MCAL MCU Module Software Design Document

Document Version : 37
Document Owner : Texas Instruments
Document Status : Published
Last Approval Date : Sep 26, 2022

1 Revision History

Version	Date	Author	Document Status	Comments
0.1	11 Apr 2022	Rohit Tiwari	DONE	First Version
0.2	25 May 2022	Rohit Tiwari	DONE	Common review Comments fixed
v.32	27 Jun 2022	Rohit Tiwari	DONE	Comola workflow added
v.33	28 Jun 2022	Rohit Tiwari	DONE	Review Comments Fixed
v.34	29 Jun 2022	Rohit Tiwari	DONE	Updated Revision history
v.35	30 Jun 2022	Rohit Tiwari	DONE	Added details on Callback function
v.36	04 Jul 2022	Rohit Tiwari	DONE	Added comments for dependency section

Version	Date	Author	Document Status	Comments
v.37	15 Sep 2022	Rohit Tiwari	DONE	Fixed wrong mappings of Design

2 Terms and Abbreviations

Abbreviation /Term	Meaning / Explanation
AUTOSAR	AUTomotive Open System ARchitecture
RTE	Runtime Environment
BSW	Basic Software
API	Application Programming Interface
ECU	Electronic Control Unit
SoC	System on Chip
DAR	Decision Analysis and Resolution

Abbreviation /Term	Meaning / Explanation
MCU	Micro Controller Unit. Here, it refers to the MCU Driver module in AUTOSAR MCAL.
CAN	Controller Area Network
ETH	Ethernet
ICU	Input Capture Unit
µC	MicroController
ISR	Interrupt Service Routine
MCAL	Microcontroller Abstraction Layer
SFR	Special Function Register (MCU register)
DEM	Diagnostic Event Manager

Abbreviation /Term	Meaning / Explanation
DET	Default Error Tracer
DMSC	Device Management Security Controller
Scilient	System Controller Interface Client
CSL	Chip Support Library
SBL	Secondary Bootloader
PLL	Phase-Locked Loop
GEL	General Extension Language file. It is loaded into the IDE. Allows direct intervention with the target device such as access to memory control, set/reset breakpoints, etc.
PMIC	Power Management Integrated Chip

Abbreviation /Term	Meaning / Explanation
MCAN	Modular Controller Area Network (hardware mapping of CAN)
ECAP	Enhanced Capture (hardware mapping of ICU)
CPSW	Common Platform Ethernet Switch (hardware mapping of Eth)
POR	Power-on-Reset
DMA	Direct Memory Access

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 Driver(CDD).

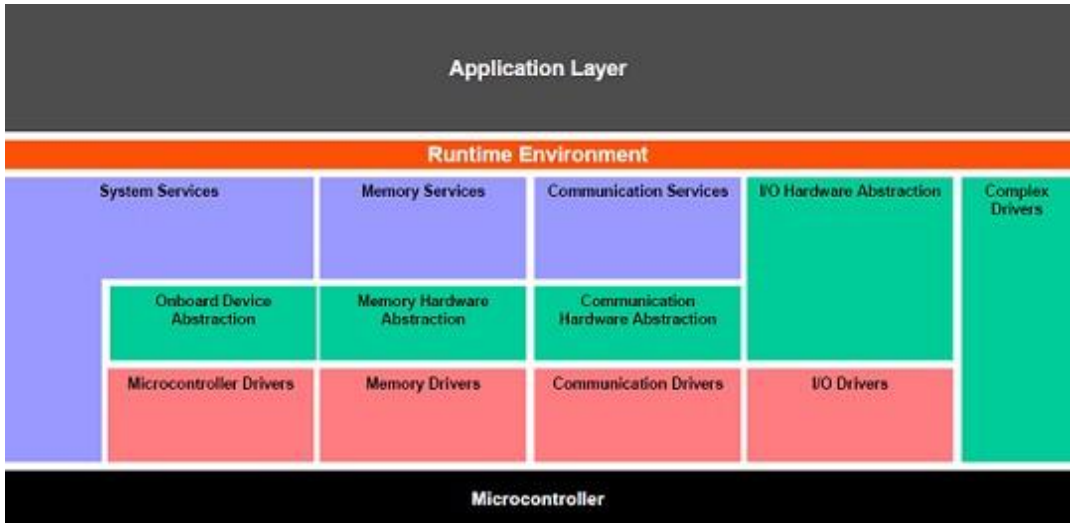


Fig: AUTOSAR Layered Architecture

3.2 AUTOSAR Architecture

MCAL is the lowest abstraction layer of the Basic Software. It contains software modules that interact with the Microcontroller and its internal peripherals directly. Mcu driver is part of the Microcontroller Drivers (block, shown above). Below figure shows the position of the Mcu driver in the AUTOSAR Architecture.

