

<b>Document Title</b>	Specification of MCU Driver
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	31
<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R21-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Removed SWS_Mcu_00131, SWS_Mcu_00054, SWS_Mcu_00035, SWS_Mcu_00030 and SWS_Mcu_00031</li> <li>Cleaned up unresolved references in traceability</li> </ul>
2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Enum and Error related modifications</li> <li>Editorial Changes</li> </ul>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Removed DRAFT status of items introduced for Multicore support</li> <li>Removed duplicated chapters McuGeneralConfiguration and McuClockSettingConfig</li> <li>Changed Document Status from Final to published</li> </ul>
2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Debugging support was removed</li> <li>Introduced support for Multicore distribution</li> </ul>
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Introduced new configuration parameter – McuRamSectionWriteSize</li> <li>Changed reentrancy of API Mcu_SetMode to Reentrant</li> </ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Removed chapter "Variants"</li> <li>Cleaned up unresolved references in traceability</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Minor change regarding DET renaming and extension Incorporation</li> <li>Clarifications regarding configuration class of symbolicNameValue parameters</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Removed requirements for NULL pointer checking as redundant with BSW General.</li> <li>Specified pass/fail criteria for extended production errors</li> </ul>
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Requirement Traceability Table revised</li> <li>Correction of requirement tag (Mcu_00146)</li> </ul>
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Mcu_GetResetReason and Mcu_GetResetRawValue return the same value if called multiple times</li> <li>RAM sector multiplicity corrected</li> <li>McuClockSettingId and McuMode range corrected</li> <li>Editorial changes</li> <li>Removed chapter(s) on change documentation</li> </ul>
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Adaptation of the Document due to the SWS General Release</li> <li>Scope Fields in all configuration parameters (chapter 10) changed as Local -&gt; impact only this module or ECU impact several modules</li> <li>Autosar Memory mapping abstraction split for each BSW</li> <li>Split Production Errors in "Pure" Production Errors and Extended Production Errors</li> <li>Changed signature of Api Mcu_DistributePllClock</li> </ul>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Mcu_SetMode assumes that all interrupts are disabled prior the call</li> </ul>
2010-09-30	3.1.5	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Corrected SWS_Mcu_00210</li> <li>Removed SWS_Mcu_00225.</li> <li>Rephrased SWS_Mcu_00125 and SWS_Mcu_00011</li> <li>Added Chapter 12</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Lots requirements rephrased to make them atomic.</li> <li>• Debugging Concept inserted.</li> <li>• Insertion of a new service (Api) to read the Status after the reset. (Affected also SRS R4.0)</li> <li>• Insertion new configuration parameters to enable/disable PLL Apis.</li> <li>• Introduction of a new container to publish all the different resets that Micro Controller support.</li> <li>• Legal disclaimer revised</li> </ul>
2008-08-13	3.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Legal disclaimer revised</li> </ul>
2007-12-21	3.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Table formatting corrected</li> </ul>
2007-01-24	2.1.15	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• 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.</li> <li>• Obsolete function Dem_ReportErrorEvent() removed.</li> <li>• Technical Office Improvements: wording improvements.</li> <li>• Re-wording of requirements for clarification</li> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2007-11-28	2.1.14	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Update to section REF _Ref158095428 \r \h : Inclusion of new file structure</li> <li>• Sections REF _Ref158095455 \r \h , REF _Ref158095460 \r \h , REF _Ref158095466 \r \h : Removal of 'const' from API type definition.</li> <li>• Section REF _Ref158095487 \r \h , REF _Ref158095489 \r \h ,10.2.5: Description detail amended</li> <li>• Section REF _Ref158095487 \r \h : Default value (0x0) for MCU_POWER_ON_RESET removed.</li> <li>• Section REF _Ref158095530 \r \h : Description updated to include reference to new pre-processor switch McuPerformResetApi.</li> <li>• Section10.2.2: Introduction of pre-processor switch McuPerformResetApi</li> <li>• Section 10.2.3: Multiplicity of sub-container Mcu Clock Setting Configuration changed to 1.</li> <li>• Legal disclaimer revised</li> <li>• Release Notes added</li> <li>• "Advice for users" revised</li> <li>• "Revision Information" added</li> </ul>
2006-05-16	2.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Document structure adapted to common Release 2.0 SWS Template.</li> <li>• Major changes in chapter 10</li> <li>• Structure of document changed partly</li> <li>• Other changes see chapter 11</li> </ul>
2005-05-31	1.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Initial Release</li> </ul>

## Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

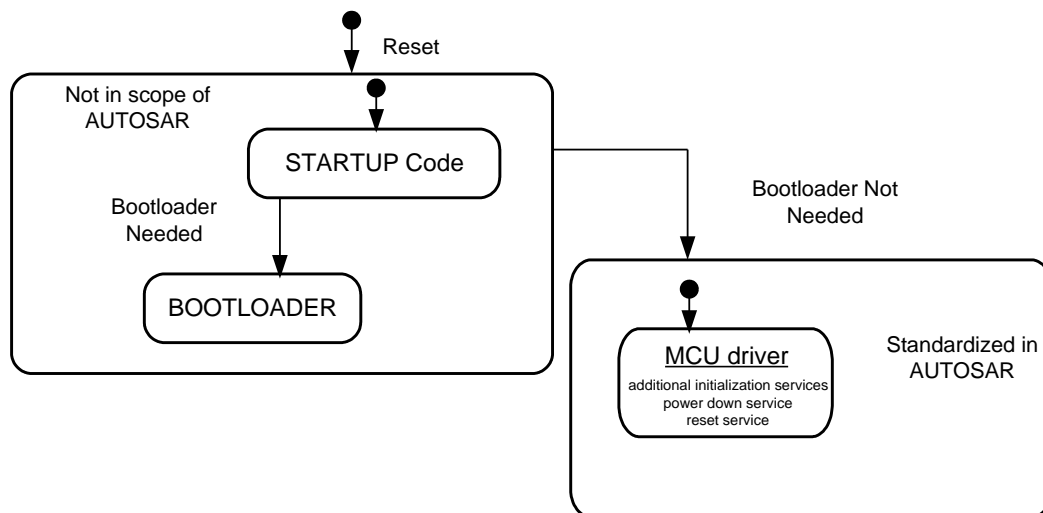
## Table of Contents

1	Introduction and functional overview .....	8
2	Acronyms and abbreviations.....	9
3	Related documentation .....	10
3.1	Input documents.....	10
3.2	Related specification.....	10
4	Constraints and assumptions.....	12
4.1	Limitations .....	12
4.2	Applicability to car domains .....	12
5	Dependencies to other modules .....	13
5.1	Start-up code.....	13
5.2	File structure .....	14
5.2.1	Code file structure .....	14
6	Requirements traceability .....	15
7	Functional specification.....	16
7.1	General Behavior .....	16
7.1.1	Background and Rationale.....	16
7.1.2	Requirements .....	16
7.2	Error classification.....	17
7.2.1	Development Errors .....	17
7.2.2	Runtime Errors .....	18
7.2.3	Transient Faults.....	18
7.2.4	Production Errors .....	18
7.2.5	Extended Production Errors.....	18
8	API specification.....	20
8.1	Imported types .....	20
8.2	Type definitions.....	20
8.2.1	Mcu_ConfigType .....	20
8.2.2	Mcu_PllStatusType .....	21
8.2.3	Mcu_ClockType.....	21
8.2.4	Mcu_ResetType .....	22
8.2.5	Mcu_RawResetType .....	22
8.2.6	Mcu_ModeType.....	23
8.2.7	Mcu_RamSectionType .....	23
8.2.8	Mcu_RamStateType.....	24
8.3	Function definitions .....	24
8.3.1	Mcu_Init .....	24
8.3.2	Mcu_InitRamSection .....	26
8.3.3	Mcu_InitClock.....	26
8.3.4	Mcu_DistributePllClock .....	27
8.3.5	Mcu_GetPllStatus.....	29
8.3.6	Mcu_GetResetReason.....	29
8.3.7	Mcu_GetResetRawValue.....	30

8.3.8	Mcu_PerformReset .....	31
8.3.9	Mcu_SetMode .....	32
8.3.10	Mcu_GetVersionInfo .....	33
8.3.11	Mcu_GetRamState .....	34
8.4	Call-back Notifications .....	34
8.5	Scheduled Functions .....	34
8.6	Expected Interfaces .....	35
8.6.1	Mandatory Interfaces.....	35
8.6.2	Optional Interfaces .....	35
8.7	API parameter checking.....	35
9	Sequence diagrams .....	37
9.1	Example Sequence for MCU initialization services .....	37
9.2	Mcu_GetResetReason.....	38
9.3	Mcu_GetResetRawValue.....	38
9.4	Mcu_PerformReset .....	39
10	Configuration specification .....	40
10.1	How to read this chapter.....	40
10.2	Containers and configuration parameters .....	40
10.2.1	Mcu.....	40
10.2.2	McuGeneralConfiguration .....	41
10.2.3	McuClockSettingConfig.....	43
10.2.4	McuModuleConfiguration .....	44
10.2.5	McuDemEventParameterRefs .....	45
10.2.6	McuModeSettingConf .....	46
10.2.7	McuRamSectorSettingConf .....	46
10.2.8	McuClockReferencePoint .....	48
10.2.9	McuPublishedInformation .....	48
10.2.10	McuResetReasonConf.....	49
10.3	Published Information .....	49

