Document Title	Specification of MCU Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	031
Document Classification	Standard
Document Version	3.4.1
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	3.4.1	AUTOSAR Release Management	• Requirement Traceability Table revised • Correction of requirement tag (Mcu_00146)
31.10.2013	3.4.0	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
27.02.2013	3.3.0	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
09.12.2012	3.2.0	AUTOSAR Administration	• Mcu_SetMode assumes that all interrupts are disabled prior the call
13.10.2010	3.1.0	AUTOSAR Administration	• Corrected SWS_Mcu_00210 • Removed SWS_Mcu_00225. • Rephrased SWS_Mcu_00125 and SWS_Mcu_00011 • Added Chapter 12

30.11.2009	3.0.0	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
23.06.2008	2.2.2	AUTOSAR Administration	Legal disclaimer revised
23.01.2008	2.2.1	AUTOSAR Administration	Table formatting corrected
11.12.2007	2.2.0	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

31.01.2007	2.1.0	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. • Section10.2.2: 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
26.01.2006	2.0.0	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
23.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification 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 specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

This specification describes the functionality and API for a MCU [**M**icro**c**ontroller **U**nit] 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.

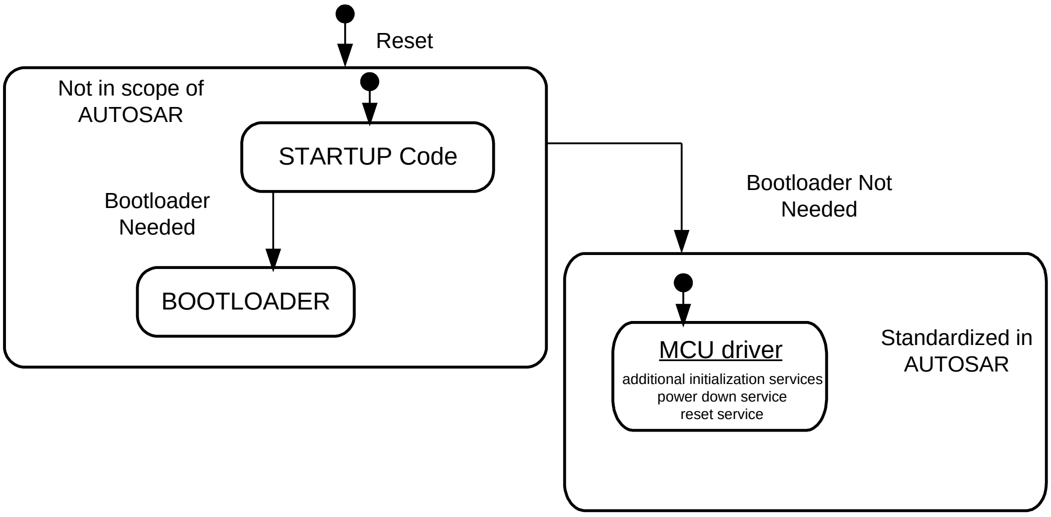

Figure 1: Scope of the MCU Driver Specification

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 µC reduced power modes
- Activation of a µC reset
- Provides a service to get the reset reason from hardware

2 Acronyms and abbreviations

Abbreviation / Acronym:	***Description:***
uC	Microcontroller
MCU	Micro Controller Unit
SFR	Special Function Register (MCU register)
DEM	Diagnostic Event Manager
DET	Development 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 Development Error Tracer,
AUTOSAR_SWS_DevelopmentErrorTracer.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_ECUStateManager.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.

5.2.2 Header file structure

The include file structure shall be as follows:

