel	Start-bit polarity used when startBitEnable is TRUE.

Parameter	Usage comment
SpiHwUnitAssignment	Reference to the HW unit used by this job
SpiEnableJobLog	Enable/disable SPI job log
SpiMaxExternalDevices	Number of different SPI hardware microcontroller peripherals (units/busses) available and handled by this SPI Handler/Driver module.
SpiMaxJobLogLength	Maximum job log entries when logging is ON
SpiMaxChannelsPerJob	Maximum channels allowed per job
SpiMaxJobsPerSequence	Maximum jobs allowed per sequence
SpiMaxChannels	Maximum channels across all jobs/sequence/hwunit
SpiMaxJobs	Maximum jobs across all sequence/hwunit
SpiMaxSequences	Maximum sequence across all hwunit
SpiMaxExternalDeviceConfig	Maximum number of HW unit.
SpiHwUnitEnabledFlag	Group configurations

Parameter	Usage comment
SpiHwUnitEnabled	Structure for storing enabled SPI HW units
SpiIrqType	Type of Isr function: void functionname(void) CAT1 see description in tool : interrupt void func(void) CAT2 see description in tool : ISR(func)
SpiNotificationHeader	Header file providing Job End and Sequence End notification functions definitions.
SpiDefaultOSCounterId	Default Os Counter Id if node reference to OsCounter ref SpiOsCounterRef is not set
SpiOsCounterRef	This parameter contains a reference to the OsCounter, which is used by the SPI driver.
SpiTimeoutDuration	SPI timeout - used in SPI busy wait
SpiRegisterReadbackApi	Enable API to readback SPI critical registers
SpiSetLoopbackModeApi	Enable or Disable Internal loopback mode of SPI.Note: Only McSPI HW units supports this feature.
SpiChannelInternalBufferMaxLength	Internal Buffer length in bytes - applicable for SpiChannelBuffer type - SPI_IB. This is the maximum length that can be allocated by each channel and it is fixed.Can vary buffer length per channel by configuring SpiIbNBuffers and SpiDataWidth. Refer SWS_Spi_00437 and ECUC_Spi_00205

4.5.2 Variant Support

The driver shall support both VARIANT-LINK-TIME , VARIANT-PRE-COMPILE & VARIANT-POST-BUILD.

Design Identifier	Description
MCAL-6591 - SpiMaxHwUnit PUBLISHED	SpiMaxHwUnit
MCAL-6712 - SpiMaxExternalDeviceConfig PUBLISHED	SpiMaxExternalDeviceConfig
MCAL-6611 - SpiCsMode PUBLISHED	SpiCsMode
MCAL-6558 - SpiCsIdleTime PUBLISHED	SpiCsIdleTime
MCAL-6436 - SpiExtDeviceClockDivider PUBLISHED	SpiExtDeviceClockDivider
MCAL-6535 - SpiExtenalDeviceConfigMCSPI PUBLISHED	SpiExtenalDeviceConfigMCSPI
MCAL-6590 - SpiExtDeviceMCSPITxRxMode PUBLISHED	SpiExtDeviceMCSPITxRxMode

Design Identifier	Description
MCAL-6667 - SpiExtDeviceMCSPIStartBitEnable PUBLISHED	SpiExtDeviceMCSPIStartBitEnable
MCAL-6732 - SpiExtDeviceMCSPIStartBitLevel PUBLISHED	SpiExtDeviceMCSPIStartBitLevel
MCAL-6603 - SpiHwUnitAssignment PUBLISHED	SpiHwUnitAssignment
MCAL-6698 - SpiEnableJobLog PUBLISHED	SpiEnableJobLog
MCAL-6662 - SpiMaxExternalDevices PUBLISHED	SpiMaxExternalDevices
MCAL-6678 - SpiMaxJobLogLength PUBLISHED	SpiMaxJobLogLength
MCAL-6554 - SpiMaxChannelsPerJob PUBLISHED	SpiMaxChannelsPerJob
MCAL-6594 - SpiMaxJobsPerSequence PUBLISHED	SpiMaxJobsPerSequence

Design Identifier	Description
MCAL-6503 - SpiMaxChannels PUBLISHED	SpiMaxChannels
MCAL-6442 - SpiMaxJobs PUBLISHED	SpiMaxJobs
MCAL-6609 - SpiMaxSequences PUBLISHED	SpiMaxSequences
MCAL-6397 - SpiHwUnitEnabledFlag PUBLISHED	SpiHwUnitEnabledFlag
MCAL-6571 - SpiHwUnitEnabled PUBLISHED	SpiHwUnitEnabled
MCAL-6652 - SpiIrqType PUBLISHED	SpiIrqType
MCAL-6738 - SpiDefaultOSCounterId PUBLISHED	SpiDefaultOSCounterId
MCAL-6595 - SpiOsCounterRef PUBLISHED	SpiOsCounterRef

Design Identifier	Description
MCAL-6456 - SpiTimeoutDuration PUBLISHED	SpiTimeoutDuration
MCAL-6664 - SpiRegisterReadbackApi PUBLISHED	SpiRegisterReadbackApi
MCAL-6389 - SpiChannelInternalBufferMaxLength PUBLISHED	SpiChannelInternalBufferMaxLength
MCAL-6604 - SWS_Spi_00056 : Configuration Variant PUBLISHED	SWS_Spi_00056 : Configuration Variant
MCAL-6472 - SWS_Spi_00148 : Configuration Variant PUBLISHED	SWS_Spi_00148 : Configuration Variant
MCAL-6374 - SWS_Spi_00076 : Configuration Variant PUBLISHED	SWS_Spi_00076 : Configuration Variant
MCAL-6737 - SWS_Spi_00109 : Functional Overview PUBLISHED	SWS_Spi_00109 : Functional Overview
MCAL-6695 - SWS_Spi_00110 : Functional Overview PUBLISHED	SWS_Spi_00110 : Functional Overview

4.6 Error Classification

Errors are classified in two categories, development error and runtime / production error. All runtime errors are reported to Det_ReportRuntimeError service of the Development Error Tracer (DET).

Design Identifier	Description
MCAL-6621 - SWS_Spi_00386 : Error classification - Extended Production Errors PUBLISHED	SWS_Spi_00386 : Error classification - Extended Production Errors
MCAL-6600 - SWS_Spi_00195 : Error classification - Extended Production Errors PUBLISHED	SWS_Spi_00195 : Error classification - Extended Production Errors
MCAL-6574 - SWS_Spi_00367 : Error classification - Debugging PUBLISHED	SWS_Spi_00367 : Error classification - Debugging
MCAL-6446 - SWS_Spi_00385 : Error classification - Extended Production Errors PUBLISHED	SWS_Spi_00385 : Error classification - Extended Production Errors
MCAL-6376 - SWS_Spi_00383 : Error classification - Extended Production Errors PUBLISHED	SWS_Spi_00383 : Error classification - Extended Production Errors

4.6.1 Error Detection

The detection of development errors is configurable (ON / OFF) at pre-compile time. The switch SpiDevErrorDetect will activate or deactivate the detection of all development errors.

4.6.2 Development Errors

Development Error Reporting

By default, development errors are reported to the DET using the service Det_ReportError(), if development error detection and reporting are enabled (i.e. checkboxes Development Mode and Development Error Reporting are checked). The reported module SPI ID is 83. The reported service IDs identify the services which are described earlier. The following table presents the service IDs and the related services:

Service ID	***Service***
0x00	Spi_Init
0x01	Spi_DeInit
0x02	Spi_WriteIB
0x03	Spi_AsyncTransmit
0x04	Spi_ReadIB

0x05	Spi_SetupEB
0x06	Spi_GetStatus
0x07	Spi_GetJobResult
0x08	Spi_GetSequenceResult
0x09	Spi_GetVersionInfo
0x0A	Spi_SyncTransmit
0x0B	Spi_GetHWUnitStatus
0x0C	Spi_Cancel
0x0D	Spi_SetAsyncMode
0x0E	Spi_MainFunction_Handling

4.6.3 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in table are internal parameter checks of the API functions. These checks are for development error reporting and can be enabled or disabled. ECUC parameters error checks are handled as per ECUC Parameter checking in configurator . The deviations should be documented in user guide.

