

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

| | |
|------------------|-------|
| Document Version | 3.0.0 |
| Document Status | Final |
| Part of Release | 4.0 |
| Revision | 1 |

| Document Change History | | | |
|-------------------------|---------|------------------------|---|
| Date | Version | Changed by | Change Description |
| 30.11.2009 | 3.0.0 | AUTOSAR Administration | <ul style="list-style-type: none"> 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 | <ul style="list-style-type: none"> 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 | <ul style="list-style-type: none">• Update to section 5.2.2: Inclusion of new file structure• Sections 8.4.2, 8.4.3, 8.4.9 : 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 8.4.8 : Description updated to include reference to new pre-processor switch McuPerformResetApi.• Section 10.2.2: Introduction of pre-processor switch McuPerformResetApi• Section 10.2.3: Multiplicity of sub-container Mcu Clock Setting Configuration changed to 1.• Legal disclaimer revised• Release Notes added• "Advice for users" revised• "Revision Information" added |
| 26.01.2006 | 2.0.0 | AUTOSAR Administration | <p>Document structure adapted to common Release 2.0 SWS Template.</p> <ul style="list-style-type: none">• 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 | 6 |
| 2 | Acronyms and abbreviations | 7 |
| 3 | Related documentation..... | 8 |
| 3.1 | Input documents..... | 8 |
| 4 | Constraints and assumptions | 9 |
| 4.1 | Limitations | 9 |
| 4.2 | Applicability to car domains | 9 |
| 5 | Dependencies to other modules | 10 |
| 5.1 | Start-up code..... | 10 |
| 5.2 | File structure | 11 |
| 5.2.1 | Code file structure | 11 |
| 5.2.2 | Header file structure..... | 11 |
| 6 | Requirements traceability | 14 |
| 7 | Functional specification | 21 |
| 7.1 | General Behavior | 21 |
| 7.1.1 | Background and Rationale | 21 |
| 7.1.2 | Requirements..... | 21 |
| 7.1.3 | Version Check..... | 22 |
| 7.2 | Error classification | 23 |
| 7.2.1 | Background and Rationale | 23 |
| 7.2.2 | Requirements..... | 23 |
| 7.3 | Error detection..... | 24 |
| 7.4 | Error notification | 24 |
| 7.5 | Debugging Support | 24 |
| 8 | API specification | 26 |
| 8.1 | Imported types..... | 26 |
| 8.2 | Type definitions | 26 |
| 8.2.1 | Mcu_ConfigType | 26 |
| 8.2.2 | Mcu_PliStatusType | 27 |
| 8.2.3 | Mcu_ClockType | 27 |
| 8.2.4 | Mcu_ResetType | 28 |
| 8.2.5 | Mcu_RawResetType | 28 |
| 8.2.6 | Mcu_ModeType | 28 |
| 8.2.7 | Mcu_RamSectionType | 29 |
| 8.2.8 | Mcu_RamStateType | 29 |
| 8.3 | | 29 |
| 8.4 | Function definitions | 29 |
| 8.4.1 | Mcu_Init | 29 |
| 8.4.2 | Mcu_InitRamSection | 30 |
| 8.4.3 | Mcu_InitClock..... | 31 |
| 8.4.4 | Mcu_DistributePliClock | 32 |

| | | |
|---------|--|----|
| 8.4.5 | Mcu_GetPllStatus | 32 |
| 8.4.6 | Mcu_GetResetReason | 33 |
| 8.4.7 | Mcu_GetResetRawValue | 34 |
| 8.4.8 | Mcu_PerformReset | 34 |
| 8.4.9 | Mcu_SetMode | 35 |
| 8.4.10 | Mcu_GetVersionInfo | 35 |
| 8.4.11 | Mcu_GetRamState | 36 |
| 8.5 | Call-back Notifications | 37 |
| 8.6 | Scheduled Functions | 37 |
| 8.7 | Expected Interfaces | 37 |
| 8.7.1 | Mandatory Interfaces | 37 |
| 8.7.2 | Optional Interfaces | 37 |
| 8.8 | API parameter checking | 38 |
| 9 | Sequence diagrams | 39 |
| 9.1 | Example Sequence for MCU initialization services | 39 |
| 9.2 | Mcu_GetResetReason | 40 |
| 9.3 | Mcu_GetResetRawValue | 40 |
| 9.4 | Mcu_PerformReset | 41 |
| 10 | Configuration specification | 42 |
| 10.1 | How to read this chapter | 42 |
| 10.1.1 | Configuration and configuration parameters | 42 |
| 10.1.2 | Containers | 42 |
| 10.2 | Containers and configuration parameters | 42 |
| 10.2.1 | Variants | 43 |
| 10.2.2 | Mcu | 43 |
| 10.2.3 | McuGeneralConfiguration | 43 |
| 10.2.4 | McuModuleConfiguration | 45 |
| 10.2.5 | McuClockSettingConfig | 46 |
| 10.2.6 | McuDemEventParameterRefs | 47 |
| 10.2.7 | McuModeSettingConf | 47 |
| 10.2.8 | McuRamSectorSettingConf | 48 |
| 10.2.9 | McuClockReferencePoint | 49 |
| 10.2.10 | McuPublishedInformation | 49 |
| 10.2.11 | McuResetReasonConf | 49 |
| 10.3 | Published Information | 50 |
| 11 | Changes to Release 3.0 | 51 |
| 11.1 | Deleted SWS Items | 51 |
| 11.2 | Replaced SWS Items | 51 |
| 11.3 | Changed SWS Items | 51 |
| 11.4 | Added SWS Items | 52 |

1 Introduction and functional overview

This specification describes the functionality and API for a MCU [Microcontroller Unit] driver. The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other MCAL software modules. The initialization services allow a flexible and application related MCU initialization in addition to the start-up code (see figure below). The start-up code is very MCU specific. The provided start-up code description in this document is for guidance and implies functionality which has to be taken into account before standardized MCU initialization is able to start.

