Document Title	Specification of MCU Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	31

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Cleaned up unresolved references in traceability. • Removed [SWS_Mcu_CONSTR_00001]
2023-11-23	R23-11	AUTOSAR Release Management	• Cleaned up unresolved references in traceability.
2022-11-24	R22-11	AUTOSAR Release Management	• Cleaned up unresolved references in traceability.
2021-11-25	R21-11	AUTOSAR Release Management	• Removed [SWS_Mcu_00131], [SWS_Mcu_00054], [SWS_Mcu_00035], [SWS_Mcu_00030] and [SWS_Mcu_00031] • Cleaned up unresolved references in traceability
2020-11-30	R20-11	AUTOSAR Release Management	• Enum and Error related modifications • Editorial Changes
2019-11-28	R19-11	AUTOSAR Release Management	• Removed DRAFT status of items introduced for Multicore support • Removed duplicated chapters McuGeneralConfiguration and McuClockSettingConfig • Changed Document Status from Final to published

▽

△

2018-10-31	4.4.0	AUTOSAR Release Management	• Debugging support was removed • Introduced support for Multicore distribution
2017-12-08	4.3.1	AUTOSAR Release Management	• Introduced new configuration parameter - McuRamSectionWriteSize • Changed reentrancy of API `Mcu_SetMode` to Reentrant
2016-11-30	4.3.0	AUTOSAR Release Management	• Removed chapter "Variants" • Cleaned up unresolved references in traceability
2015-07-31	4.2.2	AUTOSAR Release Management	• Minor change regarding DET renaming and extension Incorporation • Clarifications regarding configuration class of symbolicNameValue parameters
2014-10-31	4.2.1	AUTOSAR Release Management	• Removed requirements for NULL pointer checking as redundant with BSW General. • Specified pass/fail criteria for extended production errors
2014-03-31	4.1.3	AUTOSAR Release Management	• Requirement Traceability Table revised • Correction of requirement tag ([SWS_Mcu_00146])
2013-10-31	4.1.2	AUTOSAR Release Management	• `Mcu_GetResetReason` and `Mcu_GetResetRawValue` return the same value if called multiple times • RAM sector multiplicity corrected • McuClockSettingId and McuMode range corrected • Editorial changes • Removed chapter(s) on change documentation

▽

△

2013-03-15	4.1.1	AUTOSAR Administration	• Adaptation of the Document due to the SWS General Release • Scope Fields in all configuration parameters (chapter 10) changed as Local -> impact only this module or ECU impact several modules • Autosar Memory mapping abstraction split for each BSW • Split Production Errors in "Pure" Production Errors and Extended Production Errors • Changed signature of Api `Mcu_DistributePllClock`
2011-12-22	4.0.3	AUTOSAR Administration	• `Mcu_SetMode` assumes that all interrupts are disabled prior the call
2010-09-30	3.1.5	AUTOSAR Administration	• Corrected [SWS_Mcu_00210] • Removed [SWS_Mcu_00225]. • Rephrased [SWS_Mcu_00125] and [SWS_Mcu_00011] • Added Chapter 12
2010-02-02	3.1.4	AUTOSAR Administration	• Lots requirements rephrased to make them atomic. • Debugging Concept inserted. • Insertion of a new service (Api) to read the Status after the reset. (Affected also SRS R4.0) • Insertion new configuration parameters to enable/disable PLL Apis. • Introduction of a new container to publish all the different resets that Micro Controller support. • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Table formatting corrected

▽

△

2007-01-24	2.1.15	AUTOSAR Administration	• Wakeup concept clarified (resulted in removal of wakeup functionality and sequence diagrams in the MCU SWS). As per the concept agreed within the Startup / Wakeup Taskforce. • Obsolete function `Dem_ReportErrorEvent()` removed. • Technical Office Improvements: wording improvements. • Re-wording of requirements for clarification • Document meta information extended • Small layout adaptations made
2007-11-28	2.1.14	AUTOSAR Administration	• Update to section 5.2.2: Inclusion of new file structure • Sections 8.3.2, 8.3.3, 8.3.9 : Removal of 'const' from API type definition. • Section 8.2.4, 8.2.5,10.2.5: Description detail amended • Section 8.2.4: Default value (0x0) for `MCU_POWER_ON_RESET` removed. • Section 8.3.8 : Description updated to include reference to new pre-processor switch McuPerformResetApi. • Section 10.2.2: Introduction of pre-processor switch McuPerformResetApi • Section 10.2.3: Multiplicity of sub-container Mcu Clock Setting Configuration changed to 1. • Legal disclaimer revised • Release Notes added • "Advice for users" revised • "Revision Information" added

▽

△

2006-05-16	2.0	AUTOSAR Administration	• Document structure adapted to common Release 2.0 SWS Template. • Major changes in chapter 10 • Structure of document changed partly • Other changes see chapter 11
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module MCU [**M**icro**c**ontroller **U**nit]. The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller 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). The start-up code is very MCU specific. The provided start-up code description in this document is for guidance and implies functionality which has to be taken into account before standardized MCU initialization is able to start.

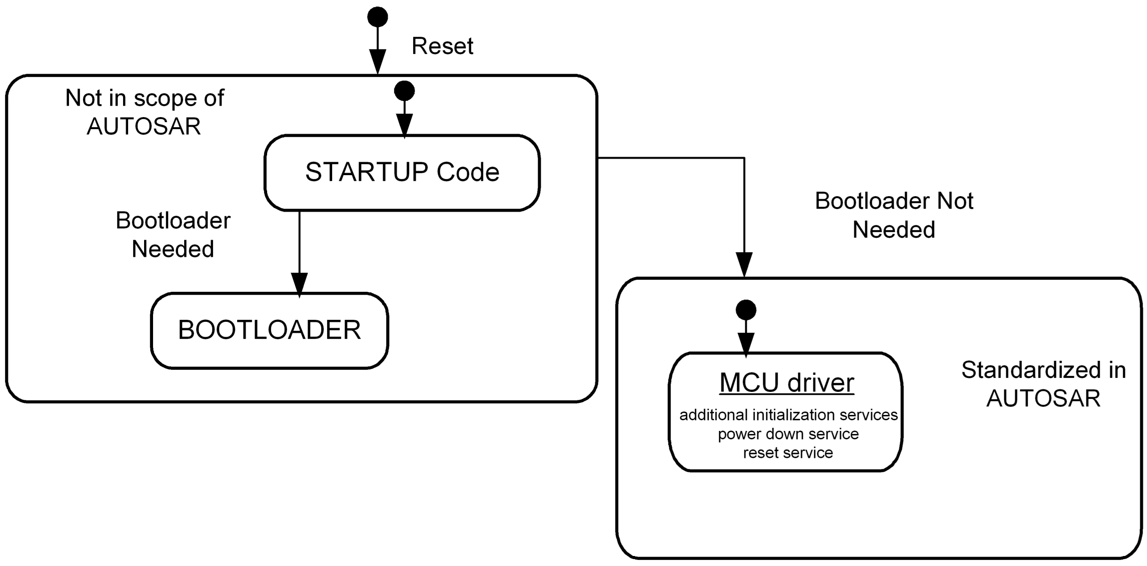

Figure 1.1: Scope of the MCU Driver Specification

The MCU driver accesses the microcontroller hardware directly and is located in the Microcontroller Abstraction Layer (MCAL).