# 1 Introduction and functional overview

This specification describes the functionality and API for a MCU [**Microcontroller Unit**] driver. 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: 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  $\mu$ C reduced power modes
- Activation of a  $\mu$ C reset
- Provides a service to get the reset reason from hardware



## 2 Acronyms and abbreviations

<b><i>Abbreviation / Acronym:</i></b>	<b><i>Description:</i></b>
uC	Microcontroller
MCU	Micro Controller Unit
SFR	Special Function Register (MCU register)
DEM	Diagnostic Event Manager
DET	Default Error Tracer

**Table 1: Acronyms and Abbreviations**

## 3 Related documentation

### 3.1 Input documents

- [1] List of Basic Software Modules,  
AUTOSAR\_TR\_BSWModuleList.pdf
- [2] Layered Software Architecture,  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [3] General Requirements on Basic Software Modules,  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [4] Specification of Default Error Tracer,  
AUTOSAR\_SWS\_DefaultErrorTracer.pdf
- [5] Specification of ECU Configuration,  
AUTOSAR\_TPS\_ECUConfiguration.pdf
- [6] Specification of Diagnostic Event Manager,  
AUTOSAR\_SWS\_DiagnosticEventManager.pdf
- [7] Specification of ECU State Manager,  
AUTOSAR\_SWS\_ECUCStateManager.pdf
- [8] General Requirements on SPAL,  
AUTOSAR\_SRS\_SPALGeneral.pdf
- [9] Requirements on MCU driver,  
AUTOSAR\_SRS\_MCUDriver.pdf
- [10] Specification of Standard Types,  
AUTOSAR\_SWS\_StandardTypes.pdf
- [11] Basic Software Module Description Template,  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [12] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

### 3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [12] (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 uC 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 traceability

Requirement	Description	Satisfied by
BSW13701	-	SWS_Mcu_00207
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_00327	Error values naming convention	SWS_Mcu_00012
SRS_BSW_00337	Classification of development errors	SWS_Mcu_00012
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_Mcu_00026

## 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 uC 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, uC 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 uC power supply.

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

## 7.2 Error classification

Section 7.x "Error Handling" of the document "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 6).」()

**[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]:**

<i>Type of error</i>	<i>Related error code</i>	<i>Error value</i>
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

](SRS\_BSW\_00327, SRS\_BSW\_00337)

## 7.2.2 Runtime Errors

There are no runtime errors.

## 7.2.3 Transient Faults

There are no transient faults.

## 7.2.4 Production Errors

There are no production errors.

## 7.2.5 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.

### 7.2.5.1 MCU\_E\_CLOCK\_FAILURE

<b>Error Name:</b>	MCU_E_CLOCK_FAILURE
<b>Short Description:</b>	Clock source failure.
<b>Long Description:</b>	If clock failure notification is enabled in the configuration set and a clock source failure error occurs, the error code <code>MCU_E_CLOCK_FAILURE</code>

	shall be reported.	
<b>Detection Criteria:</b>	Fail	See SWS_Mcu_00257.
	Pass	See SWS_Mcu_00258.
<b>Secondary Parameters:</b>	The condition under which the FAIL or PASS detection is active: Clock failure notification is enabled in the configuration set.	
<b>Time Required:</b>	Not applicable.	
<b>Monitor Frequency</b>	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」()

## 8 API specification

### 8.1 Imported types

In this chapter all types included from the following modules are listed:

[SWS\_Mcu\_00152]

<i>Module</i>	<i>Header File</i>	<i>Imported Type</i>
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]

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

]()

## 8.2.2 Mcu\_PllStatusType

[SWS\_Mcu\_00250]

<b>Name</b>	Mcu_PllStatusType		
<b>Kind</b>	Enumeration		
<b>Range</b>	MCU_PLL_LOCKED	0x00	PLL is locked
	MCU_PLL_UNLOCKED	0x01	PLL is unlocked
	MCU_PLL_STATUS_UNDEFINED	0x02	PLL Status is unknown
<b>Description</b>	This is a status value returned by the function <code>Mcu_GetPllStatus</code> of the MCU module.		
<b>Available via</b>	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]

<b>Name</b>	Mcu_ClockType		
<b>Kind</b>	Type		
<b>Derived from</b>	uint		
<b>Range</b>	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.
<b>Description</b>	Specifies the identification (ID) for a clock setting, which is configured in the configuration structure		
<b>Available via</b>	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 uC platform.」()

### 8.2.4 Mcu\_ResetType

[SWS\_Mcu\_00252][

<b>Name</b>	Mcu_ResetType		
<b>Kind</b>	Enumeration		
<b>Range</b>	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
<b>Description</b>	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.		
<b>Available via</b>	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][

<b>Name</b>	Mcu_RawResetType		
<b>Kind</b>	Type		
<b>Derived from</b>	uint		
<b>Range</b>	MCU dependent register value	--	The type shall be chosen depending on MCU platform for best performance.
<b>Description</b>	This type specifies the reset reason in raw register format read from a reset status register.		