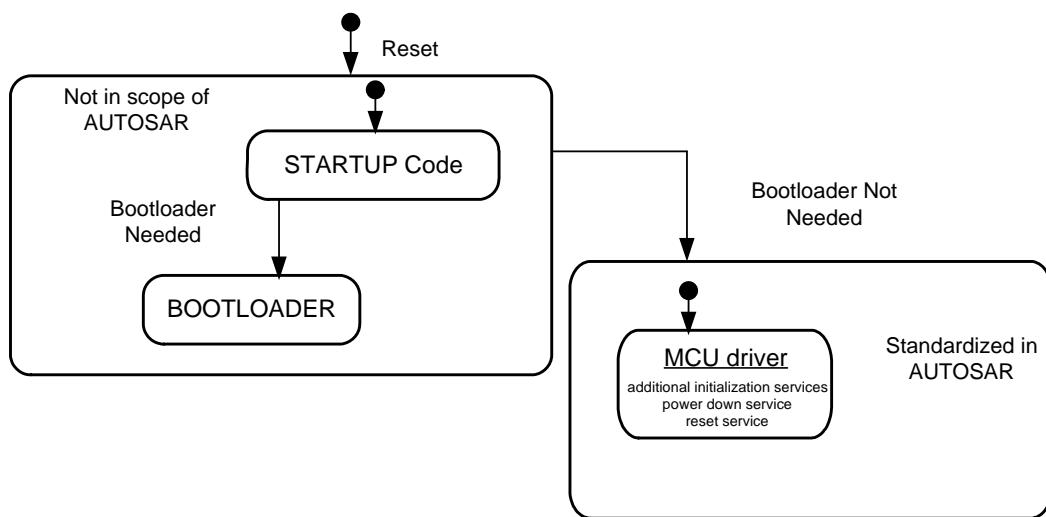


Figure 1: Scope of the MCU Driver Specification

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

MCU driver Features:

- Initialization of MCU clock, PLL, clock prescalers and MCU clock distribution
- Initialization of RAM sections
- Activation of μ C reduced power modes
- Activation of a μ C 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

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:

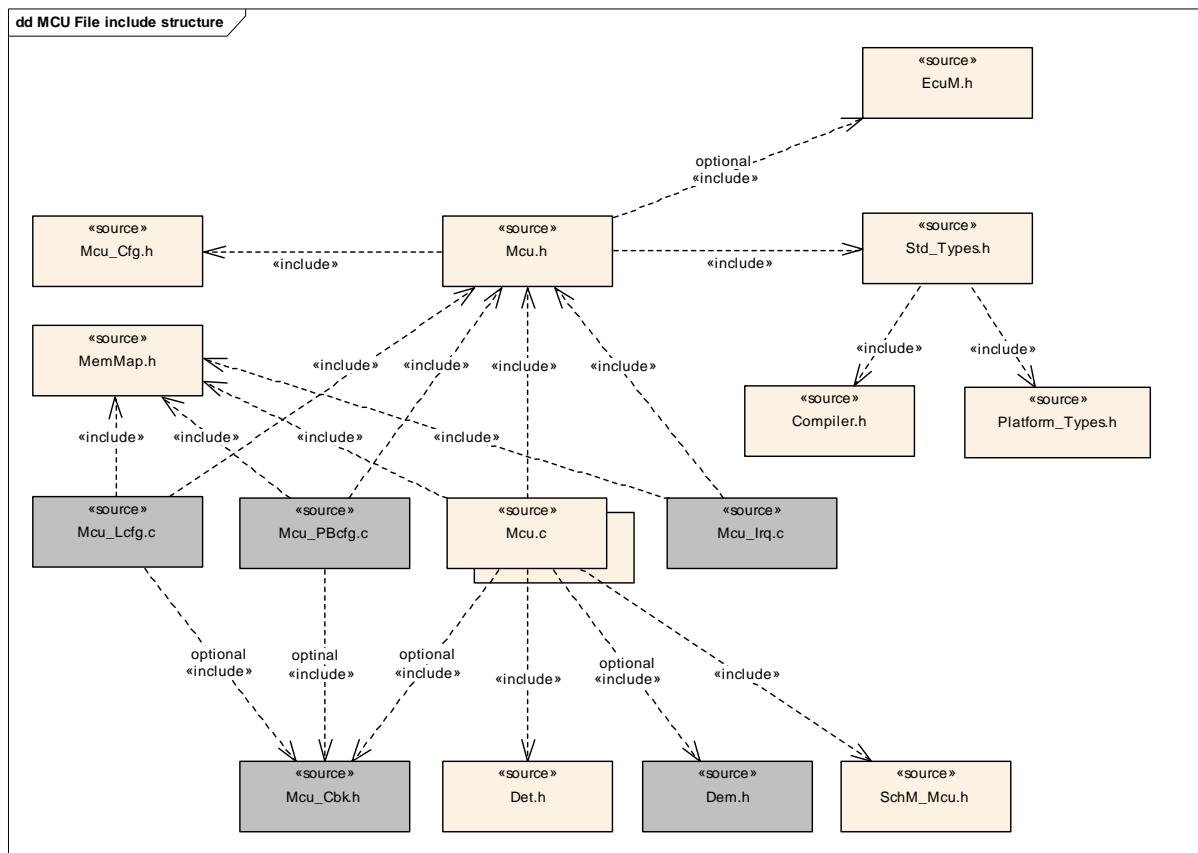


Figure 2: Header File Structure

MCU108: Mcu.c shall include Mcu.h.

MCU211: 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.