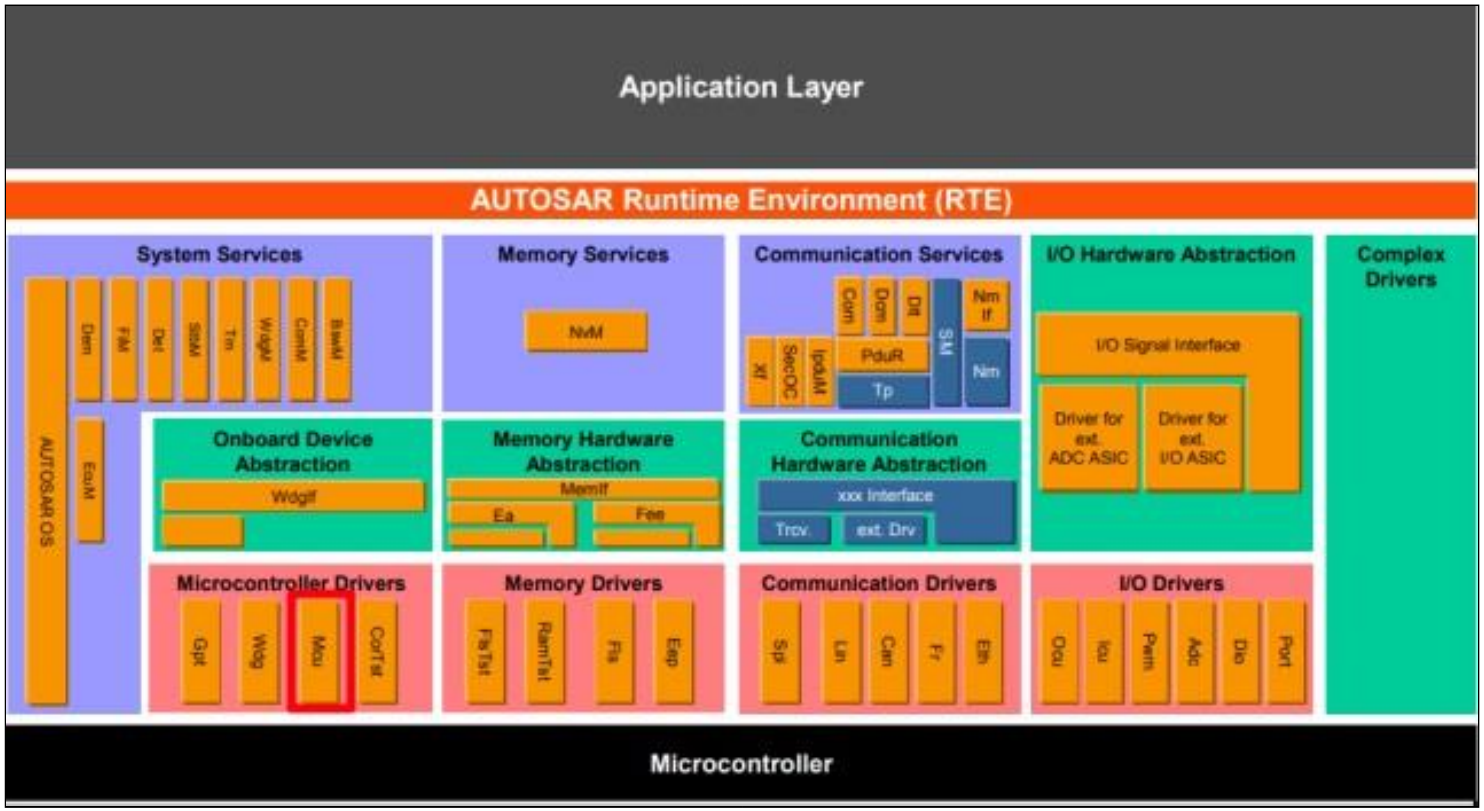

Application Layer
AUTOSAR Runtime Environment (RTE)
System Services
AUTOSAR OS
Dem
FiM
Det
StbM
Tm
WdgM
CtrlM
BswM
EcuM
Memory Services
NvM
Communication Services
Com
Dcm
Dlt
Nm If
Xf
SecOC
IpduM
PduR
Tp
SM
Nm
I/O Hardware Abstraction
I/O Signal Interface
Driver for ext. ADC ASIC
Driver for ext. I/O ASIC
Complex Drivers
Onboard Device Abstraction
WdgIf
Memory Hardware Abstraction
MemIf
Ea
Fee
Communication Hardware Abstraction
xxx Interface
Trcv
ext. Drv
Microcontroller Drivers
Gpt
Wdg
Mcu
CorTst
Memory Drivers
FlsTst
RamTst
Fls
Eep
Communication Drivers
Spi
Lin
Can
Fr
Eth
I/O Drivers
Ocu
Icu
Pwm
Adc
Dio
Port
Microcontroller

Fig: MCU Driver in AUTOSAR Layered Architecture

3.3 MCU MCAL

Mcu module provides following services.

- Initialization of MCU clock, PLL, clock prescalers and MCU clock distribution.
- Initialization of RAM sections.
- Activation of µC reduced power modes.
- Activation of a µC reset.
- Provides a service to get the reset reason from hardware.

3.4 Purpose and Scope

The Detailed Design document provides the design details of MCU 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 MCU module.

3.4.1 Module Overview

The MCU driver provides MCU services for Clock and RAM initialization. In the MCU configuration set, the MCU specific settings for the Clock (i.e. PLL setting) and RAM (i.e. section base address and size) shall be configured.

3.4.1.1 DMSC:

- DMSC hosts the DMSC firmware.
- Any request for setting up clock frequencies for a peripheral should come to DMSC via scilient.
- DMSC firmware should be loaded by the SBL.

3.4.1.2 Configure Clocks:

- The Clocks and PLLs are expected to be controlled and programmed by DMSC

3.4.1.3 Mapping of Clocks to CPU/Compute cores:

- This is to be done at start up by SBL.
- To be configured by DMSC Firmware. The request is raised by SBL via SciClient. (Fig below shows the flow)
- Not expected to change until next power cycle.