Design Identifier	Description
MCAL-6666 - SWS_Spi_00004 : Error classification - Development Errors PUBLISHED	SWS_Spi_00004 : Error classification - Development Errors
MCAL-6576 - SWS_Spi_00237 : Error classification - Development Errors PUBLISHED	SWS_Spi_00237 : Error classification - Development Errors
MCAL-6440 - SWS_Spi_00238 : Error classification - Development Errors PUBLISHED	SWS_Spi_00238 : Error classification - Development Errors
MCAL-6638 - SWS_Spi_00240 : Error classification - Development Errors PUBLISHED	SWS_Spi_00240 : Error classification - Development Errors

Design Identifier	Description
MCAL-6460 - SWS_Spi_00241 : Error classification - Development Errors PUBLISHED	SWS_Spi_00241 : Error classification - Development Errors
MCAL-6585 - SWS_Spi_00242 : Error classification - Development Errors PUBLISHED	SWS_Spi_00242 : Error classification - Development Errors
MCAL-6629 - SWS_Spi_00243 : Error classification - Development Errors PUBLISHED	SWS_Spi_00243 : Error classification - Development Error
MCAL-6706 - SWS_Spi_00245 : Error classification - Development Errors PUBLISHED	SWS_Spi_00245 : Error classification - Development Errors
MCAL-6530 - SWS_Spi_00246 : Error classification - Development Errors PUBLISHED	SWS_Spi_00246 : Error classification - Development Errors

Design Identifier	Description
MCAL-6614 - SWS_Spi_00031 : Error classification - API parameter checking PUBLISHED	SWS_Spi_00031 : Error classification - API parameter checking
MCAL-6474 - SWS_Spi_00032 : Error classification - API parameter checking PUBLISHED	SWS_Spi_00032 : Error classification - API parameter checking
MCAL-6462 - SWS_Spi_00254 : Error classification - API parameter checking PUBLISHED	SWS_Spi_00254 : Error classification - API parameter checking
MCAL-6548 - SWS_Spi_00060 : Error classification - API parameter checking PUBLISHED	SWS_Spi_00060 : Error classification - API parameter checking
MCAL-6708 - SWS_Spi_00258 : Error classification - API parameter checking PUBLISHED	SWS_Spi_00258 : Error classification - API parameter checking

Design Identifier	Description
MCAL-6633 - SWS_Spi_00143 : Error classification - API parameter checking PUBLISHED	SWS_Spi_00143 : Error classification - API parameter checking
MCAL-6431 - SWS_Spi_00288 : Error classification - API parameter checking PUBLISHED	SWS_Spi_00288 : Error classification - API parameter checking
MCAL-6490 - SWS_Spi_00046 : Error classification - SPI state checking PUBLISHED	SWS_Spi_00046 : Error classification - SPI state checking
MCAL-6492 - SWS_Spi_00256 : Error classification - SPI state checking PUBLISHED	SWS_Spi_00256 : Error classification - SPI state checking
MCAL-6587 - SWS_Spi_00233 : Error classification - SPI state checking PUBLISHED	SWS_Spi_00233 : Error classification - SPI state checking
MCAL-6658 - SWS_Spi_00389: Spi Det_ReportRuntimeError PUBLISHED	SWS_Spi_00389: Spi Det_ReportRuntimeError

Type of Error	*Related Error code*	*Value (Hex)*
Channel out of bounds, exceeds the maximum number of configured channels	SPI_E_PARAM_CHANNEL	0x0A
Job out of bounds, exceeds the maximum number of configured jobs	SPI_E_PARAM_JOB	0x0B
Sequence out of bounds, exceeds the maximum number of configured sequences	SPI_E_PARAM_SEQ	0x0C
Length out of bounds, exceeds the maximum number of configured EB- or IB- buffer length	SPI_E_PARAM_LENGTH	0x0D
The requested hardware unit does not exist	SPI_E_PARAM_UNIT	0x0E
An invalid configuration has been passed (i.e. a NULL_PTR). This is an extension to AUTOSAR.	SPI_E_PARAM_POINTER	0x10
A service was requested, but the driver has not been initialized	SPI_E_UNINIT	0x1A
The requested sequence is still pending	SPI_E_SEQ_PENDING	0x2A
Transmission of synchronous sequence in progress (not supported)	SPI_E_SEQ_IN_PROCESS	0x3A

The driver is already initialized.	SPI_E_ALREADY_INITIALIZED	0x4A

4.6.4 Debugging support

SPI driver makes Spi_JobLogObj available for debugging.

4.6.5 Error Handling : Receive FIFO/Buffer overflow

The MCSPI module supports Rx overflow interrupt generation. SPI driver uses this feature for reporting RX FIFO overflow error. This is detected when it is enabled in hardware and receiver register or FIFO becomes filled.

MCSPI module support uses of FIFO for receive and transmit operations. The FIFO is shared between Rx and TX. If FIFO is enabled receive overrun interrupt indicates FIFO full for receive. SPI driver reports this to application and stops processing any further transfers.

5 Implementation Details

5.1 Data structures and resources

MACROS, Data Types & Structures

The sections below lists some of key data structures that shall be implemented and used in driver implementation

Macro	Comments
SPI_CHANNELBUFFERS	Buffer mode - Internal or External or Both.
SPI_MAX_CHANNELS	Maximum channels across all jobs/sequence/hwunit
SPI_MAX_JOBS	Maximum jobs across all sequence/hwunit
SPI_MAX_SEQ	Maximum sequence across all hwunit
SPI_MAX_HW_UNIT	Maximum HW unit
SPI_MAX_EXT_DEV	Maximum external device cfg

Spi_ConfigType

This type of the external data structure contains the initialization data for the SPI Handler/Driver., please refer section 8.2.1 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_StatusType

This type defines a range of specific status for SPI Handler/Driver, please refer section 8.2.2 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_JobResultType

Enumeration This type defines a range of specific Jobs status for SPI Handler/Driver, Please refer section 8.2.3 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_SeqResultType

Enumeration This type defines a range of specific Sequences status for SPI Handler/Driver, Please refer section 8.2.4 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_DataBufferType

Used to specify the type of application data buffer elements, please refer section 8.2.5 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_NumberOfDataType

Used to specify Type for defining the number of data elements of the type Spi_DataBufferType to send and / or receive by Channel. please refer section 8.2.6 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_ChannelType

Used to specify the identification (ID) for a Channel. please refer section 8.2.7 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_JobType

Used to specify the identification (ID) for a Job. please refer section 8.2.8 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_SequenceType

Used to specify the identification (ID) for a sequence of jobs. please refer section 8.2.9 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_HWUnitType

Used to specify the identification (ID) for a SPI Hardware microcontroller peripheral (unit). please refer section 8.2.10 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Spi_AsyncModeType

Used to specify the asynchronous mechanism mode for SPI busses handled asynchronously in LEVEL 2. please refer section 8.2.11 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

5.1.1 Global Variables

This design expects that implementation will require to use following global variables.