<b>Available via</b>	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]**

<b>Name</b>	Mcu_ModeType		
<b>Kind</b>	Type		
<b>Derived from</b>	uint		
<b>Range</b>	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.
<b>Description</b>	This type specifies the identification (ID) for a MCU mode, which is configured in the configuration structure.		
<b>Available via</b>	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]**

<b>Name</b>	Mcu_RamSectionType		
<b>Kind</b>	Type		
<b>Derived from</b>	uint		
<b>Range</b>	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.
<b>Description</b>	This type specifies the identification (ID) for a RAM section, which is configured in the configuration structure.	
<b>Available via</b>	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]**

<b>Name</b>	<code>Mcu_RamStateType</code>		
<b>Kind</b>	Enumeration		
<b>Range</b>	<code>MCU_RAMSTATE_INVALID</code>	0x00	Ram content is not valid or unknown (default).
	<code>MCU_RAMSTATE_VALID</code>	0x01	Ram content is valid:
<b>Description</b>	This is the Ram State data type returned by the function <code>Mcu_GetRamState</code> of the <code>Mcu</code> module. It is not required that all RAM state types are supported by the hardware.		
<b>Available via</b>	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]**

<b>Service Name</b>	<code>Mcu_Init</code>
<b>Syntax</b>	<pre>void Mcu_Init (     const Mcu_ConfigType* ConfigPtr )</pre>



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

]()

**[SWS\_Mcu\_00026]:** 「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.」( SRS\_BSW\_00101, SRS\_BSW\_00406)

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]

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

<b>Service Name</b>	Mcu_InitClock	
<b>Syntax</b>	Std_ReturnType Mcu_InitClock ( Mcu_ClockType ClockSetting	

	)	
<b>Service ID [hex]</b>	0x02	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	ClockSetting	Clock setting
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: Command has been accepted E_NOT_OK: Command has not been accepted
<b>Description</b>	This service initializes the PLL and other MCU specific clock options.	
<b>Available via</b>	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]**

<b>Service Name</b>	Mcu_DistributePllClock
<b>Syntax</b>	Std_ReturnType Mcu_DistributePllClock ( void

	)	
<b>Service ID [hex]</b>	0x03	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: Command has been accepted E_NOT_OK: Command has not been accepted
<b>Description</b>	This service activates the PLL clock to the MCU clock distribution.	
<b>Available via</b>	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]

<b>Service Name</b>	Mcu_GetPllStatus	
<b>Syntax</b>	Mcu_PllStatusType Mcu_GetPllStatus ( void )	
<b>Service ID [hex]</b>	0x04	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Mcu_PllStatusType	PLL Status
<b>Description</b>	This service provides the lock status of the PLL.	
<b>Available via</b>	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]

<b>Service Name</b>	Mcu_GetResetReason
---------------------	--------------------

<b>Syntax</b>	Mcu_ResetType Mcu_GetResetReason ( void )	
<b>Service ID [hex]</b>	0x05	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Mcu_ResetType	--
<b>Description</b>	The service reads the reset type from the hardware, if supported.	
<b>Available via</b>	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]**

<b>Service Name</b>	Mcu_GetResetRawValue	
<b>Syntax</b>	Mcu_RawResetType Mcu_GetResetRawValue ( void )	
<b>Service ID [hex]</b>	0x06	

<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Mcu_RawResetType	Reset raw value
<b>Description</b>	The service reads the reset type from the hardware register, if supported.	
<b>Available via</b>	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]**

<b>Service Name</b>	Mcu_PerformReset
<b>Syntax</b>	<pre>void Mcu_PerformReset (     void )</pre>
<b>Service ID [hex]</b>	0x07
<b>Sync/Async</b>	Synchronous
<b>Reentrancy</b>	Non Reentrant
<b>Parameters (in)</b>	None
<b>Parameters (inout)</b>	None

<b>Parameters (out)</b>	None
<b>Return value</b>	None
<b>Description</b>	The service performs a microcontroller reset.
<b>Available via</b>	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. (see Section 10.2.2).」()

### 8.3.9 Mcu\_SetMode

**[SWS\_Mcu\_00161]**

<b>Service Name</b>	Mcu_SetMode	
<b>Syntax</b>	<pre>void Mcu_SetMode (     Mcu_ModeType McuMode )</pre>	
<b>Service ID [hex]</b>	0x08	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	McuMode	Set different MCU power modes configured in the configuration set
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	This service activates the MCU power modes.	