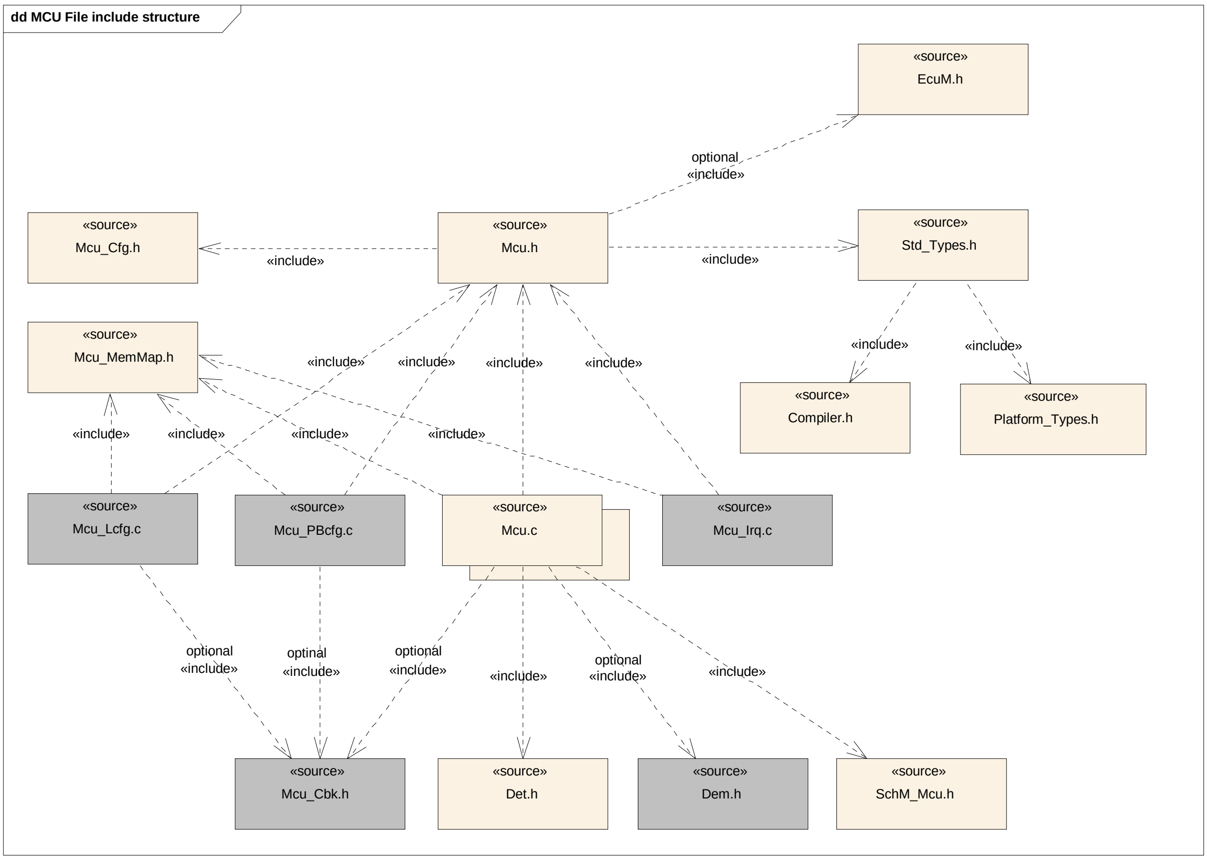

Figure 2: Header File Structure

[SWS_Mcu_00211]: ⌈`Mcu.h` shall include `Mcu_Cfg.h` for the API pre-compiler switches.⌋()

Mcu.c has access to the Mcu_Cfg.h via the implicitly included Mcu.h file.

.

[SWS_Mcu_00215]: ⌈The type definitions for `Mcu_Lcfg.c` and `Mcu_PBcfg.c` are located in the file `Mcu.h.`⌋()

Rather the implicit include of Mcu_Cfg.h via Mcu.h in the files Mcu_Lcfg.c and Mcu_PBcfg.c is necessary to solve the following construct:

```
Mcu.h
----------
#include "Mcu.h"

#ifdef xxx_VERSION_INFO_API
xxx_GetVersionInfo(...)
#endif
```

```
Mcu_Cfg.h
---------------
#define xxx_VERSION_INFO_API
```

[SWS_Mcu_00216]：⌈Mcu_Lcfg.c shall include Mcu_Cbk.h for a link time configuration if the call back function is linked to the module via the ROM structure.⌋()

[SWS_Mcu_00218]: ⌈Mcu_PBcfg.c shall include Mcu_Cbk.h for post build time configuration if the call back function is linked to the module via the ROM structure.⌋()