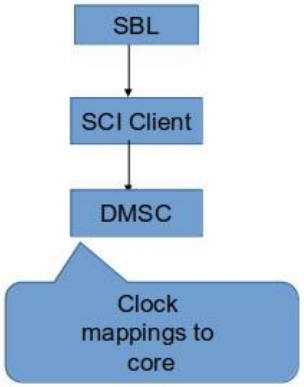
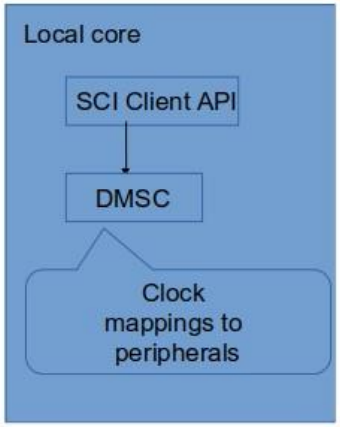


Fig: Clock Mapping to CPU/Compute cores

3.4.1.4 Mapping of Clocks to peripherals:

- Configured by DMSC Firmware.
- Request raised by a SCI Client API on local core. (Fig below shows the flow)
- Typically done at start up.
- Not expected to change until next power cycle.



Clock Mapping to peripherals

3.4.1.5 SCI Client:

- Sciclient APIs are used by application to request DMSC for clock request.

NOTE: Configure peripheral clock in Secondary Boot Loader or AUTOSAR start up code.

3.4.1.6 Reset of Device (SoC):

- Reset is performed by using the setting the appropriate bits of the hardware register for the domain. Also, Sci Client API's could be used to perform warm reset .

- Power-on reset can be done in collaboration with external PMIC . POR is performed using the software equivalent of the pin for the domain.

3.4.1.7 Reset of RAM:

- Is a shared resource for all compute cores.
- Isolation / Freedom From Interference is achieved by •Enabling firewall to allow access to carved out section.
- Carving out dedicated area to each compute core.
- Owned sections could be reset by CPU / DMA

3.5 Requirements

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

3.6 Features Supported

Below listed are some of the key features that are expected to be supported -

- The MCU module shall provide a service for software triggering of a reset.
- The MCU module shall provide services to get the reset reason of the last reset.
- Mcu shall provide a service to enable and set the MCU clock. (Peripherals - MCAN, TIMER, SPI, ADC, ECAP, CPSW)

3.7 Features Not Supported / NON Compliance

- Enabling/disabling of the ECU or uC power supply is not the task of the MCU driver. This is to be handled by the upper layer.

- Use of external PMIC for POWER_ON_RESET is not supported.
- **[NON Compliance]** The MCU module shall provide a service to activate MCU reduced power modes.
- **[NON Compliance]** McuRamSectionWriteSize is not supported since we are not supporting Mcu_InitRamSection().

3.8 Assumptions

- Before the MCU driver can be initialized, a basic initialization of the MCU has to be executed.
- Customer Applications would require to use SBL/Gels/Autosar start-up sequence for bootup/basic setup.

3.9 Constraints

Some of the critical constraints of this design are listed below -

- Number of RAM sections, always set to 0 (i.e. McuRamSectors is always 1).
- Only 1 mode is supported (i.e. McuNumberOfMcuModes is always 1).
- Start-Up Code is not in the scope of the AUTOSAR MCU Driver Module.
- The start-up code of the MCU shall be executed after power up and any kind of microcontroller reset.

3.10 Hardware and SW platforms

Hardware Platforms

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

Software Platforms

- Bare-Metal.

3.11 Dependencies

3.11.1 Start-up code

Before the MCU driver can be initialized, a basic initialization of the MCU has to be executed. This shall be done by the start-up code after every power up and uC reset.

Apart from the basic functionalities (i.e. initialization of base addresses, interrupt stack pointer, user stack pointer, memory and MCU clock system), the start up code shall shall cover MCU specific functionalities like -

- Internal watchdog shall not be serviced until the watchdog is initialized from the MCAL watchdog driver.
- Enable protection mechanisms for SFR's.
- Initialize all necessary registers that must be written once.

3.12 Stakeholders

- Developers
- Test Engineers
- Customer Integrator

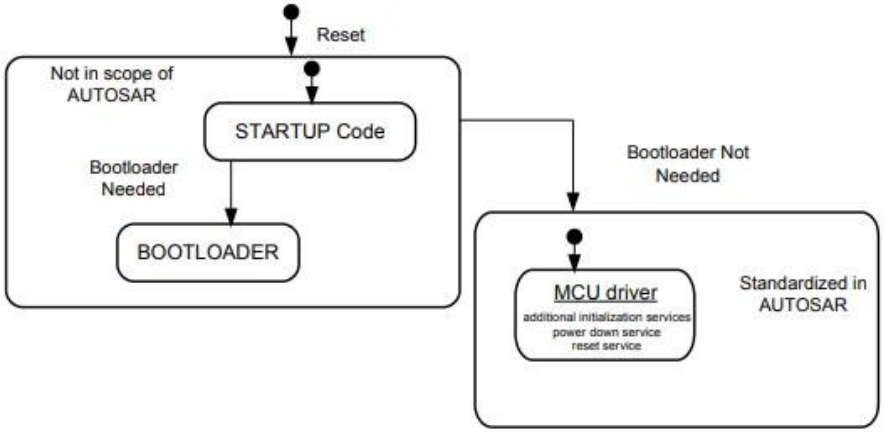
3.13 References

	Specification	Comment / Link
1	AUTOSAR 4.3.1	AUTOSAR Specification for MCU 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.

The MCU driver provides services for basic uC initialization, power down functionality, reset and uC specific functions required by other MCAL software modules. The initialization services allow a flexible and application related MCU initialization in addition to the start-up code (see figure below).