<b>Available via</b>	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]**

<b>Service Name</b>	Mcu_GetVersionInfo	
<b>Syntax</b>	<pre>void Mcu_GetVersionInfo (     Std_VersionInfoType* versioninfo )</pre>	
<b>Service ID [hex]</b>	0x09	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	versioninfo	Pointer to where to store the version information of this module.
<b>Return value</b>	None	
<b>Description</b>	This service returns the version information of this module.	
<b>Available via</b>	Mcu.h	

]()

### 8.3.11 Mcu\_GetRamState

[SWS\_Mcu\_00207]

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

](BSW13701)

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 Call-back 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

[SWS\_Mcu\_00166]

<i>API Function</i>	<i>Header File</i>	<i>Description</i>
Dem_Set-EventStatus	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 $\{\{\text{Dem/DemConfigSet/DemEventParameter/DemEventReportingType}\} == \text{STANDARD\_REPORTING}\}$

]()

### 8.6.2 Optional Interfaces

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

[SWS\_Mcu\_00163]

<i>API Function</i>	<i>Header File</i>	<i>Description</i>
Det_ReportError	Det.h	Service to report development errors.

]()

## 8.7 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`」()

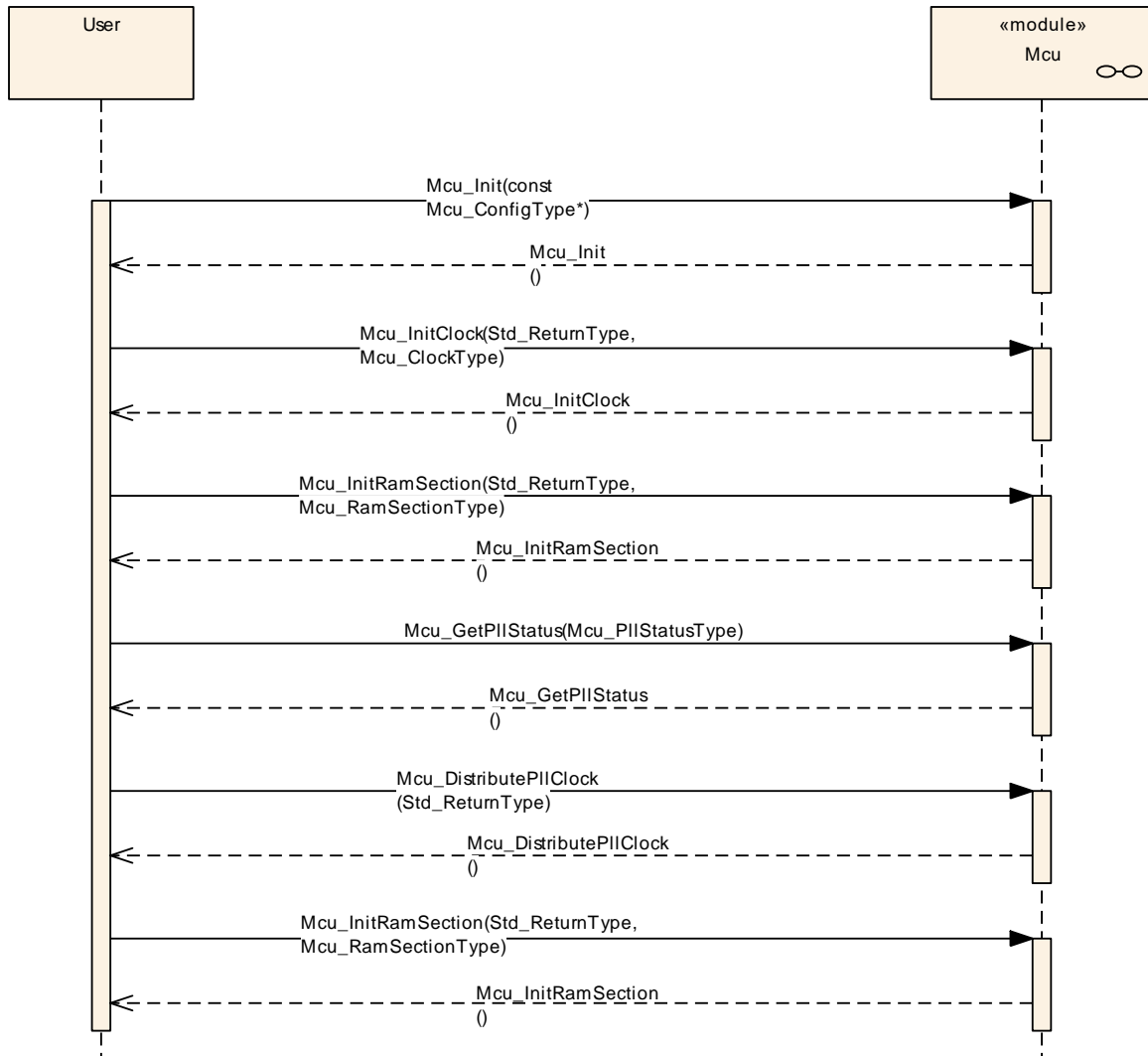
**[SWS\_Mcu\_00122]:** 「A error shall be reported if the status of the PLL is detected as not locked with the function `Mcu_DistributePllClock()`. The DET error reporting shall be used. Related error value: `MCU_E_PLL_NOT_LOCKED.`」()