Variable	Type	Description	Default Value
Spi_DrvStatus	uint32	SPI driver status	SPI_UNINIT
Spi_DrvObj	Spi_DriverObjType	SPI driver object	-
Spi_JobLogObj	Spi_JobLogType	SPI log object	-
Spi_HwUnitBaseAddr	uint32	Base address array for HW units	-

5.2 Dynamic Behavior - Control Flow Diagram

Not Applicable

5.3 Dynamic Behavior - Data Flow Diagram

Not Applicable

5.4 Application Parameters

Spi_Init

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
CfgPtr	Pointer to configuration set	0xFFFFFFFF	-	-	N.A

Spi_GetJobResult

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Job	Job ID. An invalid job ID will return an undefined result.	0-10	-	-	N.A

Spi_GetSequenceResult

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Sequence	Sequence ID. An invalid sequence ID will return an undefined result.	0-10	-	-	N.A

Spi_GetHWUnitStatus

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
HWUnit	SPI Hardware microcontroller peripheral (unit) ID	0-11	-	-	N.A

Spi_SetupEB

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Channel	Channel ID.	0-24	-	-	N.A
SrcDataBufferPtr	Pointer to source data buffer.	0xFFFFFFFF	-	-	N.A

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Length	Length (number of data elements) of the data to be transmitted from SrcDataBufferPtr and/or received from DesDataBufferPtr Min.: 1 Max.: Max of data specified at configuration for this channel	0-64	-	-	N.A
DesDataBufferPtr	Pointer to destination data buffer in RAM.	0xFFFFFFFF	-	-	N.A

Spi_AsyncTransmit

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Sequence	Sequence ID.	0-10	-	-	N.A

Spi_Cancel

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Sequence	Sequence ID.	0-10	-	-	N.A

Spi_SyncTransmit

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Sequence	Sequence ID.	0-10	-	-	N.A

Spi_SetAsyncMode

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Mode	User Mode (Non-Privileged Mode)	-	-	-	N.A

Spi_GetVersionInfo

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
versioninfo	Pointer to store the version information of this module.	-	-	-	N.A

Spi_WriteIB

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Channel	Channel ID.	0-24	-	-	N.A

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
DataBufferPtr	Pointer to source data buffer. If this pointer is null, it is assumed that the data to be transmitted is not relevant and the default transmit value of this channel will be used instead.	0xFFFFFFFF	-	-	N.A

Spi_ReadIB

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Channel	Channel ID.	0-24	-	-	N.A
DataBufferPointer	Pointer to destination data buffer in RAM	0xFFFFFFFF	-	-	N.A

Spi_RegisterReadback

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
HWUnit	SPI Hardware microcontroller peripheral unit ID. If this is invalid, then the API will return E_NOT_OK.	0-11	-	-	N.A

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
RegRbPtr	Pointer to where to store the readback values. If this pointer is NULL_PTR, then the API will return E_NOT_OK.	0xFFFFFFFF	-	-	N.A

5.5 Safety Diagnostic Features

SPI3 - Periodic Software Readback of Static Configuration Registers / SPI4 - Software Readback of Written Configuration:

Software Readback of Written Configuration ensures that the configuration register are written with the expected value. Periodic readback of configuration registers can provide a diagnostic for inadvertent writes to these registers.

The SPI MCAL driver provides the API - **Spi_RegisterReadback** to readback static and written configuration registers to implement this diagnostic feature

SPI6 - Data Overflow/ SPI7 - Underflow Detection

The MCAL Hardware logic detects the data overflow/underflow error detection, The error event is generated as interrupt to CPU. SPI MCAL driver provides the APIs to enable the interrupt and to read the status of the interrupt register to check whether Overflow/Underflow event has occurred.

SPI Mcal provides the API **Spi_dataOverflowUnderflowIntrEnable** to enable the interrupt, **Spi_dataOverflowUnderflowIntrStatusGet** to read status of the interrupt register, **Spi_dataOverflowUnderflowIntrStatusClear** to clear the interrupt status and **Spi_dataOverflowUnderflowIntrDisable** to disable the interrupt.

Design Identifier	Description
MCAL-6538 - SPI: Safety Diagnostics: SPI6: Data Overflow Detection PUBLISHED	SPI: Safety Diagnostics: SPI6: Data Overflow Detection
MCAL-6433 - SPI: Safety Diagnostics: SPI4: Software Readback of Written Configuration PUBLISHED	SPI: Safety Diagnostics: SPI4: Software Readback of Written Configuration
MCAL-6392 - SPI: Safety Diagnostics: SPI7: Data Underflow Detection PUBLISHED	SPI: Safety Diagnostics: SPI7: Data Underflow Detection
MCAL-6586 - SPI: Safety Diagnostics: SPI3: Periodic Software Readback of Static Configuration Registers PUBLISHED	SPI: Safety Diagnostics: SPI3: Periodic Software Readback of Static Configuration Registers

6 Low Level Definitions

The detailed API and interface description is available as part of of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1. This section describes the API supported by the MCAL driver and the requirements covered by each of the API.

Design Identifier	Description
MCAL-6691 - SWS_Spi_00376 : DataBuffer Type definition PUBLISHED	WS_Spi_00376 : DataBuffer Type definition
MCAL-6394 - SWS_Spi_00355 : DataBuffer Type definition PUBLISHED	SWS_Spi_00355 : DataBuffer Type definition
MCAL-6615 - SWS_Spi_00164 : DataBuffer Type definition PUBLISHED	SWS_Spi_00164 : DataBuffer Type definition
MCAL-6625 - SWS_Spi_00377 : NumberOfData Type definition PUBLISHED	SWS_Spi_00377 : NumberOfData Type definition
MCAL-6669 - SWS_Spi_00165 : NumberOfData Type definition PUBLISHED	SWS_Spi_00165 : NumberOfData Type definition
MCAL-6528 - SWS_Spi_00378 : Channel Type definition PUBLISHED	SWS_Spi_00378 : Channel Type definition

Design Identifier	Description
MCAL-6687 - SWS_Spi_00166 : Channel Type definition PUBLISHED	SWS_Spi_00356 : Channel Type definition
MCAL-6687 - SWS_Spi_00166 : Channel Type definition PUBLISHED	SWS_Spi_00166 : Channel Type definition
MCAL-6457 - SWS_Spi_00379 : Job Type definition PUBLISHED	SWS_Spi_00379 : Job Type definition
MCAL-6689 - SWS_Spi_00357 : Job Type definition PUBLISHED	SWS_Spi_00357 : Job Type definition
MCAL-6728 - SWS_Spi_00167 : Job Type definition PUBLISHED	SWS_Spi_00167 : Job Type definition
MCAL-6639 - SWS_Spi_00380 : Sequence Type definition PUBLISHED	SWS_Spi_00380 : Sequence Type definition
MCAL-6505 - SWS_Spi_00358 : Sequence Type definition PUBLISHED	SWS_Spi_00358 : Sequence Type definition
MCAL-6729 - SWS_Spi_00168 : Sequence Type definition PUBLISHED	SWS_Spi_00168 : Sequence Type definition

Design Identifier	Description
MCAL-6655 - SWS_Spi_00381 : HW Unit Type definition PUBLISHED	SWS_Spi_00381 : HW Unit Type definition
MCAL-6682 - SWS_Spi_00359 : HW Unit Type definition PUBLISHED	SWS_Spi_00359 : HW Unit Type definition
MCAL-6707 - SWS_Spi_00169 : HW Unit Type definition PUBLISHED	SWS_Spi_00169 : HW Unit Type definition
MCAL-6699 - SWS_Spi_00108 : Limitation PUBLISHED	SWS_Spi_00108 : Limitation
MCAL-6476 - SWS_Spi_00192 : Call-Back Functions PUBLISHED	SWS_Spi_00192 : Call-Back Functions
MCAL-6413 - SWS_Spi_00193 : Call-Back Functions PUBLISHED	SWS_Spi_00193 : Call-Back Functions
MCAL-6531 - SWS_Spi_00373 : Status Type definition PUBLISHED	SWS_Spi_00373 : Status Type definition
MCAL-6648 - SWS_Spi_00061 : Status Type definition PUBLISHED	SWS_Spi_00061 : Status Type definition