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_Mcu_00017
-	-	SWS_Mcu_00018
-	-	SWS_Mcu_00019
-	-	SWS_Mcu_00020
-	-	SWS_Mcu_00021
-	-	SWS_Mcu_00051
-	-	SWS_Mcu_00122
-	-	SWS_Mcu_00125
-	-	SWS_Mcu_00126
-	-	SWS_Mcu_00127
-	-	SWS_Mcu_00129
-	-	SWS_Mcu_00130
-	-	SWS_Mcu_00131
-	-	SWS_Mcu_00132
-	-	SWS_Mcu_00133
-	-	SWS_Mcu_00134
-	-	SWS_Mcu_00135
-	-	SWS_Mcu_00136
-	-	SWS_Mcu_00139
-	-	SWS_Mcu_00142
-	-	SWS_Mcu_00145
-	-	SWS_Mcu_00146
-	-	SWS_Mcu_00147
-	-	SWS_Mcu_00148
-	-	SWS_Mcu_00152
-	-	SWS_Mcu_00153
-	-	SWS_Mcu_00154
-	-	SWS_Mcu_00155
-	-	SWS_Mcu_00156
-	-	SWS_Mcu_00157
-	-	SWS_Mcu_00158
-	-	SWS_Mcu_00159
-	-	SWS_Mcu_00160
-	-	SWS_Mcu_00161
-	-	SWS_Mcu_00162
-	-	SWS_Mcu_00163

-	-	SWS_Mcu_00166
-	-	SWS_Mcu_00204
-	-	SWS_Mcu_00205
-	-	SWS_Mcu_00206
-	-	SWS_Mcu_00210
-	-	SWS_Mcu_00211
-	-	SWS_Mcu_00215
-	-	SWS_Mcu_00216
-	-	SWS_Mcu_00218
-	-	SWS_Mcu_00226
-	-	SWS_Mcu_00230
-	-	SWS_Mcu_00231
-	-	SWS_Mcu_00232
-	-	SWS_Mcu_00233
-	-	SWS_Mcu_00234
-	-	SWS_Mcu_00235
-	-	SWS_Mcu_00236
-	-	SWS_Mcu_00237
-	-	SWS_Mcu_00238
-	-	SWS_Mcu_00239
-	-	SWS_Mcu_00240
-	-	SWS_Mcu_00249
-	-	SWS_Mcu_00250
-	-	SWS_Mcu_00251
-	-	SWS_Mcu_00252
-	-	SWS_Mcu_00253
-	-	SWS_Mcu_00254
-	-	SWS_Mcu_00255
-	-	SWS_Mcu_00256
BSW00327	-	SWS_Mcu_00012
BSW00337	-	SWS_Mcu_00012
BSW00406	-	SWS_Mcu_00026
BSW101	-	SWS_Mcu_00026
BSW12000	-	SWS_Mcu_00005, SWS_Mcu_00052
BSW12057	-	SWS_Mcu_00026
BSW12063	-	SWS_Mcu_00006
BSW12125	-	SWS_Mcu_00116, SWS_Mcu_00244, SWS_Mcu_00245, SWS_Mcu_00246, SWS_Mcu_00247
BSW12207	-	SWS_Mcu_00031, SWS_Mcu_00054
BSW12208	-	SWS_Mcu_00137, SWS_Mcu_00138, SWS_Mcu_00248

BSW12215	-	SWS_Mcu_00006
BSW12268	-	SWS_Mcu_00164, SWS_Mcu_00165
BSW12277	-	SWS_Mcu_00055, SWS_Mcu_00143, SWS_Mcu_00144
BSW12331	-	SWS_Mcu_00011
BSW12336	-	SWS_Mcu_00056, SWS_Mcu_00140, SWS_Mcu_00141
BSW12350	-	SWS_Mcu_00030
BSW12392	-	SWS_Mcu_00008
BSW12394	-	SWS_Mcu_00012, SWS_Mcu_00053
BSW12421	-	SWS_Mcu_00035
BSW12461	-	SWS_Mcu_00116, SWS_Mcu_00244, SWS_Mcu_00245, SWS_Mcu_00246, SWS_Mcu_00247
BSW13701	-	SWS_Mcu_00207, SWS_Mcu_00208, SWS_Mcu_00209
BSW157	-	SWS_Mcu_00005, SWS_Mcu_00006, SWS_Mcu_00008, SWS_Mcu_00012

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.⌋(BSW12277)

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.⌋(BSW12000)

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)⌋(BSW12208)

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.⌋(BSW12268)

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.⌋ (BSW12268)

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

7.2.1 Background and Rationale

The error classification depends on the time of error occurrence according to the product life cycle:

- Development Errors:
  These errors shall be detected and fixed during the development phase. In most cases, these errors are software errors. The detection of errors that shall only occur during development can be switched off for production code (by static configuration, i.e. pre-processor switches).

- Production:
  These errors are hardware errors and software exceptions that cannot be avoided and are also expected to occur in production code.

