Document Title	SWS_MCUDriver: Complete Change Documentation 4.3.0 - 4.3.1
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	695

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Table of Contents

1 SWS_MCUDriver 3
1.1 Specification Item ECUC_Mcu_00120 3
1.2 Specification Item ECUC_Mcu_00190 6
1.3 Specification Item SWS_Mcu_00011 8
1.4 Specification Item SWS_Mcu_00017 11
1.5 Specification Item SWS_Mcu_00030 14
1.6 Specification Item SWS_Mcu_00125 17
1.7 Specification Item SWS_Mcu_00161 20

1 SWS_MCUDriver

1.1 Specification Item ECUC_Mcu_00120

Trace References:

none

Content:

Container Name	McuRamSectorSettingConfMcuRamSectorSettingConf
Description	This container contains the configuration (parameters) for the RAM Sector setting. Please see MCU030 for more information on RAM sec-tor settings.
Configuration Parameters	

Included parameters:

Included Parameters	
Parameter Name	SWS Item ID
McuRamDefaultValue	ECUC_Mcu_00177
McuRamSectionBaseAddress	ECUC_Mcu_00178
McuRamSectionSize	ECUC_Mcu_00179
McuRamSectionWriteSize	ECUC_Mcu_00190

Included containers:

No Included Containers

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #68751: MCU: Mcu_InitRamSection() and Mcu_GetRamState() API

  Problem description:

  In AUTOSAR_SWS_MCUDriver :- Specification of MCU Driver
  AUTOSAR Release 4.2.1

  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.

  In most of the implementations, the start-up code
  1. will clear the whole RAM or
  2. reads the reset factors(destructive/functional) and if destructive

reset like Power loss, Low Voltage Detection will clear the whole
RAM or if functional reset(like SW reset) may or may not clear the
RAM.
However in general, if it is an asynchronous reset, it is known
that RAM contents are not guaranteed.
3. Do min. amount of RAM initialisation of RAM and make use of
AUTOSAR API "Mcu_InitRamSection()" to do initialisation of RAM
based on Reset factors or Mcu_GetRamState() if supported by micro.

Once the above step is done in Startup, data/text sections are copied from Flash to RAM.

Latest next-gen microcontrollers have ECC(single or multiple bit) associated with RAM and requires the RAM to be initialized by performing 32-bit/64-bit write access by writing all 0's or special
kind of signature, otherwise it may lead to detection of ECC errors.

Lot of MCAL MCU implementations of Mcu_InitRamSection() uses 8-bit access to satisfy SWS_Mcu_00011 requirement, which violates the rule of ECC clearing as when they develop the source code normally the compiler generated C Code will take care of initialization of RAM.

There shall be a note under SWS_Mcu_00011 that Mcu_InitRamSection() shall be implemented based on the micro vendor implementation of RAM i.e. with ECC or without ECC i.e. Support 8-bit, 16-bit, 32-bit, 64-bit initialisation of data.

Also, based on the ECU(e.g. Display ECUs), there could be several types of RAM (with ECC) i.e.
1. Backup RAM(Protected/Retention/Battery Operated RAM or Keep Alive Memory)
2. Video RAM

SWS_Mcu_00207:Mcu_GetRamState() currently if supported by hardware returns the state of RAM alone. However, this needs to be enhanced to support other types of RAM state i.e. Backup RAM or Video RAM

Proposed Solution:

SWS_Mcu_00207: Change
Mcu_RamStateType Mcu_GetRamState(void)
to
Mcu_RamStateType Mcu_GetRamState(Mcu_RamType RamType)
where RamType
0 - SRAM

1 - Backup RAM
2 - Video RAM

SWS_Mcu_00011:
Add a note saying that Mcu_InitRamSection() shall be implemented based on the micro vendor suggestion/caution of initialising RAM with ECC.

Agreed solution:

-> New parameter "McuRamSectionWriteSize" shall be included in SWS Item ECUC_Mcu_00120, container McuRamSectorSettingConf.

Name: McuRamSectionWriteSize
Description: This parameter shall define the size in bytes of data which can be written into RAM at once.
Multiplicity: 1
Type: EcucIntegerParamDef
Default value: 8

Container McuRamSectorSettingConf in Chap. 10 to be updated to include the configuration parameter described above.

-> SWS_Mcu_00011 shall be updated to mention also the write size:
"The function Mcu_InitRamSection shall fill the memory from address McuRamSectionBaseAddress up to address McuRamSectionBaseAddress + McuRamSectionSize-1 with the byte-value contained in McuRamDefaultValue and by writing at once a number of byte defined by McuRamSectionWriteSize, where McuRamSectionBaseAddress,
McuRamSectionSize, McuRamDefaultValue and McuRamSectionWriteSize are the values of the configuration parameters for each RamSection.