Design Identifier	Description
MCAL-6537 - SWS_Spi_00011 : Status Type definition PUBLISHED	SWS_Spi_00011 : Status Type definition
MCAL-6703 - SWS_Spi_00374 : JobResult Type definition PUBLISHED	SWS_Spi_00374 : JobResult Type definition
MCAL-6425 - SWS_Spi_00062 : JobResult Type definition PUBLISHED	SWS_Spi_00062 : JobResult Type definition
MCAL-6430 - SWS_Spi_00012 : JobResult Type definition PUBLISHED	SWS_Spi_00012 : JobResult Type definition
MCAL-6512 - SWS_Spi_00375 : SeqResult Type definition PUBLISHED	SWS_Spi_00375 : SeqResult Type definition
MCAL-6607 - SWS_Spi_00019 : SeqResult Type definition PUBLISHED	SWS_Spi_00019 : SeqResult Type definition
MCAL-6686 - SWS_Spi_00017 : SeqResult Type definition PUBLISHED	SWS_Spi_00017 : SeqResult Type definition
MCAL-6502 - SWS_Spi_00382 : AsyncMode Type definition PUBLISHED	SWS_Spi_00382 : AsyncMode Type definition

Design Identifier	Description
MCAL-6659 - SWS_Spi_00360 : AsyncMode Type definition PUBLISHED	SWS_Spi_00360 : AsyncMode Type definition
MCAL-6420 - SWS_Spi_00170 : AsyncMode Type definition PUBLISHED	SWS_Spi_00170 : AsyncMode Type definition
MCAL-6475 - SWS_Spi_00150 : AsyncMode Type definition PUBLISHED	SWS_Spi_00150 : AsyncMode Type definition
MCAL-6419 - SWS_Spi_00361 : AsyncMode Type definition PUBLISHED	SWS_Spi_00361 : AsyncMode Type definition
MCAL-6517 - SWS_Spi_00362 : AsyncMode Type definition PUBLISHED	SWS_Spi_00362 : AsyncMode Type definition
MCAL-6597 - SWS_Spi_00093 : General behaviour PUBLISHED	SWS_Spi_00093 : General behaviour
MCAL-6529 - SWS_Spi_00111 : General behaviour PUBLISHED	SWS_Spi_00111 : General behaviour
MCAL-6519 - SWS_Spi_00279 : General behaviour PUBLISHED	SWS_Spi_00279 : General behaviour

Design Identifier	Description
MCAL-6649 - SWS_Spi_00112 : General behaviour PUBLISHED	SWS_Spi_00112 : General behaviour
MCAL-6716 - SWS_Spi_00280 : General behaviour PUBLISHED	SWS_Spi_00280 : General behaviour
MCAL-6619 - SWS_Spi_00063 : Config Type definition PUBLISHED	SWS_Spi_00063 : Config Type definition
MCAL-6692 - SWS_Spi_00050 : Limitation PUBLISHED	SWS_Spi_00050 : Limitation
MCAL-6437 - SWS_Spi_00066 : General behaviour PUBLISHED	SWS_Spi_00066 : General behaviour
MCAL-6717 - SWS_Spi_00263 : General behaviour PUBLISHED	SWS_Spi_00263 : General behaviour
MCAL-6650 - SWS_Spi_00370 : General behaviour PUBLISHED	SWS_Spi_00370 : General behaviour
MCAL-6684 - SWS_Spi_00368 : General behaviour PUBLISHED	SWS_Spi_00368 : General behaviour

Design Identifier	Description
MCAL-6522 - SWS_Spi_00262 : General behaviour PUBLISHED	SWS_Spi_00262 : General behaviour
MCAL-6572 - SWS_Spi_00002 : General behaviour PUBLISHED	SWS_Spi_00002 : General behaviour
MCAL-6406 - SWS_Spi_00009 : Config Type definition PUBLISHED	SWS_Spi_00009 : Config Type definition
MCAL-6632 - SWS_Spi_00010 : Config Type definition PUBLISHED	SWS_Spi_00010 : Config Type definition
MCAL-6496 - SWS_Spi_00236 : General behaviour PUBLISHED	SWS_Spi_00236 : General behaviour
MCAL-6536 - SWS_Spi_00064 : Config Type definition PUBLISHED	SWS_Spi_00064 : Config Type definition
MCAL-6742 - SWS_Spi_00121 : General behaviour PUBLISHED	SWS_Spi_00121 : General behaviour
MCAL-6570 - SWS_Spi_00372 : Config Type definition PUBLISHED	SWS_Spi_00372 : Config Type definition

Design Identifier	Description
MCAL-6423 - SWS_Spi_00344 : Config Type definition PUBLISHED	SWS_Spi_00344 : Config Type definition
MCAL-6588 - SWS_Spi_00008 : Config Type definition PUBLISHED	SWS_Spi_00008 : Config Type definition
MCAL-6485 - SWS_Spi_00044 : Call-Back Functions PUBLISHED	SWS_Spi_00044 : Call-Back Functions
MCAL-6733 - SWS_Spi_00048 : Call-Back Functions PUBLISHED	SWS_Spi_00048 : Call-Back Functions
MCAL-6426 - SWS_Spi_00345 : Status Type definition PUBLISHED	SWS_Spi_00345 : Status Type definition
MCAL-6718 - SWS_Spi_00075 : Call-Back Functions PUBLISHED	SWS_Spi_00075 : Call-Back Functions
MCAL-6735 - SWS_Spi_00071 : Call-Back Functions PUBLISHED	SWS_Spi_00071 : Call-Back Functions
MCAL-6592 - SWS_Spi_00340 : Call-Back Functions PUBLISHED	SWS_Spi_00340 : Call-Back Functions

Design Identifier	Description
MCAL-6534 - SWS_Spi_00264 : Call-Back Functions PUBLISHED	SWS_Spi_00264 : Call-Back Functions
MCAL-6483 - SWS_Spi_00054 : Call-Back Functions PUBLISHED	SWS_Spi_00054 : Call-Back Functions
MCAL-6443 - SWS_Spi_00341 : Call-Back Functions PUBLISHED	SWS_Spi_00341 : Call-Back Functions
MCAL-6390 - SWS_Spi_00265 : Call-Back Functions PUBLISHED	SWS_Spi_00265 : Call-Back Functions
MCAL-6527	[spi] QSPI config channel mode type not needed.
MCAL-6493 - SWS_Spi_00356 : Channel Type definition PUBLISHED	SWS_Spi_00356 : Channel Type definition

6.1 Driver API's

For the standard APIs please refer 8.3 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1. Sections below highlight other design considerations for the implementation.

6.1.1 Spi_Init

Refer section 8.3.1 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6613 - SWS_Spi_00299 : Spi_Init Functional PUBLISHED	SWS_Spi_00299 : Spi_Init Functional
MCAL-6620 - SWS_Spi_00151 : Spi_Init Functional PUBLISHED	SWS_Spi_00151 : Spi_Init Functional
MCAL-6466 - SWS_Spi_00015 : Spi_Init Functional PUBLISHED	SWS_Spi_00015 : Spi_Init Functional
MCAL-6481 - SWS_Spi_00298 : Spi_Init Functional PUBLISHED	SWS_Spi_00298 : Spi_Init Functional
MCAL-6654 - SWS_Spi_00013 : Spi_Init Functional PUBLISHED	SWS_Spi_00013 : Spi_Init Functional
MCAL-6411 - SWS_Spi_00082 : Spi_Init Functional PUBLISHED	SWS_Spi_00082 : Spi_Init Functional
MCAL-6382 - SWS_Spi_00235 : Configuration Variant PUBLISHED	SWS_Spi_00235 : Configuration Variant

Design Identifier	Description
MCAL-6530 - SWS_Spi_00246 : Error classification - Development Errors PUBLISHED	SWS_Spi_00246 : Error classification - Development Errors
MCAL-6587 - SWS_Spi_00233 : Error classification - SPI state checking PUBLISHED	SWS_Spi_00233 : Error classification - SPI state checking
MCAL-6567 - SWS_Spi_00024 : Spi_WriteIB Functional PUBLISHED	SWS_Spi_00024 : Spi_WriteIB Functional
MCAL-6498 - SWS_Spi_00244 : Dependency on other modules PUBLISHED	SWS_Spi_00244 : Dependency on other modules

