MCAL FLS Module Software Design Document

Document Version : 101
Document Owner : Texas Instruments
Document Status : Published
Last Approval Date : Jun 30, 2022

1 Revision History

Version	Date	Author	Document Status	Comments
0.1	18 Apr 2022	Subose Nallavelugula	DONE	Initial version
0.2	18 May 2022	Subose Nallavelugula	DONE	Updated Sec 4.5 , 5.5 and other review comments
0.3	25 May 2022	Subose Nallavelugula	DONE	Updated the safety API.
0.4	09 Jun 2022	kanak Shrinivasan	DONE	Updated review comments and added Comola workflow
v.101	22 Jun 2022	kanak Shrinivasan	DONE	Updated review comments and added Comola workflow

2 Terms and Abbreviations

Abbreviation / Term	Meaning / Explanation
FLS	Flash
AUTOSAR	AUTomotive Open System ARchitecture
RTE	Runtime Environment
BSW	Basic Software
MCAL	MicroController Abstraction Layer
SBL	Serial Bootloader
API	Application Programming Interface
DET	Default Error Tracer

Abbreviation / Term	Meaning / Explanation
DEM	Diagnostic Event Manager – module to handle diagnostic relevant events.
ECU	Electronic Control Unit
MCU	Micro Controller Unit
OS	Operating System
SoC	System on a Chip
DAR	Decision Analysis and Resolution

3 Introduction

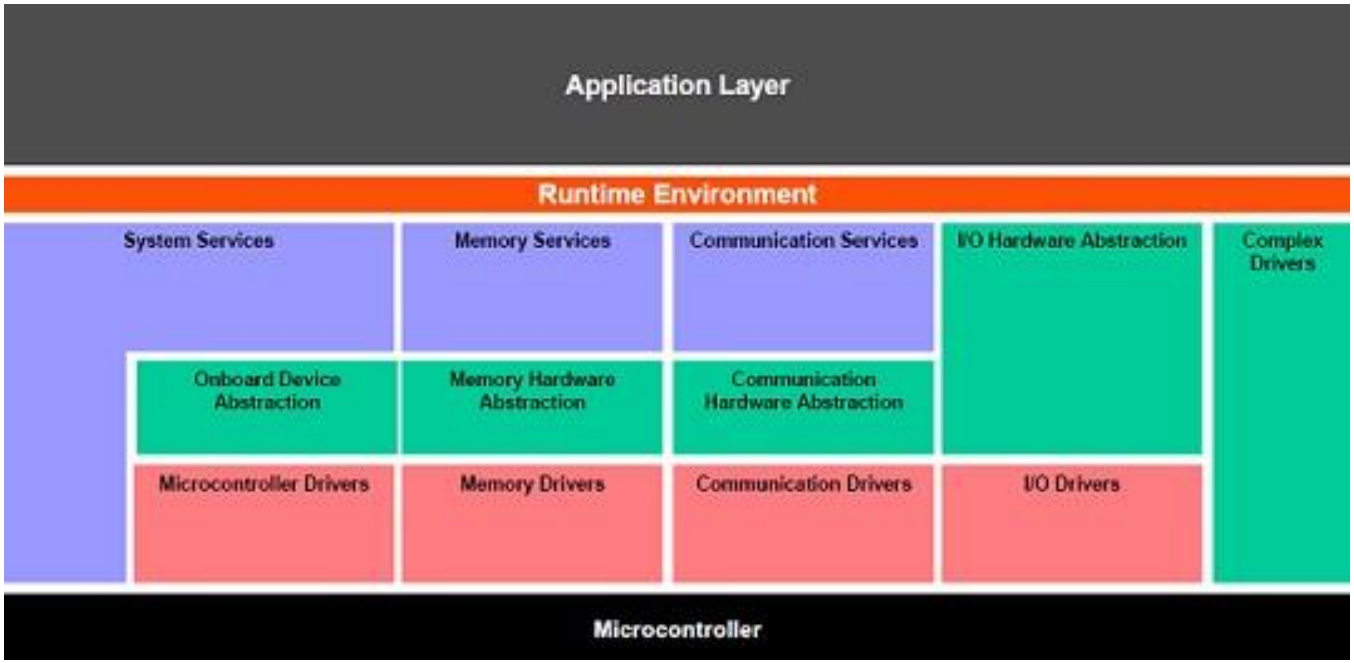
3.1 Overview

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
- MicroController 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. Flash Driver is part of the Memory Drivers module, which is part of the Basic Software. The Flash Driver provides services for reading, writing, erasing flash

memory and Execute-in-Place (XIP) mode for external Flash Device. The driver supports MT35XU512ABA1G12 and S28HS512T OSPI Flash Devices. The main tasks of the FLS driver are:

Perform storage mode applications:

- Read from flash.
- Write to flash.
- Erase Flash.
- Compare and Blank Check flash memory location.

Execute application using XIP Mode (only XIP read supported).

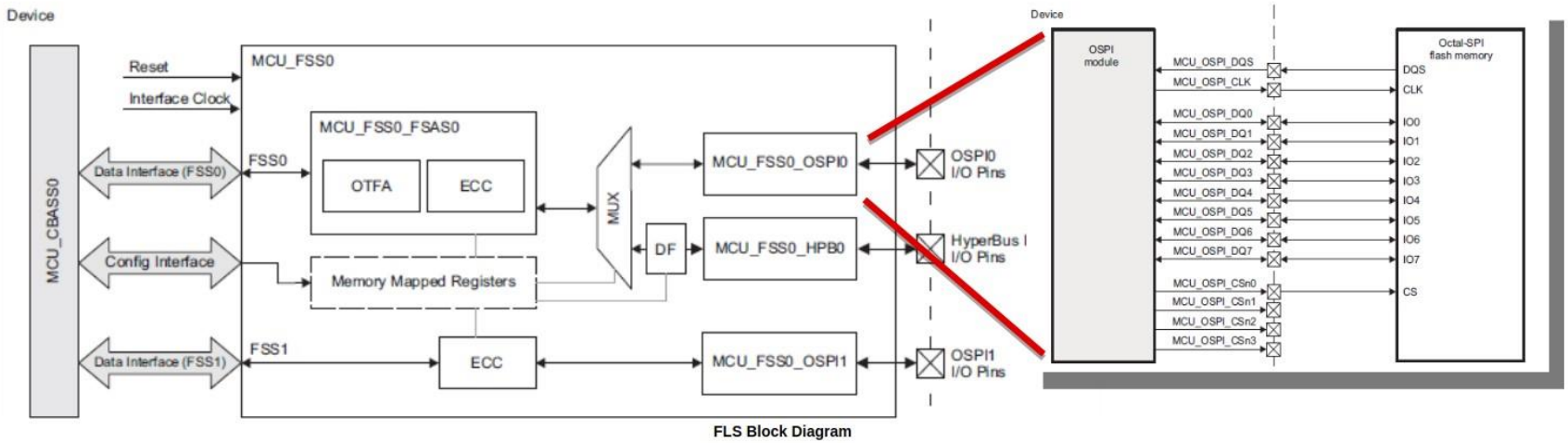
3.2 Purpose and Scope

The Detailed Design document provides the design details of FLS driver and aims to provide a guide to a design that could be implemented by a software developer, using hardware IP "ospi_10_rel.1.0.x"

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

3.3 Module Overview

This Flash driver uses OSPI module to transfer data between the system and flash device. Below diagram shows the integration between the two hardware IPs that are used by the Flash Driver: FSS and OSPI. J721-EVM has one Flash device support - MT35XU512ABA1G12, and J7200-EVM has s28HS512T. Please note that this is for just reference purpose, for other details please refer to technical reference manual.



FLS Block Diagram

The Flash Driver module includes the following main features:

- Storage Mode support to Write, Read, Erase for flash memory.
- Direct Access Mode (DAC) for Memory Mapped Flash accesses.
- Indirect Access Mode (INDAC) to make use of internal SRAM for data transfers.
- Poll Mode implementation for DAC and INDAC mode, and Interrupt Mode implementation for INDAC mode.
- Serial clock with programmable frequency.
- Single (STR/SDR) and Double Transfer Rate protocol (DTR/DDR) support.
- Execute-in-Place (XIP) Read support.

- PHY Module integration for optimized performance.

3.4 Requirements

The FLS driver shall implement as per requirements detailed in Reference 1 - AUTOSAR 4.3.1

3.4.1 Features Supported

- Configuration of Flash Device
- Supported External flash devices - MT35XU512ABA1G12 and S28HS512T.
- Customer can be able to enable their own flash device now using the FLS driver.
- Initialization of OSPI Module.
- Enabling of Development Error Detection (DET).
- Configuration of implementation features like:
- Fls DDR vs SDR
- Fls DAC vs INDAC
- Fls OSPI Clk Speed
- Fls XIP Enable
- Asynchronous Write, Read, Erase, BlankCheck, and Compare APIs
- Interrupt based implementation with INDAC using non-blocking/callback design.

3.4.2 Features Not Supported / NON Compliance

API features not supported

- [NON Compliance] BlankCheck and Compare API functionality not available in Interrupt mode.
- [NON Compliance] Erase and Write Verification not supported.
- [AUTOSAR Optional] Only one instance of Fls module supported at one time.

HW features not supported

- [NON Compliance] Supports only XIP Read mode.
- Implementation compliant with only MCU1_0 and MCU2_1.

3.5 **Assumptions**

Below listed points 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 Fls module is expected to be ON before calling any Fls service APIs. The Fls driver doesn't perform any programming to enable the module functional clock.
2. Interrupt configuration for Fls interrupt registration should be done by application. Refer to example application for reference.

3.6 **Constraints**

The driver can have unexpected behavior and loss of functionality if unsupported values are passed into configurator or unsupported combinations are used. Below is a list of such constraints.

- Configurator Input Values Constraints:
    a. maxReadNormalMode and maxWriteNormalMode values should be aligned with (multiples of) the Flash device page size.
    b. sectorList structure should not be changed, as that is specific to Flash Device.
    c. ospiClkSpeed should be either 133333333U or 166666666U.
- Feature Combination Constraints:
    a. When using Interrupt mode, dacEnable should be set to FALSE. Interrupt mode is only available when using INDAC mode.

    b. When using Interrupt mode, maxWriteNormalMode value has to be equal to page size of flash device. Please see Fls_Cfg_intr.c sample configuration file provided.
    c. If operating in XIP mode, dacEnable has to be enabled, and Interrupt mode has to be set to STD_OFF. For J7200-EVM, PHY mode has to be disabled for XIP enable flag to work.
    d. In Interrupt mode, BlankCheck API and Compare API are not supported and should be disabled.
    e. On J7200, write and erase is not functional in DAC mode. Only Read is possible in DAC mode. Please use INDAC mode for writing and erasing flash for J7200.