**[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



**Figure 3: Sequence Diagram – MCU Initialisation**

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

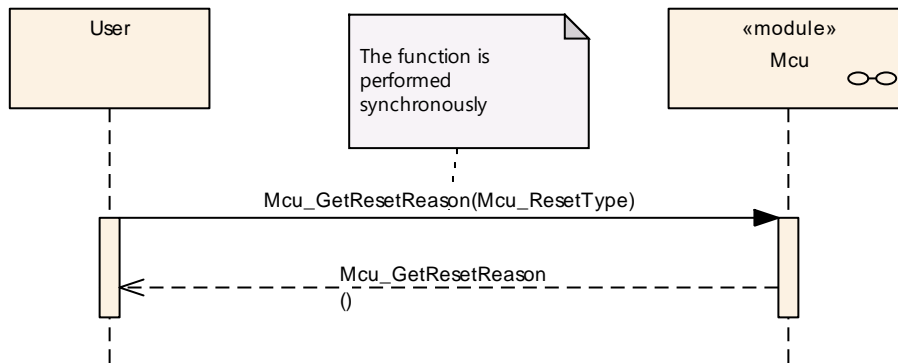


Figure 7: Sequence Diagram – MCU\_GetResetReason

### 9.3 Mcu\_GetResetRawValue

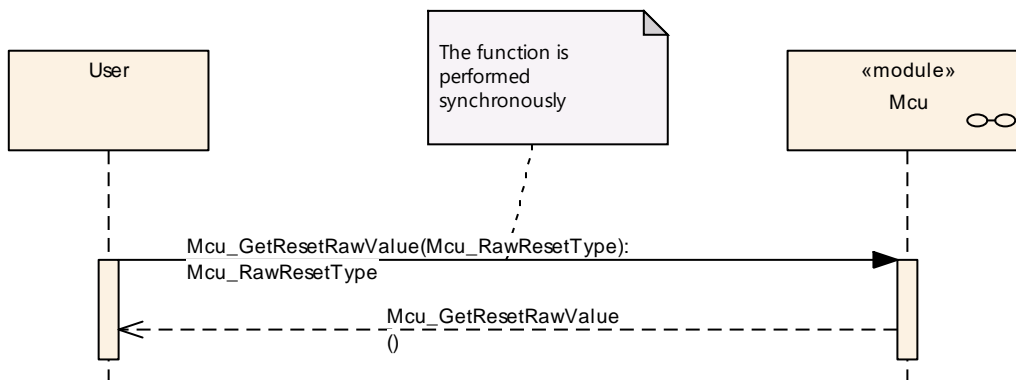
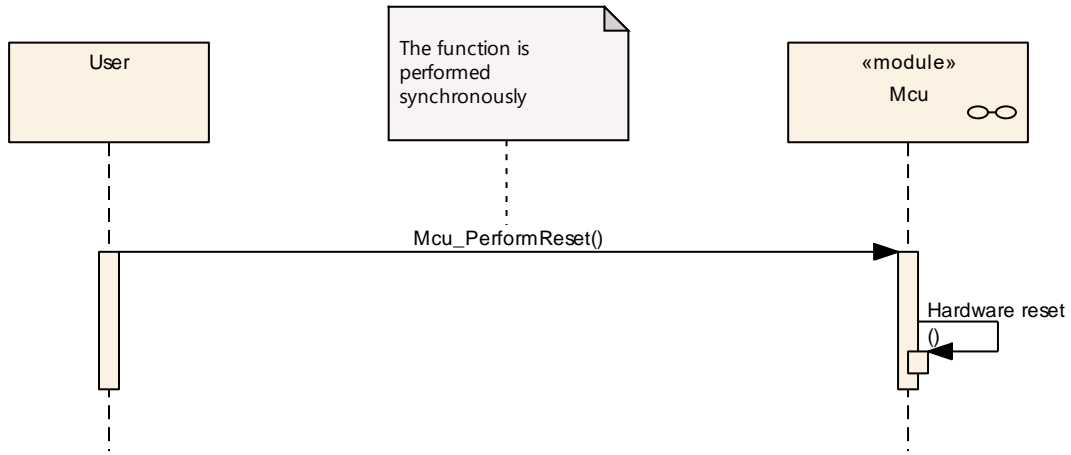