-> SWS_Mcu_00030 shall be updated: extend enumeration with "RAM write size".
–Last change on issue 68751 comment 45–

BW-C-Level:

Application	**Specification**	**Bus**
1	3	1

1.2 Specification Item ECUC_Mcu_00190

Trace References:

none

Content:

Name	McuRamSectionWriteSizeMcuRamSectorSettingConf.McuRamSectionWriteSize		
Parent Container	McuRamSectorSettingConf		
Description	This parameter shall define the size in bytes of data which can be written into RAM at once.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	8		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #68751: MCU: Mcu_InitRamSection() and Mcu_GetRamState() API

  Problem description:

  In AUTOSAR_SWS_MCUDriver :- Specification of MCU Driver
  AUTOSAR Release 4.2.1

  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.

  In most of the implementations, the start-up code
  1. will clear the whole RAM or
  2. reads the reset factors(destructive/functional) and if destructive
  reset like Power loss, Low Voltage Detection will clear the whole
  RAM or if functional reset(like SW reset) may or may not clear the
  RAM.
  However in general, if it is an asynchronous reset, it is known

that RAM contents are not guaranteed.
3. Do min. amount of RAM initialisation of RAM and make use of
AUTOSAR API "Mcu_InitRamSection()" to do initialisation of RAM
based on Reset factors or Mcu_GetRamState() if supported by micro.

Once the above step is done in Startup, data/text sections are copied from Flash to RAM.

Latest next-gen microcontrollers have ECC(single or multiple bit) associated with RAM and requires the RAM to be initialized by performing 32-bit/64-bit write access by writing all 0's or special
kind of signature, otherwise it may lead to detection of ECC errors.

Lot of MCAL MCU implementations of Mcu_InitRamSection() uses 8-bit access to satisfy SWS_Mcu_00011 requirement, which violates the rule of ECC clearing as when they develop the source code normally the compiler generated C Code will take care of initialization of RAM.

There shall be a note under SWS_Mcu_00011 that Mcu_InitRamSection() shall be implemented based on the micro vendor implementation of RAM i.e. with ECC or without ECC i.e. Support 8-bit, 16-bit, 32-bit, 64-bit initialisation of data.

Also, based on the ECU(e.g. Display ECUs), there could be several types of RAM (with ECC) i.e.
1. Backup RAM(Protected/Retention/Battery Operated RAM or Keep Alive Memory)
2. Video RAM

SWS_Mcu_00207:Mcu_GetRamState() currently if supported by hardware returns the state of RAM alone. However, this needs to be enhanced to support other types of RAM state i.e. Backup RAM or Video RAM

Proposed Solution:

SWS_Mcu_00207: Change
Mcu_RamStateType Mcu_GetRamState(void)
to
Mcu_RamStateType Mcu_GetRamState(Mcu_RamType RamType)
where RamType
0 - SRAM
1 - Backup RAM
2 - Video RAM

SWS_Mcu_00011:

Add a note saying that Mcu_InitRamSection() shall be implemented based on the micro vendor suggestion/caution of initialising RAM with ECC.

Agreed solution:

-> New parameter "McuRamSectionWriteSize" shall be included in SWS Item ECUC_Mcu_00120, container McuRamSectorSettingConf.

Name: McuRamSectionWriteSize
Description: This parameter shall define the size in bytes of data which can be written into RAM at once.
Multiplicity: 1
Type: EcucIntegerParamDef
Default value: 8

Container McuRamSectorSettingConf in Chap. 10 to be updated to include the configuration parameter described above.

-> SWS_Mcu_00011 shall be updated to mention also the write size:
"The function Mcu_InitRamSection shall fill the memory from address McuRamSectionBaseAddress up to address McuRamSectionBaseAddress + McuRamSectionSize-1 with the byte-value contained in McuRamDefaultValue and by writing at once a number of byte defined by McuRamSectionWriteSize, where McuRamSectionBaseAddress,
McuRamSectionSize, McuRamDefaultValue and McuRamSectionWriteSize are the values of the configuration parameters for each RamSection.

-> SWS_Mcu_00030 shall be updated: extend enumeration with "RAM write size".
–Last change on issue 68751 comment 45–

BW-C-Level:

Application	Specification	Bus
1	3	1

1.3 Specification Item SWS_Mcu_00011

Trace References:

none

Content:

The function Mcu_InitRamSection shall fill the memory from address McuRamSectionBaseAddress up to address McuRamSectionBaseAddress + McuRamSectionSize-1 with the byte-value contained in McuRamDefaultValue and by writing at once a number of bytes defined by McuRamSectionWriteSize, where McuRamSectionBaseAddress, McuRamSectionSizeand , McuRamDefaultValue and McuRamSectionWriteSize are the values of the configuration parameters for each RamSection (see SWS_Mcu_00030).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #68751: MCU: Mcu_InitRamSection() and Mcu_GetRamState() API

  Problem description:

  In AUTOSAR_SWS_MCUDriver :- Specification of MCU Driver
  AUTOSAR Release 4.2.1

  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.