7.2.2 Development Errors

[SWS_Mcu_00012]: ⌈The following errors and exceptions shall be detectable by the MCU module depending on its build version (development/production mode):

Type or error	**Relevance**	**Related error code**	**Value**

API service called with wrong parameter	Development	`MCU_E_PARAM_CONFIG`
		`MCU_E_PARAM_CLOCK`
		`MCU_E_PARAM_MODE`
		`MCU_E_PARAM_RAMSECTION`
		`MCU_E_PLL_NOT_LOCKED`
		`MCU_E_UNINIT`
		`MCU_E_PARAM_POINTER`
			`0x0A`
			`0x0B`
			`0x0C`
			`0x0D`
			`0x0E`
			`0x0F`
			`0x10`

Table 2: Error Classification

⌋(BSW00327, BSW00337, BSW157, BSW12394)

7.2.3 Production Errors

This module does not specify any production errors.

7.2.4 Extended Production Errors (for Release 4.1.1)

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).⌋(BSW12394)

7.3 Error detection

For details refer to the chapters 7.2 “Error classification” & 7.3 “Error Detection” in *SWS_BSWGeneral.*

7.4 Error notification

[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.⌋()

7.5 Debugging Support

For details refer to the chapter 7.1.17 “Debugging support” in *SWS_BSWGeneral.*

8 API specification

8.1 Imported types

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

[SWS_Mcu_00152]: ⌈

Module	***Imported Type***
Dem	Dem_EventIdType
	Dem_EventStatusType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋()

8.2 Type definitions

8.2.1 Mcu_ConfigType

[SWS_Mcu_00249]⌈

Name:	`Mcu_ConfigType`	
Type:	`Structure`	
Range:	`Hardware dependent structure`	A structure to hold the MCU driver configuration.
Description:	A pointer to such a structure is provided to the MCU initialization routines for configuration.	

⌋()

[SWS_Mcu_00131]: ⌈The structure `Mcu_ConfigType` is an external data structure (i.e. implementation specific) and shall contain the initialization data for the MCU module. It shall contain:

- MCU dependent properties
- Reset Configuration
- Definition of MCU modes
- Definition of Clock settings
- Definition of RAM sections

⌋()

[SWS_Mcu_00054]: ⌈The structure `Mcu_ConfigType` shall provide a configurable (enable/disable) clock failure notification if the MCU provides an interrupt for such detection.⌋(BSW12207)

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.

[SWS_Mcu_00035]: ⌈The definitions for each MCU mode within the structure `Mcu_ConfigType` shall contain: (depending on MCU)

- MCU specific properties
- Change of CPU clock
- Change of Peripheral clock
- Change of PLL settings
- Change of MCU power supply⌋(BSW12421)

[SWS_Mcu_00031]: ⌈The definitions for each Clock setting within the structure `Mcu_ConfigType` shall contain:

- MCU specific properties as, e.g., clock safety features and special clock distribution settings
- PLL settings /start lock options
- Internal oscillator setting⌋(BSW12207)

[SWS_Mcu_00030]: ⌈The definitions for each RAM section within the structure `Mcu_ConfigType` shall contain:

- RAM section base address
- Section size
- Data pre-setting to be initialized⌋(BSW12350)

Usage of linker symbols instead of scalar values is allowed.

8.2.2 Mcu_PllStatusType

[SWS_Mcu_00250]⌈

Name:	`Mcu_PllStatusType`	
Type:	`Enumeration`	
Range:	`MCU_PLL_LOCKED`	PLL is locked
	`MCU_PLL_UNLOCKED`	PLL is unlocked
	`MCU_PLL_STATUS_UNDEFINED`	PLL Status is unknown
Description:	This is a status value returned by the function Mcu_GetPllStatus of the MCU module.	

⌋()

[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]⌈

Name:	Mcu_ClockType		
Type:	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		

⌋()

[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]⌈

Name:	Mcu_ResetType	
Type:	Enumeration	
Range:	MCU_POWER_ON_RESET	Power On Reset (default)
	MCU_WATCHDOG_RESET	Internal Watchdog Timer Reset
	MCU_SW_RESET	Software Reset
	MCU_RESET_UNDEFINED	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.	

⌋()

[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]⌈

Name:	Mcu_RawResetType		
Type:	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.		

⌋()

[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]⌈

Name:	Mcu_ModeType		
Type:	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.		

⌋()

[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]⌈

Name:	Mcu_RamSectionType		
Type:	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.		

⌋()

[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]⌈

Name:	`Mcu_RamStateType`	
Type:	`Enumeration`	
Range:	`MCU_RAMSTATE_INVALID`	Ram content is not valid or unknown (default).
	`MCU_RAMSTATE_VALID`	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.	