MCU213: Mcu_Irq.c shall include Mcu.h for the function prototype which shall be called in the interrupt function.

MCU215: 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"

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

```

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

MCU216: 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.

MCU218: 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.

MCU219: Mcu.c shall include Mcu_Cbk.h for pre-compile time configuration.

MCU109: The MCU 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.

MCU220: 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 [3]

| Requirement | Satisfied by |
|---|---|
| [BSW003] Version identification | MCU037 |
| [BSW004] Version check | MCU110 , MCU225 |
| [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_Conf , MCU120_Conf |
| [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 implementer). |
| [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 | 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 | MCU049 |
| [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_Conf |
| [BSW00345] Pre-compile-time configuration | See Section 5.2.2 , MCU119_Conf |
| [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 | Not Applicable (Because the MCU driver does not provide notification of wake-up). |
| 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 | MCU037 |
| [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 | MCU049 |
| [BSW00422] Debouncing of production relevant error status | MCU049 |
| [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) |
| [BSW00435] Header file Structure for the Basic Software Scheduler | See Figure 2 |
| [BSW00436] Module Header File Structure for the Basic Software Memory Mapping | See Figure 2 |
| [BSW00442] Debugging Support | MCU200 MCU201 MCU202 MCU203 |

Document: AUTOSAR requirements on Basic Software, cluster SPAL [8] e [9]

| 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. | Not Applicable (Because the MCU driver does not provide configuration information for wake-up sources). |
| [BSW12057] Driver module initialization | MCU026 |
| [BSW12125] Initialization of hardware resources | MCU116 , MCU244 , MCU245 , MCU246 , MCU247 |
| [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 | Not Applicable (Because the MCU driver does not provide notification of wake-up). |
| [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 MCU164 , MCU165) 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 | Not Applicable (Because the MCU driver does not provide notification of wake-up). |
| [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 | MCU140 , MCU141 , MCU056 |
| [BSW12207] Configuration of clock safety features | MCU031 , MCU054 |
| [BSW12208] Initialization of MCU Clock | MCU248 MCU137 , MCU138 |
| [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 | MCU143 , MCU144 , MCU055 |
| [BSW12268] MCU Power Management Control | MCU164 , MCU165 |
| [BSW12421] Low Power Mode Configuration | MCU035 |
| [BSW12461] Responsibility for register initialisation | MCU116 , MCU244 , MCU245 , MCU246 , MCU247 |
| [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). |
| [BSW13701] Providing Ram Status | MCU207 , MCU208 , MCU209 , MCU181_Conf |

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 MCU module shall provide a service to provide software triggering of a hardware reset.

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

MCU052: The MCU module shall provide services to get the reset reason of the last reset if the hardware supports such a feature.

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

7.1.2.2 Clock

MCU248 Mcu shall provide a service to enable and set the MCU clock.
(i.e. Cpu clock, Peripheral Clock, Prescalers, Multipliers have to be configured in the MCU)

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

7.1.2.3 MCU Mode service

MCU164: The MCU module shall provide a service to activate MCU reduced power modes.

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

MCU165: The number of modes and the configuration is MCU dependent and shall be configured in the configuration set of the MCU module.

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

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

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

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

7.1.3 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

The Mcu module shall avoid the integration of incompatible files by the following pre-processor checks:

MCU110:

For included (external) header files:

- <MODULENAME>_AR_ RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_ RELEASE_MINOR_VERSION

shall be verified.

MCU225:

For the module internal files:

- MCU_SW_MAJOR_VERSION
- MCU_SW_MINOR_VERSION
- MCU_AR_RELEASE_MAJOR_VERSION
- MCU_AR_RELEASE_MINOR_VERSION
- MCU_AR_RELEASE_PATCH_VERSION

shall be verified.

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 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 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 |
| Clock source failure | Production | MCU_E_CLOCK_FAILURE | Assigned by DEM |

Table 2: Error Classification

MCU053: 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 [MCU051](#)).