MCU driver Features:

- 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

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the MCU Driver that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	**Description:**
µC	Microcontroller
MCU	Micro Controller Unit
SFR	Special Function Register (MCU register)
DEM	Diagnostic Event Manager
DET	Default Error Tracer

Table 2.1: Acronyms and abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Requirements on MCU Driver
AUTOSAR_CP_RS_MCUDriver

[4] Specification of Diagnostic Event Manager
AUTOSAR_CP_SWS_DiagnosticEventManager

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for MCU Driver.

Thus, the specification SWS BSW General shall be considered as additional and required specification for MCU Driver.

4 Constraints and assumptions

4.1 Limitations

In general the activation and configuration of MCU reduced power mode is not mandatory within AUTOSAR standardization.

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

4.2 Applicability to car domains

No restrictions

5 Dependencies to other modules

5.1 Start-up code

Before the MCU driver can be initialized, a basic initialization of the MCU has to be executed. This MCU specific initialization is typically executed in a start-up code.

The start-up code of the MCU shall be executed after power up and any kind of microcontroller reset. It shall perform very basic and microcontroller specific start-up initialization and shall be kept short because the MCU clock and PLL are not yet initialized. The start-up code shall cover MCU specific initialization which is not part of other MCU services or other MCAL drivers. The following description summarizes the basic functionality to be included in the start-up code. It is listed for guidance because some functionality might not be supported in all MCU's.

The start-up code shall initialize the base addresses for interrupt and trap vector tables. These base addresses are provided as configuration parameters or linker/locator setting.

The start-up code shall initialize the interrupt stack pointer if an interrupt stack is supported by the MCU. The interrupt stack pointer base address and the stack size are provided as configuration parameter or linker/locator setting.

The start-up code shall initialize the user stack pointer. The user stack pointer base address and the stack size are provided as configuration parameter or linker/locator setting.

If the MCU supports context save operation, the start-up code shall initialize the memory which is used for context save operation. The maximum amount of consecutive context save operations is provided as configuration parameter or linker/locator setting.

The start-up code shall ensure that the MCU internal watchdog shall not be serviced until the watchdog is initialized from the MCAL watchdog driver. This can be done for example by increasing the watchdog service time.

If the MCU supports cache memory for data and/or code, it shall be initialized and enabled in the start-up code.

The start-up code shall initialize MCU specific features with respect to internal memory as, for example, memory protection.

If external memory is used, the memory shall be initialized in the start-up code. The start-up code shall be prepared to support different memory configurations depending on code location. Different configuration options shall be taken into account for code execution from external/internal memory.

The settings of the different memories shall be provided to the start-up code as configuration parameters.

In the start-up code a default initialization of the MCU clock system shall be performed including global clock prescalers.

The start-up code shall enable protection mechanisms for special function registers (SFR's) if supported by the MCU.

The start-up code shall initialize all necessary write once registers or registers common to several drivers where one write, rather than repeated writes, to the register is required or highly desirable.

The start-up code shall initialize a minimum amount of RAM in order to allow proper execution of the MCU driver services and the caller of these services.

Note: The start-up code is ECU and MCU dependant. Details of the specification shall be described in the design specification of the MCU.

5.2 File structure

5.2.1 Code file structure

Note: The code file structure shall not be defined within this specification.

6 Requirements Tracing

The following tables reference the requirements specified in [2], [3] and links to the fulfillment of these. Please note that if column "Satisfied by" is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Mcu_00026]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_Mcu_00207]
[SRS_BSW_00327]	Error values naming convention	[SWS_Mcu_00012]
[SRS_BSW_00337]	Classification of development errors	[SWS_Mcu_00012]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_Mcu_00026]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 General Behavior

7.1.1 Background and Rationale

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.

7.1.2 Requirements

7.1.2.1 Reset

[SWS_Mcu_00055] ⌈The MCU module shall provide a service to provide software triggering of a hardware reset.⌋

Note: Only an authorized user shall be able to call this reset service function.

[SWS_Mcu_00052] ⌈The MCU module shall provide services to get the reset reason of the last reset if the hardware supports such a feature.⌋

Note: In an ECU, there are several sources which can cause a reset. Depending on the reset reason, several application scenarios might be necessary after re-initialization of the MCU.

7.1.2.2 Clock

[SWS_Mcu_00248] ⌈Mcu shall provide a service to enable and set the MCU clock (i.e. Cpu clock, Peripheral Clock, Prescalers, Multipliers have to be configured in the MCU).⌋

Note: All the available peripheral clocks have to be made available to the other BSW modules via the `McuClockReferencePoint` container.

7.1.2.3 MCU Mode service

[SWS_Mcu_00164] ⌈The MCU module shall provide a service to activate MCU reduced power modes.⌋

The service, which activates the reduced power mode, shall allow access to power modes available in the µC hardware.

[SWS_Mcu_00165] ⌈The number of modes and the configuration is MCU dependent and shall be configured in the configuration set of the MCU module.⌋

Note: The activation of MCU reduced power modes might influence the PLL, the internal oscillator, the CPU clock, µC peripheral clock and the power supply for core and peripherals.

In typical operation, MCU reduced power mode will be entered and exited frequently during ECU runtime. In this case, wake-up is performed when it is activated in one of the MCAL modules.

The upper layer is responsible for activating MCU normal operation (condition before execution of MCU power mode) or to switch off µC power supply.

For some MCU mode configuration, the MCU is able to wake up only via hardware reset.

7.2 Error Classification

Section "Error Handling" of the document [2] "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

[SWS_Mcu_00051] ⌈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 (see [4]).⌋

[SWS_Mcu_00226] ⌈Production Errors shall not be used as the return value of the called function.⌋

7.2.1 Development Errors

[SWS_Mcu_00012] Definiton of development errors in module Mcu

Upstream requirements: SRS_BSW_00327, SRS_BSW_00337

⌈

Type of error	*Related error code*	*Error value*
API service called with wrong parameter	MCU_E_PARAM_CONFIG	0x0A
API service called with wrong parameter	MCU_E_PARAM_CLOCK	0x0B
API service called with wrong parameter	MCU_E_PARAM_MODE	0x0C
API service called with wrong parameter	MCU_E_PARAM_RAMSECTION	0x0D
API service called with wrong parameter	MCU_E_PLL_NOT_LOCKED	0x0E
API service called with wrong parameter	MCU_E_UNINIT	0x0F
API service called with wrong parameter	MCU_E_PARAM_POINTER	0x10
API service called with wrong parameter	MCU_E_INIT_FAILED	0x11

⌋

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Production Errors

There are no production errors.