⌋()

8.3 Function definitions

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

8.3.1 Mcu_Init

[SWS_Mcu_00153]:

⌈

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.	

⌋()

[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.⌋(BSW101, BSW00406, BSW12057)

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.⌋(BSW12125, BSW12461)
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.⌋(BSW12125, BSW12461)
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.⌋(BSW12125, BSW12461)
4. **[SWS_Mcu_00246]:** ⌈One-time writable registers that require initialisation directly after reset shall be initialised by the startup code.⌋(BSW12125, BSW12461)
5. **[SWS_Mcu_00247]**: ⌈All other registers not mentioned before shall be initialised by the start-up code.⌋(BSW12125, BSW12461)

[SWS_Mcu_00127]: ⌈If not applicable, the MCU module's environment shall pass a NULL pointer to the function `Mcu_Init`. In this case the check for this NULL pointer has to be omitted.⌋()

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]:

⌈

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.	

⌋( )

[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`, where `McuRamSectionBaseAddress`, `McuRamSectionSize` and `McuRamDefaultValue` are the values of the configuration parameters for each RamSection (see SWS_Mcu_00030).⌋ (BSW12331)

[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]:

⌈

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.	

⌋()

[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.⌋(BSW12208)

[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.⌋(BSW12208)

[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_00182 :**).⌋()

8.3.4 Mcu_DistributePllClock

[SWS_Mcu_00156]:

⌈

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.	

⌋()

[SWS_Mcu_00140]: ⌈The function `Mcu_DistributePllClock` shall activate the PLL clock to the MCU clock distribution.⌋(BSW12336)

[SWS_Mcu_00141]: ⌈The function `Mcu_DistributePllClock` shall remove the current clock source (for example internal oscillator clock) from MCU clock distribution.⌋(BSW12336)

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.⌋(BSW12336)

[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]:

⌈

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.	

⌋()

[SWS_Mcu_00008]: ⌈The function `Mcu_GetPllStatus` shall return the lock status of the PLL.⌋(BSW157, BSW12392)

[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]:

⌈

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.	

⌋()

[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`.⌋(BSW157, BSW12000)

[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]:

⌈

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.	

⌋()

[SWS_Mcu_00135]: ⌈The function `Mcu_GetResetRawValue shall return a`n 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.⌋ (BSW157, BSW12063, BSW12215)

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]:

⌈

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.

⌋()

[SWS_Mcu_00143]: ⌈The function `Mcu_PerformReset` shall perform a microcontroller reset by using the hardware feature of the microcontroller.⌋ (BSW12277)

[SWS_Mcu_00144]: ⌈The function `Mcu_PerformReset` shall perform the reset type which is configured in the configuration set.⌋(BSW12277)

[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]:

⌈

Service name:	Mcu_SetMode	

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

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non 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.	

⌋()

[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]:

⌈

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.	

⌋()

[SWS_Mcu_00204]: ⌈if development error detection is enabled, the parameter `versioninfo` shall be checked for being NULL. The error `MCU_E_PARAM_POINTER` shall be reported in case the value is a NULL pointer.⌋()

8.3.11 Mcu_GetRamState

[SWS_Mcu_00207]:

⌈

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)	

⌋(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.`⌋ (BSW13701)

[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).⌋(BSW13701)

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]:

⌈

API function	***Description***

Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.

⌋()

8.6.2 Optional Interfaces

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

[SWS_Mcu_00163]:

⌈

API function	*Description*
Det_ReportError	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 Development Error Tracer and reject with return value `E_NOT_OK` in case the function has a standard return type.⌋()

[SWS_Mcu_00018]: ⌈If development error detection is enabled, the parameter ConfigPtr shall be checked for being NULL. Related error value: `MCU_E_PARAM_CONFIG.`⌋()