7.3 Error detection

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

MCU101: If the `McuDevErrorDetect` 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: Detected development errors shall be reported to the `Det_ReportError` service of the Development Error Tracer (DET) if the pre-processor switch `McuDevErrorDetect` is set (see chapter 10).

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).

MCU049: Production errors shall be reported to the Diagnostic Event Manager (DEM).

MCU226: Production Errors 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 to the MCU implementation specification. The classification and enumeration shall be compatible to the errors listed above (see MCU012).

7.5 Debugging Support

The following requirements deal with the definition of variables and the description of debug information.

MCU200: Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable.

MCU201: All type definitions of variables which shall be debugged, shall be accessible by the header file `Mcu.h`.

MCU202: The declaration of variables in the header file shall allow to calculate the size of the variables by C-"sizeof".

MCU203: Variables available for debugging shall be described in the respective Basic Software Module Description.

8 API specification

8.1 Imported types

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

MCU152:

| Module | Imported Type |
|-----------|---------------------|
| Dem | Dem_EventIdType |
| Std_Types | Std_ReturnType |
| | Std_VersionInfoType |

8.2 Type definitions

8.2.1 Mcu_ConfigType

| | | |
|--------------|---|---|
| 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. | |

MCU131: 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

MCU054: The structure `Mcu_ConfigType` shall provide a configurable (enable/disable) clock failure notification if the MCU provides an interrupt for such detection.

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.

MCU035: 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

MCU031: 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

MCU030: 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

Usage of linker symbols instead of scalar values is allowed.

8.2.2 `Mcu_PlStatusType`

| | | |
|---------------------|---|-----------------------|
| Name: | <code>Mcu_PlStatusType</code> | |
| Type: | Enumeration | |
| Range: | <code>MCU_PLL_LOCKED</code> | PLL is locked |
| | <code>MCU_PLL_UNLOCKED</code> | PLL is unlocked |
| | <code>MCU_PLL_STATUS_UNDEFINED</code> | PLL Status is unknown |
| Description: | This is a status value returned by the function <code>Mcu_GetPlStatus</code> of the MCU module. | |

MCU230: The type `Mcu_PlStatusType` is the type of the return value of the function `Mcu_GetPlStatus`.

MCU231: The type of `Mcu_PlStatusType` is an enumeration with the following values: `MCU_PLL_LOCKED`, `MCU_PLL_UNLOCKED`, `MCU_PLL_STATUS_UNDEFINED`.

8.2.3 `Mcu_ClockType`

| | | |
|---------------------|---|---|
| Name: | <code>Mcu_ClockType</code> | |
| Type: | uint | |
| Range: | <code>0..<number of clock settings>-1</code> | 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 | |

MCU232: The type Mcu_ClockType defines the identification (ID) for clock setting configured via the configuration structure.

MCU233: The type shall be uint8, uint16 or uint32, depending on uC platform.

8.2.4 Mcu_ResetType

| | | |
|---------------------|---|-------------------------------|
| 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. | |

MCU234: The type Mcu_ResetType, represents the different reset that a specified MCU can have.

MCU134: 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

| | | |
|---------------------|--|--|
| 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. | |

MCU235: The type Mcu_RawResetType specifies the reset reason in raw register format, read from a reset status register.

MCU236: The type shall be uint8, uint16 or uint32 based on best performance.

8.2.6 Mcu_ModeType

| | | |
|---------------|------------------------------|--|
| 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. |
|---------------------|---|

MCU237: The `Mcu_ModeType` specifies the identification (ID) for a MCU mode, configured via configuration structure.

MCU238: The type shall be `uint8`, `uint16` or `uint32`.

8.2.7 Mcu_RamSectionType

| | |
|---------------------|--|
| Name: | <code>Mcu_RamSectionType</code> |
| Type: | <code>uint</code> |
| 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. |

MCU239: The `Mcu_RamSectionType` specifies the identification (ID) for a RAM section, configured via the configuration structure.

MCU240: The type shall be `uint8`, `uint16` or `uint32`, based on best performance.

8.2.8 Mcu_RamStateType

8.3

| | |
|---------------------|---|
| Name: | <code>Mcu_RamStateType</code> |
| Type: | Enumeration |
| Range: | <code>MCU_RAMSTATE_INVALID</code> Ram content is not valid or unknown (default). <code>MCU_RAMSTATE_VALID</code> Ram content is valid: |
| Description: | This is the Ram State data type returned by the function <code>Mcu_GetRamState</code> of the Mcu module. It is not required that all RAM state types are supported by the hardware. |

8.4 Function definitions