7.2.4 Extended Production Errors

Type or error	Related error code	Value
Clock source failure	`MCU_E_CLOCK_FAILURE`	Assigned by DEM

[SWS_Mcu_00053] ⌈If clock failure notification is enabled in the configuration set and a clock source failure error occurs, the error code `MCU_E_CLOCK_FAILURE` shall be reported (see also [SWS_Mcu_00051]).⌋

If the clock failure is detected with other HW mechanisms e.g. the generation of a trap, this notification shall be disabled and the failure reporting shall be done outside the MCU driver.

7.2.4.1 MCU_E_CLOCK_FAILURE

Error Name:	MCU_E_CLOCK_FAILURE	
Short Description:	Clock source failure.	
Long Description:	If clock failure notification is enabled in the configuration set and a clock source failure error occurs, the error code `MCU_E_CLOCK_FAILURE` shall be reported.	
Detection Criteria:	Fail	See [SWS_Mcu_00257].
	Pass	See [SWS_Mcu_00258].
Secondary Parameters:	The condition under which the FAIL or PASS detection is active: Clock failure notification is enabled in the configuration set.	
Time Required:	Not applicable.	
Monitor Frequency:	continuous	

[SWS_Mcu_00257] ⌈Fail criteria for `MCU_E_CLOCK_FAILURE`: a clock source failure occurs⌋

[SWS_Mcu_00258] ⌈Pass criteria for `MCU_E_CLOCK_FAILURE`: no clock source failure occurs⌋

7.3 Security Events

The module does not report security events.

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[SWS_Mcu_00152] Definition of imported datatypes of module Mcu ⌈

Module	*Header File*	*Imported Type*
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋

8.2 Type definitions

8.2.1 Mcu_ConfigType

[SWS_Mcu_00249] Definition of datatype Mcu_ConfigType ⌈

Name	Mcu_ConfigType	
Kind	Structure	
Elements	Hardware dependent structure	
	Type	–
	Comment	A structure to hold the MCU driver configuration.
Description	A pointer to such a structure is provided to the MCU initialization routines for configuration.	
Available via	Mcu.h	

⌋

8.2.2 Mcu_PllStatusType

[SWS_Mcu_00250] Definition of datatype Mcu_PllStatusType ⌈

Name	Mcu_PllStatusType		
Kind	Enumeration		
Range	MCU_PLL_LOCKED	0x00	PLL is locked

▽

△

	MCU_PLL_UNLOCKED	0x01	PLL is unlocked
	MCU_PLL_STATUS_ UNDEFINED	0x02	PLL Status is unknown
Description	This is a status value returned by the function Mcu_GetPllStatus of the MCU module.		
Available via	Mcu.h		

⌋

[SWS_Mcu_00230] ⌈The type `Mcu_PllStatusType` is the type of the return value of the function `Mcu_GetPllStatus`.⌋

[SWS_Mcu_00231] ⌈The type of `Mcu_PllStatusType` is an enumeration with the following values: `MCU_PLL_LOCKED`, `MCU_PLL_UNLOCKED`, `MCU_PLL_STATUS_UNDEFINED`.⌋

8.2.3 Mcu_ClockType

[SWS_Mcu_00251] Definition of datatype Mcu_ClockType ⌈

Name	Mcu_ClockType		
Kind	Type		
Derived from	uint		
Range	0..<number of clock settings>- 1	–	The range is dependent on the number of different clock settings provided in the configuration structure. The type shall be chosen depending on MCU platform for best performance.
Description	Specifies the identification (ID) for a clock setting, which is configured in the configuration structure		
Available via	Mcu.h		

⌋

[SWS_Mcu_00232] ⌈The type `Mcu_ClockType` defines the identification (ID) for clock setting configured via the configuration structure.⌋

[SWS_Mcu_00233] ⌈The type shall be `uint8`, `uint16` or `uint32`, depending on µC platform.⌋

8.2.4 Mcu_ResetType

[SWS_Mcu_00252] Definition of datatype Mcu_ResetType ⌈

Name	Mcu_ResetType		
Kind	Enumeration		
Range	MCU_POWER_ON_RESET	0x00	Power On Reset (default)
	MCU_WATCHDOG_RESET	0x01	Internal Watchdog Timer Reset
	MCU_SW_RESET	0x02	Software Reset
	MCU_RESET_UNDEFINED	0x03	Reset is undefined
Description	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.		
Available via	Mcu.h		

⌋

[SWS_Mcu_00234] ⌈The type Mcu_ResetType, represents the different reset that a specified MCU can have.⌋

[SWS_Mcu_00134] ⌈The MCU module shall provide at least the values MCU_POWER_ON_RESET and MCU_RESET_UNDEFINED for the enumeration Mcu_ResetType.⌋

Note: Additional reset types of Mcu_ResetType may be added depending on MCU.

8.2.5 Mcu_RawResetType

[SWS_Mcu_00253] Definition of datatype Mcu_RawResetType ⌈

Name	Mcu_RawResetType		
Kind	Type		
Derived from	uint		
Range	MCU dependent register value	–	The type shall be chosen depending on MCU platform for best performance.
Description	This type specifies the reset reason in raw register format read from a reset status register.		
Available via	Mcu.h		

⌋

[SWS_Mcu_00235] ⌈The type Mcu_RawResetType specifies the reset reason in raw register format, read from a reset status register.⌋

[SWS_Mcu_00236] ⌈The type shall be `uint8`, `uint16` or `uint32` based on best performance.⌋

8.2.6 Mcu_ModeType

[SWS_Mcu_00254] Definition of datatype Mcu_ModeType ⌈

Name	Mcu_ModeType		
Kind	Type		
Derived from	uint		
Range	0..<number of MCU modes>-1	–	The range is dependent on the number of MCU modes provided in the configuration structure. The type shall be chosen depending on MCU platform for best performance.
Description	This type specifies the identification (ID) for a MCU mode, which is configured in the configuration structure.		
Available via	Mcu.h		

⌋

[SWS_Mcu_00237] ⌈The `Mcu_ModeType` specifies the identification (ID) for a MCU mode, configured via configuration structure.⌋

[SWS_Mcu_00238] ⌈The type shall be `uint8`, `uint16` or `uint32`.⌋

8.2.7 Mcu_RamSectionType

[SWS_Mcu_00255] Definition of datatype Mcu_RamSectionType ⌈

Name	Mcu_RamSectionType		
Kind	Type		
Derived from	uint		
Range	0..< number of RAM sections>-1	–	The range is dependent on the number of RAM sections provided in the configuration structure. The type shall be chosen depending on MCU platform for best performance.
Description	This type specifies the identification (ID) for a RAM section, which is configured in the configuration structure.		
Available via	Mcu.h		

⌋

[SWS_Mcu_00239] ⌈The `Mcu_RamSectionType` specifies the identification (ID) for a RAM section, configured via the configuration structure.⌋

[SWS_Mcu_00240] ⌈The type shall be `uint8`, `uint16` or `uint32`, based on best performance.⌋

8.2.8 Mcu_RamStateType

[SWS_Mcu_00256] Definition of datatype Mcu_RamStateType ⌈

Name	Mcu_RamStateType		
Kind	Enumeration		
Range	MCU_RAMSTATE_INVALID	0x00	Ram content is not valid or unknown (default).
	MCU_RAMSTATE_VALID	0x01	Ram content is valid:
Description	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.		
Available via	Mcu.h		

⌋

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 Mcu_Init

[SWS_Mcu_00153] Definition of API function Mcu_Init ⌈

Service Name	Mcu_Init	

Syntax	`void Mcu_Init (`
	`  const Mcu_ConfigType* ConfigPtr`
	`)`	

Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to MCU driver configuration set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service initializes the MCU driver.	
Available via	Mcu.h	

⌋

[SWS_Mcu_00026]

Upstream requirements: SRS_BSW_00101, SRS_BSW_00406

⌈The function Mcu_Init shall initialize the MCU module, i.e. make the configuration settings for power down, clock and RAM sections visible within the MCU module.⌋

Note: After the execution of the function Mcu_Init, the configuration data are accessible and can be used by the MCU module functions as, e.g., Mcu_InitRamSection.