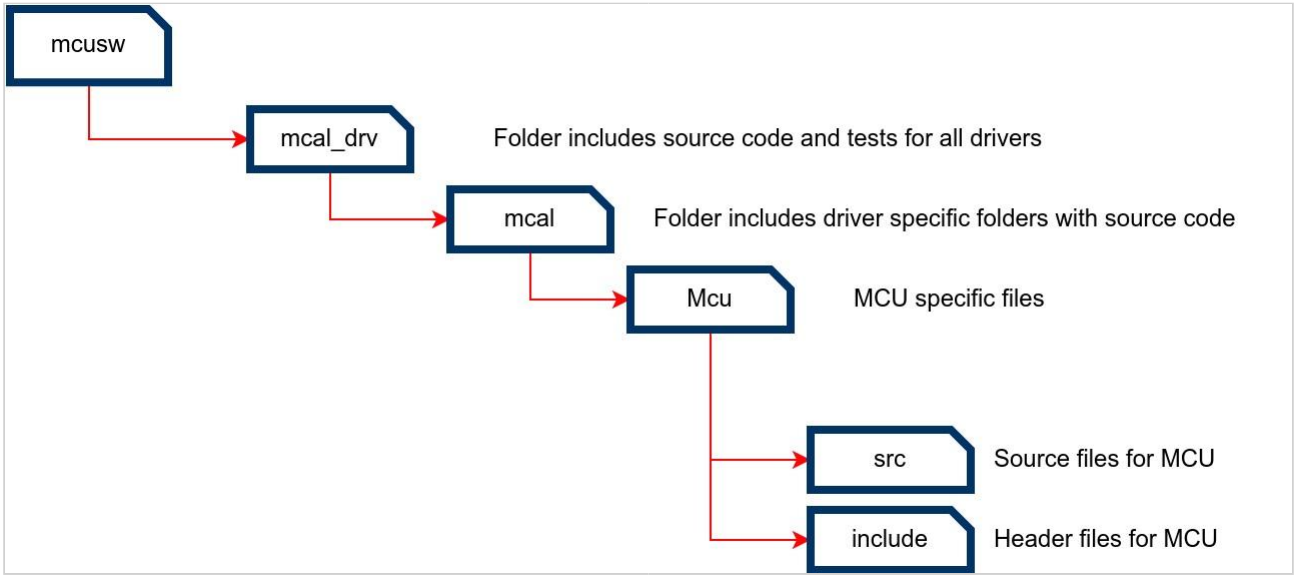


Note: Configure peripheral clock in Secondary Boot Loader or AUTOSAR start up code.

4.2 Directory Structure

The directory structure is as depicted in figures below, the source files can be categorized under “Driver Implementation” and “Example Application” **Driver Implemented by:**

- Mcu.h, Mcu_Priv.h : Shall implement the interface provided by the driver.
- Mcu.c ,Mcu_Priv.c : Shall implement the driver functionality.