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

8.4.1 Mcu_Init

MCU153:

| | |
|----------------------|---|
| Service name: | <code>Mcu_Init</code> |
| Syntax: | <code>void Mcu_Init(const Mcu_ConfigType* ConfigPtr</code> |

| | |
|----------------------------|---|
| |) |
| 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. |

MCU026: The function `Mcu_Init` shall initialize the MCU module, i.e. make the configuration settings for power down, clock and RAM sections visible within the MCU module.

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

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

1. **MCU116:** If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.
2. **MCU244:** 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. **MCU245:** 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. **MCU246:** One-time writable registers that require initialisation directly after reset shall be initialised by the startup code.
5. **MCU247:** All other registers not mentioned before shall be initialised by the start-up code.

MCU127: 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.

8.4.2 `Mcu_InitRamSection`

MCU154:

| | |
|-------------------------|---|
| Service name: | <code>Mcu_InitRamSection</code> |
| Syntax: | <code>Std_ReturnType Mcu_InitRamSection(Mcu_RamSectionType RamSection >)</code> |
| 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. | |

MCU011: 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`, `McuRamSectionBase` and `McuRamDefaultValue` are the values of the configuration parameters for each `RamSection` (see [MCU030](#)).

MCU136: The MCU module's environment shall call the function `Mcu_InitRamSection` only after the MCU module has been initialized using the function `Mcu_Init`.

8.4.3 `Mcu_InitClock`

MCU155:

| | | |
|----------------------------|---|--|
| Service name: | <code>Mcu_InitClock</code> | |
| Syntax: | <pre>Std_ReturnType Mcu_InitClock(Mcu_ClockType ClockSetting)</pre> | |
| 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. | |

MCU137: The function `Mcu_InitClock` shall initialize the PLL and other MCU specific clock options. The clock configuration parameters are provided via the configuration structure.

MCU138: 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.

MCU139: The MCU module's environment shall only call the function `Mcu_InitClock` after the MCU module has been initialized using the function `Mcu_Init`.

MCU210: The function `Mcu_InitClock` shall be disabled if the parameter `Mcu_InitClock` is set to FALSE. Instead this function is available if the former parameter is set to TRUE (see also **MCU182_Conf** :).

8.4.4 Mcu_DistributePllClock

MCU156:

| | |
|----------------------------|--|
| Service name: | <code>Mcu_DistributePllClock</code> |
| Syntax: | <code>void Mcu_DistributePllClock(void >)</code> |
| Service ID[hex]: | 0x03 |
| Sync/Async: | Synchronous |
| Reentrancy: | Non Reentrant |
| Parameters (in): | None |
| Parameters (inout): | None |
| Parameters (out): | None |
| Return value: | None |
| Description: | This service activates the PLL clock to the MCU clock distribution. |

MCU140: The function `Mcu_DistributePllClock` shall activate the PLL clock to the MCU clock distribution.

MCU141: The MCU module's environment shall execute the function `Mcu_DistributePllClock` if the MCU module needs a separate request to activate the PLL clock after the PLL is locked. In this case, the function shall remove the current clock source (for example internal oscillator clock) from MCU clock distribution.

MCU056: The function `Mcu_DistributePllClock` shall return without affecting the MCU hardware if the PLL clock has been automatically activated by the MCU hardware.

MCU142: 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`.

MCU205: 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 **MCU180_Conf** :).

8.4.5 Mcu_GetPllStatus

MCU157:

| | |
|----------------------|-------------------------------|
| Service name: | <code>Mcu_GetPllStatus</code> |
|----------------------|-------------------------------|

| | | |
|----------------------------|---|------------|
| Syntax: | Mcu_Pl1StatusType Mcu_GetPl1Status(void) | |
| Service ID[hex]: | 0x04 | |
| Sync/Async: | Synchronous | |
| Reentrancy: | Reentrant | |
| Parameters (in): | None | |
| Parameters (inout): | None | |
| Parameters (out): | None | |
| Return value: | Mcu_Pl1StatusType | PLL Status |
| Description: | This service provides the lock status of the PLL. | |

MCU008: The function `Mcu_GetPl1Status` shall return the lock status of the PLL.

MCU132: The function `Mcu_GetPl1Status` shall return `MCU_PLL_STATUS_UNDEFINED` if this function is called prior to calling of the function `Mcu_Init`.

MCU206: The function `Mcu_GetPl1Status` shall also return `MCU_PLL_STATUS_UNDEFINED` if the pre-compile parameter `McuNoPl1` is set to TRUE (see also **MCU180_Conf** :).

8.4.6 Mcu_GetResetReason

MCU158:

| | | |
|----------------------------|---|----|
| 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. | |

MCU005: 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`.

MCU133: 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.

8.4.7 Mcu_GetResetRawValue

MCU159:

| | | |
