

MCAL User Manual for BASIC package

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

This User Manual is intended to enable users to integrate the Microcontroller Abstraction Layer (MCAL) software for the TriCore™ AURIX™ family of 32-bit microcontrollers.

This document describes responsibilities of integrator in-charge of integrating MCAL software with the basic software (BSW) stack. This document also provides detailed information on safety, configuration and functions along with examples of usage of significant features.

Intended audience

This document is intended for anyone using the BASIC package of the TC3xx MCAL software.

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, sub-menus
<i>Italics</i>	Denotes variable(s) and reference(s)
Courier	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
New	
>	Indicates that a cascading sub-menu opens when you select a menu item
[cover parentID=<alpha numeric value>]	Used for traceability completeness. Reader should ignore these.

Glossary of terms

Table 2 Glossary

Term	Description
ADC	Analog-to-digital converter
ATOM	ARU connected TOM (functional block of a microcontroller peripheral)
AUTOSAR	Automotive Open System Architecture
Bank	A Flash module contains separate banks. In the DFlash, there are one or more banks. Banks support concurrent operations with some limitations due to common logic.
BIFACES	BIFACES stands for Build and Integration Framework for Automotive Controller Embedded Software. It is an internally developed build framework for Infineon automotive microcontrollers software development.
BSW	Basic software
Burst write	The maximum amount of data that can be written with one command. The programming throughput is higher than for programming single pages. For DFlash, it is 4 pages (32 bytes).
CAN	Controller Area Network
CAN FD	CAN flexible data rate

About this document
Table 2 Glossary (continued)

Term	Description
CC	Communication controller
CCM	Cluster control module (functional block of GTM)
Channel	A channel is a software exchange medium for data that are defined with the same criteria: configuration parameters, number of data elements with same size and data pointers (source, destination) or location.
CHI	Communication host interface
CMU	Clock management unit (functional block of GTM)
CRC	Cyclic redundancy check
DEM	Diagnostics event manager (MCAL module)
DER	Destination error
DET	Development error tracer
DF_EEPROM	Data Flash dedicated for EEPROM emulation
DFlash	Data Flash
DIO	Digital input output
DMA	Direct memory access
DMU	Data memory unit
Driver	A driver is a BSW module located in the MCAL layer and contains the functionality to control and access an internal or an external device.
EB	Externally defined buffer for QSPI
ECC	Error correction code
EEPROM	Electrically erasable and programmable ROM (read only memory)
ERAY	FlexRay IP module (hardware)
ERU	External request unit
ESR	External service request (microcontroller pin)
ETH	Ethernet
EVADC	Enhanced versatile analog-to-digital converter
EVER	Bit indicating erase verify error
ex_r	Exclusive read access
ex_rw	Exclusive read and write access
ex_w	Exclusive write access
Fast-Mode	Triggering the watchdog hardware has to be done with a short timeout period. This mode can be used during normal operations of the ECU. For example, the watchdog hardware is configured for the Windows mode (watchdog triggering should happen within certain minimum/maximum boundaries within the timeout period) and a timeout period of 5 ms.
FCE	Flexible CRC engine
FCFS	First come first serve
FEE	Flash EEPROM emulation

About this document
Table 2 Glossary (continued)

Term	Description
FEE sector	The DF_EEPROM area is logically divided into two FEE sectors for implementing the double sector algorithm. If the QS region exists then the part of DF_EEPROM is used for QS blocks and remaining is used for double sector algorithm.
FIFO	First in first out
FLS	Flash
GC	Garbage collection
GEth	Gigabit Ethernet MAC
GTM	Generic timer module (hardware)
HOH	Hardware object (transmit/receive) handle
HRH	Hardware receive handle
HSCT	High-speed communications tunnel
HSPDM	High-speed pulse density modulation module
HTH	Hardware transmit handle
I/O	Input/Output
IB	Driver defined internal buffer / channel
IFX	Infineon Technologies
Internal job	FEE driver internal management activities, which are not explicitly triggered by the upper layer
ISR	Interrupt service routine
Job	A job is composed of one or several channels with the same chip select (is not released during the processing of job). A job is considered atomic and therefore cannot be interrupted by another job. A job has an assigned priority.
(Logical) block	Smallest erasable unit (4 K) as seen by the modules user. Consists of one or more virtual pages.
LPdu	Datalink layer protocol data unit
LPM	Low power mode
MAC	Media access control
MC-ISAR	Microcontroller Infineon Software Architecture
MCU	Microcontroller unit
MII	Media independent interface
Module	The element is composed of various software units called modules. Typically each software driver is referred to as module. More explicitly, it is also referred to as software module.
MRST	Master receive slave transmit
MTL	MAC transaction layer
MTSR	Master transmit slave receive
(Normal) write mode	In this mode, the maximum amount of data that can be written with one command is 8 byte (1 Page).
NVM	AUTOSAR NVRAM manager
NVRAM	Non-volatile RAM (random access memory)