[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

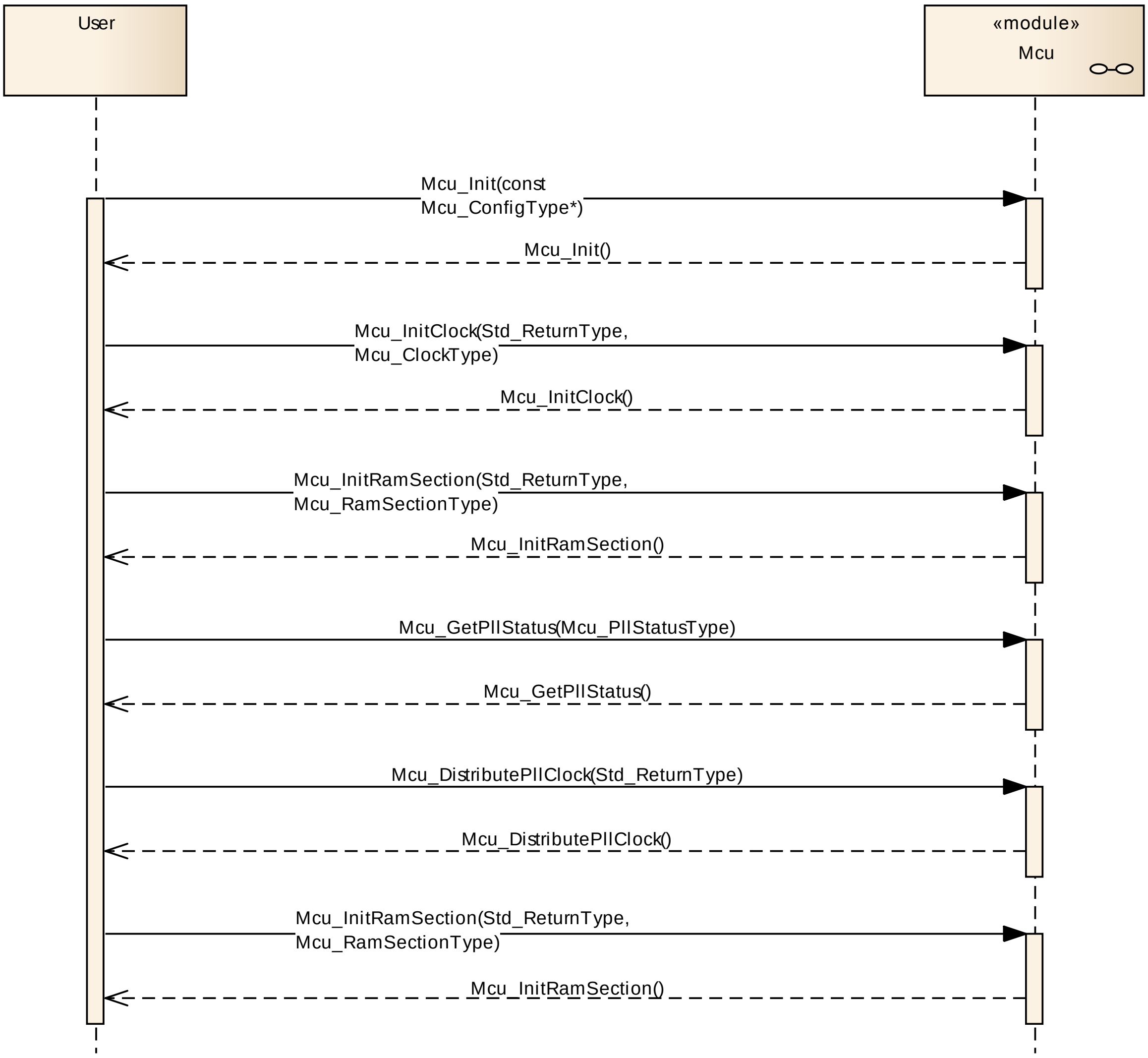

Figure 3: Sequence Diagram – MCU Initialisation

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

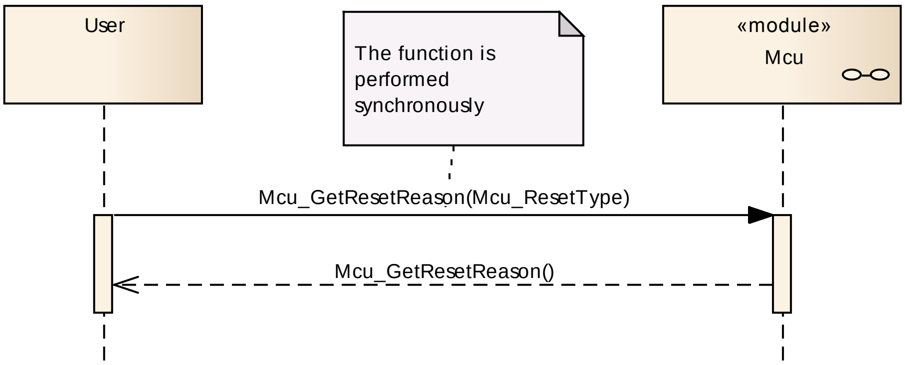
User
The function is performed synchronously
«module»
Mcu
Mcu_GetResetReason(Mcu_ResetType)
Mcu_GetResetReason()