|----------------------------|--|-----------------|
| 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. | |

MCU135: The function Mcu_GetResetRawValue shall return an implementation specific value which does not correspond to a valid value of the reset status register and is not equal to 0 if this function is called prior to calling of the function Mcu_Init, and if supported by the hardware.

MCU006: The function Mcu_GetResetRawValue shall read the reset raw value from the hardware register if the hardware supports this. If the hardware does not have a reset status register, the return value shall be 0x0.

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

8.4.8 Mcu_PerformReset

MCU160:

| | | |
|----------------------------|---|--|
| 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. | |

MCU143: The function `Mcu_PerformReset` shall perform a microcontroller reset by using the hardware feature of the microcontroller.

MCU144: The function `Mcu_PerformReset` shall perform the reset type which is configured in the configuration set.

MCU145: The MCU module's environment shall only call the function `Mcu_PerformReset` after the MCU module has been initialized by the function `Mcu_Init`.

MCU146: The function `Mcu_PerformReset` is only available if the runtime parameter `McuPerformResetApi` is set to TRUE. If set to FALSE, the function `Mcu_PerformReset` is not applicable. (see Section 10.2.2).

8.4.9 `Mcu_SetMode`

MCU161:

| | |
|----------------------------|--|
| Service name: | <code>Mcu_SetMode</code> |
| Syntax: | <code>void Mcu_SetMode(Mcu_ModeType McuMode)</code> |
| Service ID[hex]: | 0x08 |
| Sync/Async: | Synchronous |
| Reentrancy: | Non Reentrant |
| Parameters (in): | <code>McuMode</code> 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. |

MCU147: 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.

MCU148: 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.

8.4.10 `Mcu_GetVersionInfo`

MCU162:

| | |
|----------------------|---------------------------------|
| Service name: | <code>Mcu_GetVersionInfo</code> |
|----------------------|---------------------------------|

| | |
|----------------------------|---|
| Syntax: | void Mcu_GetVersionInfo(Std_VersionInfoType* versioninfo) |
| Service ID[hex]: | 0x09 |
| Sync/Async: | Synchronous |
| Reentrancy: | Non 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. |

MCU103: The function `Mcu_GetVersionInfo` shall return the version information of the MCU module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407)

MCU104: The function `Mcu_GetVersionInfo` shall be pre-compile time configurable On/Off by the configuration parameter `McuVersionInfoApi`.

MCU149: If source code for caller and callee of the function `Mcu_GetVersionInfo` is available, the MCU module should realize this function as a macro. The MCU module should define this macro in the module's header file.

MCU204: 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.

8.4.11 `Mcu_GetRamState`

MCU207:

| | |
|----------------------------|--|
| Service name: | <code>Mcu_GetRamState</code> |
| Syntax: | <code>Mcu_RamStateType Mcu_GetRamState(void)</code> |
| Service ID[hex]: | 0x04 |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | None |
| Parameters (inout): | None |
| Parameters (out): | None |
| Return value: | <code>Mcu_RamStateType</code> Status of the Ram Content |
| Description: | This service provides the actual status of the microcontroller Ram. (if supported) |

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.

MCU208: The MCU module's environment shall call this function only if the MCU module has been already initialized using the function `MCU_Init`.

MCU209: The function `Mcu_GetRamState` shall be available to the user if the pre-compile parameter `McuGetRamStateApi` is set to TRUE. Instead, if the former parameter is set to FALSE, this function shall be disabled (e.g. the hardware does not support this functionality).

8.5 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).

8.6 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

MCU166:

| API function | Description |
|------------------------------------|--|
| <code>Dem_ReportErrorStatus</code> | 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. |

8.7.2 Optional Interfaces

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

MCU163:

| API function | Description |
|------------------------------|---------------------------------------|
| <code>Det_ReportError</code> | Service to report development errors. |

8.8 API parameter checking

MCU017: 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.

MCU018: If development error detection is enabled, the parameter `ConfigPtr` shall be checked for being `NULL`. 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: 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`.

MCU125: The function `Mcu_Init` shall be called first before calling any other MCU function. If development error detection is enabled and if this sequence is not followed, the error code `MCU_E_UNINIT` shall be reported to the DET.

9 Sequence diagrams

9.1 Example Sequence for MCU initialization services

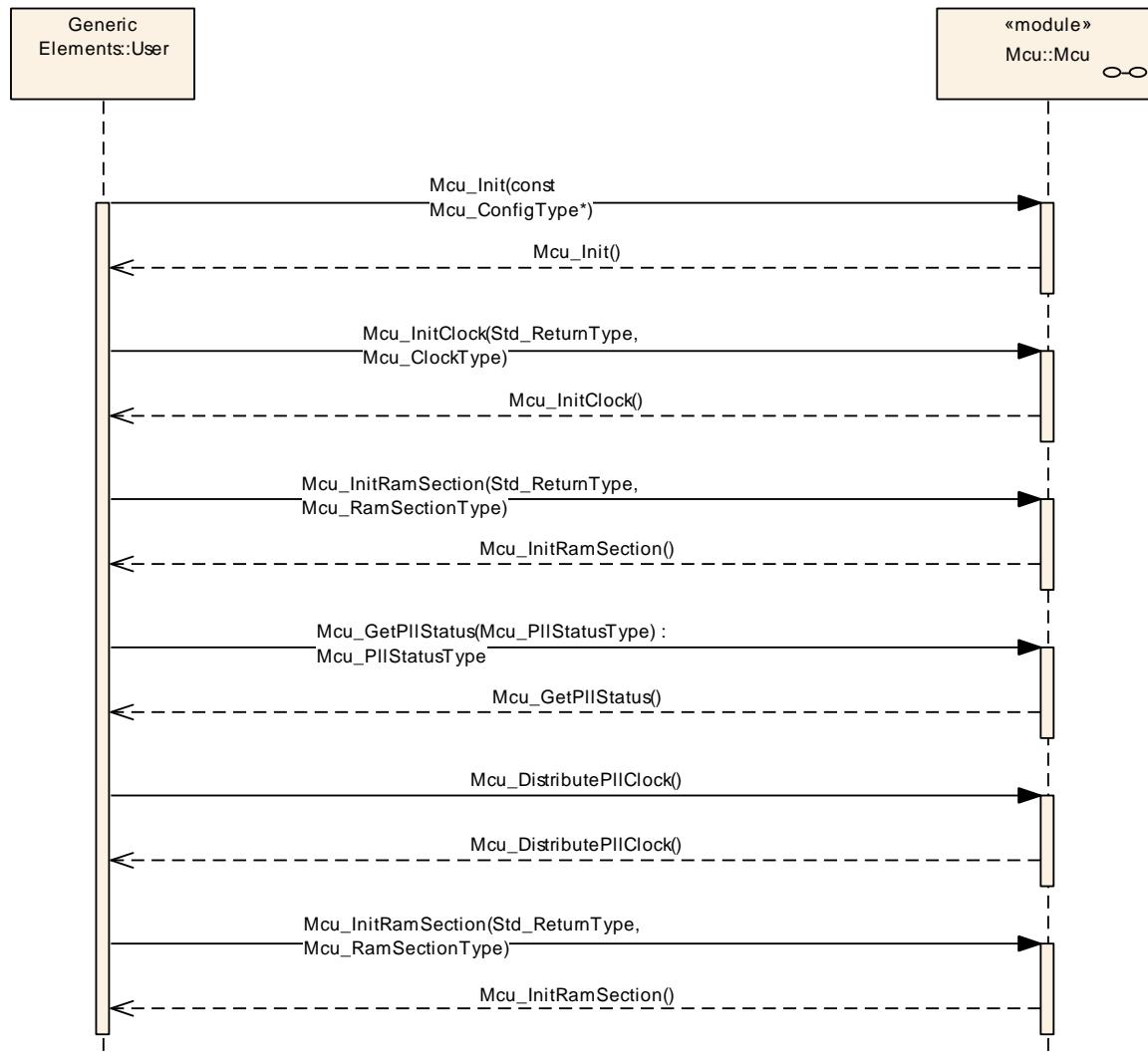


Figure 3: Sequence Diagram – MCU Initialisation

