Document Title	Specification of MCU Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	031
Document Classification	Standard

Document Version	2.2.0
Document Status	Draft
Part of Release	2.1
Revision	20

Document Change History

Date	Version	Changed by	Change Description
31.05.2010	2.2.0	AUTOSAR Administration	• Reverted Mcu_InitClock() behaviour back to AUTOSAR v2.0, i.e. allow for multiplicity 1..* of MCU Clock setting • Legal disclaimer revised
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 MCU_PERFORM_RESET_API. • Section 10.2.2: Introduction of pre-processor switch MCU_PERFORM_RESET_API • 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

Release Notes

Compatibility considerations with respect to current release

The changes to the MCU Driver specification from Release 2.0 are as follows:

- In section 8.2.5, description detail updated to correct section reference to include Mcu_GetResetRawValue().
- Removal of ‘const’ from API type definition in sections 8.3.2, 8.3.3, 8.3.9.
- MCU116: Definition added for term ‘Hardware Module’
- Section 8.2.5: New wording added to description detail for Mcu_RawResetType.
- Section 10.2.5: Description detail changed for the configuration parameter MCU_MODE.
- Section 8.2.4: New wording update to description detail for Mcu_ResetType.
- Section 10.2.2: Introduction of pre-processor switch MCU_PERFORM_RESET_API to enable / disable the use of the function Mcu_PerformReset().
- Section 8.3.8: Description detail updated to include reference to new pre-processor switch MCU_PERFORM_RESET_API.
- Section8.2.4: Element MCU_ POWER_ON_RESET default value of 0x0 removed.
- Section 10.2.3: Multiplicity of sub-container Mcu Clock Setting Configuration changed to 1 (from 1..*).
- Section 5.2.2: Inclusion of new file structure information.

Errata and known deficiencies

The wakeup concept is currently neither harmonized nor consistent throughout all wakeup related specification documents and therefore subject to change.
Affected sections in this document are:

- Section 8.4: Wake-up reason reporting
- Section 9.2: Wake-up sequence diagrams.
- Section 10.2: Wake-up reporting configurable parameters

Known and potential problems resulting from known deficiencies

Due to the fact that the wakeup concept is not harmonized, inconsistent assumptions may lead to

- the duplication of functionalities across multiple modules
- proprietary implementation extensions
- difficulties during integration

In addition, please refer to list of outstanding issues below (Changes planned for next release)

Changes planned for next release

The following changes are planned for the next release:

The harmonized wakeup concept throughout all wakeup related documents will result in

- adapted specification texts in Chapter 7 of the specification documents
- adapted APIs in Chapter 8 of the specification documents
- adapted wakeup sequences in Chapter 9 of the specification documents

Outstanding issues with the MCU module which may need introduction to future releases of the MCU Driver specification document are:

- Missing SRS General requirements in Chapter 6:
  - In accordance with AUTOSAR_SWS_Template inclusion of the following requirements to Section 6 should be added:
  - [BSW00435] Header file Structure for the Basic Software Scheduler
  - [BSW00436] Module Header File Structure for the Basic Software Memory Mapping
- Section 8.7.2: Removal of obsolete function Dem_ReportErrorEvent required
- Section 5.2.2: Inclusion of EcuM.h in file structure information

Disclaimer

Advice for users

AUTOSAR Specification Documents 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 Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, 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 from 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 have to be taken into account before standardized MCU initialization is able to start.

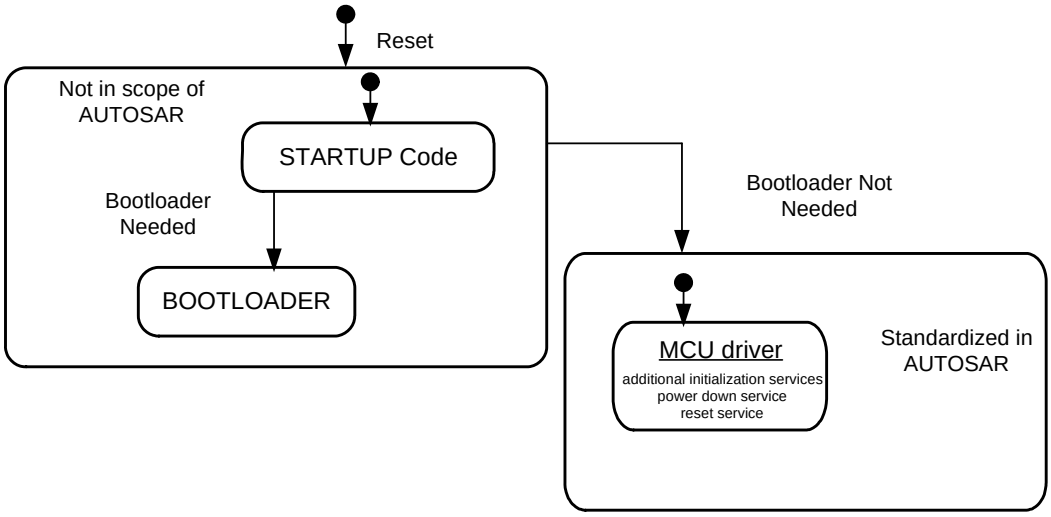

Figure 1: Scope of the MCU Driver Specification

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_BasicSoftwareModules.pdf

[2] Layered Software Architecture,
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules,
AUTOSAR_SRS_General.pdf

[4] Specification of Development Error Tracer,
AUTOSAR_SWS_Development_Error_Tracer.pdf

[5] Specification of ECU Configuration,
AUTOSAR_ECU_Configuration.pdf

[6] Specification of Diagnostic Event Manager,
AUTOSAR_SWS_DEM.pdf

[7] Specification of ECU State Manager,
AUTOSAR_SWS_ECUStateManager.pdf

[8] General Requirements on SPAL,
AUTOSAR_SRS_General.pdf

[9] Requirements on MCU driver,
AUTOSAR_SRS_MCU_Driver.pdf

[10] Specification of Standard Types,
AUTOSAR_SWS_StandardTypes.pdf

3.2 Related standards and norms

[11] IEC 7498-1 The Basic Model, IEC Norm, 1994

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 will 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 is 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 basic functionality which shall 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 of internal memory like 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 MCAL.

5.2 File structure

5.2.1 Code file structure

MCU105: The code file structure shall not be defined within this specification.

5.2.2 Header file structure

MCU108: The include file structure shall be as follows:

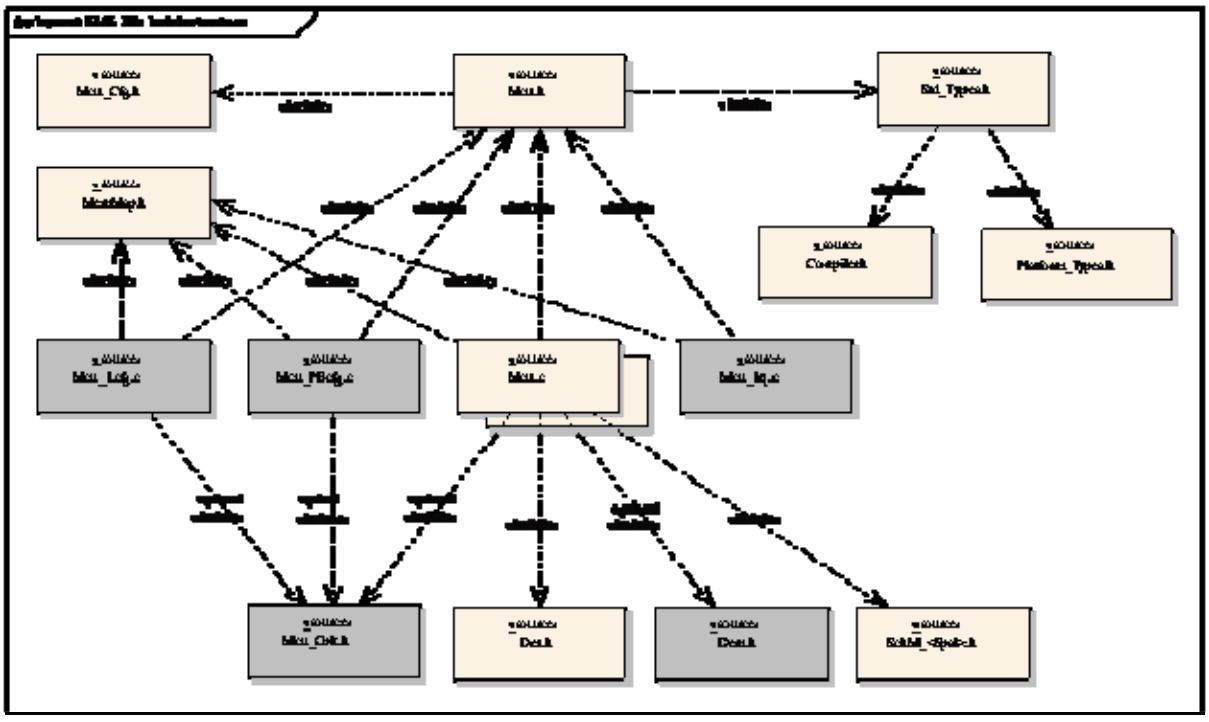

Figure 2: Header File Structure

- Mcu.c shall include Mcu.h
- Mcu.h shall include Mcu_Cfg.h for the API pre-compiler switches.
- Mcu_Xcfg.h, where X is a placeholder for 'L' or 'PB'

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

Mcu_Irq.c shall include Mcu.h for the function which shall be called in the interrupt function.