The MCU module's implementer shall apply the following rules regarding initialization of controller registers within the function Mcu_Init:

1. **[SWS_Mcu_00116]** ⌈If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.⌋

2. **[SWS_Mcu_00244]** ⌈If the register can affect several hardware modules and if it is an I/O register, it shall be initialised by the PORT driver.⌋

3. **[SWS_Mcu_00245]** ⌈If the register can affect several hardware modules and if it is not an I/O register, it shall be initialised by this MCU driver.⌋

4. **[SWS_Mcu_00246]** ⌈One-time writable registers that require initialisation directly after reset shall be initialised by the startup code.⌋

5. **[SWS_Mcu_00247]** ⌈All other registers not mentioned before shall be initialised by the start-up code.⌋

Note: The term 'Hardware Module' refers to internal modules of the MCU and not to a BSW module.

8.3.2 Mcu_InitRamSection

[SWS_Mcu_00154] Definition of API function Mcu_InitRamSection ⌈

Service Name	Mcu_InitRamSection	
Syntax	`Std_ReturnType Mcu_InitRamSection ( Mcu_RamSectionType RamSection )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	RamSection	Selects RAM memory section provided in configuration set
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: command has been accepted `E_NOT_OK`: command has not been accepted e.g. due to parameter error
Description	This service initializes the RAM section wise.	
Available via	Mcu.h	

⌋

[SWS_Mcu_00011] ⌈The function `Mcu_InitRamSection` shall fill the memory from address `McuRamSectionBaseAddress` up to address `McuRamSectionBaseAddress` + `McuRamSectionSize`-1 with the byte-value contained in `McuRamDefaultValue` and by writing at once a number of bytes defined by `McuRamSectionWriteSize`, where `McuRamSectionBaseAddress`, `McuRamSectionSize`, `McuRamDefaultValue` and `McuRamSectionWriteSize` are the values of the configuration parameters for each `RamSection`.⌋

[SWS_Mcu_00136] ⌈The MCU module's environment shall call the function `Mcu_InitRamSection` only after the MCU module has been initialized using the function `Mcu_Init`.⌋

8.3.3 Mcu_InitClock

[SWS_Mcu_00155] Definition of API function Mcu_InitClock ⌈

Service Name	Mcu_InitClock
Syntax	`Std_ReturnType Mcu_InitClock ( Mcu_ClockType ClockSetting )`
Service ID [hex]	0x02

▽

△

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ClockSetting	Clock setting
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Command has been accepted `E_NOT_OK`: Command has not been accepted
Description	This service initializes the PLL and other MCU specific clock options.	
Available via	Mcu.h	

⌋

[SWS_Mcu_00137] ⌈The function `Mcu_InitClock` shall initialize the PLL and other MCU specific clock options. The clock configuration parameters are provided via the configuration structure.⌋

[SWS_Mcu_00138] ⌈The function `Mcu_InitClock` shall start the PLL lock procedure (if PLL shall be initialized) and shall return without waiting until the PLL is locked.⌋

[SWS_Mcu_00139] ⌈The MCU module's environment shall only call the function `Mcu_InitClock` after the MCU module has been initialized using the function `Mcu_-Init`.⌋

[SWS_Mcu_00210] ⌈The function `Mcu_InitClock` shall be disabled if the parameter `McuInitClock` is set to FALSE. Instead this function is available if the former parameter is set to TRUE (see also [ECUC_Mcu_00118]).⌋

8.3.4 Mcu_DistributePllClock

[SWS_Mcu_00156] Definition of API function Mcu_DistributePllClock ⌈

Service Name	Mcu_DistributePllClock

Syntax	`Std_ReturnType Mcu_DistributePllClock (`
	`  void`
	`)`

Service ID [hex]	0x03
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None

▽

△

Return value	Std_ReturnType	E_OK: Command has been accepted E_NOT_OK: Command has not been accepted
Description	This service activates the PLL clock to the MCU clock distribution.	
Available via	Mcu.h	

⌋

[SWS_Mcu_00140] ⌈The function Mcu_DistributePllClock shall activate the PLL clock to the MCU clock distribution.⌋

[SWS_Mcu_00141] ⌈The function Mcu_DistributePllClock shall remove the current clock source (for example internal oscillator clock) from MCU clock distribution.⌋

The MCU module's environment shall only call the function Mcu_DistributePllClock after the status of the PLL has been detected as locked by the function Mcu_GetPllStatus.

[SWS_Mcu_00056] ⌈The function Mcu_DistributePllClock shall return without affecting the MCU hardware if the PLL clock has been automatically activated by the MCU hardware.⌋

[SWS_Mcu_00142] ⌈If the function Mcu_DistributePllClock is called before PLL has locked, this function shall return E_NOT_OK immediately, without any further action.⌋

[SWS_Mcu_00205] ⌈The function Mcu_DistributePllClock shall be available if the pre-compile parameter McuNoPll is set to FALSE. Otherwise, this Api has to be disabled (see also [ECUC_Mcu_00180]).⌋

8.3.5 Mcu_GetPllStatus

[SWS_Mcu_00157] Definition of API function Mcu_GetPllStatus ⌈

Service Name	Mcu_GetPllStatus

Syntax	`Mcu_PllStatusType Mcu_GetPllStatus (`
	`  void`
	`)`

Service ID [hex]	0x04
Sync/Async	Synchronous
Reentrancy	Reentrant

▽

△

Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Mcu_PllStatusType	PLL Status
Description	This service provides the lock status of the PLL.	
Available via	Mcu.h	

⌋

[SWS_Mcu_00008] ⌈The function `Mcu_GetPllStatus` shall return the lock status of the PLL.⌋

[SWS_Mcu_00132] ⌈The function `Mcu_GetPllStatus` shall return `MCU_PLL_STATUS_UNDEFINED` if this function is called prior to calling of the function `Mcu_Init`.⌋

[SWS_Mcu_00206] ⌈The function `Mcu_GetPllStatus` shall also return `MCU_PLL_STATUS_UNDEFINED` if the pre-compile parameter `McuNoPll` is set to `TRUE` (see also [ECUC_Mcu_00180]).⌋

8.3.6 Mcu_GetResetReason

[SWS_Mcu_00158] Definition of API function Mcu_GetResetReason ⌈

Service Name	Mcu_GetResetReason	

Syntax	`Mcu_ResetType Mcu_GetResetReason (`
	`  void`
	`)`	

Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Mcu_ResetType	–
Description	The service reads the reset type from the hardware, if supported.	
Available via	Mcu.h	

⌋

[SWS_Mcu_00005] : ⌈The function `Mcu_GetResetReason` shall read the reset reason from the hardware and return this reason if supported by the hardware. If the hardware does not support the hardware detection of the reset reason, the return value from the function `Mcu_GetResetReason` shall always be `MCU_POWER_ON_RESET`.⌋

[SWS_Mcu_00133] ⌈The function Mcu_GetResetReason shall return MCU_RESET_-UNDEFINED if this function is called prior to calling of the function Mcu_Init, and if supported by the hardware.⌋

The User should ensure that the reset reason is cleared once it has been read out to avoid multiple reset reasons.