6.1.2 Spi_DeInit

Refer section 8.3.2 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6605 - SWS_Spi_00253 : Spi_DeInit Functional PUBLISHED	SWS_Spi_00253 : Spi_DeInit Functional
MCAL-6680 - SWS_Spi_00303 : Spi_DeInit Functional PUBLISHED	SWS_Spi_00303 : Spi_DeInit Functional
MCAL-6453 - SWS_Spi_00022 : Spi_DeInit Functional PUBLISHED	SWS_Spi_00022 : Spi_DeInit Functional
MCAL-6653 - SWS_Spi_00021 : Spi_DeInit Functional PUBLISHED	SWS_Spi_00021 : Spi_DeInit Functional
MCAL-6722 - SWS_Spi_00300 : Spi_DeInit Functional PUBLISHED	SWS_Spi_00300 : Spi_DeInit Functional
MCAL-6378 - SWS_Spi_00301 : Spi_DeInit Functional PUBLISHED	SWS_Spi_00301 : Spi_DeInit Functional
MCAL-6568 - SWS_Spi_00302 : Spi_DeInit Functional PUBLISHED	SWS_Spi_00302 : Spi_DeInit Functional
MCAL-6734 - SWS_Spi_00252 : Spi_DeInit Functional PUBLISHED	SWS_Spi_00252 : Spi_DeInit Functional

6.1.3 Spi_GetStatus

Refer 8.3.7 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6702 - SWS_Spi_00320 : Spi_GetStatus Functional PUBLISHED	SWS_Spi_00320 : Spi_GetStatus Functional
MCAL-6582 - SWS_Spi_00319 : Spi_GetStatus Functional PUBLISHED	SWS_Spi_00319 : Spi_GetStatus Functional
MCAL-6688 - SWS_Spi_00025 : Spi_GetStatus Functional PUBLISHED	SWS_Spi_00025 : Spi_GetStatus Functional
MCAL-6426 - SWS_Spi_00345 : Status Type definition PUBLISHED	SWS_Spi_00345 : Status Type definition
MCAL-6714 - SWS_Spi_00347 : Status Type definition PUBLISHED	SWS_Spi_00347 : Status Type definition
MCAL-6467 - SWS_Spi_00259 : Status Type definition PUBLISHED	SWS_Spi_00259 : Status Type definition
MCAL-6545 - SWS_Spi_00346 : Status Type definition PUBLISHED	SWS_Spi_00346 : Status Type definition
MCAL-6748 - SWS_Spi_00134 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00134 : Spi_SyncTransmit Functional

6.1.4 Spi_GetJobResult

Refer section 8.3.8 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6618 - SWS_Spi_00321 : Spi_GetJobResult Functional PUBLISHED	SWS_Spi_00321 : Spi_GetJobResult Functional
MCAL-6486 - SWS_Spi_00038 : Spi_GetJobResult Functional PUBLISHED	SWS_Spi_00038 : Spi_GetJobResult Functional
MCAL-6386 - SWS_Spi_00322 : Spi_GetJobResult Functional PUBLISHED	SWS_Spi_00322 : Spi_GetJobResult Functional
MCAL-6626 - SWS_Spi_00026 : Spi_GetJobResult Functional PUBLISHED	SWS_Spi_00026 : Spi_GetJobResult Functional
MCAL-6563 - SWS_Spi_00261 : JobResult Type definition PUBLISHED	SWS_Spi_00261 : JobResult Type definition
MCAL-6506 - SWS_Spi_00350 : JobResult Type definition PUBLISHED	SWS_Spi_00350 : JobResult Type definition
MCAL-6623 - SWS_Spi_00286 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00286 : Spi_SyncTransmit Functional
MCAL-6525 - SWS_Spi_00194 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00194 : Spi_AsyncTransmit Functional

6.1.5 Spi_GetSequenceResult

Refer section 8.3.9 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6459 - SWS_Spi_00323 : Spi_GetSequenceResult Functional PUBLISHED	SWS_Spi_00323 : Spi_GetSequenceResult Functional
MCAL-6415 - SWS_Spi_00042 : Spi_GetSequenceResult Functional PUBLISHED	SWS_Spi_00042 : Spi_GetSequenceResult Functional
MCAL-6635 - SWS_Spi_00324 : Spi_GetSequenceResult Functional PUBLISHED	SWS_Spi_00324 : Spi_GetSequenceResult Functional
MCAL-6532 - SWS_Spi_00039 : Spi_GetSequenceResult Functional PUBLISHED	SWS_Spi_00039 : Spi_GetSequenceResult Functional
MCAL-6696 - SWS_Spi_00353 : SeqResult Type definition PUBLISHED	SWS_Spi_00353 : SeqResult Type definition
MCAL-6504 - SWS_Spi_00354 : SeqResult Type definition PUBLISHED	SWS_Spi_00354 : SeqResult Type definition
MCAL-6444 - SWS_Spi_00251 : SeqResult Type definition PUBLISHED	SWS_Spi_00251 : SeqResult Type definition
MCAL-6694 - SWS_Spi_00352 : SeqResult Type definition PUBLISHED	SWS_Spi_00352 : SeqResult Type definition

Design Identifier	Description
MCAL-6434 - SWS_Spi_00351 : SeqResult Type definition PUBLISHED	SWS_Spi_00351 : SeqResult Type definition
MCAL-6508 - SWS_Spi_00285 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00285 : Spi_SyncTransmit Functional

6.1.6 Spi_GetHWUnitStatus

Refer section 8.3.12 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6599 - SWS_Spi_00141 : Spi_GetHWUnitStatus Functional PUBLISHED	SWS_Spi_00141 : Spi_GetHWUnitStatus Functional
MCAL-6727 - SWS_Spi_00332 : Spi_GetHWUnitStatus Functional PUBLISHED	SWS_Spi_00332 : Spi_GetHWUnitStatus Functional
MCAL-6552 - SWS_Spi_00331 : Spi_GetHWUnitStatus Functional PUBLISHED	SWS_Spi_00331 : Spi_GetHWUnitStatus Functional

Design Identifier	Description
MCAL-6520 - SWS_Spi_00142 : Spi_GetHWUnitStatus Functional PUBLISHED	SWS_Spi_00142 : Spi_GetHWUnitStatus Functional
MCAL-6608 - SWS_Spi_00287 : Spi_GetHWUnitStatus Functional PUBLISHED	SWS_Spi_00287 : Spi_GetHWUnitStatus Functional
MCAL-6640 - SWS_Spi_00349 : Status Type definition PUBLISHED	SWS_Spi_00349 : Status Type definition
MCAL-6488 - SWS_Spi_00260 : Status Type definition PUBLISHED	SWS_Spi_00260 : Status Type definition
MCAL-6452 - SWS_Spi_00348 : Status Type definition PUBLISHED	SWS_Spi_00348 : Status Type definition
MCAL-6741 - ECUC_Spi_00229 : SpiHwStatusApi PUBLISHED	ECUC_Spi_00229 : SpiHwStatusApi

6.1.7 Spi_SetupEB

Refer section 8.3.6 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6700 - SWS_Spi_00316 : Spi_SetupEB Functional PUBLISHED	SWS_Spi_00316 : Spi_SetupEB Functional
MCAL-6713 - SWS_Spi_00318 : Spi_SetupEB Functional PUBLISHED	SWS_Spi_00318 : Spi_SetupEB Functional
MCAL-6387 - SWS_Spi_00058 : Spi_SetupEB Functional PUBLISHED	SWS_Spi_00058 : Spi_SetupEB Functional
MCAL-6719 - SWS_Spi_00037 : Spi_SetupEB Functional PUBLISHED	SWS_Spi_00037 : Spi_SetupEB Functional
MCAL-6601 - SWS_Spi_00139 : Spi_SetupEB Functional PUBLISHED	SWS_Spi_00139 : Spi_SetupEB Functional
MCAL-6628 - SWS_Spi_00067 : Spi_SetupEB Functional PUBLISHED	SWS_Spi_00067 : Spi_SetupEB Functional
MCAL-6417 - SWS_Spi_00317 : Spi_SetupEB Functional PUBLISHED	SWS_Spi_00317 : Spi_SetupEB Functional
MCAL-6627 - SWS_Spi_00030 : Spi_SetupEB Functional PUBLISHED	SWS_Spi_00030 : Spi_SetupEB Functional