The Type definitions for Mcu_Lcfg.c and Mcu_PBcfg.c are located in the file Mcu_Cfg.h. or Mcu.h.

The implicitly 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
----------
#ifdef xxx_VERSION_INFO_API
xxx_GetVersionInfo(...)
#endif

Mcu_Cfg.h
---------------
#include "Mcu.h"
#define xxx_VERSION_INFO_API
```

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.

Mcu_PBcfg.c shall include Mcu_Cbk.h for a post build time configuration, if the call back function is linked to the module via the ROM structure.

Mcu.c shall include Mcu_Cbk.h for a pre-compile time configuration

MCU109: The module shall include the Dem.h file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

6 Requirements traceability

Document: AUTOSAR requirements on Basic Software, general

Requirement	***Satisfied by***
[BSW003] Version identification	MCU121, MCU037
[BSW004] Version check	MCU110
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (Because the architectural AUTOSAR concept is the basis for this concept).
[BSW006] Platform independency	Not applicable (Because the MCU module is not above the MCAL).
[BSW007] HIS MISRA C	Not applicable (Because this is a requirement for implementation).
[BSW009] Module User Documentation	See Section 3.1
[BSW010] Memory resource documentation	Not applicable (Because this is a requirement for implementation).
[BSW101] Initialization interface	MCU026
[BSW158] Separation of configuration from implementation	See Section 5.2
[BSW159] Tool-based configuration	See Section 5.2
[BSW160] Human-readable configuration data	Not applicable (Because this requirements only applies to the configuration. Requirement on implementation).
[BSW161] Microcontroller abstraction	Not applicable (Because the architectural AUTOSAR concept is the basis for this concept).
[BSW162] ECU layout abstraction	Not applicable (Because the architectural AUTOSAR concept is the basis for this concept).
[BSW164] Implementation of interrupt service routines	Not applicable (Because the MCU does not provide interrupt functionality).
[BSW167] Static configuration checking	Not Applicable (Because this is a requirement for the configuration tool).
[BSW168] Diagnostic Interface of SW components	Not applicable (Because this is a requirement on SW components).
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (Because this is a requirement on SW components).
[BSW171] Configurability of optional functionality	MCU119, MCU120
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (Because the MCU does not have any special scheduling requirements).
[BSW00300] Module naming convention	See Section 5.2
[BSW00301] Limit imported information	See Section 5.2

[BSW00302] Limit exported information	See Section 5.2
[BSW00304] AUTOSAR integer data types	See Section 8.2
[BSW00305] Self-defined data types naming convention	See Section 8.2
[BSW00306] Avoid direct use of compiler and platform specific keywords	See Figure 5.2
[BSW00307] Global variables naming convention	Not applicable (Because this is a requirement on implementation).
[BSW00308] Definition of global data	Not applicable (Because this is a requirement for the implementer).
[BSW00309] Global data with read-only constraint	See Section 8.3.1
[BSW00310] API naming convention	See Section 8.3
[BSW00312] Shared code shall be reentrant	See Section 8.3
[BSW00314] Separation of interrupt frames and service routines	See Section 5.2
[BSW00318] Format of module version numbers	MCU121, MCU037
[BSW00321] Enumeration of module version numbers	Not Applicable (Because this is a requirement for the implementer).
[BSW00323] API parameter checking	MCU017
[BSW00325] Runtime of interrupt service routines	Not applicable (Because this is not a requirement of the MCU driver. Requirement for the implementer).
[BSW00326] Transition from ISRs to OS tasks	Not applicable (Because this is a requirement for the implementer).
[BSW00327] Error values naming convention	MCU012
[BSW00328] Avoid duplication of code	Not applicable (Because this is a requirement for the implementer).
[BSW00329] Avoidance of generic interfaces	Not applicable (Because there are no generic interfaces specified within this SWS).
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (Because this is a requirement for the implementer).
[BSW00331] Separation of error and status values	Not applicable (Because there are no status values specified within this SWS).
[BSW00333] Documentation of callback function context	Not applicable (Because there is no callback functionality in the MCU driver).
[BSW00334] Provision of XML file	Not applicable (Because this requirement is specified by another document).
[BSW00335] Status values naming convention	Not applicable (Because there are no status values specified within the SWS).
[BSW00336] Shutdown interface	Not applicable (Because for the MCU driver there is no need for this requirement).

[BSW00337] Classification of errors	MCU012, MCU015
[BSW00338] Detection and Reporting of development errors	MCU013
[BSW00339] Reporting of production relevant error status	MCU014
[BSW00341] Microcontroller compatibility documentation	Not applicable (Because this is a requirement for the implementer).
[BSW00342] Usage of source code and object code	Not applicable (Because this is a requirement for the implementer).
[BSW00343] Specification and configuration of time	Not applicable (Because time configuration is not a requirement of the MCU driver).
[BSW00344] Reference to Link-time configuration	MCU119
[BSW00345] Pre-compile-time configuration	See Section 5.2.2, MCU119
[BSW00346] Basic set of module files	See Section 5.2.2
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable (Because this is a requirement on implementation).
[BSW00348] Standard type header	See Section 5.2.2
[BSW00350] Development error detection keyword	MCU118
[BSW00353] Platform specific type header	See Section 5.2.2
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (Because there is no integer data types redefined in this specification).
[BSW00357] Standard API return type	Not applicable (Because this type is not used within the SWS).
[BSW00358] Return type of init() functions	See Section 8.3.1
[BSW00359] Return type of callback functions	Not applicable (Because the MCU driver does not provide a callback mechanism).