Note: In case of multiple calls to this function the return value should always be the same.

8.3.7 Mcu_GetResetRawValue

[SWS_Mcu_00159] Definition of API function Mcu_GetResetRawValue ⌈

Service Name	Mcu_GetResetRawValue	
Syntax	`Mcu_RawResetType Mcu_GetResetRawValue ( void )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Mcu_RawResetType	Reset raw value
Description	The service reads the reset type from the hardware register, if supported.	
Available via	Mcu.h	

⌋

[SWS_Mcu_00135] ⌈The function Mcu_GetResetRawValue shall return an implementation specific value which does not correspond to a valid value of the reset status register and is not equal to 0 if this function is called prior to calling of the function Mcu_Init, and if supported by the hardware.⌋

[SWS_Mcu_00006] ⌈The function Mcu_GetResetRawValue shall read the reset raw value from the hardware register if the hardware supports this. If the hardware does not have a reset status register, the return value shall be 0x0.⌋

The User should ensure that the reset reason is cleared once it has been read out to avoid multiple reset reasons.

Note: In case of multiple calls to this function the return value should always be the same.

8.3.8 Mcu_PerformReset

[SWS_Mcu_00160] Definition of API function Mcu_PerformReset ⌈

Service Name	Mcu_PerformReset

Syntax	`void Mcu_PerformReset (`
	`  void`
	`)`

Service ID [hex]	0x07
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	The service performs a microcontroller reset.
Available via	Mcu.h

⌋

[SWS_Mcu_00143] ⌈The function `Mcu_PerformReset` shall perform a microcontroller reset by using the hardware feature of the microcontroller.⌋

[SWS_Mcu_00144] ⌈The function `Mcu_PerformReset` shall perform the reset type which is configured in the configuration set.⌋

[SWS_Mcu_00145] ⌈The MCU module's environment shall only call the function `Mcu_PerformReset` after the MCU module has been initialized by the function `Mcu_Init`.⌋

[SWS_Mcu_00146] ⌈The function `Mcu_PerformReset` is only available if the pre-compile parameter McuPerformResetApi is set to `TRUE`. If set to `FALSE`, the function `Mcu_PerformReset` is not applicable.).⌋

8.3.9 Mcu_SetMode

[SWS_Mcu_00161] Definition of API function Mcu_SetMode ⌈

Service Name	Mcu_SetMode

Syntax	`void Mcu_SetMode (`
	`  Mcu_ModeType McuMode`
	`)`

▽

△

Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	McuMode	Set different MCU power modes configured in the configuration set
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service activates the MCU power modes.	
Available via	Mcu.h	

⌋

[SWS_Mcu_00147] ⌈The function Mcu_SetMode shall set the MCU power mode. In case of CPU power down mode, the function Mcu_SetMode returns after it has performed a wake-up.⌋

[SWS_Mcu_00148] ⌈The MCU module's environment shall only call the function Mcu_SetMode after the MCU module has been initialized by the function Mcu_Init.⌋

Note: The environment of the function Mcu_SetMode has to ensure that the ECU is ready for reduced power mode activation.

Note: The API Mcu_SetMode assumes that all interrupts are disabled prior the call of the API by the calling instance. The implementation has to take care that no wakeup interrupt event is lost. This could be achieved by a check whether pending wakeup interrupts already have occurred even if Mcu_SetMode has not set the controller to power down mode yet.

8.3.10 Mcu_GetVersionInfo

[SWS_Mcu_00162] Definition of API function Mcu_GetVersionInfo ⌈

Service Name	Mcu_GetVersionInfo	

Syntax	`void Mcu_GetVersionInfo (`
	`  Std_VersionInfoType* versioninfo`
	`)`	

Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	versioninfo	Pointer to where to store the version information of this module.

▽

△

Return value	None
Description	This service returns the version information of this module.
Available via	Mcu.h

⌋

8.3.11 Mcu_GetRamState

[SWS_Mcu_00207] Definition of API function Mcu_GetRamState

Upstream requirements: SRS_BSW_00171

⌈

Service Name	Mcu_GetRamState	
Syntax	`Mcu_RamStateType Mcu_GetRamState ( void )`	
Service ID [hex]	0x0a	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Mcu_RamStateType	Status of the Ram Content
Description	This service provides the actual status of the microcontroller Ram. (if supported)	
Available via	Mcu.h	

⌋

Note: Some microcontrollers offer the functionality to check if the Ram Status is valid after a reset. The function `Mcu_GetRamState` can be used for this reason.

[SWS_Mcu_00208] ⌈The MCU module's environment shall call this function only if the MCU module has been already initialized using the function `Mcu_Init`.⌋

[SWS_Mcu_00209] ⌈The function `Mcu_GetRamState` shall be available to the user if the pre-compile parameter `McuGetRamStateApi` is set to `TRUE`. Instead, if the former parameter is set to `FALSE`, this function shall be disabled (e.g. the hardware does not support this functionality).⌋

8.4 Callback notifications

There are no callback notifications for the MCU driver. The callback notifications are implemented in another module (ICU driver and/or complex drivers).

8.5 Scheduled functions

There are no scheduled functions within the MCU driver.

8.6 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.6.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

[SWS_Mcu_00166] Definition of mandatory interfaces required by module Mcu ⌈

API Function	***Header File***	***Description***
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)

⌋

8.6.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

[SWS_Mcu_00163] Definition of optional interfaces requested by module Mcu ⌈

API Function	***Header File***	***Description***
Det_ReportError	Det.h	Service to report development errors.

⌋

8.7 Service Interfaces

There are no service interfaces within the MCU driver.