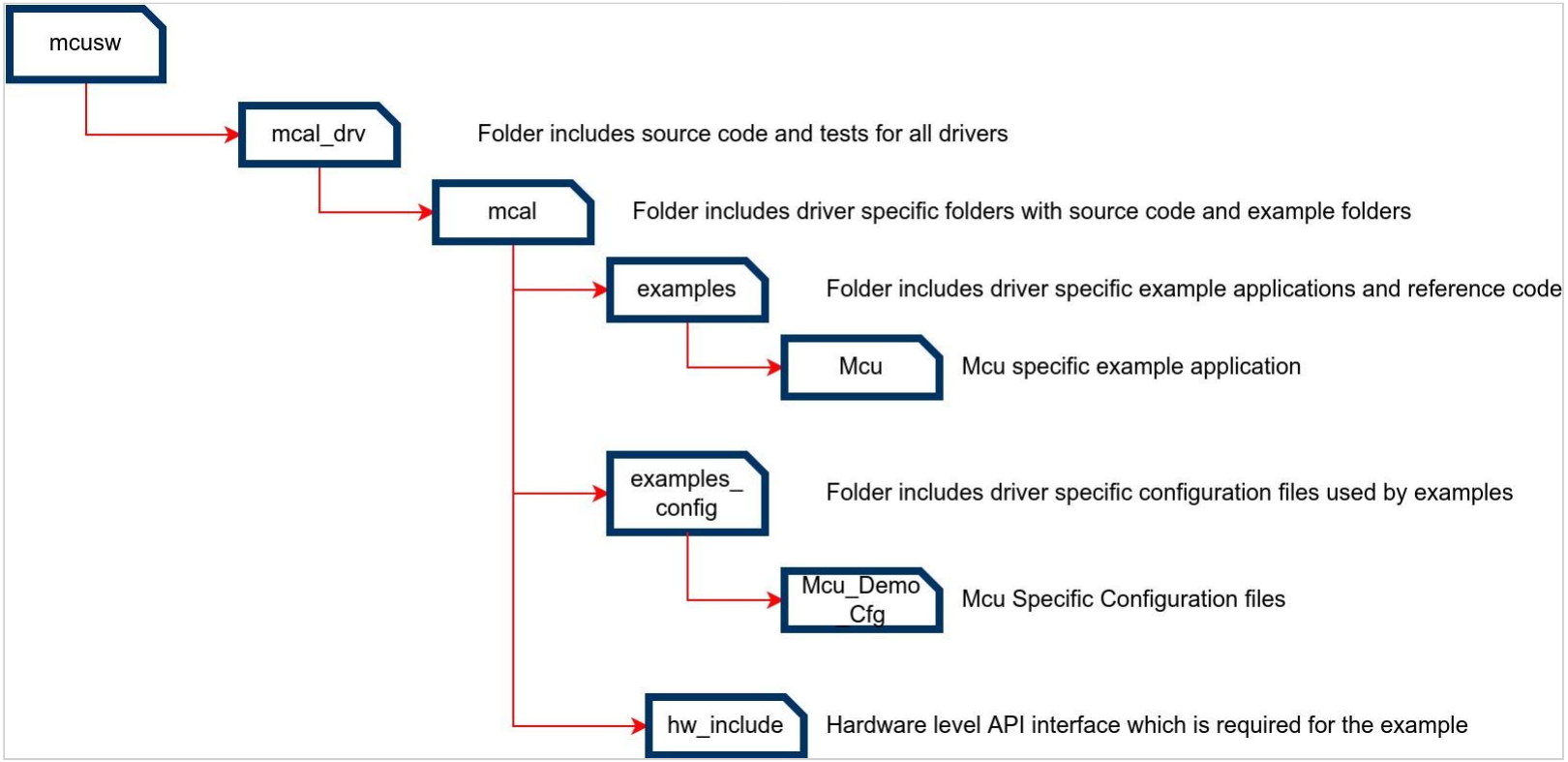


Example Application:

- Mcu_Cfg.h and Mcu_Cfg.c : Shall implement the generated configuration for pre-compile variant.

- Mcu_PBcfg.c : Shall implement the generated configuration for post-build variant.

- McuApp.c : Shall implement the example application that demonstrates the use of the driver.
- hw_include : Shall be used by example application.


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
Mcu
Mcu specific example application
examples_ config
Folder includes driver specific configuration files used by examples
Mcu_Demo Cfg
Mcu Specific Configuration files
hw_include
Hardware level API interface which is required for the example

4.3 Configurator

The AUTOSAR MCU Driver Specification details mandatory parameters that shall be configurable via the configurator. Please refer section 10 of Reference 1 - AUTOSAR 4.3.1.

Design ID	Description
MCAL-8478 - ECUC_Mcu_00118 : McuGeneralConfiguration PUBLISHED - ECUC_Mcu_00118 : Mcu General Configuration Container	ECUC_Mcu_00118 : Mcu General Configuration Container
MCAL-8411 - ECUC_Mcu_00124 : McuClockSettingConfig PUBLISHED - ECUC_Mcu_00124 : McuClockSettingConfig	ECUC_Mcu_00124 : McuClockSettingConfig
MCAL-8463 - ECUC_Mcu_00119 : McuModuleConfiguration PUBLISHED - ECUC_Mcu_00119 : McuModuleConfiguration	ECUC_Mcu_00119 : McuModuleConfiguration
MCAL-8427 - ECUC_Mcu_00123 : McuModeSettingConf PUBLISHED - ECUC_Mcu_00123 : McuModeSettingConf	ECUC_Mcu_00123 : McuModeSettingConf
MCAL-8394 - ECUC_Mcu_00120 PUBLISHED - ECUC_Mcu_00120 : McuRamSectorSettingConf	ECUC_Mcu_00120 : McuRamSectorSettingConf

Design ID	Description
MCAL-8399 - ECUC_Mcu_00185 : McuResetReasonConf PUBLISHED - ECUC_Mcu_00185 : McuResetReasonConf	ECUC_Mcu_00185 : McuResetReasonConf

4.3.1 NON Standard configurable parameters

Following lists this design's specific configurable parameters

Parameter	Usage Comment
McuDeviceVariant	This shall allow the user to select the device variant for which integration is being performed. This parameter shall be used by driver to impose device specific constraints.
Mcu_CBK_Function	This parameter is a callback function. This Callback function will be defined in the application to perform the functionality. The User has to define this function.
McuResetMode	This shall allow the user to select between or WARM/ COLD reset.

Parameter	Usage Comment
Domain	This shall allow the user to select the Domain of a peripheral. The domain selection is between MCU, MAIN and WKUP.
McuClockModuleId	This shall allow the user to select the module id for peripheral.
McuClockSourceId	This shall allow the user to select the clock source id for the selected peripheral.
McuParentSourceId	This shall allow the user to select the parent source id for the selected peripheral.
McuClockInitializationConfigFlag	This shall allow the user to switch the configuration flag ON/OFF.

4.3.2 Implementation specific parameters (computed)

The configurator shall be used to configure the Module Id, Clock Source Id, Parent Clock Source Id for the modules. Refer section (API's)

4.3.3 Variant Support

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

Design ID	Description
MCAL-8513 - ECUC_Mcu_00189 : Configuration of the Mcu (Microcontroller Unit) module PUBLISHED - ECUC_Mcu_00189 : Configuration of the Mcu (Microcontroller Unit) module	ECUC_Mcu_00189 : Configuration of the Mcu (Microcontroller Unit) module

4.4 Error Classification

Errors are classified in two categories, development error and runtime / production error.

4.4.1 Development Errors

	Relevance	Related error code	Value [hex]

Type or error

API service called with wrong parameter	Development	MCU_E_PARAM_CONFIG MCU_E_PARAM_CLOCK MCU_E_PARAM_MODE MCU_E_PARAM_RAMSECTION MCU_E_PLL_NOT_LOCKED MCU_E_UNINIT MCU_E_PARAM_POINTER MCU_E_INIT_FAILED	0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x10 0x11
Clock source failure	Extended Production Errors (for Release 4.1.1)	MCU_E_CLOCK_FAILURE	Assigned by DEM

4.4.1.1 Error Detection

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

Design ID	Description
MCAL-8450 - ECUC_Mcu_00166 PUBLISHED - ECUC_Mcu_00166 : McuDevErrorDetection	ECUC_Mcu_00166 : McuDevErrorDetection
MCAL-8507 - SWS_Mcu_00012 : Development Errors PUBLISHED - SWS_Mcu_00012 : Development Errors	SWS_Mcu_00012 : Development Errors

4.4.1.2 Error notification (DET)

The MCU driver follows the standardized AUTOSAR concept to report production errors. The provided callback routines are specified in the Diagnostic Event Manager (DEM) specification

Note: Production Errors shall not be used as the return value of the called function.