[BSW00360] Parameters of callback functions	Not applicable (Because the MCU driver does not provide a callback mechanism).
[BSW00361] Compiler specific language extension header	See Section 5.2.2
[BSW00369] Do not return development error codes via API	MCU013, MCU015
[BSW00370] Separation of callback interface from API	Not applicable (Because the MCU driver does not provide a callback mechanism).
[BSW00371] Do not pass function pointers via API	Not applicable (Because no function pointers are passed via API in this SWS).
[BSW00373] Main processing function naming convention	Not applicable (Because there is no main processing function is specified).
[BSW00374] Module vendor identification	MCU121, MCU037
[BSW00375] Notification of wake-up reason	MCU057, MCU058, MCU059, MCU060

[BSW00376] Return type and parameters of main processing functions	Not applicable (Because there is no main processing function specified).
[BSW00377] Module specific API return types	Not applicable (Because this type is not used within the SWS).
[BSW00378] AUTOSAR boolean type	See Section 10.2.2
[BSW00379] Module identification	MCU121
[BSW00380] Separate C-File for configuration parameters	See Section 5.2.2
[BSW00381] Separate configuration header file for pre-compile time parameters	See Section 5.2.2
[BSW00382] Not-used configuration elements need to be listed	Not applicable (Because this is an implementation requirement).
[BSW00383] List dependencies of configuration files	See Section 5.2.2
[BSW00384] List dependencies to other modules	See Section 5.2.2
[BSW00385] List possible error notifications	See Section 7.4
[BSW00386] Configuration for detecting an error	See Section 7.2.2
[BSW00387] Specify the configuration class of callback function	Not applicable (Because the MCU driver does not have any callback capability).
[BSW00388] Introduce containers	See Section 10.2.2
[BSW00389] Containers shall have names	See Section 10.2.2
[BSW00390] Parameter content shall be unique within the module	See Chapter 8.3
[BSW00391] Parameter shall have unique names	See Chapter 8.3
[BSW00392] Parameters shall have a type	See Chapter 8.3
[BSW00393] Parameters shall have a range	See Chapter 8.3
[BSW00394] Specify the scope of the parameters	See Chapter 8.3
[BSW00395] List the required parameters (per parameter)	Not applicable (Because none of the parameters of the MCU driver are dependent on other modules).
[BSW00396] Configuration classes	See Chapter 10.2.2
[BSW00397] Pre-compile-time parameters	See Chapter 10.2.2
[BSW00398] Link-time parameters	See Chapter 10.2.2
[BSW00399] Loadable Post-build time parameters	See Chapter 10.2.2
[BSW00400] Selectable Post-build time parameters	See Chapter 10.2.2
[BSW00401] Documentation of multiple instances of configuration parameters	See Chapter 10.2.2
[BSW00402] Published information	See Chapter 10.3
[BSW00404] Reference to post build time configuration	See Chapter 10.2.2
[BSW00405] Reference to multiple configuration sets	See Chapter 10.2.2
[BSW00406] Check module initialization	MCU026
[BSW00407] Function to read out published parameters	MCU103
[BSW00408] Configuration parameter naming convention	See Chapter 10.2.2
[BSW00409] Header files for production code error IDs	See Section 5.2.2
[BSW00410] Compiler switches shall have defined values	See Chapter 10.2.2
[BSW00411] Get version info keyword	MCU103, MCU104
[BSW00412] Separate H-File for configuration parameters	See Section 5.2.2
[BSW00413] Accessing instances of BSW modules	Not applicable (Because this is a requirement on implementation).
[BSW00414] Parameter of init function	MCU126
[BSW00415] User dependent include files	See Section 5.2.2
[BSW00416] Sequence of Initialization	Not applicable (Because this requirement describes the initialization of the whole SPAL layer).
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (Because this driver is part of the basic software

	layer, applies only for non-BSW modules).
[BSW00419] Separate C-Files for pre-compile time configuration parameters	See Section 5.2.2
[BSW00420] Production relevant error event rate detection	Not applicable (Because this requirement applies only to DEM).
[BSW00421] Reporting of production relevant error events	MCU014
[BSW00422] Debouncing of production relevant error status	MCU014
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (Because it is a non functional requirement. The MCU driver has no AUTOSAR interface).
[BSW00424] BSW main processing function task allocation	Not applicable (Because the MCU driver does not contain any main processing functions).
[BSW00425] Trigger conditions for schedulable objects	Not applicable (Because the MCU driver does not contain any schedulable objects/services. This is a requirement for the implementer).
[BSW00426] Exclusive areas in BSW modules	Not applicable (Because this requirement applies only for the module descriptions template).
[BSW00427] ISR description for BSW modules	Not applicable (Because this is a requirement for the implementer).
[BSW00428] Execution order dependencies of main processing functions	Not applicable (Because the MCU driver does not contain any main processing functions).
[BSW00429] Restricted BSW OS functionality access	Not applicable (Because the MCU driver is not dependent on the OS driver).
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (Because this requirement is for an upper layer. There is no scheduling functionality in the MCU driver).
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (Because the MCU driver does not contain any main processing functions).
[BSW00433] Calling of main processing functions	Not applicable (Because the MCU driver does not contain any main processing functions).
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (Because it is a non-functional requirement. There is no scheduling functionality in the MCU driver)

Document: AUTOSAR requirements on Basic Software, cluster SPAL

Requirement	***Satisfied by***
[BSW12263] Object code compatible configuration concept	SeeSection 10.2.2
[BSW12056] Configuration of notification mechanisms	SeeSection 10.2.2
[BSW12267] Configuration of wake-up sources.	SeeSection 10.2.2
[BSW12057] Driver module initialization	MCU026
[BSW12125] Initialization of hardware resources	MCU116