  In most of the implementations, the start-up code
  1. will clear the whole RAM or
  2. reads the reset factors(destructive/functional) and if destructive
  reset like Power loss, Low Voltage Detection will clear the whole
  RAM or if functional reset(like SW reset) may or may not clear the
  RAM.
  However in general, if it is an asynchronous reset, it is known
  that RAM contents are not guaranteed.
  3. Do min. amount of RAM initialisation of RAM and make use of
  AUTOSAR API "Mcu_InitRamSection()" to do initialisation of RAM
  based on Reset factors or Mcu_GetRamState() if supported by micro.

  Once the above step is done in Startup, data/text sections are copied from Flash to RAM.

  Latest next-gen microcontrollers have ECC(single or multiple bit) associated with RAM and requires the RAM to be initialized by performing 32-bit/64-bit write access by writing all 0's or special
  kind of signature, otherwise it may lead to detection of ECC errors.

  Lot of MCAL MCU implementations of Mcu_InitRamSection() uses 8-bit access to satisfy SWS_Mcu_00011 requirement, which violates the rule of ECC clearing as when they develop the source code normally the compiler generated C Code will take care of initialization of RAM.

There shall be a note under SWS_Mcu_00011 that Mcu_InitRamSection() shall be implemented based on the micro vendor implementation of RAM i.e. with ECC or without ECC i.e. Support 8-bit, 16-bit, 32-bit, 64-bit initialisation of data.

Also, based on the ECU(e.g. Display ECUs), there could be several types of RAM (with ECC) i.e.
1. Backup RAM(Protected/Retention/Battery Operated RAM or Keep Alive Memory)
2. Video RAM

SWS_Mcu_00207:Mcu_GetRamState() currently if supported by hardware returns the state of RAM alone. However, this needs to be enhanced to support other types of RAM state i.e. Backup RAM or Video RAM

Proposed Solution:

SWS_Mcu_00207: Change
Mcu_RamStateType Mcu_GetRamState(void)
to
Mcu_RamStateType Mcu_GetRamState(Mcu_RamType RamType)
where RamType
0 - SRAM
1 - Backup RAM
2 - Video RAM

SWS_Mcu_00011:
Add a note saying that Mcu_InitRamSection() shall be implemented based on the micro vendor suggestion/caution of initialising RAM with ECC.

Agreed solution:

-> New parameter "McuRamSectionWriteSize" shall be included in SWS Item ECUC_Mcu_00120, container McuRamSectorSettingConf.

Name: McuRamSectionWriteSize
Description: This parameter shall define the size in bytes of data which can be written into RAM at once.
Multiplicity: 1
Type: EcucIntegerParamDef
Default value: 8

Container McuRamSectorSettingConf in Chap. 10 to be updated to include the configuration parameter described above.

-> SWS_Mcu_00011 shall be updated to mention also the write size:
"The function Mcu_InitRamSection shall fill the memory from address McuRamSectionBaseAddress up to address McuRamSectionBaseAddress + McuRamSectionSize-1 with the byte-value contained in McuRamDefaultValue and by writing at once a number of byte defined by McuRamSectionWriteSize, where McuRamSectionBaseAddress,
McuRamSectionSize, McuRamDefaultValue and McuRamSectionWriteSize are the values of the configuration parameters for each RamSection.

-> SWS_Mcu_00030 shall be updated: extend enumeration with "RAM write size".
–Last change on issue 68751 comment 45–

BW-C-Level:

Application	Specification	Bus
1	3	1

1.4 Specification Item SWS_Mcu_00017

Trace References:

none

Content:

If the default 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.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"

- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.5 Specification Item SWS_Mcu_00030

Trace References:

none

Content:

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
- RAM write size

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #68751: MCU: Mcu_InitRamSection() and Mcu_GetRamState() API

  Problem description:

  In AUTOSAR_SWS_MCUDriver :- Specification of MCU Driver
  AUTOSAR Release 4.2.1

  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.

  In most of the implementations, the start-up code
  1. will clear the whole RAM or
  2. reads the reset factors(destructive/functional) and if destructive
  reset like Power loss, Low Voltage Detection will clear the whole
  RAM or if functional reset(like SW reset) may or may not clear the
  RAM.
  However in general, if it is an asynchronous reset, it is known
  that RAM contents are not guaranteed.
  3. Do min. amount of RAM initialisation of RAM and make use of
  AUTOSAR API "Mcu_InitRamSection()" to do initialisation of RAM
  based on Reset factors or Mcu_GetRamState() if supported by micro.

  Once the above step is done in Startup, data/text sections are copied from Flash to RAM.