4.4.2 Runtime Errors

There are no runtime errors.

4.4.2.1 Error notification (DEM)

All detected run time errors shall be reported to Dem_ReportErrorStatus () service of the Diagnostic Event Manager (DEM).

4.4.3 Transient Faults

There are no transient faults.

5 Implementation Details

5.1 Data structures and resources

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

5.1.1 Mcu_ConfigType

A pointer to such a structure is provided to the MCU initialization routines for configuration, please refer section 8.2.1 of the MCU Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

5.1.2 Mcu_PllStatusType

This is a status value returned by the function Mcu_GetPllStatus of the MCU module, refer section 8.2.2 of the MCU Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

5.1.3 Mcu_ClockType

Specifies the identification (ID) for a clock setting, which is configured in the configuration structure, refer section 8.2.3 of the MCU Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

5.1.4 **Mcu_ResetType**

This is the type of the reset enumerator containing the subset of reset types. It is not required that all reset types are supported by hardware. Refer section 8.2.4 of the MCU Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

5.1.5 **Mcu_RawResetType**

This type specifies the reset reason in raw register format read from a reset status register. Refer section 8.2.5 of the MCU Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

5.1.6 **Mcu_ModeType**

This type specifies the identification (ID) for a MCU mode, which is configured in the configuration structure. Refer section 8.2.6 of the MCU Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

5.1.7 **Mcu_RamSectionType**

This type specifies the identification (ID) for a RAM section, which is configured in the configuration structure. Refer section 8.2.7 of the MCU Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

5.1.8 **Mcu_RamStateType**

This is the Ram State data type returned by the function Mcu_GetRamState of the Mcu module. It is not required that all RAM state types are supported by the hardware. Refer section 8.2.8 of the MCU Driver AUTOSAR Specification as listed in Reference 1 - AUTOSAR 4.3.1.

5.2 Dynamic Behavior - Control Flow Diagram

Not Applicable

5.3 Dynamic Behavior - Data Flow Diagram

Not Applicable

5.4 Application Parameters

Mcu_Init

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

Mcu_InitRamSection

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
RamSection	Selects RAM memory section provided in configuration set	0	-	-	N.A

Mcu_InitClock

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
ClockSetting	Clock setting	0-255	uint8	0	N.A

Mcu_DistributePllClock

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
N.A	N.A	-	-	-	N.A

Mcu_GetPllStatus

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
N.A	N.A	-	-	-	N.A

Mcu_GetResetReason

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
N.A	N.A	0-5	-	0	N.A

Mcu_GetResetRawValue

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
N.A	N.A	-	-	-	N.A

Mcu_PerformReset

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant

N.A	N.A	-	-	-	N.A

Mcu_SetMode

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
McuMode	Mcu_ModeType	0-0	-	0	N.A

Mcu_GetVersionInfo

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
versioninfo	Std_VersionInfoType*	0xFFFFFFFF	-	-	N.A

Mcu_GetRamState

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
N.A	N.A	-	-	-	N.A

6 Low Level Definitions

6.1 Driver API's

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

6.1.1 Mcu_Init

Refer section 8.3.1 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8454 - SWS_Mcu_00026 : McuInit PUBLISHED - Mcu Driver : SWS_Mcu_00026 : McuInit	Mcu Driver : SWS_Mcu_00026 : McuInit
MCAL-8484 - SWS_Mcu_00116 : McuInit : Initializing register PUBLISHED - Mcu Driver : SWS_Mcu_00116 : McuInit : Initializing register	Mcu Driver : SWS_Mcu_00116 : McuInit : Initializing register
MCAL-8426 - SWS_Mcu_00244 : Init controller registers PUBLISHED - Mcu Driver : SWS_Mcu_00244 : Init controller registers	Mcu Driver : SWS_Mcu_00244 : Init controller registers
MCAL-8469 - SWS_Mcu_00245 : Init controller registers PUBLISHED - Mcu Driver : SWS_Mcu_00245 : Init controller registers	Mcu Driver : SWS_Mcu_00245 : Init controller registers
MCAL-8467 - SWS_Mcu_00246 : Init controller registers PUBLISHED - Mcu Driver : SWS_Mcu_00246 : Init controller registers	Mcu Driver : SWS_Mcu_00246 : Init controller registers
MCAL-8389 - SWS_Mcu_00247 : Init controller registers PUBLISHED - Mcu Driver : SWS_Mcu_00247 : Init controller registers	Mcu Driver : SWS_Mcu_00247 : Init controller registers

6.1.2 **Mcu_InitRamSection**

Implementation - Dummy API. No actions performed, always returns E_OK.