[BSW12163] Driver module deinitialization	Not applicable (Because the MCU driver cannot be de-initialized).

[BSW12068] MCAL initialization sequence	Not applicable (Because this is a general software integration requirement).
[BSW12069] Wake-up notification of ECU State Manager	MCU057
[BSW157] Notification mechanisms of drivers and handlers	MCU008, MCU005, MCU006, MCU012
[BSW12155] Prototypes of callback functions	Not applicable (Because there is no callback functionality in the MCU driver).
[BSW12169] Control of operation mode	Not applicable (Because there are no special operation modes defined for the MCU driver. The specified MCU mode interface (see MCU001) is able to change operation modes of the whole MCU, which is not meant with this requirement).
[BSW12063] Raw value mode	MCU006
[BSW12075] Use of application buffers	Not applicable (Because there is no random streaming capability).
[BSW12129] Resetting of interrupt flags	Not applicable (Because the interrupt functionality not part of the MCU driver)
[BSW12064] Change of operation mode during running operation	Not applicable (Because this is a non-functional requirement concerning system design).
[BSW12067] Setting of wake-up conditions	See Section 8.4
[BSW12448] Behaviour after development error detection	MCU013
[BSW12077] Non-blocking implementation	Not Applicable (Because this is a requirement for the implementer).
[BSW12078] Runtime and memory efficiency	Not Applicable (Because this is a requirement for the implementer).
[BSW12092] Access to drivers	Not Applicable (Because this is a requirement concerning system design).
[BSW12265] Configuration data shall be kept constant	Not Applicable (Because this is a requirement for the implementer).
[BSW12264] Specification of configuration items	See Section 10.2.2
[BSW12350] Configuration of RAM segments	MCU030
[BSW12331] RAM Initialization	MCU011
[BSW12392] Provide lock status of PLL	MCU008
[BSW12336] Activate PLL Clock distribution	MCU028, MCU056
[BSW12207] Configuration of clock safety features	MCU031, MCU054
[BSW12208] Initialization of MCU Clock	MCU009,
[BSW12394] Fault condition handling of clock safety features	MCU012, MCU053;
[BSW12000] Provide standardized reset reason	MCU005, MCU052
[BSW12215] Provide raw reset status	MCU006
[BSW12277] Reset trigger function	MCU007, MCU055
[BSW12268] MCU Power Management Control	MCU001
[BSW12421] Low Power Mode Configuration	MCU035
[BSW12461] Responsibility for register initialisation	MCU116
[BSW12462] Provide settings for register initialisation.	See Section 10.3
[BSW12463] Combine and forward settings for register initialisation.	Not applicable (Because this is a requirement for a configuration tool).

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

MCU055: The service to provide software triggering of a hardware reset is provided by the MCU driver. A call to this reset service function requires an authorized user.

The authorization of the user shall not be checked in the MCU driver. This shall be the responsibility of the upper layer or the software design.

MCU052: 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. Therefore the MCU driver provides services to get the reset reason of the last reset occurred, if the hardware supports such a feature.

7.1.2.2 MCU Mode service

MCU001: The MCU driver provides a service to activate MCU reduced power modes. It allows access to power modes implemented in the uC hardware. The number of modes and the configuration is MCU dependent and shall be configured in the configuration set of the MCU driver. 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.

7.1.3 Version Check

7.1.3.1 Background and Rationale

The integration of incompatible files shall be avoided. Minimum implementation is the version check of the header file inside the C file (version numbers of C and H files shall be identical).

7.1.3.2 Requirements

MCU110: The integration of incompatible files shall be avoided. Minimum implementation is the version check of the header file.
For included header files:

- MCU_AR_MAJOR_VERSION
- MCU_AR_MINOR_VERSION

shall be identical.

For the module internal c and h files:

- MCU_SW_MAJOR_VERSION
- MCU_SW_MINOR_VERSION
- MCU_AR_MAJOR_VERSION
- MCU_AR_MINOR_VERSION
- MCU_AR_PATCH_VERSION

shall be identical.

7.2 Error classification

7.2.1 Background and Rationale

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

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

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

7.2.2 Requirements

MCU111: Values for production code Event IDs are assigned externally by the configuration of the DEM. They are published in the file Dem_IntErrId.h and included via Dem.h.

MCU112: Development error values are of type uint8.

MCU012: The following errors and exceptions shall be detectable by the MCU driver 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	0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
Clock source failure	Production	MCU_E_CLOCK_FAILURE	Assigned by DEM

Table 2: Error Classification

MCU053: MCU_E_CLOCK_FAILURE is activated if clock failure notification is enabled in the configuration set (see also MCU051).

7.3 Error detection

MCU100: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time.
The switch MCU_DEV_ERROR_DETECT (see chapter 10) shall activate or deactivate the detection of all development errors.

MCU101: If the MCU_DEV_ERROR_DETECT switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.2 and chapter 8.

MCU102: The detection of production code errors cannot be switched off.

7.4 Error notification

MCU016: The MCU driver follows the standardized AUTOSAR concept to report development errors. The provided routines are specified in the Development Error Tracer (DET) specification.

MCU013: The detection of all development errors shall be configurable (on/off) with the pre-processor switch MCU_DEV_ERROR_DETECT. Detected development errors shall be reported to the error hook of the Development Error Tracer (see 5) if the pre-processor switch MCU_DEV_ERROR_DETECT is set.

MCU051: 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).

MCU0149: Production relevant errors shall be reported to the Diagnostic Event Manager (DEM). They shall not be used as the return value of the called function.