8.8 API parameter checking

[SWS_Mcu_00017] ⌈If the development error detection is enabled for the MCU module, the MCU functions shall check the following API parameters, report detected errors to the Default Error Tracer and reject with return value `E_NOT_OK` in case the function has a standard return type.⌋

[SWS_Mcu_00019] ⌈ClockSetting shall be within the settings defined in the configuration data structure. Related error value: `MCU_E_PARAM_CLOCK`⌋

[SWS_Mcu_00020] ⌈McuMode shall be within the modes defined in the configuration data structure. Related error value: `MCU_E_PARAM_MODE`⌋

[SWS_Mcu_00021] ⌈RamSection shall be within the sections defined in the configuration data structure. Related error value: `MCU_E_PARAM_RAMSECTION`⌋

[SWS_Mcu_00122] ⌈A error shall be reported if the status of the PLL is detected as not locked with the function `Mcu_DistributePllClock`. The DET error reporting shall be used. Related error value: `MCU_E_PLL_NOT_LOCKED`.⌋

[SWS_Mcu_00125] ⌈If development error detection is enabled and if any other function (except `Mcu_GetVersionInfo`) of the MCU module is called before `Mcu_Init` function, the error code `MCU_E_UNINIT` shall be reported to the DET.⌋

9 Sequence diagrams

9.1 Example Sequence for Mcu initialization services

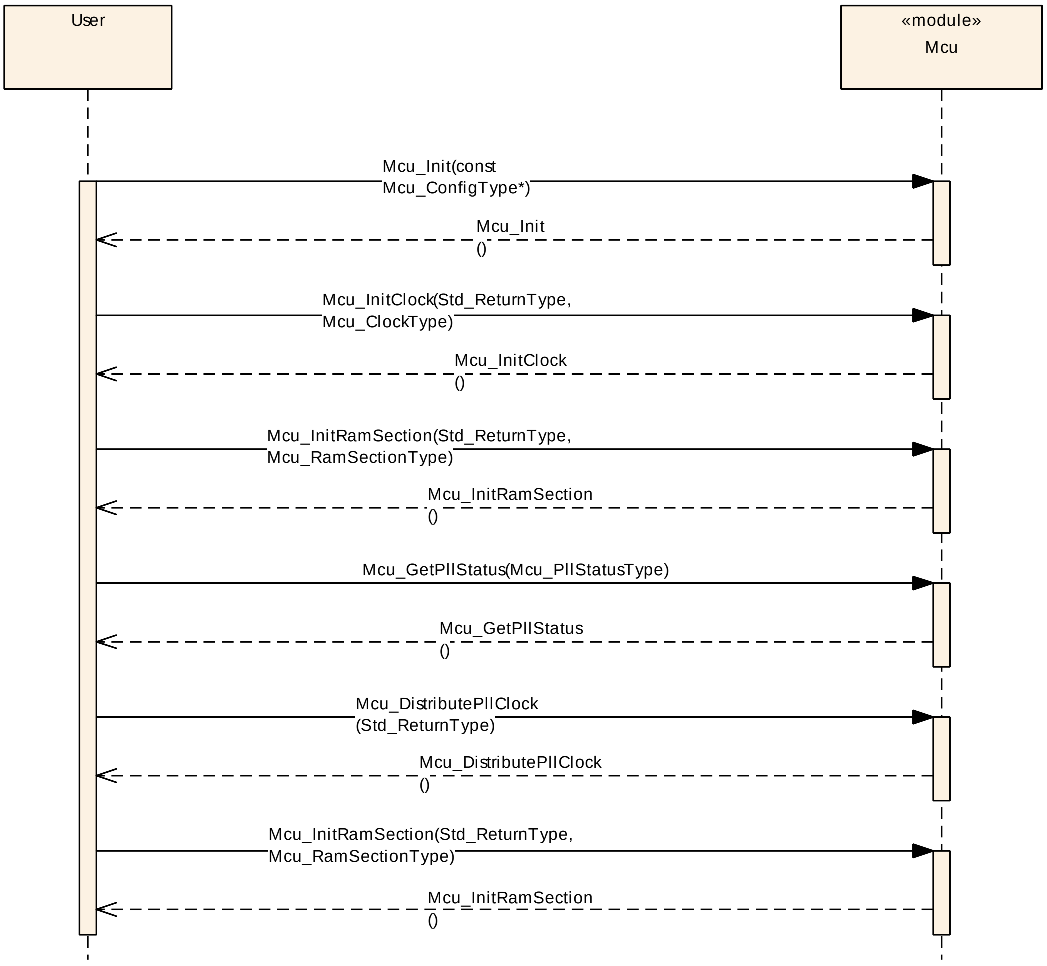
User
«module»
Mcu
Mcu_Init(const Mcu_ConfigType*)
Mcu_Init ()
Mcu_InitClock(Std_ReturnType, Mcu_ClockType)
Mcu_InitClock ()
Mcu_InitRamSection(Std_ReturnType, Mcu_RamSectionType)
Mcu_InitRamSection ()
Mcu_GetPllStatus(Mcu_PllStatusType)
Mcu_GetPllStatus ()
Mcu_DistributePllClock (Std_ReturnType)
Mcu_DistributePllClock ()
Mcu_InitRamSection(Std_ReturnType, Mcu_RamSectionType)
Mcu_InitRamSection ()

Figure 9.1: Sequence Diagram - Mcu Initialization

The order of services is just an example and might differ depending on the user. Mcu_Init shall be executed first after power-up. The user takes care that the PLL is locked by executing Mcu_GetPllStatus.

9.2 Mcu_GetResetReason

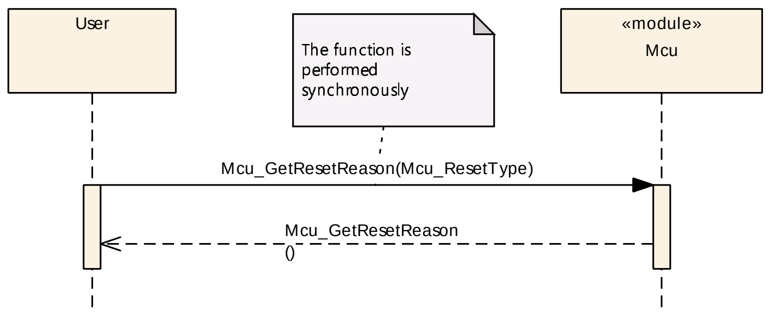
User
The function is performed synchronously
«module»
Mcu
Mcu_GetResetReason(Mcu_ResetType)
Mcu_GetResetReason
()

Figure 9.2: Sequence Diagram - Mcu_GetResetReason

9.3 Mcu_GetResetRawValue

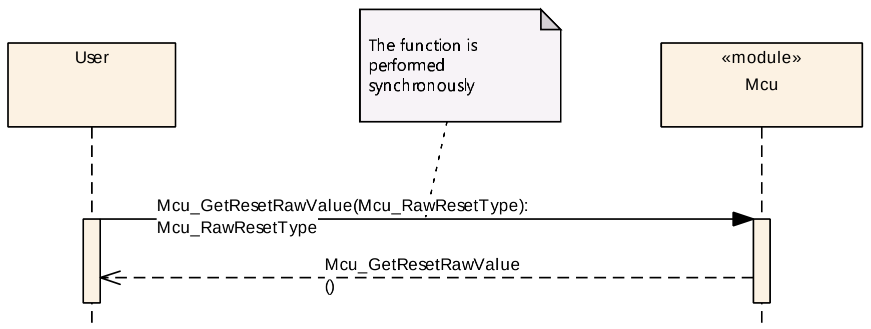
User
The function is performed synchronously
«module»
Mcu
Mcu_GetResetRawValue(Mcu_RawResetType):
Mcu_RawResetType
Mcu_GetResetRawValue
()