Figure 8: Sequence Diagram – Mcu\_GetResetRawValue

### 9.4 Mcu\_PerformReset



**Figure 9: 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.

Chapter 10.3 specifies published information of the module MCU.

### 10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS\_BSWGeneral*.

### 10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 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.」()

**[SWS\_Mcu\_CONSTR\_00001]:** 「The module will operate as an independent instance in each of the partitions, means the called API will only target the partition it is called in.」()

#### 10.2.1 Mcu

<b>SWS Item</b>	<b>ECUC_Mcu_00189 :</b>
<b>Module Name</b>	<i>Mcu</i>
<b>Module Description</b>	Configuration of the Mcu (Microcontroller Unit) module.
<b>Post-Build Variant Support</b>	true
<b>Supported Config Variants</b>	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

<b>Included Containers</b>
----------------------------



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

SWS Item	ECUC_Mcu_00118 :
Container Name	McuGeneralConfiguration
Parent Container	Mcu
Description	This container contains the configuration (parameters) of the MCU driver.
<b>Configuration Parameters</b>	

SWS Item	ECUC_Mcu_00166 :		
Name	McuDevErrorDetect		
Parent Container	McuGeneralConfiguration		
Description	Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>true: detection and notification is enabled.</li> <li>false: detection and notification is disabled.</li> </ul>		
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		

SWS Item	ECUC_Mcu_00181 :		
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		

SWS Item	ECUC_Mcu_00182 :
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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	true		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00180 :</b>		
<b>Name</b>	McuNoPll		
<b>Parent Container</b>	McuGeneralConfiguration		
<b>Description</b>	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_DistributePllClock has to be disabled and MCU_GetPllStatus has to return MCU_PLL_STATUS_UNDEFINED. Otherwise this parameters has to be set False		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	true		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00167 :</b>		
<b>Name</b>	McuPerformResetApi		
<b>Parent Container</b>	McuGeneralConfiguration		
<b>Description</b>	Pre-processor switch to enable / disable the use of the function Mcu_PerformReset()		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00168 :</b>		
<b>Name</b>	McuVersionInfoApi		
<b>Parent Container</b>	McuGeneralConfiguration		
<b>Description</b>	Pre-processor switch to enable / disable the API to read out the modules version information.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00191 :</b>		
-----------------	-------------------------	--	--

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

**No Included Containers**

### 10.2.3 McuClockSettingConfig

<b>SWS Item</b>	<b>ECUC_Mcu_00124 :</b>		
<b>Container Name</b>	McuClockSettingConfig		
<b>Parent Container</b>	McuModuleConfiguration		
<b>Description</b>	This container contains the configuration (parameters) for the Clock settings of the MCU. Please see MCU031 for more information on the MCU clock settings.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_Mcu_00183 :</b>		
<b>Name</b>	McuClockSettingId		
<b>Parent Container</b>	McuClockSettingConfig		
<b>Description</b>	The Id of this McuClockSettingConfig to be used as argument for the API call "Mcu_InitClock".		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
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.

### 10.2.4 McuModuleConfiguration

<b>SWS Item</b>	<b>ECUC_Mcu_00119 :</b>
<b>Container Name</b>	McuModuleConfiguration
<b>Parent Container</b>	Mcu
<b>Description</b>	This container contains the configuration (parameters) of the MCU driver
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Mcu_00170 :</b>		
<b>Name</b>	McuClockSrcFailureNotification		
<b>Parent Container</b>	McuModuleConfiguration		
<b>Description</b>	Enables/Disables clock failure notification. In case this feature is not supported by HW the setting should be disabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	DISABLED	--	
	ENABLED	--	
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00171 :</b>		
<b>Name</b>	McuNumberOfMcuModes		
<b>Parent Container</b>	McuModuleConfiguration		
<b>Description</b>	This parameter shall represent the number of Modes available for the MCU. calculationFormula = Number of configured McuModeSettingConf		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00172 :</b>		
<b>Name</b>	McuRamSectors		
<b>Parent Container</b>	McuModuleConfiguration		
<b>Description</b>	This parameter shall represent the number of RAM sectors available for the MCU. calculationFormula = Number of configured McuRamSectorSettingConf		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 4294967295		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00173 :</b>		
<b>Name</b>	McuResetSetting		
<b>Parent Container</b>	McuModuleConfiguration		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
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.
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 sector settings.