About this document
Table 2 Glossary (continued)

Term	Description
NVRAM block	Management unit as seen by the NVRAM manager
OCU	Output compare unit
Off-Mode	Watchdog hardware is disabled / shut-down
OS	Operating system
Page	<p>A page is an aligned group of data double words plus an ECC extension. It is the smallest unit that can be programmed.</p> <p>DFlash: 1 data double word (8 bytes) plus 22-bit ECC extension.</p>
Peripheral	Hardware module used by a driver. A driver can use one or more peripherals.
PHY	Physical layer device (Ethernet transceiver)
Physical address	Address information in device-specific format (depending on the underlying Flash driver and device) that is used to access a logical block.
PLL	Phase lock loop
Pn_xxxxx	Port n register(xxxxx is register name)
POC	Protocol operation control
PORST	Power-on reset
PVER	Bit indicating program verify error
QS	Quasi-static
QSPI	Queued serial peripheral interface
r	Read access
RGMII	Reduced Gigabit Media Independent Interface
RMII	Reduced Media Independent Interface
rw	Read and write access
SchM	Scheduler manager (AUTOSAR module)
SCU	System control unit
Sequence	A sequence is a number of consecutive Jobs to transmit but it can be rescheduled between jobs using a priority mechanism. A sequence transmission is interruptible (by another sequence transmission) or not depending on a static configuration.
SER	Source error
Slow-Mode	Triggering the watchdog hardware can be done with a long timeout period. This mode can be used during system start-up / initialization phase. For example, the watchdog hardware is configured for toggle mode (no constraints on the point in time at which the triggering is done) and a timeout period of 10 ms.
SLSO	Programmable slave select outputs
SMU	Safety management unit
SPB	System peripheral bus
SRC	Service request control register
SRI	System resource interconnect
SRN	Service request node
STC	Set trigger condition API

About this document
Table 2 Glossary (continued)

Term	Description
STM	System timer (hardware)
TBU	Time base unit (functional block of GTM)
TECQED	Triple-bit error correction and quad-bit error detection
TIM	Timer input module (functional block of a microcontroller peripheral)
TOM	Timer output module (functional block of a microcontroller peripheral)
Unconfigured block	Data block which is stored in the DFlash (DF_EEPROM), but is not contained in the currently active configuration
User Job	User requested read/write/invalidate job
VariantLT	This variant allows a mix of pre-compile time, link-time configuration parameters. The intention of this variant is to optimize the parameters configuration for an object code delivery.
VariantPB	This variant allows a mix of pre-compile time, post-build-time and link time configuration parameters. The intention of this variant is to optimize the parameters configuration for a re-loadable binary
VarinatPC	This variant allows only pre-compile configuration parameters. The intention of this variant is to optimize the parameters configuration for a source code delivery.
Virtual address	Consisting of 16-bit block number and 16-bit offset inside the logical block
Virtual page	May consist of one or several physical pages to ease handling of logical blocks and address calculation
w	Write access
WDG	Watchdog
WDGx	Watchdog timer for CPUx
WDT	Watchdog timer (hardware)
Word-line (WL)	An aligned group of bytes. In DFLASH, 512 bytes are in the single-ended mode and 256 bytes are in the complement-sensing mode.
WUT	WakeUp timer

Reference documents

This User Manual should be read in conjunction with the following documents:

- AURIX™ TC3xx User Manual, V1.2.0, 2019-04, Infineon Technologies Munich AG
- AURIX™ TC38x Appendix to User Manual, V1.2.0, 2019-04, Infineon Technologies Munich AG
- AURIX™ TC39x-B Appendix to User Manual, V1.2.0, 2019-04, Infineon Technologies Munich AG
- AURIX™ TC35x Appendix to User Manual, V1.2.0, 2019-04, Infineon Technologies Munich AG
- AURIX™ TC37xEXT Appendix to User Manual, V1.2.0, 2019-04, Infineon Technologies Munich AG
- AURIX™ TC37x Appendix to User Manual, V1.2.0, 2019-04, Infineon Technologies Munich AG
- AURIX™ TC36x Appendix to User Manual, V1.2.0, 2019-04, Infineon Technologies Munich AG
- TC35x_AA_Errata_Sheet, Rel1.2, 2019-07-19, Infineon Technologies Munich AG
- TC35x_AB_Errata_Sheet, Rel1.1, 2019-07-19, Infineon Technologies Munich AG
- TC37xEXT_AA_Errata_Sheet, Rel1.2, 2019-07-19, Infineon Technologies Munich AG
- TC37xEXT_AB_Errata_Sheet, Rel1.1, 2019-07-19, Infineon Technologies Munich AG
- TC39x_BC_Errata_Sheet, Rel1.2, 2019-07-19, Infineon Technologies Munich AG
- TC39x_BB_Errata_Sheet, Rel1.3, 2019-07-19, Infineon Technologies Munich AG

About this document

- TC39x_BA_Errata_Sheet, Rel1.5, 2019-07-19, Infineon Technologies Munich AG
- TC39x_AA_Errata_Sheet, Rel1.8, 2019-03-29, Infineon Technologies Munich AG
- TC38x_AD_Errata_Sheet, Rel1.2, 2019-07-19, Infineon Technologies Munich AG
- TC38x_AC_Errata_Sheet, Rel1.2, 2019-07-19, Infineon Technologies Munich AG
- TC38x_AB_Errata_Sheet, Rel1.3, 2019-07-19, Infineon Technologies Munich AG
- TC38x_AA_Errata_Sheet, Rel1.5, 2019-07-19, Infineon Technologies Munich AG
- Note: *The errata analysis (TC3xx_SW_MCAL_HWErrataAnalysis.xlsx, v1.30.0_12.0) for TC3xx devices is placed inside the BASIC package.*
- Specification of ADC Driver, AUTOSAR_SWS_ADCDriver.pdf, AUTOSAR Release 4.2.2
- Specification of CAN Driver, AUTOSAR_SWS_CANDriver.pdf, AUTOSAR Release 4.2.2
- TLE9251V (Datasheet of CanTrcv_17_W9251), Rev. 1.1, 2018-05-23, Infineon Technologies Munich AG
- TLE9255W (Datasheet of CanTrcv_17_W9255), Rev. 1.03, 2018-10-25, Infineon Technologies Munich AG
- Specification of CRC Driver, AUTOSAR_SWS_CRCLibrary.pdf, AUTOSAR Release 4.2.2
- Specification of DIO Driver, AUTOSAR_SWS_DIODriver.pdf, AUTOSAR Release 4.2.2
- Specification of Flash EEPROM Emulation, AUTOSAR_SWS_Flash_EEPROM_Emulation.pdf, AUTOSAR Release 4.2.2
- Specification of GPT Driver, AUTOSAR_SWS_GPTDriver.pdf, AUTOSAR Release 4.2.2
- Specification of Flash Driver, AUTOSAR_SWS_FlashDriver.pdf, AUTOSAR Release 4.2.2
- Specification of ICU Driver, AUTOSAR_SWS_ICUDriver.pdf, AUTOSAR Release 4.2.2
- Specification of MCU Driver, AUTOSAR_SWS_MCUDriver.pdf, AUTOSAR Release 4.2.2
- Specification of OCU Driver, AUTOSAR_SWS_OCUDriver.pdf, AUTOSAR Release 4.2.2
- Specification of PORT Driver, AUTOSAR_SWS_PORTDriver.pdf, AUTOSAR Release 4.2.2
- Specification of PWM Driver, AUTOSAR_SWS_PWMDriver.pdf, AUTOSAR Release 4.2.2
- Specification of SPI Driver, AUTOSAR_SWS_SPIDriver.pdf, AUTOSAR Release 4.2.2
- Specification of WDG Driver, AUTOSAR_SWS_WatchdogDriver.pdf, AUTOSAR Release 4.2.2