  Latest next-gen microcontrollers have ECC(single or multiple bit) associated with RAM and requires the RAM to be initialized by performing 32-bit/64-bit write access by writing all 0's or special
  kind of signature, otherwise it may lead to detection of ECC errors.

  Lot of MCAL MCU implementations of Mcu_InitRamSection() uses 8-bit access to satisfy SWS_Mcu_00011 requirement, which violates the rule of ECC clearing as when they develop the source code normally the compiler generated C

Code will take care of initialization of RAM.

There shall be a note under SWS_Mcu_00011 that Mcu_InitRamSection() shall be implemented based on the micro vendor implementation of RAM i.e. with ECC or without ECC i.e. Support 8-bit, 16-bit, 32-bit, 64-bit initialisation of data.

Also, based on the ECU(e.g. Display ECUs), there could be several types of RAM (with ECC) i.e.
1. Backup RAM(Protected/Retention/Battery Operated RAM or Keep Alive Memory)
2. Video RAM

SWS_Mcu_00207:Mcu_GetRamState() currently if supported by hardware returns the state of RAM alone. However, this needs to be enhanced to support other types of RAM state i.e. Backup RAM or Video RAM

Proposed Solution:

SWS_Mcu_00207: Change
Mcu_RamStateType Mcu_GetRamState(void)
to
Mcu_RamStateType Mcu_GetRamState(Mcu_RamType RamType)
where RamType
0 - SRAM
1 - Backup RAM
2 - Video RAM

SWS_Mcu_00011:
Add a note saying that Mcu_InitRamSection() shall be implemented based on the micro vendor suggestion/caution of initialising RAM with ECC.

Agreed solution:

-> New parameter "McuRamSectionWriteSize" shall be included in SWS Item ECUC_Mcu_00120, container McuRamSectorSettingConf.

Name: McuRamSectionWriteSize
Description: This parameter shall define the size in bytes of data which can be written into RAM at once.
Multiplicity: 1
Type: EcucIntegerParamDef
Default value: 8

Container McuRamSectorSettingConf in Chap. 10 to be updated to include the configuration parameter described above.

-> SWS_Mcu_00011 shall be updated to mention also the write size:
"The function Mcu_InitRamSection shall fill the memory from address McuRamSectionBaseAddress up to address McuRamSectionBaseAddress + McuRamSectionSize-1 with the byte-value contained in McuRamDefaultValue and by writing at once a number of byte defined by McuRamSectionWriteSize, where McuRamSectionBaseAddress,
McuRamSectionSize, McuRamDefaultValue and McuRamSectionWriteSize are the values of the configuration parameters for each RamSection.

-> SWS_Mcu_00030 shall be updated: extend enumeration with "RAM write size".
–Last change on issue 68751 comment 45–

BW-C-Level:

Application	Specification	Bus
1	3	1

1.6 Specification Item SWS_Mcu_00125

Trace References:

none

Content:

If default 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.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."

- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer"

to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.7 Specification Item SWS_Mcu_00161

Trace References:

none

Content:

Service name:	Mcu_SetModeMcu_SetMode	
Syntax:	void Mcu_SetMode(Mcu_ModeType McuMode)	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	McuModeMcu_SetMode.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.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #72565: Harmonization of the EcuM and McuDriver software specifications

  Problem description:

  Name: Joel Thurlby
  Phone:
  Role: EcuM Developer

  Description/Motivation:

  The EcuM SWS specifies that the Mcu_SetMode function must be concurrently called by the EcuM running on the master and slave cores:
  - [SWS_EcuM_04024] Master Core Halt Sequence,
  - [SWS_EcuM_04025] Master Core Poll Sequence,
  - [SWS_EcuM_04026] Master Core WakeupRestart Sequence,
  - [SWS_EcuM_04028] Slave Core Halt Sequence,
  - [SWS_EcuM_04029] Slave Core Poll Sequence and
  - [SWS_EcuM_04030] Slave Core WakeupRestart Sequence

  The McuDriver SWS specifies though that the reentrancy of the Mcu_SetMode function is non-reentrant ([SWS_Mcu_00161]). This conflicts with the Bsw General Specification requirement [SWS_BSW_00191] Multi-core safety:

  If a BSW module entity shall be executable on multiple partitions (e.g. multiple cores), then the whole module entity code shall be concurrency safe

Proposed Solution:
- Set the Reentrancy level of Mcu_SetMode to Concurrency Safe.
- Specify the expected multicore behavior within the McuDriver SWS (which is currently not covered).

Was there already a decision? No.

Agreed solution:

Update requirement SWS_Mcu_00161 by changing the Reentrancy field for Mcu_SetMode API to "Reentrant"
–Last change on issue 72565 comment 16–

BW-C-Level:

Application	**Specification**	**Bus**
1	1	1