Figure 7: Sequence Diagram – MCU_GetResetReason

9.3 Mcu_GetResetRawValue

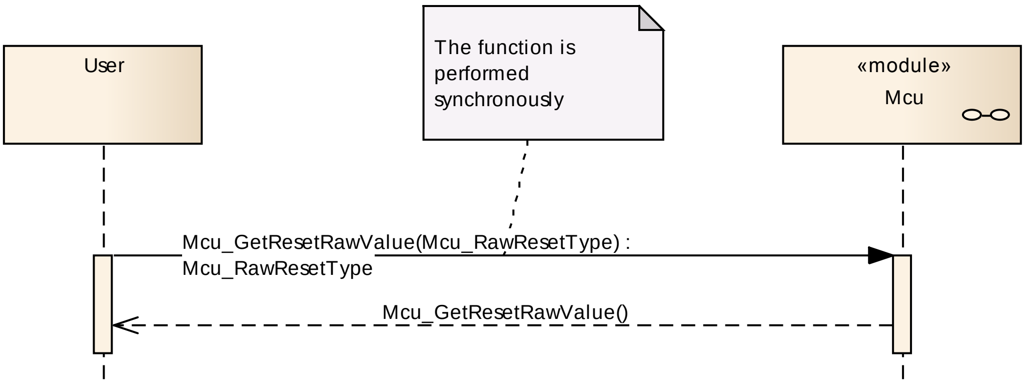
User
The function is performed synchronously
«module»
Mcu
Mcu_GetResetRawValue(Mcu_RawResetType) : Mcu_RawResetType
Mcu_GetResetRawValue()

Figure 8: Sequence Diagram – Mcu_GetResetRawValue

9.4 Mcu_PerformReset

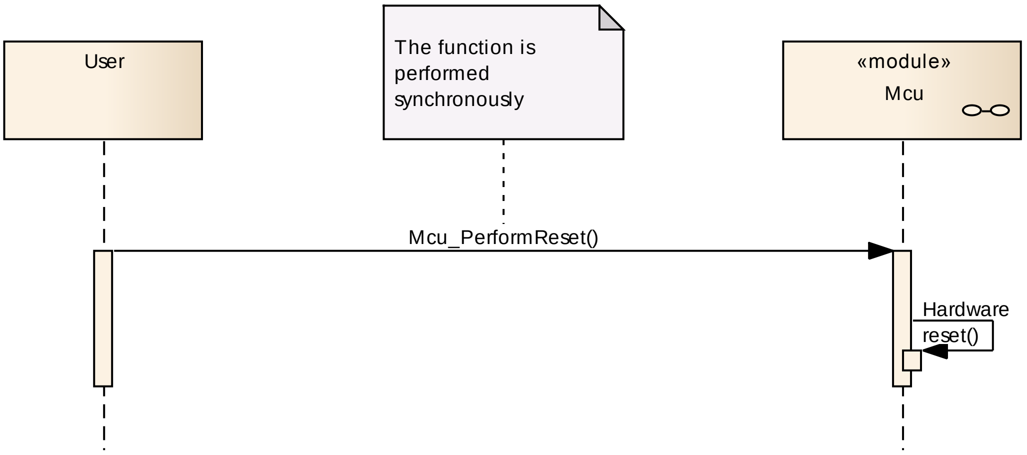

Figure 9: Sequence Diagram – Mcu_PerformReset

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1describes 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.

10.2.1 Variants

[SWS_Mcu_00129]: ⌈VARIANT-PRE-COMPILE.
Only parameters with "Pre-compile time" configuration are allowed in this variant.
The intention of this variant is to optimize the parameters configuration for a source code delivery.⌋()

[SWS_Mcu_00130]: ⌈VARIANT-POST-BUILD.
Parameters with "Pre-compile time", "Link time" and "Post-build time" are
allowed in this variant.
The intention of this variant is to optimize the parameters configuration for a re-loadable binary.⌋()

[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.⌋()

10.2.2 Mcu

Module Name	*Mcu*
Module Description	Configuration of the Mcu (Microcontroller Unit) module.