3.7 Hardware and SW platforms

Hardware Platforms

- Refer to specified SoC User Manual to check if Flash module is supported. **Software Platforms**
- Bare-Metal

3.8 Dependencies

DET

This implementation depends on the DET in order to report development errors and can be turned OFF. Refer section 4.4 for detailed error codes.

SchM

This implementation requires one level of exclusive access to guard critical sections. Invokes SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_0 (), SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_0 () to enter critical section and exit.

In the example implementation SchM_Fls.c all the interrupts on CPU are disabled. However, disabling of the enabled Fls interrupt should suffice.

MemIf

This implementation depends on MemIf module and uses its imported types such as MemIf_JobResultType, MemIf_ModeType and MemIf_StatusType.

Fee

This implementation depends on Fee module for callback notification to notify the module environment about job end and job error.

System clock

If the hardware of the internal flash memory depends on the system clock, changes to the system clock (e.g. PLL on PLL off) may also affect the clock settings of the flash memory hardware.

Communication or I/O drivers

If the flash memory is located in an external device, the access to this device shall be enacted via the corresponding communication respectively I/O driver.

3.9 Stakeholders

- Developers
- Test Engineers
- Customer Integrator

3.10 References

	Specification	Comment/Link
1	AUTOSAR 4.3.1	AUTOSAR Specification for FLS Driver
2	BSW General Requirements / Coding guidelines	Autosar and Coding guidelines for the Mcal drivers.
3	Software Product Specification (SPS)	Product Functional requirements.
4	Software Architecture	Mcal Software Architecture.

4 Design Description

4.1 Fundamental Operation

Fls module uses OSPI interface to transfer data between the system and the flash device. To be able to write to the flash, the flash has to be erased first. The erase operation internally programs 0xFF to all memory cells.

4.1.1 DAC Mode:

- Direct Access Mode allows data interface accesses directly trigger read or write to flash memory. It is memory-mapped, and can be used to both access and directly execute code from external flash.

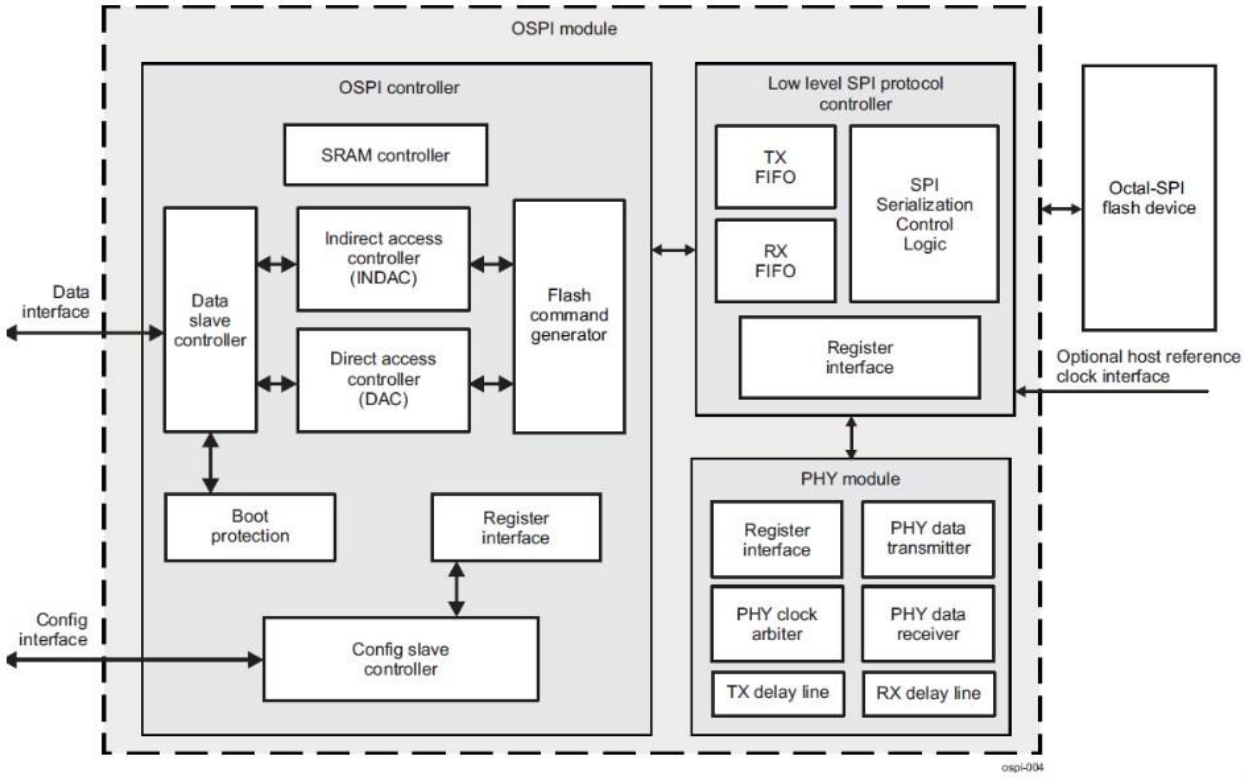
4.1.2 INDAC Mode:

- Indirect Access Mode used the internal OSPI SRAM to make data transfers. Indirect trigger access address does not have any relationship with flash address. It should take the SRAM as source (instead of Flash memory).

4.1.3 STIG:

- Software Triggered Instruction Generator - While DAC and INDAC are used for data transfer, STIG is used to access the volatile and non-volatile configuration registers, status registers, and for erase functions.

This Flash Driver Module Write, Read, Erase, BlankCheck and Compare APIs are asynchronous such that control will return to the application after the job from one of the APIs has been accepted. A "job" describes the action that will perform actual transactions to the hardware. The driver's Fls_MainFunction () API will do the job processing internally, after the APIs have been called and the job has been accepted. During job processing, the module will perform the programming of the flash device, and will make data transfers between system and flash device.


OSPI module
OSPI controller
SRAM controller
Data slave controller
Indirect access controller (INDAC)
Direct access controller (DAC)
Flash command generator
Boot protection
Register interface
Config slave controller
Data interface
Config interface
Low level SPI protocol controller
TX FIFO
RX FIFO
SPI Serialization Control Logic
Register interface
PHY module
Register interface
PHY data transmitter
PHY clock arbiter
PHY data receiver
TX delay line
RX delay line
Octal-SPI flash device
Optional host reference clock interface
ospi-004

4.1.4 External Flash Device Hardware:

- Flash is composed of Sectors and Pages. Sectors are subdivided into Pages.
- Writes happen in complete page blocks, erases happen in complete sectors.
- Write Enable command sent before a write or erase operation.
- Read ID command sent to detect the connected flash device.
- MT35XU512ABA1G12 NOR Flash Device:
    a. Micron Tech, 64MB density
    b. Sector Size: 4k Bytes
    c. Page Size: 256 Bytes
- S28HS512T NOR Flash Device:
    a. Cypress, 64MB density
    b. Sector Size: 256K Bytes
    c. Page Size: 256 Bytes
- Other OSPI NOR flash devices can be used as well.

4.1.5 **Sequence Diagrams**

The Flash driver follows the code sequence outlined by the Autosar Spec for FLASH Module. Please refer to Reference 1 - AUTOSAR 4.3.1

4.1.6 **Interrupt Service Routines**

For the Flash module instance, one interrupt service routine has been mapped. Note that Interrupt mode is only available when operating in INDAC mode. BlankCheck and Compare API are not functional with interrupt mode. The supported ISR is part of the Fls_Irq.h and Fls_Irq.c files.

Handling Flash OSPI FIFO Interrupt: OSPI Hardware Behavior: The hardware uses SRAM Watermark levels to trigger interrupts. There are two interrupts generated for every write/read call from SRAM to/from Flash memory. One interrupt notifies driver that the SRAM FIFO fill level is below a watermark level, which allows more data to be transferred into the SRAM. Another interrupt occurs when the transfer from SRAM into Flash memory, or to system memory, completes. Both these interrupts trigger the same ISR, and are handled accordingly within the ISR.

4.2 Directory Structure

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

The Driver API are mentioned in the two different folder.

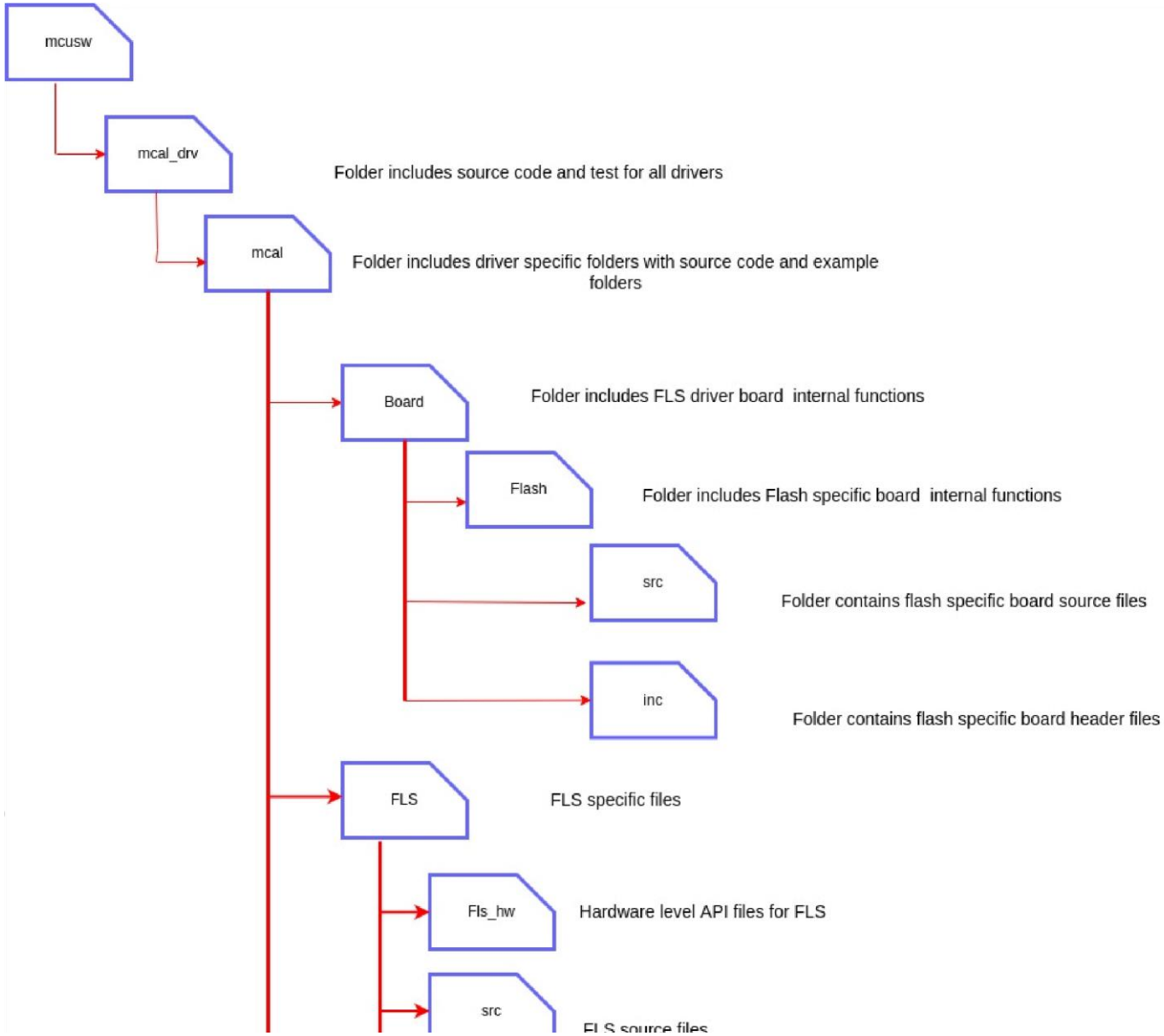
Board Folder : Fls_Brd_Nor_Ospi.c,Fls_Brd_Nor_Xspi.c,nor_spi_patterns.c,nor_spi_phy_tune.c shall have the shall have the internal functions and data structures for board level APIs and PHY tune algorithm files.