Figure 9.3: Sequence Diagram - Mcu_GetResetRawValue

9.4 Mcu_PerformReset

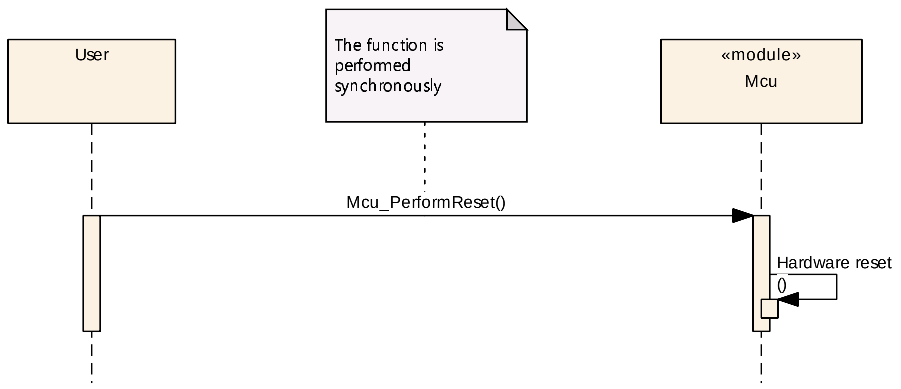

Figure 9.4: Sequence Diagram - Mcu_PerformReset

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module MCU Driver.

Chapter 10.3 specifies published information of the module MCU Driver.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in [2].

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

[SWS_Mcu_00126] ⌈The initialization function of this module shall always have a pointer as a parameter, even though for `VARIANT-PRE-COMPILE` no configuration set shall be given. Instead a `NULL` pointer shall be passed to the initialization function.⌋

[SWS_Mcu_00259] ⌈The MCU Driver module shall reject configurations with partition mappings which are not supported by the implementation.⌋

10.2.1 Mcu

[ECUC_Mcu_00189] Definition of EcucModuleDef Mcu ⌈

Module Name	Mcu
Description	Configuration of the Mcu (Microcontroller Unit) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
McuGeneralConfiguration	1	This container contains the configuration (parameters) of the MCU driver.
McuModuleConfiguration	1	This container contains the configuration (parameters) of the MCU driver
McuPublishedInformation	1	Container holding all MCU specific published information parameters

⌋

10.2.2 McuGeneralConfiguration

[ECUC_Mcu_00118] Definition of EcucParamConfContainerDef McuGeneralConfiguration ⌈

Container Name	McuGeneralConfiguration
Parent Container	Mcu
Description	This container contains the configuration (parameters) of the MCU driver.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
McuDevErrorDetect	1	[ECUC_Mcu_00166]
McuGetRamStateApi	1	[ECUC_Mcu_00181]
McuInitClock	1	[ECUC_Mcu_00182]
McuNoPll	1	[ECUC_Mcu_00180]
McuPerformResetApi	1	[ECUC_Mcu_00167]
McuVersionInfoApi	1	[ECUC_Mcu_00168]
McuEcucPartitionRef	0..*	[ECUC_Mcu_00191]

No Included Containers

⌋

[ECUC_Mcu_00166] Definition of EcucBooleanParamDef McuDevErrorDetect ⌈

Parameter Name	McuDevErrorDetect
Parent Container	McuGeneralConfiguration
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00181] Definition of EcucBooleanParamDef McuGetRamStateApi ⌈

Parameter Name	McuGetRamStateApi		
Parent Container	McuGeneralConfiguration		
Description	Pre-processor switch to enable/disable the API Mcu_GetRamState. (e.g. If the H/W does not support the functionality, this parameter can be used to disable the Api).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00182] Definition of EcucBooleanParamDef McuInitClock ⌈

Parameter Name	McuInitClock		
Parent Container	McuGeneralConfiguration		
Description	If this parameter is set to FALSE, the clock initialization has to be disabled from the MCU driver. This concept applies when there are some write once clock registers and a bootloader is present. If this parameter is set to TRUE, the MCU driver is responsible of the clock initialization.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00180] Definition of EcucBooleanParamDef McuNoPll ⌈

Parameter Name	McuNoPll		
Parent Container	McuGeneralConfiguration		
Description	This parameter shall be set True, if the H/W does not have a PLL or the PLL circuitry is enabled after the power on without S/W intervention. In this case MCU_DistributePll Clock has to be disabled and MCU_GetPllStatus has to return MCU_PLL_STATUS_ UNDEFINED. Otherwise this parameters has to be set False		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00167] Definition of EcucBooleanParamDef McuPerformResetApi ⌈

Parameter Name	McuPerformResetApi		
Parent Container	McuGeneralConfiguration		
Description	Pre-processor switch to enable / disable the use of the function Mcu_PerformReset()		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00168] Definition of EcucBooleanParamDef McuVersionInfoApi ⌈

Parameter Name	McuVersionInfoApi		
Parent Container	McuGeneralConfiguration		
Description	Pre-processor switch to enable / disable the API to read out the modules version information.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Scope / Dependency	scope: local

⌋

[ECUC_Mcu_00191] Definition of EcucReferenceDef McuEcucPartitionRef ⌈

Parameter Name	McuEcucPartitionRef		
Parent Container	McuGeneralConfiguration		
Description	Maps the MCU driver to zero or multiple ECUC partition to make the driver API available in this partition.		
Multiplicity	0..*		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

10.2.3 McuClockSettingConfig

[ECUC_Mcu_00124] Definition of EcucParamConfContainerDef McuClockSettingConfig ⌈

Container Name	McuClockSettingConfig
Parent Container	McuModuleConfiguration
Description	This container contains the configuration (parameters) for the Clock settings of the MCU. Please see MCU031 for more information on the MCU clock settings.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
McuClockSettingId	1	[ECUC_Mcu_00183]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
McuClockReferencePoint	1..*	This container defines a reference point in the Mcu Clock tree. It defines the frequency which then can be used by other modules as an input value. Lower multiplicity is 1, as even in the simplest case (only one frequency is used), there is one frequency to be defined.

⌋

[ECUC_Mcu_00183] Definition of EcucIntegerParamDef McuClockSettingId ⌈

Parameter Name	McuClockSettingId		
Parent Container	McuClockSettingConfig		
Description	The Id of this McuClockSettingConfig to be used as argument for the API call "Mcu_Init Clock".		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.4 McuModuleConfiguration

[ECUC_Mcu_00119] Definition of EcucParamConfContainerDef McuModuleConfiguration ⌈

Container Name	McuModuleConfiguration
Parent Container	Mcu
Description	This container contains the configuration (parameters) of the MCU driver
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
McuClockSrcFailureNotification	1	[ECUC_Mcu_00170]
McuNumberOfMcuModes	1	[ECUC_Mcu_00171]
McuRamSectors	1	[ECUC_Mcu_00172]
McuResetSetting	0..1	[ECUC_Mcu_00173]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
McuClockSettingConfig	1..*	This container contains the configuration (parameters) for the Clock settings of the MCU. Please see MCU031 for more information on the MCU clock settings.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
McuDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
McuModeSettingConf	1..*	This container contains the configuration (parameters) for the Mode setting of the MCU. Please see MCU035 for more information on the MCU mode settings.
McuRamSectorSettingConf	0..*	This container contains the configuration (parameters) for the RAM Sector setting. Please see MCU030 for more information on RAM sec-tor settings.