Refer section 8.3.2 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8497 - SWS_Mcu_00011 : InitRamSection PUBLISHED - Mcu Driver : SWS_Mcu_00011 : InitRamSection	Mcu Driver : SWS_Mcu_00011 : InitRamSection
MCAL-8430 - SWS_Mcu_00136 : InitRamSection : Usability PUBLISHED - Mcu Driver : SWS_Mcu_00136 : InitRamSection : Usability	Mcu Driver : SWS_Mcu_00136 : InitRamSection : Usability

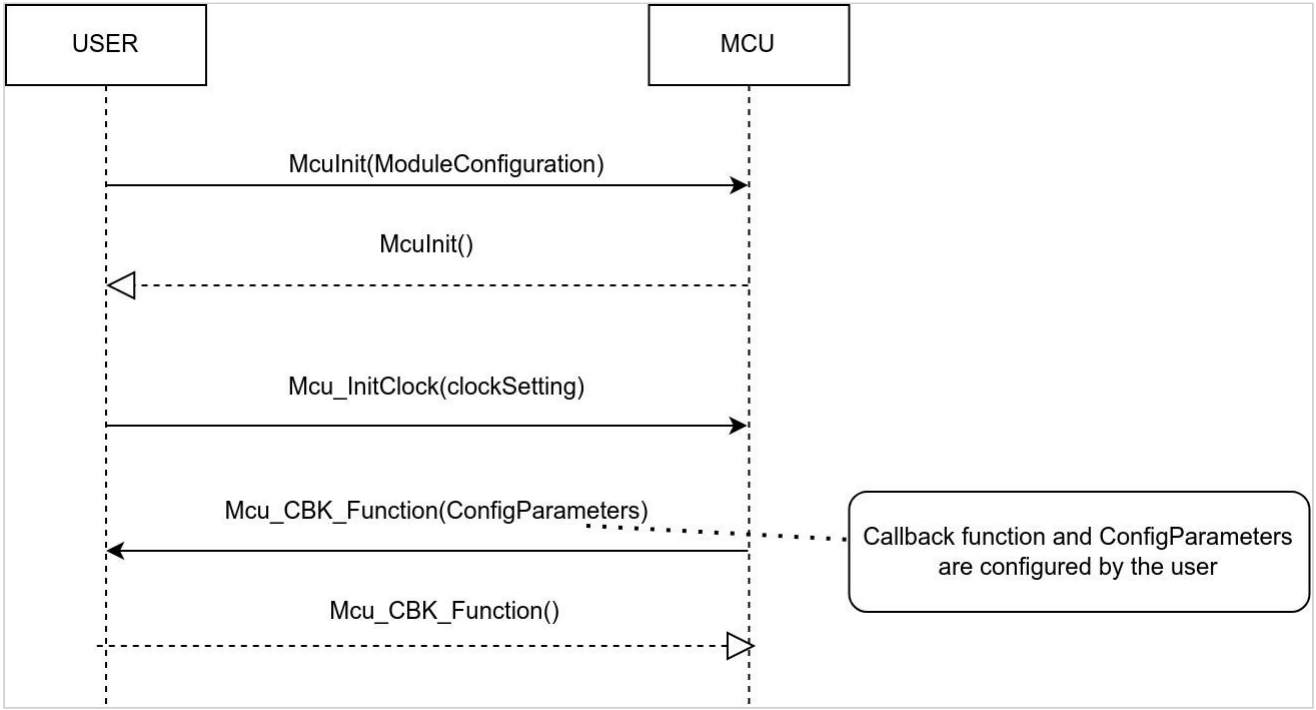
6.1.3 **Mcu_InitClock**

Refer section 8.3.3 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8442 - SWS_Mcu_00137 : InitClock : Functionality PUBLISHED - Mcu Driver : SWS_Mcu_00137 : InitClock : Functionality	Mcu Driver : SWS_Mcu_00137 : InitClock : Functionality
MCAL-8455 - SWS_Mcu_00138 : InitClock : Functionality PUBLISHED - Mcu Driver : SWS_Mcu_00138 : InitClock : Functionality	Mcu Driver : SWS_Mcu_00138 : InitClock : Functionality
MCAL-8416 - SWS_Mcu_00139 : InitClock : Usability PUBLISHED - Mcu Driver : SWS_Mcu_00139 : InitClock : Usability	Mcu Driver : SWS_Mcu_00139 : InitClock : Usability
MCAL-8420 - SWS_Mcu_00210 : InitClock PUBLISHED - Mcu Driver : SWS_Mcu_00210 : InitClock	Mcu Driver : SWS_Mcu_00210 : InitClock

6.1.3.1 Callback function in Mcu_InitClock

- Mcu_InitClock() initialises the clocks for the module configuration that comes from the configuration file. This is done by using the callback function.
- The callback function Mcu_CBK_Function() is provided and defined by the user.


USER
MCU
McuInit(ModuleConfiguration)
McuInit()
Mcu_InitClock(clockSetting)
Mcu_CBK_Function(ConfigParameters)
Callback function and ConfigParameters are configured by the user
Mcu_CBK_Function()

6.1.4 Mcu_DistributePllClock

Implementation - Dummy API. No actions performed.

Refer section 8.3.4 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8508 - SWS_Mcu_00140 : DistributePllClock : Functionality PUBLISHED - Mcu Driver : SWS_Mcu_00140 : DistributePllClock : Functionality	Mcu Driver : SWS_Mcu_00140 : DistributePllClock : Functionality
MCAL-8470 - SWS_Mcu_00141 : DistributePllClock : Functionality PUBLISHED - Mcu Driver : SWS_Mcu_00141 : DistributePllClock : Functionality	Mcu Driver : SWS_Mcu_00141 : DistributePllClock : Functionality
MCAL-8445 - SWS_Mcu_00056 : DistributePllClock : No effect PUBLISHED - Mcu Driver : SWS_Mcu_00056 : DistributePllClock : No effect	Mcu Driver : SWS_Mcu_00056 : DistributePllClock : No effect
MCAL-8475 - SWS_Mcu_00142 : DistributePllClock : Usability PUBLISHED - Mcu Driver : SWS_Mcu_00142 : DistributePllClock : Usability	Mcu Driver : SWS_Mcu_00142 : DistributePllClock : Usability

Design Identifier	Description
MCAL-8405 - SWS_Mcu_00205 : DistributePllClock PUBLISHED - Mcu Driver : SWS_Mcu_00205 : DistributePllClock	Mcu Driver : SWS_Mcu_00205 : DistributePllClock

6.1.5 Mcu_GetPllStatus

Implementation - Dummy API. No actions performed.