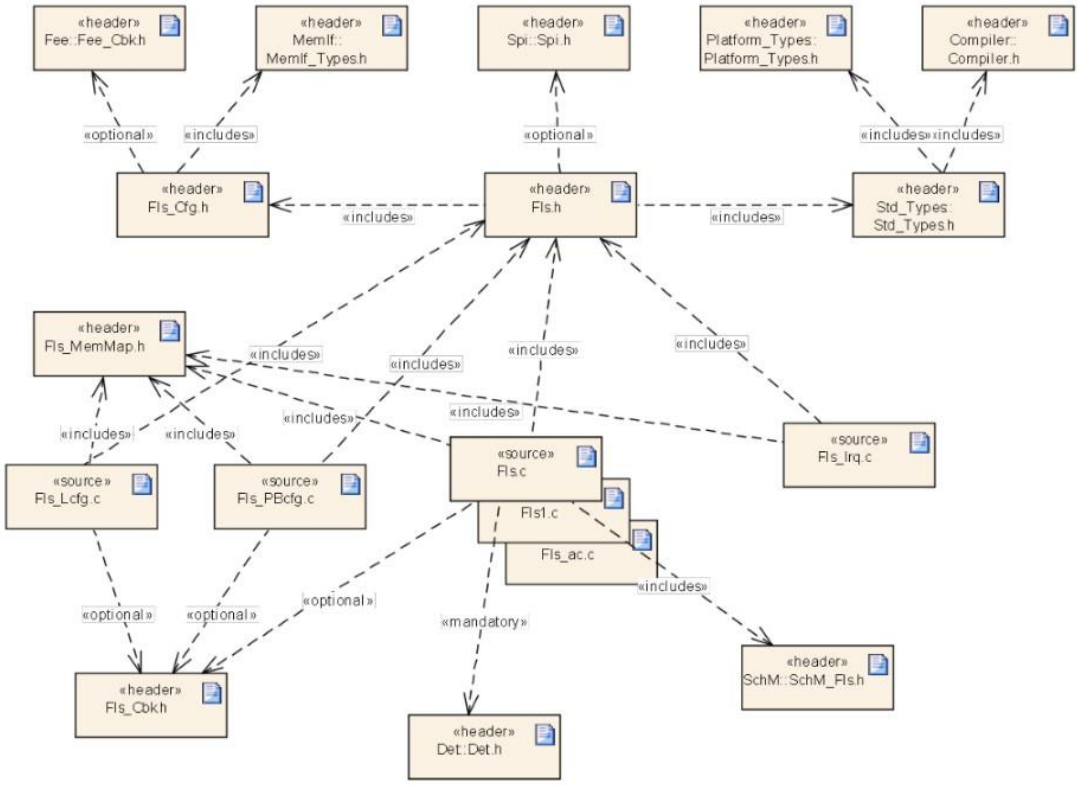
Fls Folder: Fls_Ospi.c,Fls_Ospi.h Shall have the OSPI interface APIs, functions and data structures.

Driver Implemented by

- Fls.h: Shall implement the interface provided by the Fls driver
- Fls_Irq.h Contains ISR function declaration.
- Fls.c, Fls_Irq.c : Shall implement the driver functionality
- Fls_Brd_Nor_Ospi.c, Fls_Brd_Nor_Xspi.c and Fls_Brd_Nor.h : shall have the internal functions and data structures for board level APIs. J721E-EVM board will include Fls_Brd_Nor_Ospi.c, while J7200-EVM will include Fls_Brd_Nor_Xspi.c.
- Fls_Ospi.c, Fls_Ospi.h, Fls_Spi_Intf.c, Fls_Spi_Intf.h : Shall have the OSPI interface APIs, functions and data structures.
- Fls_Soc.c, Fls_Soc.h, Fls_NOR_s28hs512t.h and Fls_NOR_m35xu512.h : Shall contain the SOC and Flash device specific data structures.

- nor_spi_patterns.c, nor_spi_patterns.h, nor_spi_phy_tune.c and nor_spi_phy_tune.h are PHY tune algorithm files.
- hw_include is used for Example Application

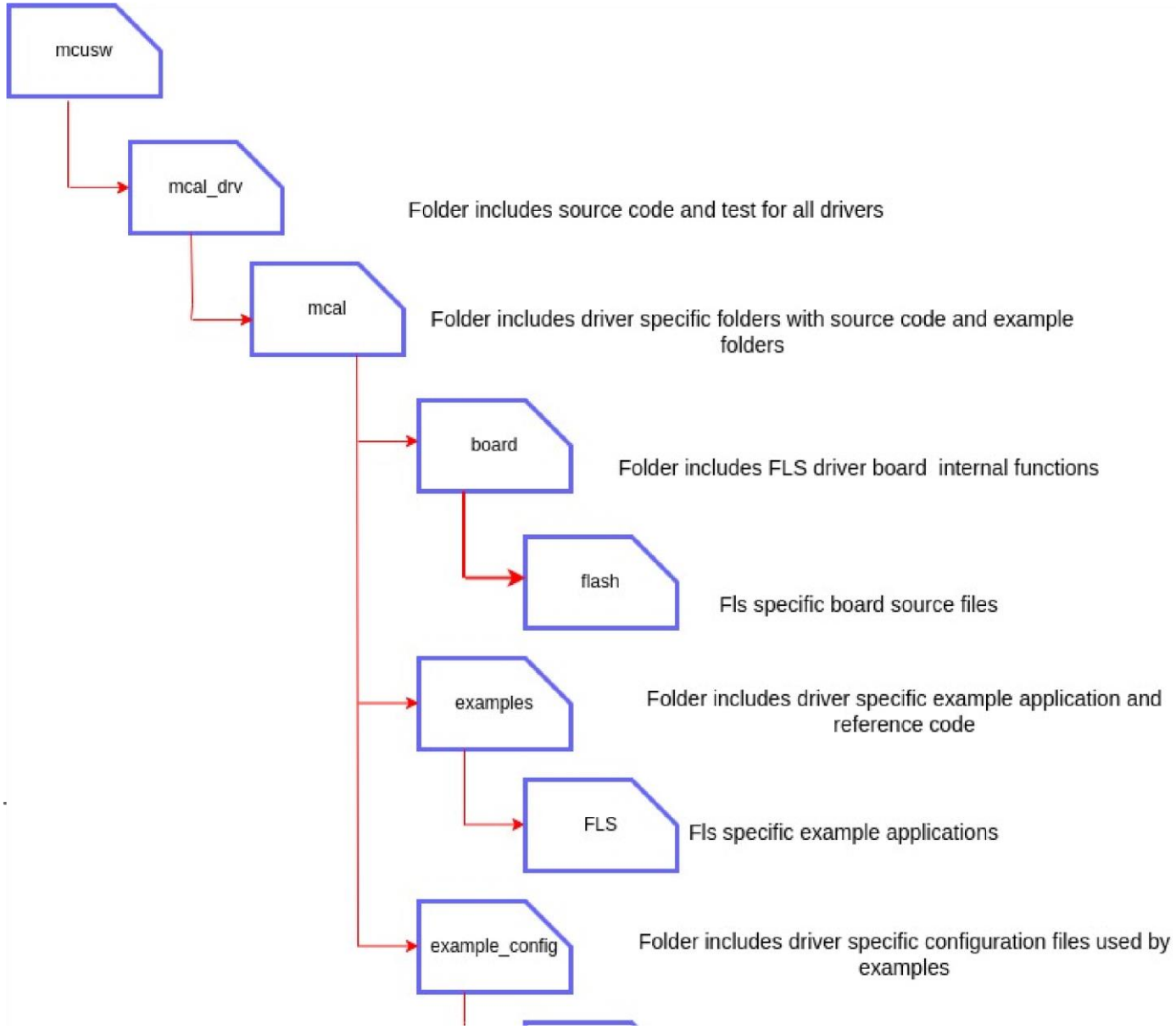

mcusw
mcal_drv
Folder includes source code and test for all drivers
mcal
Folder includes driver specific folders with source code and example folders
Board
Folder includes FLS driver board internal functions
Flash
Folder includes Flash specific board internal functions
src
Folder contains flash specific board source files
inc
Folder contains flash specific board header files
FLS
FLS specific files
Fls_hw
Hardware level API files for FLS
src
FLS source files


«header»
Fee::Fee_Cbk.h
«header»
MemIf::
MemIf_Types.h
«header»
Spi::Spi.h
«header»
Platform_Types:
Platform_Types.h
«header»
Compiler::
Compiler.h
«optional»
«includes»
«optional»
«includes»
«includes»
«header»
Fls_Cfg.h
«includes»
«header»
Fls.h
«includes»
«header»
Std_Types:
Std_Types.h
«header»
Fls_MemMap.h
«includes»
«includes»
«includes»
«includes»
«includes»
«includes»
«includes»
«includes»
«source»
Fls_Irq.c
«source»
Fls_Lcfg.c
«source»
Fls_PBcfg.c
«source»
Fls.c
Fls1.c
Fls_ac.c
«includes»
«optional»
«optional»
«optional»
«mandatory»
«header»
SchM::SchM_Fls.h
«header»
Fls_Cbk.h
«header»
Det::Det.h