Design Identifier	Description
MCAL-6584 - SWS_Spi_00028 : Spi_SetupEB Functional PUBLISHED	SWS_Spi_00028 : Spi_SetupEB Functional
MCAL-6421 - SWS_Spi_00173 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00173 : Spi_AsyncTransmit Functional
MCAL-6665 - SWS_Spi_00077 : Setup/AsyncTransmit (EB) PUBLISHED	SWS_Spi_00077 : Setup/AsyncTransmit (EB)
MCAL-6385 - SWS_Spi_00078 : Setup/AsyncTransmit (EB) PUBLISHED	SWS_Spi_00078 : Setup/AsyncTransmit (EB)
MCAL-6602 - ECUC_Spi_00203 : SpiDefaultData PUBLISHED	ECUC_Spi_00203 : SpiDefaultData
MCAL-6524 - SWS_Spi_00036 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00036 : Spi_AsyncTransmit Functional

6.1.8 Spi_AsyncTransmit

Refer section 8.3.4 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6395 - SWS_Spi_00311 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00311 : Spi_AsyncTransmit Functional
MCAL-6606 - SWS_Spi_00308 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00308 : Spi_AsyncTransmit Functional
MCAL-6421 - SWS_Spi_00173 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00173 : Spi_AsyncTransmit Functional
MCAL-6721 - SWS_Spi_00157 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00157 : Spi_AsyncTransmit Functional
MCAL-6598 - SWS_Spi_00292 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00292 : Spi_AsyncTransmit Functional
MCAL-6589 - SWS_Spi_00266 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00266 : Spi_AsyncTransmit Functional
MCAL-6525 - SWS_Spi_00194 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00194 : Spi_AsyncTransmit Functional
MCAL-6637 - SWS_Spi_00020 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00020 : Spi_AsyncTransmit Functional

Design Identifier	Description
MCAL-6720 - SWS_Spi_00081 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00081 : Spi_AsyncTransmit Functional
MCAL-6524 - SWS_Spi_00036 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00036 : Spi_AsyncTransmit Functional
MCAL-6373 - SWS_Spi_00309 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00309 : Spi_AsyncTransmit Functional
MCAL-6566 - SWS_Spi_00310 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00310 : Spi_AsyncTransmit Functional
MCAL-6645 - SWS_Spi_00133 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00133 : Spi_AsyncTransmit Functional
MCAL-6644 - SWS_Spi_00057 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00057 : Spi_AsyncTransmit Functional
MCAL-6711 - SWS_Spi_00086 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00086 : Spi_AsyncTransmit Functional
MCAL-6410 - SWS_Spi_00055 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00055 : Spi_AsyncTransmit Functional

Design Identifier	Description
MCAL-6414 - SWS_Spi_00035 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00035 : Spi_AsyncTransmit Functional
MCAL-6665 - SWS_Spi_00077 : Setup/AsyncTransmit (EB) PUBLISHED	SWS_Spi_00077 : Setup/AsyncTransmit (EB)
MCAL-6385 - SWS_Spi_00078 : Setup/AsyncTransmit (EB) PUBLISHED	SWS_Spi_00078 : Setup/AsyncTransmit (EB)

6.1.9 Spi_Cancel

Refer section 8.3.13 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6740 - SWS_Spi_00144 : Spi_Cancel Functional PUBLISHED	SWS_Spi_00144 : Spi_Cancel Functional
MCAL-6631 - SWS_Spi_00145 : Spi_Cancel Functional PUBLISHED	SWS_Spi_00145 : Spi_Cancel Functional

Design Identifier	Description
MCAL-6579 - SWS_Spi_00334 : Spi_Cancel Functional PUBLISHED	SWS_Spi_00334 : Spi_Cancel Functional
MCAL-6514 - SWS_Spi_00333 : Spi_Cancel Functional PUBLISHED	SWS_Spi_00333 : Spi_Cancel Functional
MCAL-6739 - SWS_Spi_00146 : Spi_Cancel Functional PUBLISHED	SWS_Spi_00146 : Spi_Cancel Functional
MCAL-6749 - ECUC_Spi_00226 : SpiCancelApi PUBLISHED	ECUC_Spi_00226 : SpiCancelApi
MCAL-6512 - SWS_Spi_00375 : SeqResult Type definition PUBLISHED	SWS_Spi_00375 : SeqResult Type definition
MCAL-6616 - SWS_Spi_00073 : Call-Back Functions PUBLISHED	SWS_Spi_00073 : Call-Back Functions

6.1.10 Spi_SyncTransmit

Refer section 8.3.11 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6409 - SWS_Spi_00330 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00330 : Spi_SyncTransmit Functional
MCAL-6634 - SWS_Spi_00327 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00327 : Spi_SyncTransmit Functional
MCAL-6508 - SWS_Spi_00285 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00285 : Spi_SyncTransmit Functional
MCAL-6623 - SWS_Spi_00286 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00286 : Spi_SyncTransmit Functional
MCAL-6748 - SWS_Spi_00134 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00134 : Spi_SyncTransmit Functional
MCAL-6523 - SWS_Spi_00329 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00329 : Spi_SyncTransmit Functional
MCAL-6509 - SWS_Spi_00328 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00328 : Spi_SyncTransmit Functional
MCAL-6465 - SWS_Spi_00136 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00136 : Spi_SyncTransmit Functional

Design Identifier	Description
MCAL-6371 - SWS_Spi_00135 : Spi_SyncTransmit Functional PUBLISHED	SWS_Spi_00135 : Spi_SyncTransmit Functional
MCAL-6402 - ECUC_Spi_00237 : SpiSupportConcurrentSyncTransmit PUBLISHED	ECUC_Spi_00237 : SpiSupportConcurrentSyncTransmit
MCAL-6745 - SWS_Spi_00114 : General behaviour PUBLISHED	SWS_Spi_00114 : General behaviour
MCAL-6414 - SWS_Spi_00035 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00035 : Spi_AsyncTransmit Functional

6.1.11 Spi_SetAsyncMode

Refer section 8.3.14 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6553 - SWS_Spi_00338 : Spi_SetAsyncMode Functional PUBLISHED	SWS_Spi_00338 : Spi_SetAsyncMode Functional

Design Identifier	Description
MCAL-6491 - SWS_Spi_00335 : Spi_SetAsyncMode Functional PUBLISHED	SWS_Spi_00335 : Spi_SetAsyncMode Functional
MCAL-6510 - SWS_Spi_00152 : Spi_SetAsyncMode Functional PUBLISHED	SWS_Spi_00152 : Spi_SetAsyncMode Functional
MCAL-6416 - SWS_Spi_00171 : Spi_SetAsyncMode Functional PUBLISHED	SWS_Spi_00171 : Spi_SetAsyncMode Functional
MCAL-6379 - SWS_Spi_00172 : Spi_SetAsyncMode Functional PUBLISHED	SWS_Spi_00172 : Spi_SetAsyncMode Functional
MCAL-6617 - SWS_Spi_00337 : Spi_SetAsyncMode Functional PUBLISHED	SWS_Spi_00337 : Spi_SetAsyncMode Functional
MCAL-6660 - SWS_Spi_00336 : Spi_SetAsyncMode Functional PUBLISHED	SWS_Spi_00336 : Spi_SetAsyncMode Functional
MCAL-6675 - SWS_Spi_00154 : Spi_SetAsyncMode Functional PUBLISHED	SWS_Spi_00154 : Spi_SetAsyncMode Functional
MCAL-6659 - SWS_Spi_00360 : AsyncMode Type definition PUBLISHED	SWS_Spi_00360 : AsyncMode Type definition

Design Identifier	Description
MCAL-6419 - SWS_Spi_00361 : AsyncMode Type definition PUBLISHED	SWS_Spi_00361 : AsyncMode Type definition
MCAL-6517 - SWS_Spi_00362 : AsyncMode Type definition PUBLISHED	SWS_Spi_00362 : AsyncMode Type definition