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

9.2 Mcu_GetResetReason

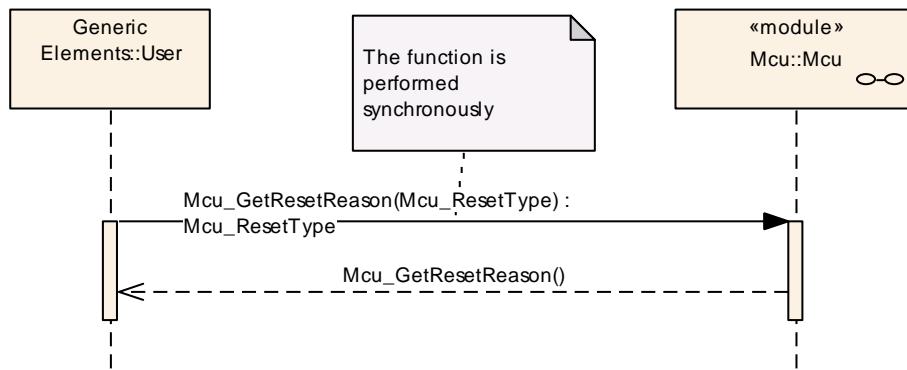


Figure 7: Sequence Diagram – MCU_GetResetReason

9.3 Mcu_GetResetRawValue

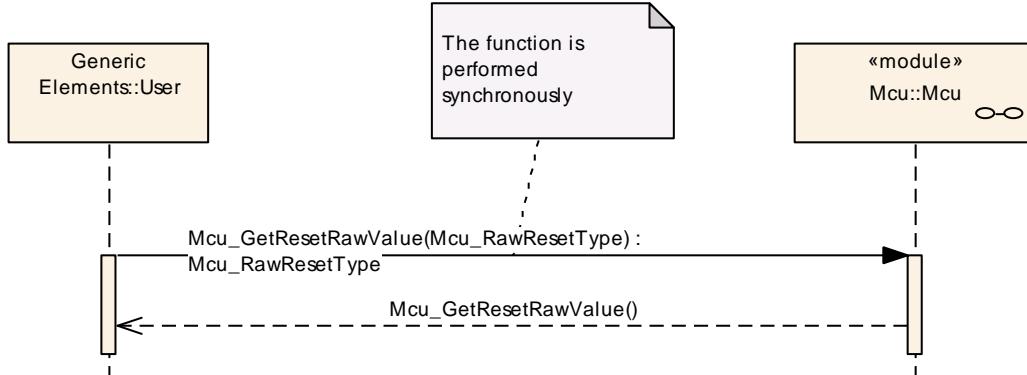


Figure 8: Sequence Diagram – Mcu_GetResetRawValue

9.4 Mcu_PerformReset

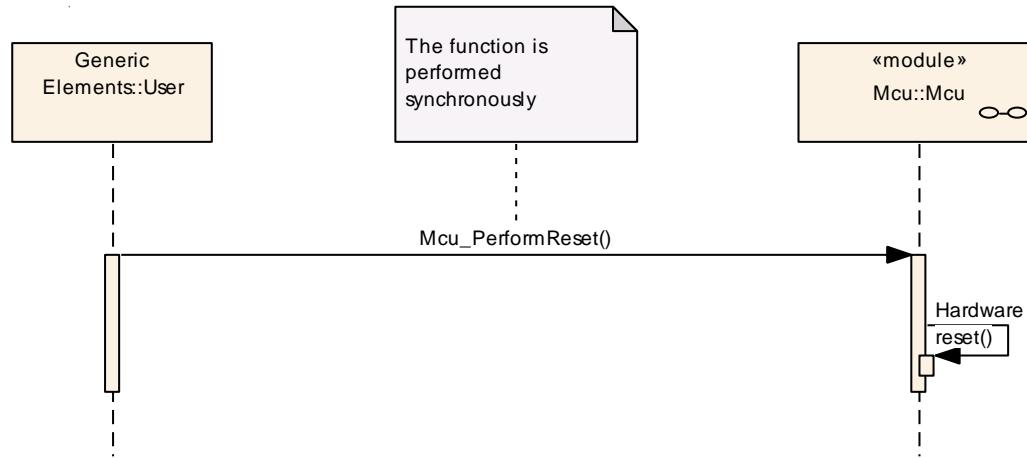


Figure 9: Sequence Diagram – Mcu_PerformReset

10 Configuration specification

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

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

Chapter 10.3 specifies published information of the module MCU.

10.1 How to read this chapter

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

MCU129: VARIANT-PRE-COMPIL.

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.

MCU130: 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.

MCU126: The initialization function of this module shall always have a pointer as a parameter, even though for VARIANT-PRE-COMPIL 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 (Microcontroler 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 | MCU118_Conf : |
| Container Name | McuGeneralConfiguration{MCU General Configuration} |
| Description | This container contains the configuration (parameters) of the MCU driver. |
| Configuration Parameters | |

| | | | | | | | | | | |
|---------------------------|--|-------------------------|---|--------------|------------------|----|--|------------------------|----|--|
| SWS Item | MCU166_Conf : | | | | | | | | | |
| 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 | <table border="1"> <tr> <td>Pre-compile time</td> <td>X</td> <td>All Variants</td> </tr> <tr> <td>Link time</td> <td>--</td> <td></td> </tr> <tr> <td>Post-build time</td> <td>--</td> <td></td> </tr> </table> | Pre-compile time | X | All Variants | Link time | -- | | Post-build time | -- | |
| Pre-compile time | X | All Variants | | | | | | | | |
| Link time | -- | | | | | | | | | |
| Post-build time | -- | | | | | | | | | |
| Scope / Dependency | scope: Module | | | | | | | | | |

| | | | |
|---------------------------|--|----|--------------|
| SWS Item | MCU181_Conf : | | |
| 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: Module | | |

| | | | |
|---------------------------|---|----|--------------|
| SWS Item | MCU182_Conf : | | |
| 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: Module | | |

| | | | |
|---------------------------|---|----|--------------|
| SWS Item | MCU180_Conf : | | |
| 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: Module | | |

| | | | |
|---------------------------|---|----|--------------|
| SWS Item | MCU167_Conf : | | |
| 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 | | | |

| | | | |
|-----------------|--|--|--|
| SWS Item | MCU168_Conf : | | |
| 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: Module | | |

No Included Containers

10.2.4 McuModuleConfiguration

| | | | |
|---------------------------------|---|--|--|
| SWS Item | MCU119_Conf : | | |
| Container Name | McuModuleConfiguration{MCU Module Configuration} [Multi Config Container] | | |
| Description | This container contains the configuration (parameters) of the MCU driver | | |
| Configuration Parameters | | | |

| | | | |
|---------------------------|--|----|--------------------|
| SWS Item | MCU170_Conf : | | |
| 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-COMPIL |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: Module | | |

| | | | |
|---------------------------|--|----|--------------------|
| SWS Item | MCU171_Conf : | | |
| 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-COMPIL |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: Module | | |

| | | | |
|---------------------|---|--|--|
| SWS Item | MCU172_Conf : | | |
| 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 | 1 .. 4294967295 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPIL |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: Module | | |

| | | | |
|---------------------------|--|----|--------------|
| SWS Item | MCU173_Conf : | | |
| 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: Module | | |

| 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 value. 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 | MCU124_Conf : | | |
| 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 | MCU183_Conf : | | |
| 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 | 1 .. 255 | | |

| | | | |
|---------------------------|-------------------------|----|--------------------|
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPIL |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: Module | | |

| 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 multiplicitiy 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 | MCU187_Conf : | | |
| 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 value. The standardized errors are provided in the container and can be extended by vendor specific error references. | | |
| Configuration Parameters | | | |

| | | | |
|---------------------------|---|----|--------------|
| SWS Item | MCU188_Conf : | | |
| Name | MCU_E_CLOCK_FAILURE {MCU_E_CLOCK_FAILURE} | | |
| Description | Reference to configured DEM event to report "Clock source failure". | | |