Example Application

- Fls_Cfg.h: Shall implement the generated configuration for Pre-Compile variant
- FlsApp.c: Shall implement the example application that demonstrates the use of the driver
- Please note the below configuration and test application associations:

1. fls_app uses Fls_Cfg.c,Fls_Cfg.h
2. fls_app_dac uses Fls_Cfg_dac.c,Fls_Cfg.h
3. fls_app_indac uses Fls_Cfg_indac.c,Fls_Cfg.h
4. fls_app_xip uses Fls_Cfg_xip.c,Fls_Cfg.h
5. fls_app for Intr Mode uses Fls_Cfg_indac.c, Fls_Cfg_Intr_Sample.h


mcusw
mcal_drv
Folder includes source code and test for all drivers
mcal
Folder includes driver specific folders with source code and example folders
board
Folder includes FLS driver board internal functions
flash
Fls specific board source files
examples
Folder includes driver specific example application and reference code
FLS
Fls specific example applications
example_config
Folder includes driver specific configuration files used by examples

Design Identifier	Description
MCAL-7352 - MCAL Module : Flash Driver : header file should include the specific definitons for one flash driver PUBLISHED	MCAL Module : Flash Driver : header file should include the specific definitons for one flash driver

4.3 Configurator

The AUTOSAR Flash Driver Specification details mandatory parameters that shall be configurable via the configurator. Please refer section 10 1 Error Checks are common for variants of devices and we can do this in xdm or generator code.

Design Identifier	Description
MCAL-7414 - MCAL Module : Flash Driver : Config : FlsConfigSet : OSPI Clock Speed PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : OSPI Clock Speed

Design Identifier	Description
MCAL-7325 - MCAL Module : Flash Driver : Config : FlsConfigSet : XIP Read Mode Enable/Disable PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : XIP Read Mode Enable/Disable
MCAL-7296 - MCAL Module : Flash Driver : Config : FlsConfigSet : Direct Access Mode Enable/Disable PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : Direct Access Mode Enable/ Disable
MCAL-7428 - MCAL Module : Flash Driver : Config : FlsSector : start address of sector PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : start address of sector
MCAL-7339 - MCAL Module : Flash Driver : Config : FlsSector : size of this sector PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : size of this sector
MCAL-7298 - MCAL Module : Flash Driver : Config : FlsSector : size of one page of this sector PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : size of one page of this sector

Design Identifier	Description
MCAL-7415 - MCAL Module : Flash Driver : Config : FlsSector : number of continous sectors PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : number of continous sectors
MCAL-7328 - MCAL Module : Flash Driver : Config : FlsSector : flashable sector def PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : flashable sector def
MCAL-7290 - MCAL Module : Flash Driver : Config : FlsSectorList : list of flashable sectors and pages PUBLISHED	MCAL Module : Flash Driver : Config : FlsSectorList : list of flashable sectors and pages
MCAL-7404 - MCAL Module : Flash Driver : Config : FlsConfigSet : max number of bytes to write in normal mode PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : max number of bytes to write in normal mode
MCAL-7302 - MCAL Module : Flash Driver : Config : FlsConfigSet : max number of bytes to read in normal mode PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : max number of bytes to read in normal mode

Design Identifier	Description
MCAL-7355 - MCAL Module : Flash Driver : Config : FlsConfigSet : mapped to job error notification PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : mapped to job error notification
MCAL-7343 - MCAL Module : Flash Driver : Config : FlsConfigSet : Mapped to job end notification PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : Mapped to job end notification
MCAL-7417 - MCAL Module : Flash Driver : Config : FlsConfigSet : container for runtime configuration params PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : container for runtime configuration params
MCAL-7430 - MCAL Module : Flash Driver : Config : FlsGeneral : switch for FlsVersonInfoApi PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : switch for FlsVersonInfoApi
MCAL-7378 - MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_GetStatus PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_GetStatus

Design Identifier	Description
MCAL-7320 - MCAL Module : Flash Driver : Config : FlsGeneral : Job processing triggered by hardware interrupt PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : Job processing triggered by hardware interrupt
MCAL-7280 - MCAL Module : Flash Driver : Config : FlsGeneral : total amount of flash mem in bytes PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : total amount of flash mem in bytes
MCAL-7410 - MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_GetJobResult PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_GetJobResult
MCAL-7384 - MCAL Module : Flash Driver : Config : FlsGeneral : index of driver PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : index of driver
MCAL-7429 - MCAL Module : Flash Driver : Config : FlsGeneral : Switch for development error detection and notification PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : Switch for development error detection and notification

Design Identifier	Description
MCAL-7263 - MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_Compare PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_Compare
MCAL-7256 - MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_BlankCheck PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_BlankCheck
MCAL-7276 - MCAL Module : Flash Driver : Config : FlsGeneral : Flash Memory start address - lower bound PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : Flash Memory start address - lower bound
MCAL-7454 - MCAL Module : Flash Driver : Config : FlsGeneral : Container for general parameters of the flash driver PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : Container for general parameters of the flash driver

4.3.1 **NON Standard configurable parameters**

The design's specific configurable parameters are as follows:

Parameter	Usage Comment
FlsDacEnable	Switch between INDAC and DAC mode.
FlsDtrEnable	Switch between DTR and STR mode.
FlsOspiClkSpeed	Clock Speed Freq Input (133MHz or 166MHz)
FlsXipEnable	Switch to turn on and off XIP mode.
FlsPhyEnable	Switch to turn on and off PHY mode.
FlsDeviceVariant	The driver shall support both VARIANT-POST-BUILD & VARIANT-PRE-COMPILE

Please note that: The supported clock speeds are as follows:

1. 133Mhz (133333333U) - This Clock freq works with all modes of the Flash Driver.
2. 166Mhz (166666666U) - This Clock freq is fully supported in INDAC and DAC mode of operations.

4.3.2 Implementation specific parameters (computed)

The configurator shall determine the maximum number of channels that are configured and generate a macro to define the same. This shall be used to perform range checks on channel configurations and channel ID provided at driver initialization time. Refer section 5.1

4.3.3 Variant Support

The driver shall support both VARIANT-POST-BUILD & VARIANT-PRE-COMPILE

4.4 Error Classification

Errors are classified in two categories, development error and Transient Faults.

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	FLS_SID_INIT
0x01	FLS_SID_ERASE

Service ID	*Service*
0x02	FLS_SID_WRITE
0x03	FLS_SID_CANCEL
0x04	FLS_SID_GET_STATUS
0x05	FLS_SID_GET_JOB_RESULT
0x06	FLS_SID_MAIN_FUNCTION
0x07	FLS_SID_READ
0x08	FLS_SID_COMPARE
0x09	FLS_SID_SET_MODE
0x10	FLS_SID_GET_VERSION_INFO

Service ID	*Service*
0x0A	FLS_SID_BLANK_CHECK

4.4.1 Development Errors

AUTOSAR requires that API functions check the validity of their parameters and module status. 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.

Design Identifier	Description
MCAL-7275 - MCAL Module : Flash Driver : Report error codes when module is busy PUBLISHED	MCAL Module : Flash Driver : Report error codes when module is busy
MCAL-7369 - MCAL Module : Flash Driver : Report error code for read function fail PUBLISHED	MCAL Module : Flash Driver : Report error code for read function fail
MCAL-7308 - MCAL Module : Flash Driver : Report error code for unexpected flash ID PUBLISHED	MCAL Module : Flash Driver : Report error code for unexpected flash ID

Design Identifier	Description
MCAL-7267 - MCAL Module : Flash Driver : Report error code for compare function fail PUBLISHED	MCAL Module : Flash Driver : Report error code for compare function fail
MCAL-7427 - MCAL Module : Flash Driver : Report error code for write function fail PUBLISHED	MCAL Module : Flash Driver : Report error code for write function fail
MCAL-7269 - MCAL Module : Flash Driver : Report error code for erase function fail PUBLISHED	MCAL Module : Flash Driver : Report error code for erase function fail
MCAL-7394 - MCAL Module : Flash Driver : Report error codes for API calls with wrong params PUBLISHED	MCAL Module : Flash Driver : Report error codes for API calls with wrong params
MCAL-7332 - MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization PUBLISHED	MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization

Development Error Reporting

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

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).

Type of Error	Related Error code	Value (Hex)
API service called with wrong parameter	FLS_E_PARAM_CONFIG	0x01
API service called with wrong parameter	FLS_E_PARAM_ADDRESS	0x02
API service called with wrong parameter	FLS_E_PARAM_LENGTH	0x03
API service called with wrong parameter	FLS_E_PARAM_DATA	0x04
API service used without module initialization	FLS_E_UNINIT	0x05
API called when module is busy	FLS_E_BUSY	0x06

Type of Error	Related Error code	Value (Hex)
API called with a Null Pointer	FLS_E_PARAM_POINTER	0x0A

Transient Faults

Transient errors are reported to the DET using the service Det_reportDetTransientFault(). The driver interface Fls.h File Structure lists the SID.

Type of Error	*Related Error code*	*Value (Hex)*
Flash Erase Failed in HW	FLS_E_ERASE_FAILED	0x01
Flash Write Failed in HW	FLS_E_WRITE_FAILED	0x02
Flash Read Failed in HW	FLS_E_READ_FAILED	0x03
Flash Compare Failed in HW	FLS_E_COMPARE_FAILED	0x04
Expected HW ID not matched	FLS_E_UNEXPECTED_FLASH_ID	0x05

Debugging support

Flash driver makes driver status available for debugging. The states MEMIF_UNINIT, MEMIF_IDLE, MEMIF_BUSY can be probed. Driver status variable is defined by the status element in the Fls_DrvObj.