6.1.12 Spi_MainFunction_Handling

Refer section 8.5.1 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6555 - ECUC_Spi_00242 : SpiMainFunctionPeriod PUBLISHED	ECUC_Spi_00242 : SpiMainFunctionPeriod

6.1.13 Spi_GetVersionInfo

Refer section 8.3.10 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6562 - SWS_Spi_00325 : Spi_GetVersionInfo Functional PUBLISHED	SWS_Spi_00325 : Spi_GetVersionInfo Functional
MCAL-6630 - SWS_Spi_00371 : Spi_GetVersionInfo Functional PUBLISHED	SWS_Spi_00371 : Spi_GetVersionInfo Functional
MCAL-6479 - ECUC_Spi_00232 : SpiVersionInfoApi PUBLISHED	ECUC_Spi_00232 : SpiVersionInfoApi

6.1.14 Spi_WriteIB

Refer section 8.3.3 of of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1. The maximum internal buffer length that can be allocated by each channel is fixed. Can vary buffer length per channel by configuring SpiIbNBuffers and SpiDataWidth. Refer MCAL-1255 and MCAL-1203.Size of 256 bytes covers nowadays requirements. By default ECUC parameter SpiChannelInternalBufferMaxLength is configured with value 64 in configurator

Design Identifier	Description
MCAL-6477 - SWS_Spi_00304 : Spi_WriteIB Functional PUBLISHED	SWS_Spi_00304 : Spi_WriteIB Functional

Design Identifier	Description
MCAL-6407 - SWS_Spi_00307 : Spi_WriteIB Functional PUBLISHED	SWS_Spi_00307 : Spi_WriteIB Functional
MCAL-6567 - SWS_Spi_00024 : Spi_WriteIB Functional PUBLISHED	SWS_Spi_00024 : Spi_WriteIB Functional
MCAL-6473 - SWS_Spi_00018 : Spi_WriteIB Functional PUBLISHED	SWS_Spi_00018 : Spi_WriteIB Functional
MCAL-6676 - SWS_Spi_00306 : Spi_WriteIB Functional PUBLISHED	SWS_Spi_00306 : Spi_WriteIB Functional
MCAL-6725 - SWS_Spi_00305 : Spi_WriteIB Functional PUBLISHED	SWS_Spi_00305 : Spi_WriteIB Functional
MCAL-6564 - SWS_Spi_00137 : Spi_WriteIB Functional PUBLISHED	SWS_Spi_00137 : Spi_WriteIB Functional
MCAL-6724 - SWS_Spi_00023 : Spi_WriteIB Functional PUBLISHED	SWS_Spi_00023 : Spi_WriteIB Functional
MCAL-6421 - SWS_Spi_00173 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00173 : Spi_AsyncTransmit Functional

Design Identifier	Description
MCAL-6602 - ECUC_Spi_00203 : SpiDefaultData PUBLISHED	ECUC_Spi_00203 : SpiDefaultData

6.1.15 Spi_ReadIB

Refer section 8.3.5 of the SPI Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-6482 - SWS_Spi_00312 : Spi_ReadIB Functional PUBLISHED	SWS_Spi_00312 : Spi_ReadIB Functional
MCAL-6469 - SWS_Spi_00027 : Spi_ReadIB Functional PUBLISHED	SWS_Spi_00027 : Spi_ReadIB Functional
MCAL-6464 - SWS_Spi_00315 : Spi_ReadIB Functional PUBLISHED	SWS_Spi_00315 : Spi_ReadIB Functional
MCAL-6400 - SWS_Spi_00016 : Spi_ReadIB Functional PUBLISHED	SWS_Spi_00016 : Spi_ReadIB Functional

Design Identifier	Description
MCAL-6501 - SWS_Spi_00313 : Spi_ReadIB Functional PUBLISHED	SWS_Spi_00313 : Spi_ReadIB Functional
MCAL-6447 - SWS_Spi_00314 : Spi_ReadIB Functional PUBLISHED	SWS_Spi_00314 : Spi_ReadIB Functional
MCAL-6731 - SWS_Spi_00138 : Spi_ReadIB Functional PUBLISHED	SWS_Spi_00138 : Spi_ReadIB Functional
MCAL-6672 - SWS_Spi_00085 : Call-Back Functions PUBLISHED	SWS_Spi_00085 : Call-Back Functions
MCAL-6421 - SWS_Spi_00173 : Spi_AsyncTransmit Functional PUBLISHED	SWS_Spi_00173 : Spi_AsyncTransmit Functional

6.1.16 Spi_RegisterReadback

As noted from previous implementation, some of the critical configuration registers could potentially be corrupted by other entities (s/w or h/w). One of the recommended detection methods would be to periodically read-back the configuration and confirm configuration is consistent. The service API defined below shall be implemented to enable this detection

	Description	***Comments***

Service Name	Spi_RegisterReadback	Can potentially be turned OFF (NON Standard configurable parameters)
Syntax	uint32 Spi_RegisterReadback (Spi_HWUnitType HWUnit, P2VAR(Spi_RegisterReadbackType, AUTOMATIC, SPI_APPL_DATA) RegRbPtr)	E_OK: Register read back has been done, E_NOT_OK: Register read back failed
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	HWUnit	SPI Hardware microcontroller peripheral unit ID. If this is invalid, then the API will return E_NOT_OK.
Parameters out	RegRbPtr - Pointer to where to store the readback values. If this pointer is NULL_PTR, then the API will return E_NOT_OK.	
Return Value	Std_ReturnType	E_OK, E_NOT_OK

Design Identifier	Description
MCAL-6433 - SPI: Safety Diagnostics: SPI4: Software Readback of Written Configuration PUBLISHED	SPI: Safety Diagnostics: SPI4: Software Readback of Written Configuration
MCAL-6586 - SPI: Safety Diagnostics: SPI3: Periodic Software Readback of Static Configuration Registers PUBLISHED	SPI: Safety Diagnostics: SPI3: Periodic Software Readback of Static Configuration Registers

6.1.17 Spi_dataOverflowUnderflowIntrEnable

This API Enable Under/Overflow Interrupts of the hardware unit and returns the status.

	Description	***Comments***
Service Name	Spi_dataOverflowUnderflowIntrEnable	Spi_dataOverflowUnderflowIntrEnable Enable Under/Overflow Interrupts of the hardware unit
Syntax	Std_ReturnType Spi_dataOverflowUnderflowIntrEnable(Spi_HWUnitType HWUnit, uint32 intFlags)	

	Description	*Comments*
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	HWUnit	SPI Hardware microcontroller peripheral unit ID. If this is invalid, then the API will return E_NOT_OK.
Parameters out	NONE	
Return Value	Std_ReturnType	E_OK: Interrupt Enabled E_NOT_OK: Interrupt Enable failed

Design Identifier	**Description**
MCAL-6538 - SPI: Safety Diagnostics: SPI6: Data Overflow Detection **PUBLISHED**	SPI: Safety Diagnostics: SPI6: Data Overflow Detection
MCAL-6392 - SPI: Safety Diagnostics: SPI7: Data Underflow Detection **PUBLISHED**	SPI: Safety Diagnostics: SPI7: Data Underflow Detection

6.1.18 Spi_dataOverflowUnderflowIntrDisable

This API Disable Under/Overflow Interrupts of the hardware unit and returns the status.

	Description	***Comments***
Service Name	Spi_dataOverflowUnderflowIntrDisable	Spi_dataOverflowUnderflowIntrDisable Disable Under/Overflow Interrupts of the hardware unit and returns the status.
Syntax	Std_ReturnType Spi_dataOverflowUnderflowIntrDisable(Spi_HWUnitType HWUnit, uint32 intFlags)	
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	HWUnit	SPI Hardware microcontroller peripheral unit ID. If this is invalid, then the API will return E_NOT_OK.
Parameters out	NONE	
Return Value	Std_ReturnType	E_OK: Interrupt Disabled E_NOT_OK: Interrupt Disable failed

Design Identifier	Description
MCAL-6538 - SPI: Safety Diagnostics: SPI6: Data Overflow Detection PUBLISHED	SPI: Safety Diagnostics: SPI6: Data Overflow Detection
MCAL-6392 - SPI: Safety Diagnostics: SPI7: Data Underflow Detection PUBLISHED	SPI: Safety Diagnostics: SPI7: Data Underflow Detection

6.1.19 Spi_dataOverflowUnderflowIntrStatusGet

This API gets status of Under/Overflow Interrupts of the hardware unit and returns the status.