| Multiplicity | 0..1 | | |
| Type | Reference to [DemEventParameter] | | |
| ConfigurationClass | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: ECU dependency: Dem | | |

No Included Containers

10.2.7 McuModeSettingConf

| | | | |
|---------------------------------|--|--|--|
| SWS Item | MCU123_Conf : | | |
| 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 | MCU176_Conf : | | |
| Name | McuMode {MCU_MODE_NORMAL} | | |
| Description | The parameter represents the MCU Mode settings. | | |
| Multiplicity | 1 | | |

| | | | |
|--------------------|--|----|---------------------|
| Type | EcucIntegerParamDef (Symbolic Name generated for this parameter) | | |
| 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: Module | | |

No Included Containers

10.2.8 McuRamSectorSettingConf

| | | | |
|--------------------------|---|--|--|
| SWS Item | MCU120_Conf : | | |
| 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 sector settings. | | |
| Configuration Parameters | | | |

| | | | |
|--------------------|---|----|---------------------|
| SWS Item | MCU177_Conf : | | |
| 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: Module | | |

| | | | |
|--------------------|---|----|---------------------|
| SWS Item | MCU178_Conf : | | |
| 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: Module | | |

| | | | |
|--------------------|--|----|---------------------|
| SWS Item | MCU179_Conf : | | |
| 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: Module | | |

No Included Containers

10.2.9 McuClockReferencePoint

| | | | |
|---------------------------------|--|--|--|
| SWS Item | MCU174_Conf : | | |
| 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 simpliest case (only one frequency is used), there is one frequency to be defined. | | |
| Configuration Parameters | | | |

| | | | |
|---------------------------|---|----|--------------------|
| SWS Item | MCU175_Conf : | | |
| 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 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPIL |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: Module | | |

No Included Containers

10.2.10 McuPublishedInformation

| | | | |
|---------------------------------|---|--|--|
| SWS Item | MCU184_Conf : | | |
| 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 | MCU185_Conf : | | |
| 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 | MCU186_Conf : | | |
| 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: Module | |

No Included Containers

10.3 Published Information

[MCU001_PI] The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules[3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1].

Additional module-specific published parameters are listed below if applicable.

11 Changes to Release 3.0

11.1 Deleted SWS Items

| SWS Item | Rationale |
|-----------------|-----------------------|
| MCU105 | No Requirement of Mcu |
| | |
| | |

11.2 Replaced SWS Items

| SWS Item | replaced by SWS Item | Rationale |
|-----------------|--------------------------------------|----------------------------|
| MCU0149 | MCU049 | Wrong Id number |
| MCU166 | MCU166_Conf | Corrected the duplication. |
| MCU108 | MCU108, 211, 213, 215, 216, 218, 219 | Made Req atomic |
| MCU109 | MCU109, 220 | // |
| MCU110 | MCU110, 225 | // |
| MCU049 | MCU049, 226 | // |
| MCU116 | MCU116,244,245,246,247 | // |

All the SWS IDs in chapter10 have been modified by TO. It was added the suffix _Conf to avoid names duplicated.

11.3 Changed SWS Items

| SWS Item | Rationale |
|---|--|
| MCU147 | Rephrased |
| MCU053 | Rephrased |
| MCU055 | Made Atomic |
| MCU134 | Reformulated |
| MCU011 | // |
| MCU110 | // |
| MCU129 | // |
| MCU130 | // |
| MCU171_Conf | Changes in the configuration class |
| MCU179_Conf | Rephrased to increase the understandability- |
| MCU164 | Made Atomic |
| MCU054 | // |
| MCU171_Conf, MCU172_Conf, MCU173_Conf, MCU176_Conf, MCU177_Conf, MCU178_Conf, MCU179_Conf | Min and Max values in the Range field changed. |

11.4 Added SWS Items

| SWS Item | Rationale |
|-----------------------------|---|
| MCU200,201,202,203 | Inserted to cover the debugging concept |
| MCU204 | Null Pointer checking in GetVersionInfo |
| MCU205,206,180_Conf | Inserted to disable the PLL related APIs |
| MCU207,208,209,181_Conf | Inserted to read the Ram Status |
| MCU182_Conf,MCU210 | Switch to enable/disable the Mcu_InitClock Api |
| MCU230,231 | Added atomic IDs in the datatype section |
| MCU232,233 | // |
| MCU234 | // |
| MCU235,236 | // |
| MCU237,238 | // |
| MCU239,240 | // |
| MCU242,243 | // |
| MCU248 | Inserted a new section for the clock in chapter 7. A new note for the ClockReference concept was inserted as well |
| MCU183_Conf | Inserted new ID for the clock to be passed to Mcu_InitClock |
| MCU187_Conf, MCU188_Conf | Inserted reference to DemEventParameter container in the Dem module |
| MCU184_Conf | Inserted container McuPublishedInformation to group all the MCU specific published information |
| MCU185_Conf,186_Conf | Inserted a new container and a new configuration parameter to specify all the different type of reset that a specified Micro supports |
| MCU001_PI | Rework of Published Information |