⌋

[ECUC_Mcu_00170] Definition of EcucEnumerationParamDef McuClockSrcFailureNotification ⌈

Parameter Name	McuClockSrcFailureNotification		
Parent Container	McuModuleConfiguration		
Description	Enables/Disables clock failure notification. In case this feature is not supported by HW the setting should be disabled.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DISABLED	–	
	ENABLED	–	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00171] Definition of EcucIntegerParamDef McuNumberOfMcuModes ⌈

Parameter Name	McuNumberOfMcuModes	
Parent Container	McuModuleConfiguration	
Description	This parameter shall represent the number of Modes available for the MCU. calculation Formula = Number of configured McuModeSettingConf	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default value	–	
Post-Build Variant Value	true	

▽

△

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00172] Definition of EcucIntegerParamDef McuRamSectors ⌈

Parameter Name	McuRamSectors		
Parent Container	McuModuleConfiguration		
Description	This parameter shall represent the number of RAM sectors available for the MCU. calculationFormula = Number of configured McuRamSectorSettingConf		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00173] Definition of EcucIntegerParamDef McuResetSetting ⌈

Parameter Name	McuResetSetting		
Parent Container	McuModuleConfiguration		
Description	This parameter relates to the MCU specific reset configuration. This applies to the function Mcu_PerformReset, which performs a microcontroller reset using the hardware feature of the microcontroller.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.5 McuDemEventParameterRefs

[ECUC_Mcu_00187] Definition of EcucParamConfContainerDef McuDemEventParameterRefs ⌈

Container Name	McuDemEventParameterRefs
Parent Container	McuModuleConfiguration
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MCU_E_CLOCK_FAILURE	0..1	[ECUC_Mcu_00188]

No Included Containers

⌋

[ECUC_Mcu_00188] Definition of EcucReferenceDef MCU_E_CLOCK_FAILURE ⌈

Parameter Name	MCU_E_CLOCK_FAILURE		
Parent Container	McuDemEventParameterRefs		
Description	Reference to configured DEM event to report "Clock source failure".		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: Dem		

⌋

10.2.6 McuModeSettingConf

[ECUC_Mcu_00123] Definition of EcucParamConfContainerDef McuModeSettingConf ⌈

Container Name	McuModeSettingConf
Parent Container	McuModuleConfiguration
Description	This container contains the configuration (parameters) for the Mode setting of the MCU. Please see MCU035 for more information on the MCU mode settings.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
McuMode	1	[ECUC_Mcu_00176]

No Included Containers

⌋

[ECUC_Mcu_00176] Definition of EcucIntegerParamDef McuMode ⌈

Parameter Name	McuMode		
Parent Container	McuModeSettingConf		
Description	The parameter represents the MCU Mode settings.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.7 McuRamSectorSettingConf

[ECUC_Mcu_00120] Definition of EcucParamConfContainerDef McuRamSectorSettingConf ⌈

Container Name	McuRamSectorSettingConf
Parent Container	McuModuleConfiguration
Description	This container contains the configuration (parameters) for the RAM Sector setting. Please see MCU030 for more information on RAM sec-tor settings.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
McuRamDefaultValue	1	[ECUC_Mcu_00177]
McuRamSectionBaseAddress	1	[ECUC_Mcu_00178]
McuRamSectionSize	1	[ECUC_Mcu_00179]
McuRamSectionWriteSize	1	[ECUC_Mcu_00190]

No Included Containers

⌋

[ECUC_Mcu_00177] Definition of EcucIntegerParamDef McuRamDefaultValue ⌈

Parameter Name	McuRamDefaultValue		
Parent Container	McuRamSectorSettingConf		
Description	This parameter shall represent the Data pre-setting to be initialized		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00178] Definition of EcucIntegerParamDef McuRamSectionBaseAddress ⌈

Parameter Name	McuRamSectionBaseAddress		
Parent Container	McuRamSectorSettingConf		
Description	This parameter shall represent the MCU RAM section base address		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00179] Definition of EcucIntegerParamDef McuRamSectionSize ⌈

Parameter Name	McuRamSectionSize		
Parent Container	McuRamSectorSettingConf		
Description	This parameter represents the MCU RAM Section size in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Mcu_00190] Definition of EcucIntegerParamDef McuRamSectionWriteSize ⌈

Parameter Name	McuRamSectionWriteSize		
Parent Container	McuRamSectorSettingConf		
Description	This parameter shall define the size in bytes of data which can be written into RAM at once.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	8		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.8 McuClockReferencePoint

[ECUC_Mcu_00174] Definition of EcucParamConfContainerDef McuClockReferencePoint ⌈

Container Name	McuClockReferencePoint
Parent Container	McuClockSettingConfig
Description	This container defines a reference point in the Mcu Clock tree. It defines the frequency which then can be used by other modules as an input value. Lower multiplicity is 1, as even in the simplest case (only one frequency is used), there is one frequency to be defined.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
McuClockReferencePointFrequency	1	[ECUC_Mcu_00175]

No Included Containers

⌋

[ECUC_Mcu_00175] Definition of EcucFloatParamDef McuClockReferencePoint Frequency ⌈

Parameter Name	McuClockReferencePointFrequency		
Parent Container	McuClockReferencePoint		
Description	This is the frequency for the specific instance of the McuClockReferencePoint container. It shall be given in Hz.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

⌋

10.2.9 McuPublishedInformation

[ECUC_Mcu_00184] Definition of EcucParamConfContainerDef McuPublishedInformation ⌈

Container Name	McuPublishedInformation
Parent Container	Mcu
Description	Container holding all MCU specific published information parameters
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
McuResetReasonConf	1..*	This container contains the configuration for the different type of reset reason that can be retrieved from Mcu_GetResetReason Api.

⌋

10.2.10 McuResetReasonConf

[ECUC_Mcu_00185] Definition of EcucParamConfContainerDef McuResetReasonConf ⌈

Container Name	McuResetReasonConf
Parent Container	McuPublishedInformation
Description	This container contains the configuration for the different type of reset reason that can be retrieved from Mcu_GetResetReason Api.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
McuResetReason	1	[ECUC_Mcu_00186]

No Included Containers

⌋

[ECUC_Mcu_00186] Definition of EcucIntegerParamDef McuResetReason ⌈

Parameter Name	McuResetReason		
Parent Container	McuResetReasonConf		
Description	The parameter represents the different type of reset that a Micro supports. This parameter is referenced by the parameter EcuMResetReason in the ECU State manager module.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Published Information**	X	All Variants
Scope / Dependency	scope: ECU		

⌋

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in [2].