5 Implementation Details

5.1 Data structures and resources

MACROS, Data Types & Structures

Design Identifier	Description
MCAL-7339 - MCAL Module : Flash Driver : Config : FlsSector : size of this sector PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : size of this sector
MCAL-7328 - MCAL Module : Flash Driver : Config : FlsSector : flashable sector def PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : flashable sector def
MCAL-7428 - MCAL Module : Flash Driver : Config : FlsSector : start address of sector PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : start address of sector
MCAL-7298 - MCAL Module : Flash Driver : Config : FlsSector : size of one page of this sector PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : size of one page of this sector

Design Identifier	Description
MCAL-7415 - MCAL Module : Flash Driver : Config : FlsSector : number of continous sectors PUBLISHED	MCAL Module : Flash Driver : Config : FlsSector : number of continous sectors
MCAL-7343 - MCAL Module : Flash Driver : Config : FlsConfigSet : Mapped to job end notification PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : Mapped to job end notification
MCAL-7355 - MCAL Module : Flash Driver : Config : FlsConfigSet : mapped to job error notification PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : mapped to job error notification
MCAL-7302 - MCAL Module : Flash Driver : Config : FlsConfigSet : max number of bytes to read in normal mode PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : max number of bytes to read in normal mode
MCAL-7404 - MCAL Module : Flash Driver : Config : FlsConfigSet : max number of bytes to write in normal mode PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : max number of bytes to write in normal mode

Design Identifier	Description
MCAL-7290 - MCAL Module : Flash Driver : Config : FlsSectorList : list of flashable sectors and pages PUBLISHED	MCAL Module : Flash Driver : Config : FlsSectorList : list of flashable sectors and pages
MCAL-7296 - MCAL Module : Flash Driver : Config : FlsConfigSet : Direct Access Mode Enable/Disable PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : Direct Access Mode Enable/ Disable
MCAL-7325 - MCAL Module : Flash Driver : Config : FlsConfigSet : XIP Read Mode Enable/Disable PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : XIP Read Mode Enable/Disable
MCAL-7414 - MCAL Module : Flash Driver : Config : FlsConfigSet : OSPI Clock Speed PUBLISHED	MCAL Module : Flash Driver : Config : FlsConfigSet : OSPI Clock Speed
MCAL-7396 - MCAL Module : Flash Driver : Type Defs : Fls_configType PUBLISHED	MCAL Module : Flash Driver : Type Defs : Fls_configType

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

Macro	Comments
FLS_BASE_ADDRESS	Flash Memory start address, defines lower boundary for read/writ/erase/compare jobs.
FLS_TOTAL_SIZE	Total amount of flash memory in bytes.
FLS_DRIVER_INDEX	Index of driver, used by FEE - 0 for driver
FLS_BLANK_CHECK_API	Enable/disable FLS Blank Check API
FLS_COMPARE_API	Enable/disable FLS compare API
FLS_DEV_ERROR_DETECT	Enable/disable FLS dev detect error
FLS_GET_JOB_RESULT_API	Enable/disable FLS Fls_GetJobResult function API
FLS_GET_STATUS_API	Enable/disable FLS Fls_GetStatus function API
FLS_USE_INTERRUPTS	Job processing triggered by hardware interrupt.

Macro	Comments
FLS_VERSION_INFO_API	Enable/disable FLS get version info API

5.1.1 Fls_AddressType and Fls_LengthType

TypeDefs of address Type used by driver.

5.1.2 **Fls_ConfigType**

This type of the external data structure contains the initialization data for the Flash Driver

5.1.3 Fls_SectorType

This structure contain information about the Flash Device in use. For this driver design, and internal structure is used instead.

Global Variables

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

Variable	Type	Description	Default Value
Fls_NorOspiInfo	Fls_NOR_InfoType	Flash Device Struct	-
Fls_DrvObj	Fls_DriverObjType	Flash driver object	-

5.2 Dynamic Behavior - Control Flow Diagram

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

- MEMIF_UNINIT: The Flash Driver is not initialized. This is the state before starting driver initialization.
- MEMIF_IDLE: The Flash Driver is not currently transferring any Job. This is the state before the transfer is started or after the transfer is completed.
- MEMIF_BUSY: This is the state after a transfer has started i.e. Job execution is on going.

5.3 Dynamic Behavior - Data Flow Diagram

Not Applicable

5.4 Application Parameters

Fls_Init

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
ConfigPtr	Pointer to flash driver configuration set	0xFFFFFFFF	-	-	NA

Fls_Erase

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
TargetAddress	Target address in flash memory. This address offset will be added to the flash memory base address.	0xFFFFFFFF	-	-	NA
Length	Number of bytes to erase	0-64	-	-	

Fls_Write

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
TargetAddress	Target address in flash memory. This address offset will be added to the flash memory base address.	0xFFFFFFFF	-	-	NA

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
SourceAddress	Pointer to source data buffer	0xFFFFFFFF	-	-	NA
Length	Number of bytes to erase	0-64	-	-	NA

Fls_GetStatus

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	NA

Fls_GetJobResult

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	NA

Fls_Read

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
SourceAddress	source address in flash memory. This address offset will be added to the flash memory base address.	0xFFFFFFFF	-	-	NA
TargetAddress	Pointer to source data buffer	0xFFFFFFFF	-	-	NA
Length	Number of bytes to erase	0-64	-	-	NA

Fls_Compare

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
SourceAddress	Target address in flash memory. This address offset will be added to the flash memory base address.	0xFFFFFFFF	-	-	NA
TargetAddress	Pointer to source data buffer	0xFFFFFFFF	-	-	NA

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Length	Number of bytes to erase	0-64	-	-	NA

Fls_GetVersionInfo

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
versioninfo	Pointer to where to store the version information of this module	-	-	-	NA

Fls_BlankCheck

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
TargetAddress	Target address in flash memory. This address offset will be added to the flash memory base address.	0xFFFFFFFF	-	-	NA

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
SourceAddress	Pointer to source data buffer	0xFFFFFFFF	-	-	NA
Length	Number of bytes to erase	0-64	-	-	NA

Fls_MainFunction

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	NA

Internal, Private Functions that aid FLS module in Job Processing

processJobs

	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	NA

Parameter

Fls_norRead

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	NA

Fls_norWrite

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	NA

Fls_norErase

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	NA

Fls_norCompare

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	NA

Fls_norBlankCheck

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	NA

5.5 Safety Diagnostic Features

OSPI3 - Periodic Software Readback of Static Configuration Registers/OSPI4 - 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 FLS MCAL driver provides the API - **Ospi_RegisterReadback** to readback static and written configuration registers to implement this diagnostic feature **OSPI6 -**

Data Overflow Detection/OSPI7 - Data Underflow Detection

The MCAL Hardware logic detects the data overflow/underflow error detection, The error event is generated as interrupt to CPU. FLS 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.

FLS Mcal provides the API **Fls_dataOverflowUnderflowIntrEnable** to enable the interrupt, **Fls_dataOverflowUnderflowIntrGetStatus** to read status of the interrupt register, **Fls_dataOverflowUnderflowIntrStatusClear** to clear the interrupt status and **Fls_dataOverflowUnderflowIntrDisable** to disable the interrupt.

Design Identifier	Description
MCAL-7401 - FLS: Safety Diagnostics: OSPI6: Data underflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data underflow detection
MCAL-7388 - FLS: Safety Diagnostics: OSPI6: Data overflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data overflow detection
MCAL-7375 - FLS: Safety Diagnostics: OSPI4: Software Readback of Written Configuration PUBLISHED	FLS: Safety Diagnostics: OSPI4: Software Readback of Written Configuration
MCAL-7300 - FLS: Safety Diagnostics: OSPI3: Periodic Software Readback of Static Configuration Registers PUBLISHED	FLS: Safety Diagnostics: OSPI3: Periodic Software Readback of Static Configuration Registers

Design Identifier	Description
MCAL-7423 - FLS: Safety Diagnostics: OSPI1A: Software test of basic functionality using external memory. PUBLISHED	FLS: Safety Diagnostics: OSPI1A: Software test of basic functionality using external memory.
MCAL-7371 - FLS: Safety Diagnostics: OSPI1B: Software Test of Basic Functionality Using Digital I/O Loopback PUBLISHED	FLS: Safety Diagnostics: OSPI1B: Software Test of Basic Functionality Using Digital I/O Loopback

6 Low Level Definitions

6.1 Driver API's

For the standard API's please refer 8.3 of Reference 1 - AUTOSAR 4.3.1. Sections below highlight other design considerations for the implementation.

6.1.1 Fls_Init

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

Design Identifier	Description
MCAL-7254 - MCAL Module : Flash Driver : Func Defs : Fls_Init check to see if module is busy PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Init check to see if module is busy
MCAL-7366 - MCAL Module : Flash Driver : Func Defs : Fls_Init set FLS module state to idle after init PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Init set FLS module state to idle after init
MCAL-7312 - MCAL Module : Flash Driver : Func Defs : Fls_Init store pointer to config set in variable PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Init store pointer to config set in variable

Design Identifier	Description
MCAL-7425 - MCAL Module : Flash Driver : Func Defs : Fls_Init set flash job state result to OK after init PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Init set flash job state result to OK after init
MCAL-7289 - MCAL Module : Flash Driver : Func Defs : Fls_Init initialize global variables and controller registers PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Init initialize global variables and controller registers
MCAL-7259 - MCAL Module : Flash Driver : Func Defs : Fls_Init check contents of given config set PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Init check contents of given config set
MCAL-7330 - MCAL Module : Flash Driver : Func Defs : Fls_Init initialize FLS module with given params from config set PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Init initialize FLS module with given params from config set
MCAL-7332 - MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization PUBLISHED	MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization

Design Identifier	Description
MCAL-7394 - MCAL Module : Flash Driver : Report error codes for API calls with wrong params PUBLISHED	MCAL Module : Flash Driver : Report error codes for API calls with wrong params
MCAL-7275 - MCAL Module : Flash Driver : Report error codes when module is busy PUBLISHED	MCAL Module : Flash Driver : Report error codes when module is busy

6.1.2 Fls_Erase

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

Design Identifier	Description
MCAL-7445 - MCAL Module : Flash Driver : Func Defs : FLs_Erase return okay PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase return okay

Design Identifier	Description
MCAL-7464 - MCAL Module : Flash Driver : Func Defs : FLs_Erase check initialization PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase check initialization
MCAL-7327 - MCAL Module : Flash Driver : Func Defs : FLs_Erase can erase one or more complete flash sectors PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase can erase one or more complete flash sectors
MCAL-7286 - MCAL Module : Flash Driver : Func Defs : FLs_Erase check is module is busy PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase check is module is busy
MCAL-7249 - MCAL Module : Flash Driver : Func Defs : FLs_Erase execute erase async in main function PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase execute erase async in main function
MCAL-7307 - MCAL Module : Flash Driver : Func Defs : FLs_Erase initiate erase job PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase initiate erase job