MCU015: Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the MCU implementation specification. The classification and enumeration shall be compatible to the errors listed above (see **MCU012**).

8 API specification

8.1 Imported types

8.1.1 Standard types

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

- Std_Types.h
- Std_ReturnType
- Std_VersionInfoType

8.1.2 Other Module Types

- EcuM_WakeUpSourceType

8.2 Type definitions

8.2.1 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. This type of the external data structure shall contain the initialization data for the MCU Driver. It shall contain: • MCU dependent properties • Reset Configuration • Definition of MCU modes • Definition of Clock settings • Definition of RAM sections **MCU054:** A clock failure notification shall be configurable (enable/disable), if the MCU provides an interrupt for such detection. Error reporting follows the DEM procedures (see also **MCU051**). In case of other HW detection mechanisms like the generation of a trap, this notification shall be disabled and the failure reporting is done outside the MCU driver. **MCU060:** Configuration for possible wake-up reasons. This configuration is taken into account, if MCU_REPORT_WAKEUP_REASON is set. (see also **MCU057, MCU058, MCU059)** **MCU035:** The definition for each MCU mode 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 **MCU031:** The definition for each Clock setting shall contain: • MCU specific properties like clock safety features and special clock

	distribution settings • PLL settings /start lock options • Internal oscillator setting **MCU030:** The definition for each RAM section shall contain: • RAM section base address • Section size • Data pre-setting to be initialized • Usage of linker symbols is allowed instead of scalar values.

8.2.2 Mcu_PllStatusType

Type:	`enumeration`	
Ranges:	`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 API service `Mcu_GetPllStatus()` of MCU driver. MCU_PLL_STATUS_UNDEFINED is returned if the function `Mcu_GetPllStatus()` is called prior to calling the `Mcu_Init()` function.	

8.2.3 Mcu_ClockType

Type:	`uint8...uint32`	
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 performing
Description:	Specifies the identification (ID) for a clock setting, which is configured in the configuration structure	

8.2.4 Mcu_ResetType

Type:	`enumeration`	
Ranges:	`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
		Additional reset types may be added depending on MCU.
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. For implementation at least the MCU_POWER_ON_RESET and UNDEFINED_RESET should be included. Where applicable (for some but not all MCU's) MCU_RESET_UNDEFINED is returned if the function `Mcu_GetResetReason()` is called prior to calling `Mcu_Init()`.	

8.2.5 Mcu_RawResetType

Type:	`uint8...uint32`	
Range:	MCU dependent register value	The type shall be chosen depending on MCU platform for best performing
Description:	This type specifies the reset reason in raw register format read from a reset status register. If hardware does not support a reset status register, the value shall be “0”. In the case where the function `Mcu_GetResetRawValue()` is called prior to the function `Mcu_Init()`, an implementation specific value, which does not correspond to a valid value of the reset status register and is not equal to 0, is returned. This is applicable to some but not all MCU’s.	

8.2.6 Mcu_ModeType

Type:	`uint8...uint32`	
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 performing
Description:	This type specifies the identification (ID) for a MCU mode, which is configured in the configuration structure	

8.2.7 Mcu_RamSectionType

Type:	`uint8...uint32`	
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 performing
Description:	This type specifies the identification (ID) for a RAM section, which is configured in the configuration structure	

8.3 Function definitions

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

8.3.1 Mcu_Init

Service name:	Mcu_Init
Syntax:	`void Mcu_Init ( const Mcu_ConfigType *ConfigPtr )`
Service ID hex:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	`ConfigPtr` Pointer to MCU driver configuration set.
Parameters (out):	None
Return value:	None
Description:	**MCU026:** This routine initializes the MCU driver. The intention of this function is to make the configuration settings for power down, clock and RAM sections visible within the MCU driver. After execution of `Mcu_Init()` the configuration data are accessible and can be used from the Mcu services like `Mcu_InitRamSection()`. MCU116: The following rules regarding initialisation of controller registers shall apply to the MCU driver implementation: 1. If the hardware allows for only one usage of the register, the driver modules implementing that functionality is responsible for initializing the register. 2. 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. 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. One-time writable registers that require initialisation directly after reset shall be initialised by the startup code. 5. All other registers shall be initialised by the start-up code. NOTE: The term 'Hardware Module' refers to internal modules of the MCU and not a BSW module.
Caveats:	None
Configuration:	None

8.3.2 Mcu_InitRamSection

Service name:	Mcu_InitRamSection
Syntax:	`Std_ReturnType Mcu_InitRamSection ( Mcu_RamSectionType RamSection )`
Service ID:	0x01
Sync/Async:	Synchronous
Reentrancy:	Non re-entrant
Parameters (in):	RamSection Selects RAM memory section provided in configuration

		set
Parameters (out):	None	
Return value:	`E_OK`:	command has been accepted
	`E_NOT_OK`	command has not been accepted e.g. due to parameter error
Description:	**MCU011:** This function initializes the RAM section wise. The definition of the section and the initialization value is provided from the configuration structure (see **MCU030**) .	
Caveats:	This function requires an execution of `Mcu_Init()` before it can be used.	
Configuration:	None	

8.3.3 Mcu_InitClock

Service name:	Mcu_InitClock	
Syntax:	`Std_ReturnType Mcu_InitClock ( Mcu_ClockType ClockSetting )`	
Service ID:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non reentrant	