Table of contents**Table of contents**

About this document	1
Table of contents	7
1 Generic information.....	78
1.1 Application integration	78
1.1.1 Package installation and project creation	78
1.1.1.1 Installing MCAL package	78
1.1.1.2 EB tresos™: license handling	82
1.1.1.3 EB tresos: Eclipse handling	83
1.1.1.4 Creating an EB tresos project	84
1.1.2 Typical module parameter configuration	89
1.1.2.1 Configuring parameters	89
1.1.2.2 Generating configuration files	91
1.1.3 Building and linking to generate final application binary	92
1.1.3.1 Building package	92
1.1.3.2 Release folder structure	93
1.1.3.3 Makefile adaptations	95
1.1.3.4 Linker script adaptations	96
1.1.3.5 Initiation of compiling and linking	97
1.1.3.6 Flashing the image	99
1.2 Multicore support	106
1.2.1 Multicore initialization	107
1.2.2 Multicore de-initialization	108
1.2.3 Multicore runtime	109
1.2.4 Multicore interrupts	110
1.2.5 Resource Manager (ResourceM)	111
1.2.5.1 Description	111
1.2.5.2 Configuration containers and parameters	112
1.2.5.2.1 Container: ResourceM	112
1.2.5.2.2 Container: ResourceMAllocation	112
1.2.5.2.3 Container: ResourceMGeneral	114
1.2.5.2.4 Container: ResourceMMcalConfig	114
1.2.5.2.5 Container: ResourceMMcalCore	115
1.2.5.3 Code generator plugin files	115
1.2.5.4 Example usage	116
1.2.5.4.1 Configuring master core	116
1.2.5.4.2 Configuring cores	117
1.2.5.4.3 Configuring resources to core	117
1.3 User mode execution	118
1.3.1 Atomic update of the Supervisor mode register when CPU is in the User-1 mode	118

Table of contents

1.3.2	Protected Supervisor mode register update when CPU is in the Supervisor mode	119
1.3.3	Protected Supervisor mode register update when CPU is in the User-1 mode	120
1.3.4	Read Supervisor mode register when CPU is in the User-1 mode	121
1.3.5	Write to Supervisor mode register when CPU is in the User-1 mode	122
1.4	Execution context of functions	123
1.5	Memory mapping	124
1.5.1	Memory mapping for code	125
1.5.2	Memory mapping for callout code	125
1.5.3	Memory mapping for configuration data	126
1.5.4	Memory mapping for constants	126
1.5.5	Memory mapping for variables	126
1.6	Variation point support	127
1.6.1	Configuring the Sidebar view	127
1.6.2	Configuring EcuC module	130
1.6.3	Configuring modules	132
1.6.4	Post-build variant multiplicity and Multiplicity configuration class	134
2	Safety view	136
2.1	Safety objectives	136
2.1.1	Safety objective 1	136
2.1.1.1	Safety realization	136
2.1.1.1.1	McDrivers, I/O drivers, ComDrivers, Complex drivers, AUTOSAR libraries	137
2.1.1.1.2	QM ComDrivers	137
2.1.1.1.3	MemDrivers	138
2.1.2	Safety objective 2	138
2.1.2.1	Safety realization	139
2.2	Common Assumptions of Use (AoUs)	139
2.3	Mapping of MCAL to external safety mechanism (ESM)	143
2.3.1	Reporting of unintended service requests	144
3	ADC driver	147
3.1	User information	147
3.1.1	Description	147
3.1.2	Hardware-software mapping	147
3.1.2.1	EVADC: primary hardware peripheral	148
3.1.2.2	SRC: dependent hardware peripheral	149
3.1.2.3	Port: dependent hardware peripheral	149
3.1.2.4	SCU: dependent hardware peripheral	150
3.1.2.5	EICR/IGCR: primary hardware peripheral	150
3.1.2.6	GTM: dependent hardware peripheral	150
3.1.2.7	CONVERTER: primary hardware peripheral	151
3.1.2.8	CCU6: dependent hardware peripheral	151
3.1.2.9	DMA: dependent hardware peripheral	151