Design Identifier	Description
MCAL-7399 - MCAL Module : Flash Driver : Func Defs : FLs_Erase set job results to pending PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase set job results to pending
MCAL-7398 - MCAL Module : Flash Driver : Func Defs : FLs_Erase set FLS module stats to budy PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase set FLS module stats to budy
MCAL-7418 - MCAL Module : Flash Driver : Func Defs : FLs_Erase check the specified memory region with end address PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase check the specified memory region with end address
MCAL-7421 - MCAL Module : Flash Driver : Func Defs : FLs_Erase check the specified memory region with start address PUBLISHED	MCAL Module : Flash Driver : Func Defs : FLs_Erase check the specified memory region with start address
MCAL-7394 - MCAL Module : Flash Driver : Report error codes for API calls with wrong params PUBLISHED	MCAL Module : Flash Driver : Report error codes for API calls with wrong params

Design Identifier	Description
MCAL-7332 - MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization PUBLISHED	MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization

6.1.3 Fls_Write

Refer section 8.3.3 of the FLASH *Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-7350 - MCAL Module : Flash Driver : Func Defs : Fls_Write return OK PUBLISHED	Flash Driver : Func Defs : Fls_Write return OK
MCAL-7295 - MCAL Module : Flash Driver : Func Defs : Fls_Write check module is initialized PUBLISHED	Flash Driver : Func Defs : Fls_Write check module is initialized

Design Identifier	Description
MCAL-7265 - MCAL Module : Flash Driver : Func Defs : Fls_Write check if module is busy PUBLISHED	Flash Driver : Func Defs : Fls_Write check if module is busy
MCAL-7368 - MCAL Module : Flash Driver : Func Defs : Fls_Write set the job result to pending PUBLISHED	Flash Driver : Func Defs : Fls_Write set the job result to pending
MCAL-7248 - MCAL Module : Flash Driver : Func Defs : Fls_Write set FLS module status to busy PUBLISHED	Flash Driver : Fls_MainFunction issue one sector erase command in each cycle
MCAL-7288 - MCAL Module : Flash Driver : Func Defs : Fls_Write execute write async in main PUBLISHED	Flash Driver : Func Defs : Fls_Write execute write async in main
MCAL-7261 - MCAL Module : Flash Driver : Func Defs : Fls_Write check given data is not null pointer PUBLISHED	Flash Driver : Func Defs : Fls_Write check given data is not null pointer

Design Identifier	Description
MCAL-7271 - MCAL Module : Flash Driver : Func Defs : Fls_Write check write end address PUBLISHED	Flash Driver : Func Defs : Fls_Write check write end address
MCAL-7359 - MCAL Module : Flash Driver : Func Defs : Fls_Write check write start address PUBLISHED	Flash Driver : Func Defs : Fls_Write check write start address
MCAL-7394 - MCAL Module : Flash Driver : Report error codes for API calls with wrong params PUBLISHED	Flash Driver : Report error codes for API calls with wrong params
MCAL-7332 - MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization PUBLISHED	Flash Driver : Report error codes for API calls prior to module initilization
MCAL-7275 - MCAL Module : Flash Driver : Report error codes when module is busy PUBLISHED	Flash Driver : Report error codes when module is busy

Design Identifier	Description
MCAL-7390 - MCAL Module : Flash Driver : Func Defs : Fls_Write intiate the write job PUBLISHED	Flash Driver : Func Defs : Fls_Write intiate the write job
MCAL-7442 - MCAL Module : Flash Driver : Func Defs : Fls_Write reduce overall time PUBLISHED	Flash Driver : Func Defs : Fls_Write reduce overall time

6.1.4 Fls_GetStatus

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

Design Identifier	Description
MCAL-7439 - MCAL Module : Flash Driver : Func Defs : Fls_GetStatus return module status with sync PUBLISHED	Flash Driver : Func Defs : Fls_GetStatus return module status with sync

Design Identifier	Description
MCAL-7408 - MCAL Module : Flash Driver : Func Defs : Fls_GetStatus turn on/off pre-compile with config set PUBLISHED	Flash Driver : Func Defs : Fls_GetStatus turn on/off pre-compile with config set

6.1.5 Fls_GetJobResult

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

Design Identifier	Description
MCAL-7340 - MCAL Module : Flash Driver : Func Defs : Fls_GetJobResult check for initialization PUBLISHED	Flash Driver : Func Defs : Fls_GetJobResult check for initialization
MCAL-7416 - MCAL Module : Flash Driver : Func Defs : Fls_GetJobResult turn on/off pre-compile with config set PUBLISHED	Flash Driver : Func Defs : Fls_GetJobResult turn on/off pre-compile with config set

Design Identifier	Description
MCAL-7413 - MCAL Module : Flash Driver : Func Defs : Fls_GetJobResult return result of last job with sync PUBLISHED	Flash Driver : Func Defs : Fls_GetJobResult return result of last job with sync
MCAL-7353 - MCAL Module : Flash Driver : Func Defs : Fls_GetJobResult last job results only PUBLISHED	Flash Driver : Func Defs : Fls_GetJobResult last job results only
MCAL-7332 - MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization PUBLISHED	Flash Driver : Report error codes for API calls prior to module initilization

6.1.6 Fls_Read

Refer section 8.3.7 of the FLASH *Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-7337 - MCAL Module : Flash Driver : Func Defs : Fls_Read return OK PUBLISHED	Flash Driver : Func Defs : Fls_Read return OK
MCAL-7381 - MCAL Module : Flash Driver : Func Defs : Fls_Read check if initialized PUBLISHED	Flash Driver : Func Defs : Fls_Read check if initialized
MCAL-7341 - MCAL Module : Flash Driver : Func Defs : Fls_Read read from flash mem PUBLISHED	Flash Driver : Func Defs : Fls_Read read from flash mem
MCAL-7297 - MCAL Module : Flash Driver : Func Defs : Fls_Read check that driver is busy PUBLISHED	Flash Driver : Func Defs : Fls_Read check that driver is busy
MCAL-7247 - MCAL Module : Flash Driver : Func Defs : Fls_Read initiate the read job PUBLISHED	Flash Driver : Func Defs : Fls_Read initiate the read job

Design Identifier	Description
MCAL-7304 - MCAL Module : Flash Driver : Func Defs : Fls_Read set job result to pending PUBLISHED	Flash Driver : Func Defs : Fls_Read set job result to pending
MCAL-7301 - MCAL Module : Flash Driver : Func Defs : Fls_Read set the module status to busy PUBLISHED	Func Defs : Fls_Read set the module status to busy
MCAL-7453 - MCAL Module : Flash Driver : Func Defs : Fls_Read call read only after initialization PUBLISHED	Func Defs : Fls_Read call read only after initialization
MCAL-7285 - MCAL Module : Flash Driver : Func Defs : Fls_Read check the end address PUBLISHED	Flash Driver : Func Defs : Fls_Read check the end address
MCAL-7282 - MCAL Module : Flash Driver : Func Defs : Fls_Read check the start address PUBLISHED	Func Defs : Fls_Read check the start address

Design Identifier	Description
MCAL-7462 - MCAL Module : Flash Driver : Func Defs : Fls_Read check that given data pointer is not null PUBLISHED	Func Defs : Fls_Read check that given data pointer is not null
MCAL-7394 - MCAL Module : Flash Driver : Report error codes for API calls with wrong params PUBLISHED	Flash Driver : Report error codes for API calls with wrong params
MCAL-7275 - MCAL Module : Flash Driver : Report error codes when module is busy PUBLISHED	Flash Driver : Report error codes when module is busy

6.1.7 Fls_Compare

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

Design Identifier	Description
MCAL-7281 - MCAL Module : Flash Driver : Func Defs : Fls_Compare return OK PUBLISHED	Flash Driver : Func Defs : Fls_Compare return OK
MCAL-7258 - MCAL Module : Flash Driver : Func Defs : Fls_Compare initiate the job PUBLISHED	Flash Driver : Func Defs : Fls_Compare initiate the job
MCAL-7345 - MCAL Module : Flash Driver : Func Defs : Fls_Compare set status to busy PUBLISHED	Flash Driver : Func Defs : Fls_Compare set status to busy
MCAL-7321 - MCAL Module : Flash Driver : Func Defs : Fls_Compare compare data in flash mem to buffer PUBLISHED	Flash Driver : Func Defs : Fls_Compare compare data in flash mem to buffer
MCAL-7380 - MCAL Module : Flash Driver : Func Defs : Fls_Compare check that module is initialized PUBLISHED	Flash Driver : Func Defs : Fls_Compare check that module is initialized

Design Identifier	Description
MCAL-7336 - MCAL Module : Flash Driver : Func Defs : Fls_Compare check that driver is not busy PUBLISHED	Flash Driver : Func Defs : Fls_Compare check that driver is not busy
MCAL-7393 - MCAL Module : Flash Driver : Func Defs : Fls_Compare execute job async in main PUBLISHED	Flash Driver : Func Defs : Fls_Compare execute job async in main
MCAL-7459 - MCAL Module : Flash Driver : Func Defs : Fls_Compare set job result to pending PUBLISHED	Flash Driver : Func Defs : Fls_Compare set job result to pending
MCAL-7293 - MCAL Module : Flash Driver : Func Defs : Fls_Compare turn on/off pre-compule with config set PUBLISHED	Flash Driver : Func Defs : Fls_Compare turn on/off pre-compule with config set
MCAL-7463 - MCAL Module : Flash Driver : Func Defs : Fls_Compare check the end address PUBLISHED	Flash Driver : Func Defs : Fls_Compare check the end address

Design Identifier	Description
MCAL-7420 - MCAL Module : Flash Driver : Func Defs : Fls_Compare check the start address PUBLISHED	Flash Driver : Func Defs : Fls_Compare check the start address
MCAL-7435 - MCAL Module : Flash Driver : Func Defs : Fls_Compare check that given data pointer is no null PUBLISHED	Flash Driver : Func Defs : Fls_Compare check that given data pointer is no null
MCAL-7394 - MCAL Module : Flash Driver : Report error codes for API calls with wrong params PUBLISHED	Flash Driver : Report error codes for API calls with wrong params
MCAL-7332 - MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization PUBLISHED	Flash Driver : Report error codes for API calls prior to module initilization
MCAL-7275 - MCAL Module : Flash Driver : Report error codes when module is busy PUBLISHED	Flash Driver : Report error codes when module is busy

6.1.8 Fls_GetVersionInfo

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

Design Identifier	Description
MCAL-7252 - MCAL Module : Flash Driver : Func Defs : Fls_GetVersionInfo check for null pointer PUBLISHED	Flash Driver : Func Defs : Fls_GetVersionInfo check for null pointer
MCAL-7394 - MCAL Module : Flash Driver : Report error codes for API calls with wrong params PUBLISHED	Flash Driver : Report error codes for API calls with wrong params