Parameters (in):	ClockSetting	Clock setting
Parameters (out):	None	
Return value:	`E_OK`:	Command has been accepted
	`E_NOT_OK:`	Command has not been accepted
Description:	**MCU009:** This function initializes the PLL and other MCU specific clock options. The clock configuration parameters are provided via the configuration structure. In this function the PLL lock procedure is started (if PLL shall be initialized) but the function does not wait until the PLL is locked.	
Caveats:	This function requires an execution of `Mcu_Init()` before it can be used.	
Configuration:	None	

8.3.4 Mcu_DistributePllClock

Service name:	Mcu_DistributePllClock
Syntax:	`void Mcu_DistributePllClock ( void )`
Service ID:	0x03
Sync/Async:	Synchronous
Reentrancy:	Non reentrant
Parameters (in):	None
Parameters (out):	None
Return value:	None
Description:	**MCU028:** This function activates the PLL clock to the MCU clock distribution. This function shall be executed, if the MCU needs a separate request to activate the PLL clock after the PLL is locked. In this case the current clock source (for example internal oscillator clock) is removed from MCU clock distribution.

	MCU056: If the PLL clock is automatically activated by the MCU hardware, this function shall return without affecting the MCU hardware.
Caveats:	This function has to be called only after the status of the PLL is detected as locked with Mcu_GetPllStatus().
Configuration:	None

8.3.5 Mcu_GetPllStatus

Service name:	Mcu_GetPllStatus
Syntax:	`Mcu_PllStatusType Mcu_GetPllStatus ( void )`
Service ID:	0x04
Sync/Async:	Synchronous
Reentrancy:	reentrant
Parameters (in):	None
Parameters (out):	None
Return value:	Mcu_PllStatusType
Description:	**MCU008:** This function provides the lock status of the PLL.
Caveats:	None
Configuration:	None

8.3.6 Mcu_GetResetReason

Service name:	Mcu_GetResetReason
Syntax:	`Mcu_ResetType Mcu_GetResetReason ( void )`
Service ID:	0x05
Sync/Async:	Synchronous
Reentrancy:	reentrant
Parameters (in):	None
Parameters (out):	None
Return value:	Mcu_ResetType
Description:	**MCU005:** The function reads the reset type from the hardware, if supported. If the hardware doesn't support the hardware detection of the reset reason, the return value for those resets is always MCU_POWER_ON_RESET.
Caveats:	None
Configuration:	None

8.3.7 Mcu_GetResetRawValue

Service name:	Mcu_GetResetRawValue
Syntax:	`Mcu_RawResetType Mcu_GetResetRawValue ( void`

	)
Service ID:	0x06
Sync/Async:	Synchronous
Reentrancy:	reentrant
Parameters (in):	None
Parameters (out):	None
Return value:	Mcu_RawResetType
Description:	**MCU006:** The function reads the reset type from the hardware register, if supported. If the hardware doesn't have a reset status register, the return value shall be 0x0.
Caveats:	A call to the `Mcu_Init()` function is required before executing this function.
Configuration:	None

8.3.8 Mcu_PerformReset

Service name:	Mcu_PerformReset

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

Service ID:	0x07
Sync/Async:	Synchronous
Reentrancy:	Non reentrant
Parameters (in):	None
Parameters (out):	None
Return value:	None
Description:	**MCU007:** The function performs a microcontroller reset. This is done by using the hardware feature of the microcontroller. The MCU specific reset type to be performed by this service shall be configured in the configuration set.
Caveats:	This function requires an execution of `Mcu_Init()` before it can be used. This function is only available if the runtime parameter MCU_PERFORM_RESET_API is set to TRUE. If set to FALSE, this function is not applicable. (see Section 10.2.2)
Configuration:	None

8.3.9 Mcu_SetMode

Service name:	Mcu_SetMode	

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

Service ID:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	`McuMode`	Set different MCU power modes configured in the configuration set
Parameters (out):	None	

Return value:	None
Description:	**MCU001:** This function activates the MCU power modes. In case of the CPU is switched off, the function is returned after a wake-up was performed.
Caveats:	This function requires an execution of Mcu_Init() before it can be used. In addition the user of this function has to ensure that the ECU is ready for reduced power mode activation.
Configuration:	None

8.3.10 Mcu_GetVersionInfo

Service name:	Mcu_ GetVersionInfo
Syntax:	`void Mcu_GetVersionInfo ( Std_VersionInfoType *versioninfo )`
Service ID hex:	0x09
Sync/Async:	Synchronous
Reentrancy:	non reentrant
Parameters (in):	none
Parameters (out):	`versioninfo` Pointer to where to store the version information of this module.
Return value:	none
Description:	**MCU103**:This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). **MCU104**: : This function shall be pre compile time configurable `On/Off` by the configuration parameter: `MCU_VERSION_INFO_API` Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.
Caveats:	-
Configuration:	This service is only available if enabled by the pre-processor switch `MCU_VERSION_INFO_API`.

8.4 Wake-up reason reporting

MCU057: After the occurrence of a hardware wake-up the function EcuM_SetWakeupReason shall be called if the static configuration parameter MCU_REPORT_WAKE_UP_REASON is set.

MCU058: The values for the wake-up reasons shall be configurable in the configuration structure. The function EcuM_SetWakeupReason shall be called with the parameter from the type EcuM_WakeupSourceType. The passed value is the configured wake-up reason from the assigned wake-up source.

MCU059: If there is no specific wake-up reason configured in the configuration structure, the parameter of EcuM_SetWakeupReason shall be WAKEUP_REASON_NONE.

8.5 Call-back Notifications

MCU117: There are no callback notifications for the MCU driver. The callback notifications are implemented in another module (ICU driver and/or complex drivers).

8.6 Scheduled Functions

There are no scheduled functions within the MCU driver.

8.7 Expected Interfaces

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

8.7.1 Mandatory Interfaces

None

8.7.2 Optional Interfaces

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

API function	`Det_ReportError`
Module	Development Error Tracer (DET, see section 3.1)
Description	Development error notification