### 10.2.5 McuDemEventParameterRefs

<b>SWS Item</b>	<b>ECUC_Mcu_00187 :</b>
<b>Container Name</b>	McuDemEventParameterRefs
<b>Parent Container</b>	McuModuleConfiguration
<b>Description</b>	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.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Mcu_00188 :</b>
-----------------	-------------------------

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

**No Included Containers**

### 10.2.6 McuModeSettingConf

<b>SWS Item</b>	<b>ECUC_Mcu_00123 :</b>		
<b>Container Name</b>	McuModeSettingConf		
<b>Parent Container</b>	McuModuleConfiguration		
<b>Description</b>	This container contains the configuration (parameters) for the Mode setting of the MCU. Please see MCU035 for more information on the MCU mode settings.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_Mcu_00176 :</b>		
<b>Name</b>	McuMode		
<b>Parent Container</b>	McuModeSettingConf		
<b>Description</b>	The parameter represents the MCU Mode settings.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.2.7 McuRamSectorSettingConf

<b>SWS Item</b>	<b>ECUC_Mcu_00120 :</b>		
<b>Container Name</b>	McuRamSectorSettingConf		
<b>Parent Container</b>	McuModuleConfiguration		
<b>Description</b>	This container contains the configuration (parameters) for the RAM Sector setting. Please see MCU030 for more information on RAM sec-tor settings.		

**Configuration Parameters**

<b>SWS Item</b>	<b>ECUC_Mcu_00177 :</b>		
<b>Name</b>	McuRamDefaultValue		
<b>Parent Container</b>	McuRamSectorSettingConf		
<b>Description</b>	This parameter shall represent the Data pre-setting to be initialized		
<b>Multiplicity</b>	1		
<b>Type</b>	EcuIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00178 :</b>		
<b>Name</b>	McuRamSectionBaseAddress		
<b>Parent Container</b>	McuRamSectorSettingConf		
<b>Description</b>	This parameter shall represent the MCU RAM section base address		
<b>Multiplicity</b>	1		
<b>Type</b>	EcuIntegerParamDef		
<b>Range</b>	0 .. 4294967295		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00179 :</b>		
<b>Name</b>	McuRamSectionSize		
<b>Parent Container</b>	McuRamSectorSettingConf		
<b>Description</b>	This parameter represents the MCU RAM Section size in bytes.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcuIntegerParamDef		
<b>Range</b>	0 .. 4294967295		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Mcu_00190 :</b>		
<b>Name</b>	McuRamSectionWriteSize		
<b>Parent Container</b>	McuRamSectorSettingConf		
<b>Description</b>	This parameter shall define the size in bytes of data which can be written into RAM at once.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcuIntegerParamDef		
<b>Range</b>	0 .. 4294967295		
<b>Default value</b>	8		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE



	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.2.8 McuClockReferencePoint

<b>SWS Item</b>	<b>ECUC_Mcu_00174 :</b>		
<b>Container Name</b>	McuClockReferencePoint		
<b>Parent Container</b>	McuClockSettingConfig		
<b>Description</b>	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.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_Mcu_00175 :</b>		
<b>Name</b>	McuClockReferencePointFrequency		
<b>Parent Container</b>	McuClockReferencePoint		
<b>Description</b>	This is the frequency for the specific instance of the McuClockReferencePoint container. It shall be given in Hz.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

**No Included Containers**

### 10.2.9 McuPublishedInformation

<b>SWS Item</b>	<b>ECUC_Mcu_00184 :</b>		
<b>Container Name</b>	McuPublishedInformation		
<b>Parent Container</b>	Mcu		
<b>Description</b>	Container holding all MCU specific published information parameters		
<b>Configuration Parameters</b>			

<b>Included Containers</b>			
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>	
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

<b>SWS Item</b>	<b>ECUC_Mcu_00185 :</b>
<b>Container Name</b>	McuResetReasonConf
<b>Parent Container</b>	McuPublishedInformation
<b>Description</b>	This container contains the configuration for the different type of reset reason that can be retrieved from Mcu_GetResetReason Api.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Mcu_00186 :</b>		
<b>Name</b>	McuResetReason		
<b>Parent Container</b>	McuResetReasonConf		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Published Information</b>	X	All Variants
<b>Scope / Dependency</b>	scope: ECU		

<b>No Included Containers</b>
-------------------------------

## 10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS\_BSWGeneral*.