6.1.9 Fls_BlankCheck

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

Design Identifier	Description
MCAL-7383 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck return OK PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck return OK

Design Identifier	Description
MCAL-7456 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck initiate verification PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck initiate verification
MCAL-7294 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck check driver is initialized PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck check driver is initialized
MCAL-7306 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck check that driver is not busy PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck check that driver is not busy
MCAL-7279 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck set the job result to pending PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck set the job result to pending
MCAL-7310 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck execute the job async in main PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck execute the job async in main

Design Identifier	Description
MCAL-7361 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck set module status to busy PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck set module status to busy
MCAL-7433 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck check start address PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck check start address
MCAL-7394 - MCAL Module : Flash Driver : Report error codes for API calls with wrong params PUBLISHED	Flash Driver : Report error codes for API calls with wrong params
MCAL-7332 - MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization PUBLISHED	Flash Driver : Report error codes for API calls prior to module initilization
MCAL-7275 - MCAL Module : Flash Driver : Report error codes when module is busy PUBLISHED	Flash Driver : Report error codes when module is busy

6.1.10 Fls_MainFunction

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

Design Identifier	Description
MCAL-7403 - MCAL Module : Flash Driver : Fls_MainFunction check module is initialized PUBLISHED	Flash Driver : Fls_MainFunction check module is initialized
MCAL-7376 - MCAL Module : Flash Driver : Fls_MainFunction only process specified data PUBLISHED	Flash Driver : Fls_MainFunction only process specified data
MCAL-7305 - MCAL Module : Flash Driver : Fls_MainFunction shall process jobs w/0 HW interrupt support PUBLISHED	Flash Driver : Fls_MainFunction shall process jobs w/0 HW interrupt support
MCAL-7443 - MCAL Module : Flash Driver : Fls_MainFunction return without any action when no job is pending PUBLISHED	Flash Driver : Fls_MainFunction return without any action when no job is pending

Design Identifier	Description
MCAL-7268 - MCAL Module : Flash Driver : Fls_MainFunction issue one sector erase command in each cycle PUBLISHED	Flash Driver : Fls_MainFunction issue one sector erase command in each cycle
MCAL-7448 - MCAL Module : Flash Driver : Fls_MainFunction set job results to inconsistent is mismatch while compare PUBLISHED	Flash Driver : Fls_MainFunction set job results to inconsistent is mismatch while compare
MCAL-7332 - MCAL Module : Flash Driver : Report error codes for API calls prior to module initilization PUBLISHED	Flash Driver : Report error codes for API calls prior to module initilization

6.1.11 Fls_Cancel

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

Design Identifier	Description
MCAL-7412 - MCAL Module : Flash Driver : Func Defs : Fls_Cancel cancel on going flash job PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Cancel cancel on going flash job
MCAL-7458 - MCAL Module : Flash Driver : Func Defs : Fls_Cancel set job result to cancelled PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Cancel set job result to cancelled
MCAL-7270 - MCAL Module : Flash Driver : Func Defs : Fls_Cancel check for initialization PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Cancel check for initialization
MCAL-7274 - MCAL Module : Flash Driver : Func Defs : Fls_Cancel set module status to idle PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Cancel set module status to idle
MCAL-7319 - MCAL Module : Flash Driver : Func Defs : Fls_Cancel reset processing job variables PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Cancel reset processing job variables

Design Identifier	Description
MCAL-7260 - MCAL Module : Flash Driver : Func Defs : Fls_Cancel abort running job with sync PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Cancel abort running job with sync
MCAL-7299 - MCAL Module : Flash Driver : Func Defs : Fls_Cancel call error notification about cancellation PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Cancel call error notification about cancellation
MCAL-7274 - MCAL Module : Flash Driver : Func Defs : Fls_Cancel set module status to idle PUBLISHED	MCAL Module : Flash Driver : Func Defs : Fls_Cancel turn on/off pre-compile for config set
MCAL-7289 - MCAL Module : Flash Driver : Func Defs : Fls_Init initialize global variables and controller registers PUBLISHED	MCAL Module : Flash Driver : Fls_JobErrorNotification called when job is cancelled
MCAL-7326 - MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_CancelApi PUBLISHED	MCAL Module : Flash Driver : Config : FlsGeneral : switch for Fls_CancelApi

6.1.12 Internal, Private Functions that aid FLS module in Job Processing

6.1.12.1 processJobs

Design Identifier	Description
MCAL-7278 - MCAL Module : Flash Driver : Fls_MainFunction after jobs complete, set job result to OK PUBLISHED	Flash Driver : Fls_MainFunction after jobs complete, set job result to OK
MCAL-7409 - MCAL Module : Flash Driver : Fls_MainFunction after jobs complete, set modle state to idle PUBLISHED	Flash Driver : Fls_MainFunction after jobs complete, set modle state to idle
MCAL-7448 - MCAL Module : Flash Driver : Fls_MainFunction set job results to inconsistent is mismatch while compare PUBLISHED	Flash Driver : Fls_MainFunction set job results to inconsistent is mismatch while compare
MCAL-7269 - MCAL Module : Flash Driver : Report error code for erase function fail PUBLISHED	Flash Driver : Report error code for erase function fail

Design Identifier	Description
MCAL-7369 - MCAL Module : Flash Driver : Report error code for read function fail PUBLISHED	Flash Driver : Report error code for read function fail
MCAL-7267 - MCAL Module : Flash Driver : Report error code for compare function fail PUBLISHED	Flash Driver : Report error code for compare function fail
MCAL-7356 - MCAL Module : Flash Driver : Fls_JobEndNotification called when jobs have positive results PUBLISHED	Flash Driver : Fls_JobEndNotification called when jobs have positive results
MCAL-7266 - MCAL Module : Flash Driver : Fls_JobErrorNotification called when compare differs PUBLISHED	Flash Driver : Fls_JobErrorNotification called when compare differs

6.1.12.2 Fls_norRead

Design Identifier	Description
MCAL-7327 - MCAL Module : Flash Driver : Func Defs : FLs_Erase can erase one or more complete flash sectors PUBLISHED	Flash Driver : Func Defs : FLs_Erase can erase one or more complete flash sectors
MCAL-7341 - MCAL Module : Flash Driver : Func Defs : Fls_Read read from flash mem PUBLISHED	Flash Driver : Func Defs : Fls_Read read from flash mem

6.1.12.3 Fls_norWrite

Design Identifier	Description
MCAL-7303 - MCAL Module : Flash Driver : Func Defs : Fls_Write write one or more complete flash pages PUBLISHED	Flash Driver : Func Defs : Fls_Write write one or more complete flash pages

Design Identifier	Description
MCAL-7315 - MCAL Module : Flash Driver : Func Defs : Fls_Write program flash mem block PUBLISHED	Flash Driver : Func Defs : Fls_Write program flash mem block

6.1.12.4 Fls_norErase

Design Identifier	Description
MCAL-7327 - MCAL Module : Flash Driver : Func Defs : FLs_Erase can erase one or more complete flash sectors PUBLISHED	Flash Driver : Func Defs : FLs_Erase can erase one or more complete flash sectors
MCAL-7249 - MCAL Module : Flash Driver : Func Defs : FLs_Erase execute erase async in main function PUBLISHED	Flash Driver : Func Defs : FLs_Erase execute erase async in main function
MCAL-7400 - MCAL Module : Flash Driver : Func Defs : FLs_Erase erase sepcified memory block PUBLISHED	Flash Driver : Func Defs : FLs_Erase erase sepcified memory block

6.1.12.5 Fls_norCompare

Design Identifier	Description
MCAL-7406 - MCAL Module : Flash Driver : Func Defs : Fls_Compare compare the correct block of memory PUBLISHED	Flash Driver : Func Defs : Fls_Compare compare the correct block of memory
MCAL-7258 - MCAL Module : Flash Driver : Func Defs : Fls_Compare initiate the job PUBLISHED	Flash Driver : Func Defs : Fls_Compare initiate the job

6.1.12.6 Fls_norBlankCheck

Design Identifier	Description
MCAL-7456 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck initiate verification PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck initiate verification

Design Identifier	Description
MCAL-7437 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck check mem area has been erased PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck check mem area has been erased
MCAL-7310 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck execute the job async in main PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck execute the job async in main
MCAL-7419 - MCAL Module : Flash Driver : Func_Defs : Fls_BlankCheck check that mem area is erased PUBLISHED	Flash Driver : Func_Defs : Fls_BlankCheck check that mem area is erased

6.1.11.7.Ospi_RegisterReadback

	Description	Comments
Service Name	Ospi_RegisterReadback	Can potentially be turned OFF
Syntax	Ospi_RegisterReadback(P2VAR(CSL_ospi_flash_cfgRegs, AUTOMATIC, SPI_APPL_DATA) RegRbPtr)	E_OK: Register read back has been done, E_NOT_OK: Register read back failed

	Description	Comments
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter inout	RegRbPtr - Pointer to where to store the readback values. If this pointer is NULL_PTR, then the API will return E_NOT_OK	A pointer of type **Ospi_RegisterReadbackType** which holds the read back values
Return Value	Standard return type	E_OK, E_NOT_OK

Design Identifier	Description
MCAL-7375 - FLS: Safety Diagnostics: OSPI4: Software Readback of Written Configuration PUBLISHED	FLS: Safety Diagnostics: OSPI4: Software Readback of Written Configuration
MCAL-7300 - FLS: Safety Diagnostics: OSPI3: Periodic Software Readback of Static Configuration Registers PUBLISHED	FLS: Safety Diagnostics: OSPI3: Periodic Software Readback of Static Configuration Registers

6.1.12.8.Fls_dataOverflowUnderflowIntrEnable

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

	Description	Comments
Service Name	Fls_dataOverflowUnderflowIntrEnable	Fls_dataOverflowUnderflowIntrEnable Enable Under/Overflow Interrupts of the hardware unit
Syntax	(Std_ReturnType, FLS_CODE) Fls_dataOverflowUnderflowIntrEnable(uint32 intrFlag)	
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	intrFlag	Interrupt Enable flag for Underflow=2U Interrupt Enable flag for Overflow=128U

	Description	Comments
Parameter out	NA	
Return Value	Std_ReturnType	E_OK: Interrupt Enabled E_NOT_OK: Interrupt Enable failed

Design Identifier	Description
MCAL-7401 - FLS: Safety Diagnostics: OSPI6: Data underflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data underflow detection
MCAL-7388 - FLS: Safety Diagnostics: OSPI6: Data overflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data overflow detection