Refer section 8.3.5 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8402 - SWS_Mcu_00008 : GetPllStatus PUBLISHED - Mcu Driver : SWS_Mcu_00008 : GetPllStatus	Mcu Driver : SWS_Mcu_00008 : GetPllStatus
MCAL-8503 - SWS_Mcu_00132 : GetPllStatus : Return value PUBLISHED - Mcu Driver : SWS_Mcu_00132 : GetPllStatus : Return value	Mcu Driver : SWS_Mcu_00132 : GetPllStatus : Return value

Design Identifier	Description
MCAL-8466 - SWS_Mcu_00206 : GetPllStatus : MCU_PLL_STATUS_UNDEFINED PUBLISHED - Mcu Driver : SWS_Mcu_00206 : GetPllStatus : MCU_PLL_STATUS_UNDEFINED	Mcu Driver : SWS_Mcu_00206 : GetPllStatus : MCU_PLL_STATUS_UNDEFINED

6.1.6 Mcu_GetResetReason

Refer section 8.3.6 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8493 - SWS_Mcu_00005 : GetResetReason PUBLISHED - Mcu Driver : SWS_Mcu_00005 : GetResetReason	Mcu Driver : SWS_Mcu_00005 : GetResetReason
MCAL-8425 - SWS_Mcu_00133 : GetResetReason : Return value PUBLISHED - Mcu Driver : SWS_Mcu_00133 : GetResetReason : Return value	Mcu Driver : SWS_Mcu_00133 : GetResetReason : Return value

6.1.7 **Mcu_GetResetRawValue**

Refer section 8.3.7 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8393 - SWS_Mcu_00135 : GetResetRawValue : Functionality PUBLISHED - Mcu Driver : SWS_Mcu_00135 : GetResetRawValue : Functionality	Mcu Driver : SWS_Mcu_00135 : GetResetRawValue : Functionality
MCAL-8458 - SWS_Mcu_00006 : GetResetRawValue PUBLISHED - Mcu Driver : SWS_Mcu_00006 : GetResetRawValue	Mcu Driver : SWS_Mcu_00006 : GetResetRawValue

6.1.8 **Mcu_PerformReset**

Refer section 8.3.8 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8444 - SWS_Mcu_00143 : PerformReset : Functionality PUBLISHED - Mcu Driver : SWS_Mcu_00143 : PerformReset : Functionality	Mcu Driver : SWS_Mcu_00143 : PerformReset : Functionality

Design Identifier	Description
MCAL-8452 - SWS_Mcu_00144 : PerformReset : Config PUBLISHED - Mcu Driver : SWS_Mcu_00144 : PerformReset : Config	Mcu Driver : SWS_Mcu_00144 : PerformReset : Config
MCAL-8448 - SWS_Mcu_00145 : PerformReset : Usability PUBLISHED - Mcu Driver : SWS_Mcu_00145 : PerformReset : Usability	Mcu Driver : SWS_Mcu_00145 : PerformReset : Usability
MCAL-8423 - SWS_Mcu_00146 : PerformReset : Availability PUBLISHED - Mcu Driver : SWS_Mcu_00146 : PerformReset : Availability	Mcu Driver : SWS_Mcu_00146 : PerformReset : Availability

6.1.9 **Mcu_SetMode**

Implementation - only NOMINAL (default) mode is supported.

Refer section 8.3.9 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8462 - SWS_Mcu_00147 : SetMode PUBLISHED - Mcu Driver : SWS_Mcu_00147 : SetMode	Mcu Driver : SWS_Mcu_00147 : SetMode
MCAL-8396 - SWS_Mcu_00148 : SetMode PUBLISHED - Mcu Driver : SWS_Mcu_00148 : SetMode	Mcu Driver : SWS_Mcu_00148 : SetMode

6.1.10 **Mcu_GetVersionInfo**

Refer section 8.3.10 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8468 - ECUC_Mcu_00168 PUBLISHED - Mcu Driver : ECUC_Mcu_00168 : McuVersionInfoApi	Mcu Driver : ECUC_Mcu_00168 : McuVersionInfoApi Pre-processor switch to enable / disable the API

6.1.11 **Mcu_GetRamState**

Implementation - Dummy API. No actions performed, always returns E_OK.

Refer section 8.3.11 of Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-8395 - SWS_Mcu_00208 : Init PUBLISHED - Mcu Driver : SWS_Mcu_00208 : Init	Mcu Driver : SWS_Mcu_00208 : Init
MCAL-8447 - SWS_Mcu_00209 : GetRamState PUBLISHED - Mcu Driver : SWS_Mcu_00209 : GetRamState	Mcu Driver : SWS_Mcu_00209 : GetRamState

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
0.6	NA	NA	1	2.73	NA	NA

7.2 Critical timing and Performance

Not Applicable

8 Decision Analysis & Resolution (DAR)

Not Applicable

9 Test Criteria

Not Applicable

10 Template Revision History

Author Name	Description	Version	Date
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
Yaniv Machani	Initial version	0.1	03 Oct 2018
Yaniv Machani	Updated to include EP views	0.4	02 Nov 2018

Author Name	Description	Version	Date
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
Yaniv Weizman	Adding tables for Testing guidelines	0.7	18 Nov 2019