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

10.2.3 McuGeneralConfiguration

SWS Item	**ECUC_Mcu_00118 :**
Container Name	McuGeneralConfiguration{MCU General Configuration}
Description	This container contains the configuration (parameters) of the MCU driver.
Configuration Parameters	

SWS Item	**ECUC_Mcu_00166 :**		
Name	McuDevErrorDetect {MCU_DEV_ERROR_DETECT}		
Description	Pre-processor switch for enabling the development error detection and reporting.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00181 :**		
Name	McuGetRamStateApi {MCU_ GET_RAM_STATE_API}		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00182 :**
Name	McuInitClock {MCU_INIT_CLOCK}
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		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00180 :**		
Name	McuNoPll {MCU_NO_PLL}		
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_DistributePllClock 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		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00167 :**		
Name	McuPerformResetApi {MCU_PERFORM_RESET_API}		
Description	Pre-processor switch to enable / disable the use of the function Mcu_PerformReset()		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00168 :**		
Name	McuVersionInfoApi {MCU_VERSION_INFO_API}		
Description	Pre-processor switch to enable / disable the API to read out the modules version information.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.4 McuModuleConfiguration

SWS Item	**ECUC_Mcu_00119 :**
Container Name	McuModuleConfiguration{MCU Module Configuration} [Multi Config Container]
Description	This container contains the configuration (parameters) of the MCU driver
Configuration Parameters	

SWS Item	**ECUC_Mcu_00170 :**		
Name	McuClockSrcFailureNotification {MCU_CLOCK_SOURCE_FAILURE_NOTIFICATION}		
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	--	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00171 :**		
Name	McuNumberOfMcuModes {Mcu_Number_Of_Modes}		
Description	This parameter shall represent the number of Modes available for the MCU. calculationFormula = Number of configured McuModeSettingConf		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00172 :**		
Name	McuRamSectors {MCU_RAM_SECTORS}		
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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00173 :**		
Name	McuResetSetting {MCU_RESET_SETTING}		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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.
McuDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic name. The standardized errors are provided in the 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.

10.2.5 McuClockSettingConfig

SWS Item	**ECUC_Mcu_00124 :**
Container Name	McuClockSettingConfig{MCU Clock Setting Configuration}
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	

SWS Item	**ECUC_Mcu_00183 :**		
Name	McuClockSettingId {MCU_MODE_NORMAL}		
Description	The Id of this McuClockSettingConfig to be used as argument for the API call "Mcu_InitClock".		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

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.

10.2.6 McuDemEventParameterRefs

SWS Item	**ECUC_Mcu_00187 :**

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

SWS Item	**ECUC_Mcu_00188 :**		
Name	MCU_E_CLOCK_FAILURE {MCU_E_CLOCK_FAILURE}		
Description	Reference to configured DEM event to report "Clock source failure".		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: Dem		

No Included Containers

10.2.7 McuModeSettingConf

SWS Item	**ECUC_Mcu_00123 :**
Container Name	McuModeSettingConf{MCU Mode Setting Configuration}
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	

SWS Item	**ECUC_Mcu_00176 :**		
Name	McuMode {MCU_MODE_NORMAL}		
Description	The parameter represents the MCU Mode settings.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.8 McuRamSectorSettingConf

SWS Item	**ECUC_Mcu_00120 :**
Container Name	McuRamSectorSettingConf{MCU RAM Sector Setting Configuration}
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	

SWS Item	**ECUC_Mcu_00177 :**		
Name	McuRamDefaultValue {MCU_RAM_DEFAULT_VALUE}		
Description	This parameter shall represent the Data pre-setting to be initialized		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00178 :**		
Name	McuRamSectionBaseAddress {MCU_RAM_SECTION_BASE_ADDRESS}		
Description	This parameter shall represent the MCU RAM section base address		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Mcu_00179 :**		
Name	McuRamSectionSize {MCU_RAM_SECTION_SIZE}		
Description	This parameter represents the MCU RAM Section size in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.9 McuClockReferencePoint

SWS Item	**ECUC_Mcu_00174 :**
Container Name	McuClockReferencePoint
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	

SWS Item	**ECUC_Mcu_00175 :**
Name	McuClockReferencePointFrequency
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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.10 McuPublishedInformation

SWS Item	**ECUC_Mcu_00184 :**
Container Name	McuPublishedInformation
Description	Container holding all MCU specific published information parameters
Configuration 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.

10.2.11 McuResetReasonConf

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

SWS Item	**ECUC_Mcu_00186 :**		
Name	McuResetReason {MCU_POWER_ON_RESET}		
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	--		
ConfigurationClass	***Published Information***	X	All Variants
Scope / Dependency	scope: ECU		

No Included Containers

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*