6.1.12.9.Fls_dataOverflowUnderflowIntrDisable

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

	Description	Comments
Service Name	Fls_dataOverflowUnderflowIntrDisable	Spi_dataOverflowUnderflowIntrDisable Disable Under/Overflow Interrupts of the hardware unit and returns the status.
Syntax	(Std_ReturnType, FLS_CODE) Fls_dataOverflowUnderflowIntrDisable(uint32 intrFlag)	
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	intrFlag	Interrupt Disable flag for Underflow=2U Interrupt Disable flag for Overflow=128U
Parameter out	NA	

	Description	Comments
Return Value	Std_ReturnType	E_OK: Interrupt Disabled E_NOT_OK: Interrupt Disabled failed

Design Identifier	Description
MCAL-7401 - FLS: Safety Diagnostics: OSPI6: Data underflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data underflow detection
MCAL-7388 - FLS: Safety Diagnostics: OSPI6: Data overflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data overflow detection

6.1.12.10.Fls_dataOverflowUnderflowIntrGetStatus

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

	Description	Comments
Service Name	Fls_dataOverflowUnderflowIntrGetStatus	Fls_dataOverflowUnderflowIntrGetStatus gets status of Under/Overflow Interrupts of the hardware unit

	Description	Comments
Syntax	(Ospi_IrqStatusType, FLS_CODE) Fls_dataOverflowUnderflowIntrGetStatus(uint32 intFlags)	
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	intrFlag	Interrupt Enable flag
Parameter out	NA	
Return Value	ospiIrqstatus	OSPI_NO_EVENT: No underflow event OSPI_EVENT_PENDING: Underflow Event OSPI_STATUS_READ_FAIL: Status read fail

Design Identifier	Description
MCAL-7401 - FLS: Safety Diagnostics: OSPI6: Data underflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data underflow detection
MCAL-7388 - FLS: Safety Diagnostics: OSPI6: Data overflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data overflow detection

6.1.12.11. Fls_dataOverflowUnderflowIntrStatusClear

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

	Description	Comments
Service Name	Fls_dataOverflowUnderflowIntrStatusClear	Fls_dataOverflowUnderflowIntrStatusClear clear Under/Overflow Interrupts of the hardware unit and returns the status.
Syntax	(Std_ReturnType, FLS_CODE) Fls_dataOverflowUnderflowIntrStatusClear(uint32 intrFlag)	
Service ID	NA	

	Description	Comments
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	intrFlag	Interrupt Enable flag for Underflow=2U Interrupt Enable flag for Overflow=128U
Parameter out	NA	
Return Value	Std_ReturnType	E_OK: Interrupt status clears E_NOT_OK: Interrupt status clear failed

Design Identifier	Description
MCAL-7401 - FLS: Safety Diagnostics: OSPI6: Data underflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data underflow detection
MCAL-7388 - FLS: Safety Diagnostics: OSPI6: Data overflow detection PUBLISHED	FLS: Safety Diagnostics: OSPI6: Data overflow detection

7 Performance Objectives

7.1 Resource Consumption Objectives

	ROM - Data(KB)	RAM - Program(KB)	RAM - Data(KB)	Stack Size (KB)	EEPROM (KB)	% CPU Utilization
				5		

ROM - Program(KB)

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

The OSPI hardware module provides a couple ways to transfer data: DAC mode and INDAC mode.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
1	CPU must have minimal restrictions while the flash driver is processing. Polling vs Interrupt mode should be allowed in the implementation.	• **DAC Mode** Direct Access Controller (or DAC) refers to the operation where data interface accesses directly trigger a read or write to Flash memory. It is memory mapped and can be used to both access and directly execute code from external Flash memory. Access that use DAC do not use the embedded SRAM within the OSPI module. • **Advantages:** • Flash memory is memory mapped, so Execute-in-place is supported.		In case of a execute-in-place/ memory mapped use case, DAC Poll mode should be used. Since DAC mode enables the PHY Module, it gives higher performance results for read operations. For better performance for write operations, INDAC Interrupt mode should be used.	

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs

		• Integration with the PHY Module gives optimized performance.
		• **Disadvantages:**
		  • Needs a data interface access to trigger for any number of bytes transfer.
		  • Only supports poll mode implementation as it does not use SRAM (which is used to trigger interrupts otherwise).			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		• Design implementation is blocking, since in poll mode, control stays within the driver until all the data has been transfered.			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		• **INDAC Mode** Indirect Access Controller (INDAC) uses the internal SRAM of the OSPI module. Data transfers then happen between the SRAM and Flash memory or SRAM and system memory, so no memory mapping is needed. The aim of the indirect mode of operation is to transfer significant numbers of bytes to/from Flash memory. Indirect operations are controlled and triggered by software via specific read/write transfer registers. • **Advantages:**			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		• Bulk transfers take place between system and flash memory in the most efficient manner. Fewest possible write cycles carried out inside flash device to maximize life of device.			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		• INDAC supports poll and interrupt based implementation. The embedded SRAM used by INDAC provides interrupt mechanism. Interrupts are triggered when SRAM levels fall below a watermark, and also when a write/read operation finishes from within the SRAM.			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		• Design implementation is non-blocking and allows the control to return to the application (CPU) while the data is still transferring in hardware. • **Disadvantages:** • Execute-in-place not supported.			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		• When using interrupt based INDAC mode, the FlsMaxWriteNormalMode has to be set to the page size of the flash device. • PHY module is not integrated.			

8.2 Selecting Flash device information structure

The Flash device information should be available to the Flash Driver. Information such as Flash device sector size, number of sectors, page size, etc. is needed.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs

1	Avoid user input errors.	• **Use device specific, local structure**
		Only one Flash device is connected to the J7EVM, and this driver implementation only supported one device.
		  • **Advantages:**
		  • Simpler design.
		  • The values of flash device are constant and are stored in local, internal structure. This information should not be changed by user.
		  • **Disadvantages:**
		  • Does not follow Autosar Spec.
				Option 1 is selected. The configuration parameter for FlsSector is not editable by the user, to avoid any error. Option 2 is useful when there are various flash devices connect, with varied size details. Since this Flash device contains sectors with all same sizes, Option 1 makes a simpler and more efficient design.	

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs

		• **Use spec specific structure** Autosar Spec provide a specific structure that would come as input from user as a configuration parameter.
		  • **Advantages:**
		  • This would follow the Autosar Spec requirement.
		  • **Disadvantages:**
		  • Increased chance of user input error, and complexity, as only one Flash device is connected and supported for J7EVM.			

8.3 Selecting the SPI Driver

The Flash driver module needs to use SPI protocol to perform the data transfers between system and Flash device. Specifically, this Flash driver requires OSPI interface and driver.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs

1	SPI driver should have least impact on the latency and performance of the Flash driver.	• **Use PDK SPI Driver**
		TI's PDK package provides several different SPI drivers: OSPI, QSPI, NAND, or NOR. This driver could directly be used by MCAL Flash Driver.
		  • **Advantages:**
		  • Simpler design. PDK SPI driver APIs could be called by Flash driver for data transfers.
		  • Reduced Flash Driver Size.
		  • **Disadvantages:**
		  • The calls to PDK SPI driver would add latency and decrease overall performance of Flash Driver, since the driver is not completely integrated.
			Use of Merge OSPI Driver	Option 3 is selected as it reduces the latency and would give better performance. Also, this option decreases the dependency on other modules and packages.	

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		• Adds dependency on separate TI provided package. • The PDK SPI driver design is not suitable for Autosar Spec Flash Driver.			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs

		Autosar Spec specific MCAL SPI Driver is already present in MCUSW package.
		• **Advantages:**
		  • This would follow the Autosar Spec requirements.
		• **Disadvantages:**
		  • OSPI driver has not been implemented, so would require additional steps to integrate into Flash Driver.
		  • The calls to MCAL SPI driver would add latency and decrease overall performance of Flash Driver, since the driver is not completely integrated.			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs

		• **Merge OSPI Driver**
		The required OSPI driver API and functions can be implemented inside the Flash driver itself. Therefore, the OSPI driver would be part of the Flash Driver module.
		  • **Advantages:**
		  • Reduced latency as the OSPI API are within the Flash Driver module.
		  • Narrows dependency on other modules and packages (with the exception of CSL APIs).
		  • **Disadvantages:**
		  • Flash Driver code size increased.			

8.4 Fls driver and Functional safety

No.	Decision Criteria	Alternatives	Selected alternative	Rationale

1	FLS driver needs to be split into 2 layers:
	1. Generic for handling MCALnuance, common OSPI, QSPI read/write functions. (OSPI driver)
	2. And then a flash specific layer which deals with commands, write nuances, etc. (Flash device driver)
		• The flash specific layer is just an example (not safety certified). As it is practically not possible to implement everything for the various flash we have in market. So when we do safety for FLS, we should just focus on the generic layer and not on the flash specific layer.
		• **Advantages:**
		  ◦ We would only have to safety certify OSPI/XSPI driver.
		  ◦ FLS driver will become customizable and scalable for other devices.
		  ◦ TI will only be able to test FLS driver with flash devices which are available to us. Other test effort is moved to customer side.
		• **Disadvantages:**
				We will go with option 1→ if flash device is not common, we will not certify it. But, if it is a common one, maybe we can go and certify for just that device.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale
		• Flash device driver will not be safety certified. • We will have to limit flash driver deliverables to OSPI/xSPI. • Flash will not be safety certified		

		• Continue with currently planned approach, but let customer make customization for FLS driver.
		
		Customer will need to take FLS driver through safety certification.
		
		• **Advantages:**
		  • TI will not require to take FLS driver through functional safety.
		• **Disadvantages:**
		  • Customers will require to take FLS driver through safety.		

No.	Decision Criteria	Alternatives	Selected alternative	Rationale

		• Continue with currently planned approach, but restrict customizations.
		  • **Advantages:**
		  • Entire FLS driver will be safety certified.
		  • **Disadvantages:**
		  • It will break safety certification if FLS driver is modified.		

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 • Performance Testing**
  - Validating ECUC Parameter: Configuration for each test case shall be generated by EB Tresos command line.
  - While testing the Write, Read, Erase, BlankCheck and Compare APIs, care should be taken to check the return value to see if job is finished before sending the next job. This can be checked using the Fee_Cbk callback functions: Fls_JobEndNotification and Fls_JobErrorNotification.

- **Test Verification**
    - To verify that the flash memory is correctly programmed, BlankCheck API should be used after performing Erase operation and Compare API should be used after a Write operation to ensure data integrity.

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