Configuration parameter (description see chapter 10)	MCU_DEV_ERROR_DETECT

API function	`Dem_ReportErrorStatus`
Module	Diagnostic Event Manager (DEM, see section 3.1)
Description	Reporting of production relevant error status

API function	`Dem_ReportErrorEvent`
Module	Diagnostic Event Manager (DEM, see section 3.1)
Description	Reporting of production relevant error events

8.8 API parameter checking

MCU017: If the development error detection is enabled for this module, the following API parameter shall be checked. Detected errors shall be reported to the Development Error Tracer. The called service shall be rejected with E_NOT_OK, if the service provides a standard return type

MCU018: `ConfigPtr` shall not be NULL pointer. Also the (hardware specific) contents of the given configuration set shall be checked for being within the allowed boundaries. Related error value: MCU_E_PARAM_CONFIG.

MCU019: `ClockSetting` shall be within the settings defined in the configuration data structure. Related error value: MCU_E_PARAM_CLOCK

MCU020: `McuMode` shall be within the modes defined in the configuration data structure. Related error value: MCU_E_PARAM_MODE

MCU021: `RamSection` shall be within the sections defined in the configuration data structure. Related error value: MCU_E_PARAM_RAMSECTION

MCU122: An error shall be reported if the status of the PLL is detected as not locked with the function Mcu_DistributePllClock(). The DET reporting error shall be used. Related error value: MCU_E_PLL_NOT_LOCKED.

MCU125: The service Mcu_Init() shall be called first before calling any other MCU services. If development error detection is enabled and if this sequence is not respected, the following error code MCU_E_UNINIT shall be reported to the DET.

9 Sequence diagrams

9.1 Example Sequence for MCU initialization services

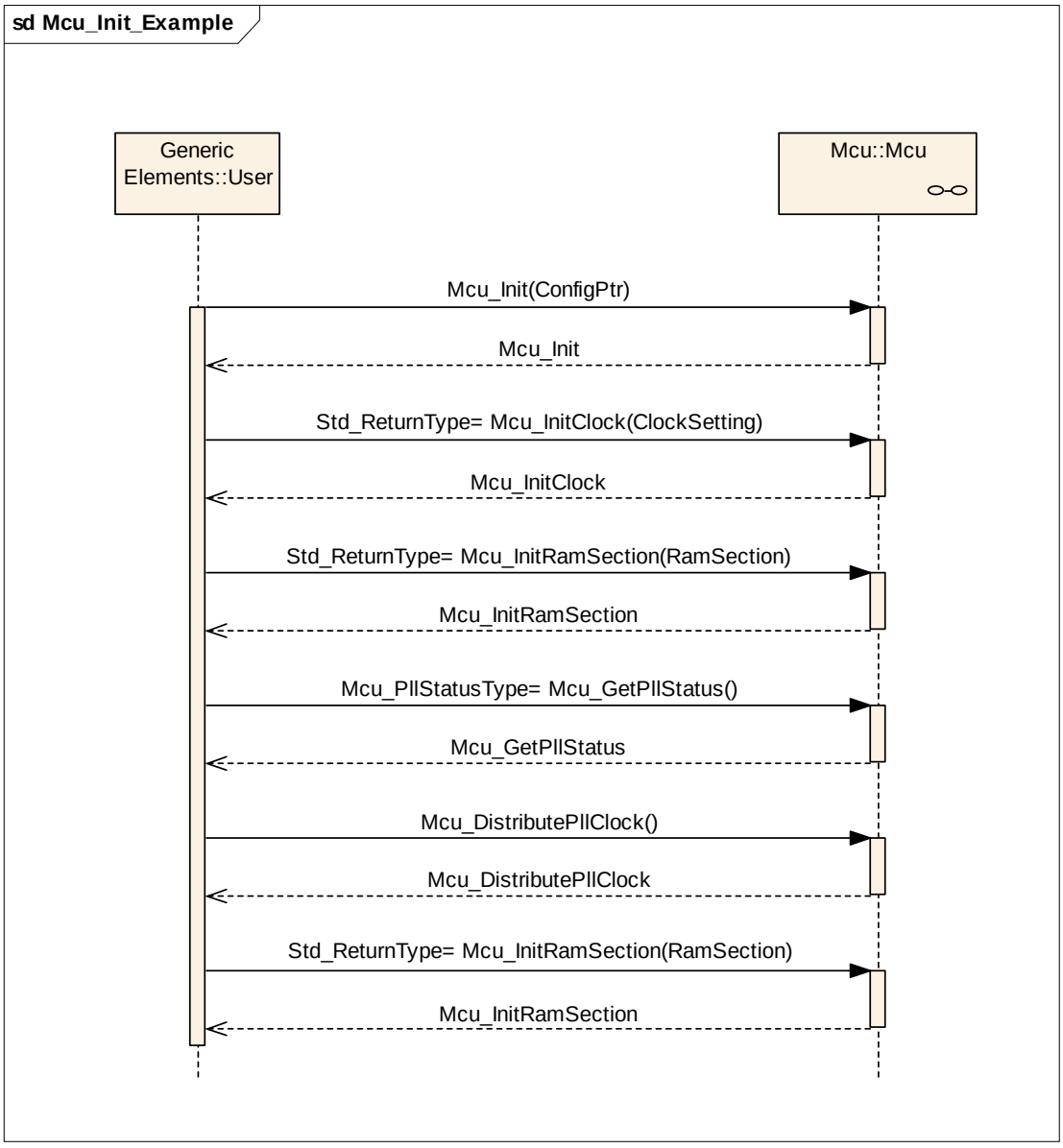

Figure 3: Sequence Diagram – MCU Initialisation

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_SetMode & Wake_up Mechanism

9.2.1 Wake-up reporting outside the MCU driver

The following diagrams identify wake-up handling in different microcontrollers

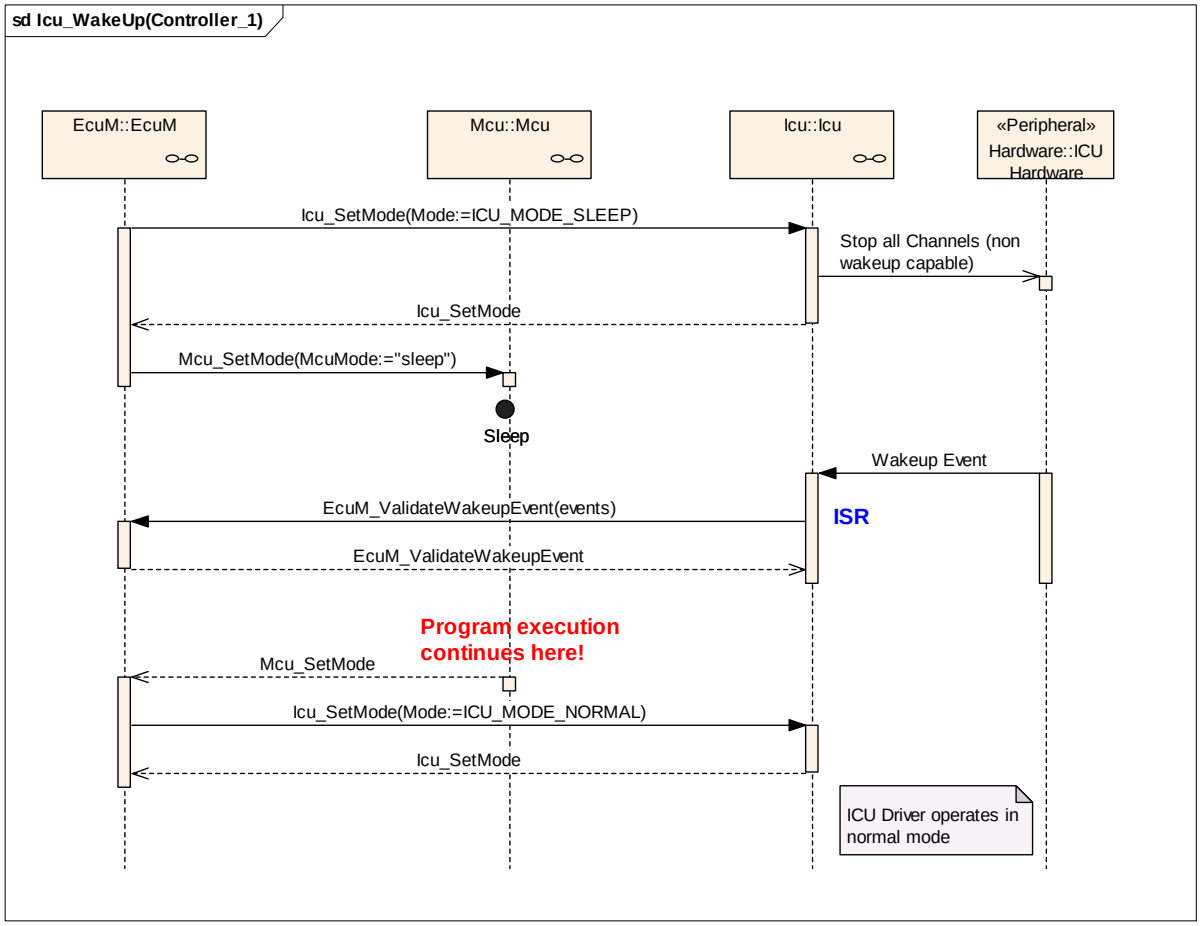

Figure 4: Sequence Diagram – Wake-up Reporting outside the MCU Driver

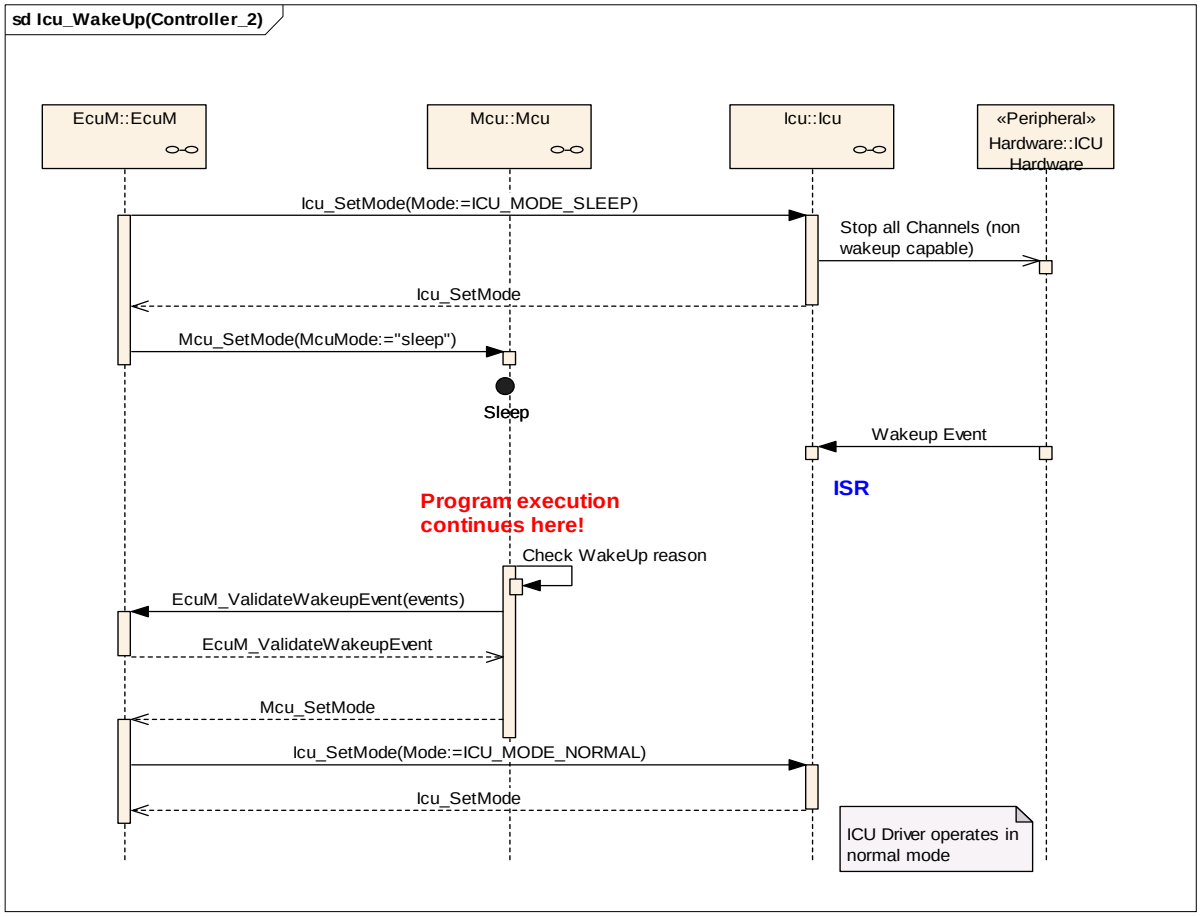

Figure 5: Sequence Diagram – Wake-up Reporting outside the MCU Driver

Before activating a low power mode of the MCU Driver, a callout to an ECU specific module is done to prepare sleep mode (Wakeup/Sleep Control). After that, the MCU is set to low power mode.
After some time, the ECU is woken up. The activated ISR checks the wake-up flag, detects that the ISR is caused by a wake-up and reports the wake-up reason to the ECU StateManager.

9.2.2 Wake-up reporting from MCU driver

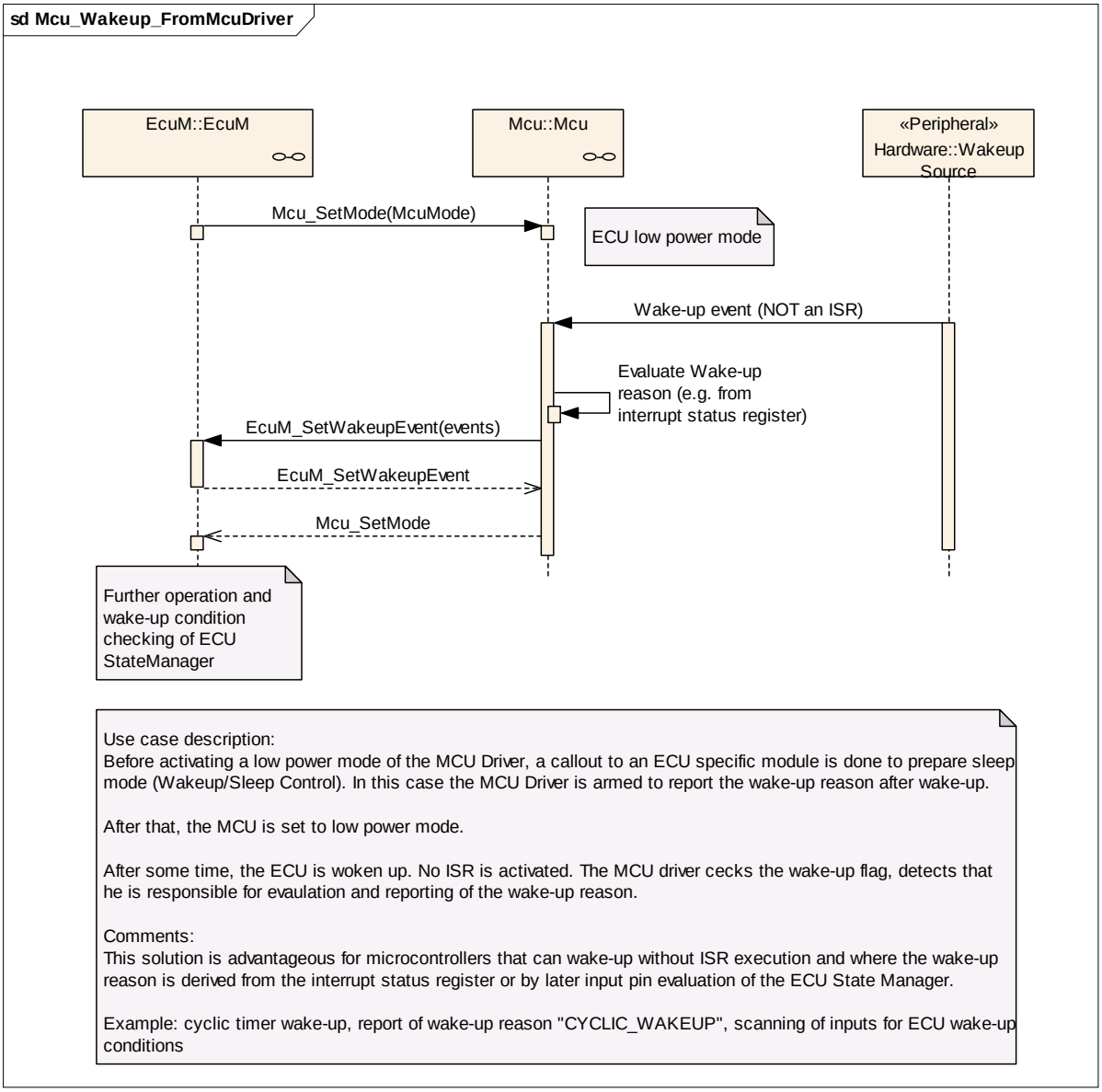

Figure 6: Sequence Diagram – Wake-up Reporting from the MCU Driver

Before activating a low power mode of the MCU Driver, a callout to an ECU specific module is done to prepare sleep mode (Wakeup/Sleep Control). In this case the MCU Driver is armed to report the wake-up reason after wake-up.

After that, the MCU is set to low power mode.

After some time, the ECU is woken up. No ISR is activated. The MCU driver checks the wake-up flag, detects that it is responsible for evaluation and reporting of the wake-up reason.

This solution is advantageous for microcontrollers that can wake-up without ISR execution and where the wake-up reason is derived from the interrupt status register or by later input pin evaluation of the ECU State Manager.

9.3 Mcu_GetResetReason

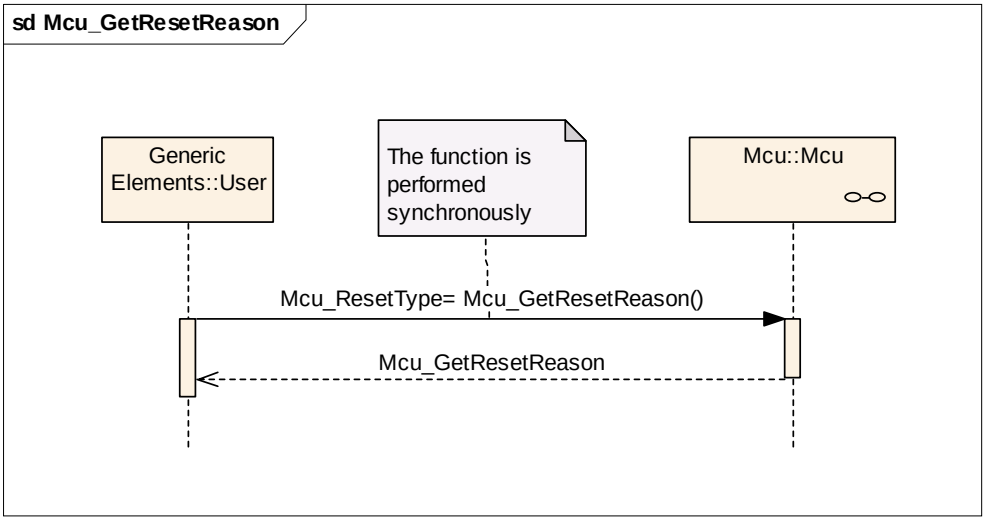

Figure 7: Sequence Diagram – MCU_GetResetReason

9.4 Mcu_GetResetRawValue

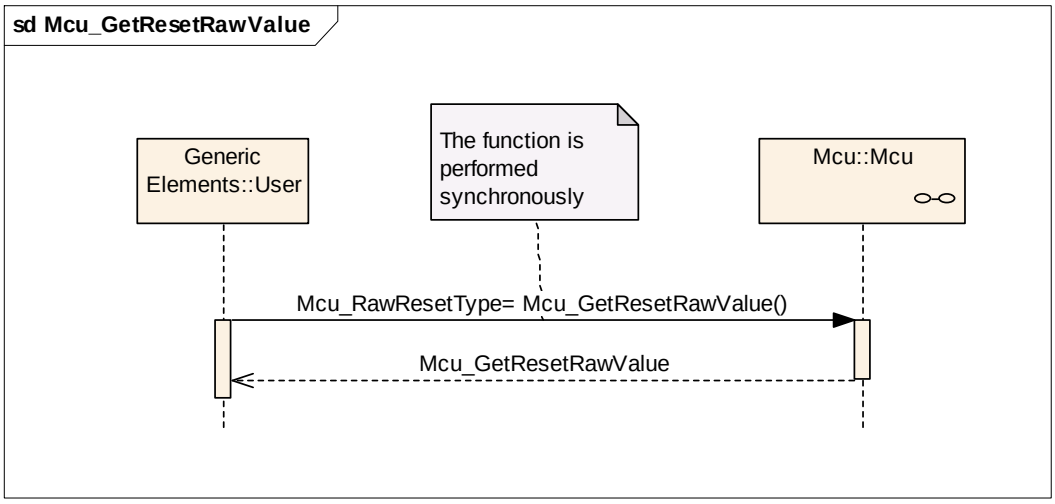

Figure 8: Sequence Diagram – Mcu_GetResetRawValue

9.5 Mcu_PerformReset

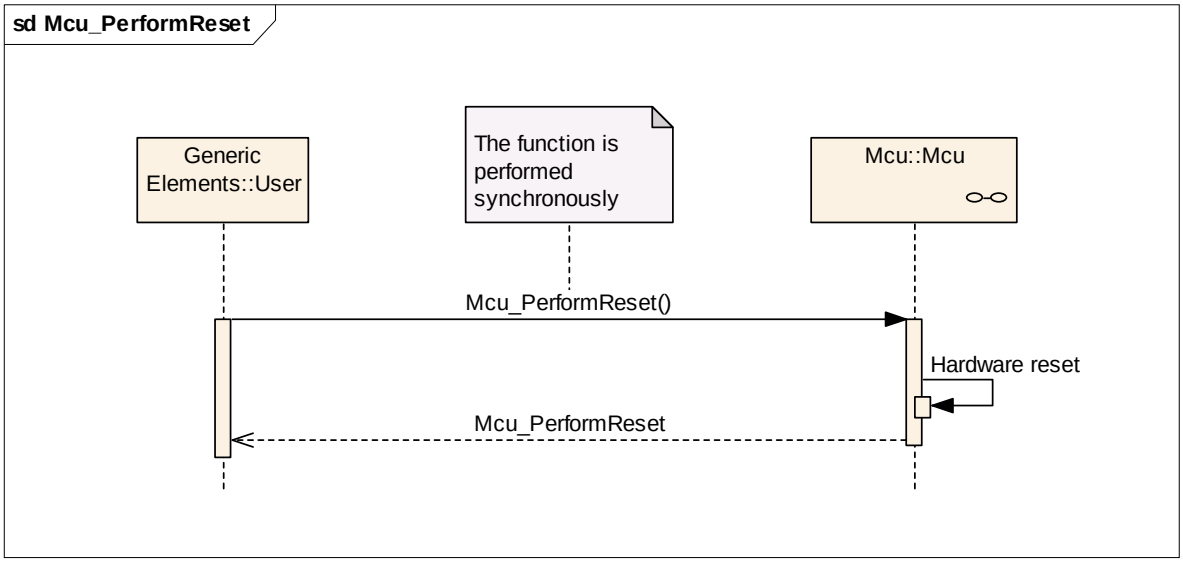

Figure 9: Sequence Diagram – Mcu_PerformReset

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

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [2].
- AUTOSAR ECU Configuration Specification [5].
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.

(sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

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

VariantPC: This variant is limited to pre-compile-configuration parameters only. The intention of this variant is to optimize the parameters configuration for a source code delivery.

VariantPB: This variant allows a mix of pre-compile time-, post build-time configuration parameters. The intention of this variant is to optimize the parameters configuration for a re-loadable binary.

MCU126: The initialization function of this module shall always have a pointer as a parameter, even though for VariantPC no configuration set shall be given. Instead a NULL pointer shall be passed to the initialization function. This means that, in contradiction to BSW00414 only one interface for initialization shall be implemented and it shall not depend on the modules configuration, which interface the calling software module shall use.

10.2.2 General Configuration

SWS Item	**MCU118**:
Container Name	MCU General Configuration
Description	This container contains the configuration (parameters) of the MCU driver.
Configuration Parameters	

Name	MCU_DEV_ERROR_DETECT		
Description	Pre-processor switch for enabling the development error detection and reporting.		
Type	#define		
Unit	-		
Range	ON	Development error detection enabled	
	OFF	Development error detection disabled	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	MCU_VERSION_INFO_API
Description	Pre-processor switch to enable / disable the API to read out the modules version information.
Type	#define
Unit	--

Range	ON	Version info API enabled.	
	OFF	Version info API disabled,	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	MCU_REPORT_WAKE_UP_REASON		
Description	Pre-processor switch for enabling Wake-up reason notification.		
Type or Unit	#define		
Range	ON	Wake-up reason notification enabled	
	OFF	Wake-up reason notification disabled	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	--		

Name	MCU_PERFORM_RESET_API		
Description	Pre-processor switch to enable / disable the use of the function Mcu_PerformReset() (section 8.3.8) .		
