1. Introduction

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]

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

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.

5. Dependencies to other modules

7. Functional specification

7.1 General behavior

7.1.1 The MCU driver provides MCU services for Clock and RAM initialization. In the MCU configuration set, the MCU specific settings for the Clock (i.e. PLL setting) and RAM (i.e. section base address and size) shall be configured.

7.1.2 Requirements

7.1.2.1 Reset

SWS_Mcu_00055. The MCU module shall provide a service to provide software triggering of a hardware reset. Note: Only an authorized user shall be able to call this reset service function.

SWS_Mcu_00052. The MCU module shall provide services to get the reset reason of the last reset if the hardware supports such a feature. Note: In an ECU, there are several sources which can cause a reset. Depending on the reset reason, several application scenarios might be necessary after reinitialization of the MCU.

7.1.2.2 Clock

SWS_Mcu_00248. Mcu shall provide a service to enable and set the MCU clock. (i.e. Cpu clock, Peripheral Clock, Prescalers, Multipliers have to be configured in the MCU). Note: All the available peripheral clocks have to be made available to the other BSW modules via the McuClockReferencePoint container.

7.1.2.3 MCU Mode service

SWS_Mcu_00164. The MCU module shall provide a service to activate MCU reduced power modes. The service, which activates the reduced power mode, shall allow access to power modes available in the uC hardware.

SWS_Mcu_00165. The number of modes and the configuration is MCU dependent and shall be configured in the configuration set of the MCU module. Note: The activation of MCU reduced power modes might influence the PLL, the internal oscillator, the CPU clock, uC peripheral clock and the power supply for core and peripherals.

7.2 Error clasification

SWS_Mcu_00051. The MCU driver follows the standardized AUTOSAR concept to report production errors. The provided callback routines are specified in the Diagnostic Event Manager (DEM) specification (see 6).

SWS_Mcu_00226. Production Errors shall not be used as the return value of the called function.

7.2.1 Development Errors

SWS_Mcu_00012

Type of error	Related error code	Error value
API service called with wrong parameter	MCU_E_PARAM_CONFIG	0x0A
API service called with wrong parameter	MCU_E_PARAM_CLOCK	0x0B

API service called with wrong parameter	MCU_E_PARAM_MODE	0x0C
API service called with wrong parameter	MCU_E_PARAM_RAMSECTION	0x0D
API service called with wrong parameter	MCU_E_PLL_NOT_LOCKED	0x0E
API service called with wrong parameter	MCU_E_UNINIT	0x0F
API service called with wrong parameter	MCU_E_PARAM_POINTER	0x10
API service called with wrong parameter	MCU_E_INIT_FAILED	0x11

7.2.5 Extended Production Errors

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

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

7.2.5.1 MCU_E_CLOCK_FAILURE

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

SWS_Mcu_00257. Fail criteria for MCU_E_CLOCK_FAILURE: a clock source failure occurs.

SWS_Mcu_00258. Pass criteria for MCU_E_CLOCK_FAILURE: no clock source failure occurs.

8. API Specification

8.1 Imported types

SWS_Mcu_00152

Module	Header file	Imported type
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

8.2 Type definitions

8.2.1 Mcu_ConfigType

SWS_Mcu_00249

8.3.5 Mcu_GetPllStatus

SWS_Mcu_00157

Service Name	Mcu_GetPllStatus
Syntax	Mcu_PllStatusType Mcu_GetPllStatus (void)
Service ID [hex]	0x04
Sync/Async	Synchronous
Reentrancy	Reentrant

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

SWS_Mcu_00008
[The function Mcu_GetPllStatus shall return the lock status of the PLL.

SWS_Mcu_00132
The function Mcu_GetPllStatus shall return MCU_PLL_STATUS_UNDEFINED if this function is called prior to calling of the function Mcu_Init.

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

8.3.6 Mcu_GetResetReason

SWS_Mcu_00158

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

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

SWS_Mcu_00133
The function Mcu_GetResetReason shall return MCU_RESET_UNDEFINED if this function is called prior to calling of the function Mcu_Init, and if supported by the hardware.

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

8.3.7 Mcu_GetResetRawValue

SWS_Mcu_00159

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

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

SWS_Mcu_00006
The function Mcu_GetResetRawValue shall read the reset raw value from the hardware register if the hardware supports this. If the hardware does not have a reset status register, the return value shall be 0x0.
Note: In case of multiple calls to this function the return value should always be the same.

8.3.8 Mcu_PerformReset

SWS_Mcu_00160

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

SWS_Mcu_00143
The function Mcu_PerformReset shall perform a microcontroller reset by using the hardware feature of the microcontroller.

SWS_Mcu_00144
The function Mcu_PerformReset shall perform the reset type which is configured in the configuration set.

SWS_Mcu_00145
The MCU module’s environment shall only call the function Mcu_PerformReset after the MCU module has been initialized by the function Mcu_Init.

SWS_Mcu_00146
[The function Mcu_PerformReset is only available if the precompile parameter McuPerformResetApi is set to TRUE. If set to FALSE, the function Mcu_PerformReset is not applicable. (see Section 10.2.2).

8.3.9 Mcu_SetMode

SWS_Mcu_00161

Service Name	Mcu_SetMode
Syntax	void Mcu_SetMode (Mcu_ModeType McuMode)
Service ID [hex]	0x08
Sync/Async	Synchronous
Reentrancy	Reentrant

Parameters (in)	McuMode	Set different MCU power modes configured in the configuration set
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service activates the MCU power modes.	
Available via	Mcu.h	

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

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

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

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

8.3.10 Mcu_GetVersionInfo

SWS_Mcu_00162

Service Name	Mcu_GetVersionInfo	
Syntax	void Mcu_GetVersionInfo (Std_VersionInfoType* versioninfo)	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	versioninfo	Pointer to where to store the version information of this module
Return value	None	

Description	This service returns the version information of this module.
Available via	Mcu.h

8.3.11 Mcu_GetRamState

SWS_Mcu_00207

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

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

SWS_Mcu_00208
The MCU module’s environment shall call this function only if the MCU module has been already initialized using the function MCU_Init.

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

8.6 Expected Interfaces

8.6.1 Mandatory Interfaces

SWS_Mcu_00166

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

8.6.2 Optional Interfaces

SWS_Mcu_00163

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

8.7 API parameter checking

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

SWS_Mcu_00019
ClockSetting shall be within the settings defined in the configuration data structure. Related error value: MCU_E_PARAM_CLOCK

SWS_Mcu_00020
McuMode shall be within the modes defined in the configuration data structure. Related error value: MCU_E_PARAM_MODE

SWS_Mcu_00021
RamSection shall be within the sections defined in the configuration data structure. Related error value: MCU_E_PARAM_RAMSECTION

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

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

10. Configuration specification

10.2 Containers and configuration parameters

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

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

SWS_Mcu_CONSTR_00001
The module will operate as an independent instance in each of the partitions, means the called API will only target the partition it is called in.