	Description	***Comments***
Service Name	Spi_dataOverflowUnderflowIntrStatusGet	Spi_dataOverflowUnderflowIntrStatusGet gets status of Under/ Overflow Interrupts of the hardware unit
Syntax	Mcspi_IrqStatusType Spi_dataOverflowUnderflowIntrStatusGet(Spi_HWUnitType HWUnit, uint32 intFlags)	
Service ID	NA	

	Description	*Comments*
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	HWUnit	SPI Hardware microcontroller peripheral unit ID. If this is invalid, then the API will return E_NOT_OK.
Parameters out	NO	
Return Value	Mcspi_IrqStatusType	SPI_NO_EVENT: No underflow event SPI_EVENT_PENDING: Underflow Event SPI_STATUS_READ_FAIL: Status read fail

Design Identifier	**Description**
MCAL-6538 - SPI: Safety Diagnostics: SPI6: Data Overflow Detection **PUBLISHED**	SPI: Safety Diagnostics: SPI6: Data Overflow Detection
MCAL-6392 - SPI: Safety Diagnostics: SPI7: Data Underflow Detection **PUBLISHED**	SPI: Safety Diagnostics: SPI7: Data Underflow Detection

6.1.20 Spi_dataOverflowUnderflowIntrStatusClear

This API clear Under/Overflow Interrupts of the hardware unit and returns the status.

	Description	***Comments***
Service Name	Spi_dataOverflowUnderflowIntrStatusClear	Spi_dataOverflowUnderflowIntrStatusClear clear Under/Overflow Interrupts of the hardware unit and returns the status.
Syntax	Std_ReturnType Spi_dataOverflowUnderflowIntrStatusClear(Spi_HWUnitType HWUnit, uint32 intFlags)	
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	HWUnit	SPI Hardware microcontroller peripheral unit ID. If this is invalid, then the API will return E_NOT_OK.
Parameters out	NONE	

	Description	*Comments*
Return Value	Std_ReturnType	E_OK: Interrupt status clears E_NOT_OK: Interrupt status clear failed

Design Identifier	**Description**
MCAL-6538 - SPI: Safety Diagnostics: SPI6: Data Overflow Detection **PUBLISHED**	SPI: Safety Diagnostics: SPI6: Data Overflow Detection
MCAL-6392 - SPI: Safety Diagnostics: SPI7: Data Underflow Detection **PUBLISHED**	SPI: Safety Diagnostics: SPI7: Data Underflow Detection

7 Performance Objectives

7.1 Resource Consumption Objectives

ROM - Program(KB)	ROM - Data(KB)	RAM - Program(KB)	RAM - Data(KB)	Stack Size (KB)	EEPROM (KB)	% CPU Utilization
30	NA	NA	1	4	NA	NA

7.2 Critical timing and Performance

Not Applicable

8 Decision Analysis & Resolution (DAR)

Sections below list some of the important design decisions and rational behind those decision.

8.1 Data transfer mode (MCSPI)

The SPI asynchronous data transfer can be achieved by DMA or through interrupt (CPU) mode. The mode chosen can have system wide effect and it's important to choose the right method.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
1	Minimal restrictions on the system and guaranteed total throughput in system.	**DMA Mode** The MCSPI module generates DMA events to the system DMA as soon as data is available in RX register for reception and if TX register is empty for transmission. In this case DMA needs to be initiated by CPU. **Advantages:** • CPU loading is low and constant irrespective of the SPI data transfer rate • Less probability of FIFO overflow in MCSPI as the DMA copy happens without CPU intervention **Disadvantages:** • Cache coherency needs to be taken care. This will result in Cache module dependency in driver or in the AUTOSAR stack • Need of a common DMA complex driver with resource management as the DMA is at system level and is common across peripherals	Interrupt/ pooling(CPU) Mode	In case of ADAS use case, the MCSPI will be used for low bandwidth data transfer and the simplicity of CPU based data transfer makes it preferable to DMA. The CPU based throughput is expected to be sufficient to meet the use cases	• CPU loading increases with increasing SPI data transfer rate • High probability of FIFO overflow in MCSPI if data transfer rate is high

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		CPU (Interrupt/Polling) Mode The MCSPI module can trigger interrupts on data transfer completion. **Advantages:** • Simple implementation • No cache coherency is needed and no dependency on cache APIs **Disadvantages:** • CPU loading increases with increasing SPI data transfer rate • High probability of FIFO overflow in MCSPI if data transfer rate is high			

8.2 Selecting SPI Instances in Configurator

The number of SPI HW units will vary based on device variant, power domain and cores that can access these. Selecting specific SPI instance in configurator can be done in following two ways.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
1	Minimal error checks on the configurator.	**Multiple ECUC Parameters** Provide 4 ECUC parameters Device Variant, Domain, Core and SPI HW Unit **Advantages:** • This will help in selecting right SPI HW Unit which is available in Device Variants, Domain and accessible from Cores. **Disadvantages:** • Increases complexity	Based on device variant	All SPI HW Units are accessible from all cores in TDA4X families. Also it is simple to design configurator with less error checks.	None
		Select based on device variant Based on Device Variant select SPI HW Unit, no need of Domain and Core provided all SPI HW Units are accessible from all cores. **Advantages:** • Simpler **Disadvantages:** • -			

9 Testing Guidelines

The sections below identify some of the aspects of design that would require emphasis during testing of this design implementation

- **Validating ECUC parameters**
  - Validating ECUC Parameter: Configuration for each test case shall be generated by EB Tresos command line.
- **Performance Testing**
  - While measuring time taken to send data, care should be taken to use a timer (and not rely on CPU ticks & conversion). Time shall be measured between invoke of transmit API and return of this function call for both Asynctransmit and Synctransmit cases.

Design Identifier	Description
MCAL-6710 - MCAL: McSPI Performance PUBLISHED	MCAL: McSPI Performance

- **Loopback Test**
  - The loopback transmit test for all SPI instances
- **Transmit Test with different configurations**
  - Multichannel transmit test with different channel parameter configurations
  - Multijob transmit test with different job parameter configurations
  - Multisequence transmit test with Non interruptible sequence configuration
  - Multisequence transmit test with Interruptible sequence configuration
  - Transmit test with different device configurations like modes, polarity and phase.
  - Asynchronous and Synchronous mode of transmission test
  - Asynchronous transmission test with interrupt and polling mode
  - Transmit test with IB/EB channel buffering modes

- **Transmit test with different baud rates**
    - Transmit test with different clock bit rates obtained for data transfer by programming the clock divider.

10 Template Revision History

Author Name	Description	Version	Date
Yaniv Machani	Initial version	0.1	03 Oct 2018
Yaniv Machani	Updated to include EP views	0.4	02 Nov 2018
Yaniv Weizman	Restructuring and editing to further meet the A-SPICE and EP requirements	0.5	27 Dec 2018
Yaniv Weizman	Adding link to Architecture review template	0.6	22 Oct 2019
Yaniv Weizman	Adding requirement type column for requirements table (Functional/Non-Functional). Adding DAR table	0.65	13 Nov 2019

Author Name	Description	Version	Date
Yaniv Weizman	Adding tables for Testing guidelines	0.7	18 Nov 2019
Krishna	Updated based on ASPICE requirements	0.8	20 Aug 2020
Krishna	Updated based on the feedback from Jon N	0.9	09 Oct 2020
Krishna	Updated the traceability scheme	1.0	17 Dec 2020