Type	#define		
Unit	--		
Range	TRUE	Enabled: The function Mcu_PerformReset() is available.	
	FALSE	Disabled: The function Mcu_PerformReset() is not available.	
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	-	-
	Post Build	-	-
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
MCU Module configuration	1..*	--

10.2.3 Mcu Module Configuration

SWS Item	**MCU119**:
Container Name	MCU Module Configuration
Description	This container contains the configuration (parameters) of the MCU driver.
Configuration Parameters	

Name	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.

Type	#define		
Unit	--		
Range	Implementation Specific		
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	MCU_CLOCK_SOURCE_FAILURE_NOTIFICATION		
Description	Enables/Disables clock failure notification		
Type	enumeration		
Unit	--		
Range	Implementation Specific		
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	-	-
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

Name	MCU_WAKEUP_REASON		
Description	Wake-up reasons to be reported		
Type	EcuM_WakeupReasonType		
Unit	--		
Range	--		
	--		
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	-	-
	Post Build	M	VariantPB
Scope	ECU		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
MCU RAM Sector Setting Configuration	1..*	--
MCU Mode Setting Configuration	1..*	--
MCU Clock Setting Configuration	1..*	--

10.2.4 Mcu RAM Sector Setting Configuration

SWS Item	**MCU120**:
Container Name	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 sector settings.
Configuration Parameters	

Name	MCU_RAM_SECTORS		
Description	This parameter shall represent the number of RAM sectors available for the MCU.		
Type	Mcu_RamSectionType		
Unit	--		
Range	Implementation Specific		
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	-	-
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

Name	MCU_RAM_SECTION_BASE_ADDRESS		
Description	This parameter shall represent the MCU RAM section base address.		
Type	Implementation Specific		
Unit	--		
Range	Implementation Specific		
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	-	-
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

Name	MCU_RAM_SECTION_SIZE		
Description	This parameter shall represent the MCU RAM Section size.		
Type	Implementation Specific		
Unit	--		
Range	Implementation Specific		
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	-	-
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

Name	MCU_RAM_DEFAULT_VALUE		
Description	This parameter shall represent the Data pre-setting to be initialized.		
Type	Implementation Specific		
Unit	--		
Range	Implementation Specific		
	Pre-compile	X	VariantPC

	Link time	-	-
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

10.2.5 MCU Mode Setting Configuration

SWS Item	**MCU123**:		
Container Name	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			

Name	MCU_MODE		
Description	The parameter is to represent the different MCU Modes available to an MCU		
Type	Mcu_ModeType		
Unit	--		
Range	Implementation Specific		
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	-	-
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

10.2.6 MCU Clock Setting Configuration

SWS Item	**MCU124**:		
Container Name	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			

Name	MCU_NUMBER_OF_CLOCK_SETTINGS		
Description	This parameter shall represent the number of clock settings available for the MCU.		
Type	Mcu_ClockType		
Unit	--		
Range	Implementation Specific		
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	-	-
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

MCU037: The following table specifies parameters that shall be published in the modules header file Mcu.h and also in the modules description file.

SWS Item	**MCU121**:	
Information elements		
Information element name	***Type / Range***	***Information element description***
`MCU_VENDOR_ID`	`#define /uint16`	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
`MCU_MODULE_ID`	`#define /uint8`	Module ID of this module from Module List
`MCU_AR_MAJOR_VERSION`	`#define /uint8`	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
`MCU_AR_MINOR_VERSION`	`#define /uint8`	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
`MCU_AR_PATCH_VERSION`	`#define /uint8`	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
`MCU_SW_MAJOR_VERSION`	`#define /uint8`	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
`MCU_SW_MINOR_VERSION`	`#define /uint8`	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
`MCU_SW_PATCH_VERSION`	`#define /uint8`	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

11 Changes to Release 1

11.1 Deleted SWS Items

SWS Item	*Rationale*
MCU022	Removed due to template change in Section 10.
MCU023	Removed due to template change in Section 10.
MCU061	Removed due to template change in Section 10.

11.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
MCU036 (SWS2098)	MCU110	New template structure.

11.3 Changed SWS Items

No changed SWS Items to Release 1.

11.4 Added SWS Items

SWS Item	*Rationale*
MCU100	Inclusion of switch for detection of development errors.
MCU101	Inclusion of MCU_DEV_ERROR_DETECT switch
MCU102	Detection of production errors.
MCU103	New API service for Mcu_GetVersionInfo()
MCU104	New API service.
MCU105	New section – code file structure
MCU108	New section - code file structure
MCU109	New section – code file structure
MCU110	New requirement of this template
MCU111	New requirement of this template
MCU112	New requirement of this template
MCU113	New requirement [BSW12461] introduced in SPAL_General.SRS [3]
MCU114	New requirement [BSW12461] introduced in SPAL_General.SRS [3]
MCU115	New requirement [BSW12461] introduced in SPAL_General.SRS [3]
MCU116	New requirement [BSW12461] introduced in SPAL_General.SRS [3]
MCU117	New requirement of this template
MCU118	New requirement of this template
MCU119	New requirement of this template
MCU120	New requirement of this template
MCU121	New requirement of this template
MCU122	New DET error MCU_E_PLL_NOT_LOCKED.
MCU123	New requirement of this template
MCU124	New requirement of this template
MCU125	MCU_E_UNINIT added to API Parameter checking (section 8.8)
MCU126	Additional requirement introduced in line with BSW00414 (SPAL decision, 42nd meeting, minutes day2, issue 5)