Table of contents

3.1.3	File structure	152
3.1.3.1	C file structure	152
3.1.3.2	Code generator plugin files	153
3.1.4	Integration hints	154
3.1.4.1	Integration with AUTOSAR stack	154
3.1.4.2	Multicore and Resource Manager	160
3.1.4.3	MCU support	161
3.1.4.4	Port support	161
3.1.4.5	DMA support	161
3.1.4.6	Interrupt connections	162
3.1.4.7	Example usage	166
3.1.5	Key architectural considerations	174
3.1.5.1	Modes of operation: priority and queuing	174
3.1.5.2	Modes of operation: result handling	175
3.1.5.3	Re-entrancy of APIs	176
3.1.5.4	Accessing shared SFR	176
3.1.5.5	Hardware trigger and gate mechanism	176
3.2	Assumptions of Use (AoUs)	177
3.3	Reference information	179
3.3.1	Configuration interfaces	179
3.3.1.1	Container: Adc	180
3.3.1.1.1	Config Variant	180
3.3.1.2	Container: AdcChannel	180
3.3.1.2.1	AdcAnChannelNum	180
3.3.1.2.2	AdcBWDEnable	181
3.3.1.2.3	AdcBWDPrechargeLevel	181
3.3.1.2.4	AdcChannelConvTime	182
3.3.1.2.5	AdcChannelHighLimit	182
3.3.1.2.6	AdcChannelId	183
3.3.1.2.7	AdcChannelLimitCheck	183
3.3.1.2.8	AdcChannelLowLimit	184
3.3.1.2.9	AdcChannelRangeSelect	185
3.3.1.2.10	AdcChannelRefVoltsrcHigh	185
3.3.1.2.11	AdcChannelRefVoltsrcLow	186
3.3.1.2.12	AdcChannelResolution	186
3.3.1.2.13	AdcChannelSampTime	187
3.3.1.2.14	AdcInputClassSelection	187
3.3.1.2.15	AdcSyncConvChannelEnable	188
3.3.1.3	Container: AdcConfigSet	188
3.3.1.3.1	AdcSyncClockDisable	189
3.3.1.3.2	AdcSystemClock	189
3.3.1.4	Container: AdcDemEventParameterRefs	190

Table of contents

3.3.1.4.1	AdcClcFailureNotification	190
3.3.1.4.2	AdcConvStopTimeNotification	190
3.3.1.5	Container: AdcGeneral	191
3.3.1.5.1	AdcDeInitApi	191
3.3.1.5.2	AdcDevErrorDetect	191
3.3.1.5.3	AdcEnableLimitCheck	192
3.3.1.5.4	AdcEnableQueueing	192
3.3.1.5.5	AdcEnableStartStopGroupApi	193
3.3.1.5.6	AdcGrpNotifCapability	193
3.3.1.5.7	AdcHwTriggerApi	194
3.3.1.5.8	AdcInitCheckApi	195
3.3.1.5.9	AdcInitDeInitApiMode	195
3.3.1.5.10	AdcLowPowerStatesSupport	196
3.3.1.5.11	AdcMaxChConvTimeCount	196
3.3.1.5.12	AdcMultiCoreErrorDetect	197
3.3.1.5.13	AdcPowerStateAsynchTransitionMode	197
3.3.1.5.14	AdcPriorityImplementation	198
3.3.1.5.15	AdcReadGroupApi	198
3.3.1.5.16	AdcResultAlignment	199
3.3.1.5.17	AdcResultHandlingImplementation	199
3.3.1.5.18	AdcRuntimeApiMode	200
3.3.1.5.19	AdcSafetyEnable	200
3.3.1.5.20	AdcSleepMode	201
3.3.1.5.21	AdcStartupCalibApi	201
3.3.1.5.22	AdcSupplyVoltage	202
3.3.1.5.23	AdcSyncConvEnable	203
3.3.1.5.24	AdcTriggerOneConversionEnable	203
3.3.1.5.25	AdcVersionInfoApi	204
3.3.1.6	Container: AdcGlobalInputClass	204
3.3.1.6.1	AdcChConvMode	204
3.3.1.6.2	AdcChPreChargeClkCycles	205
3.3.1.6.3	AdcChSESPSEnable	205
3.3.1.6.4	AdcChSampleTime	206
3.3.1.7	Container: AdcGroup	206
3.3.1.7.1	AdcChannel0Alias	207
3.3.1.7.2	AdcChannel1Alias	207
3.3.1.7.3	AdcGroupAccessMode	208
3.3.1.7.4	AdcGroupConversionMode	208
3.3.1.7.5	AdcGroupDefinition	209
3.3.1.7.6	AdcGroupId	209
3.3.1.7.7	AdcGroupPriority	210
3.3.1.7.8	AdcGroupReplacement	211

Table of contents

3.3.1.7.9	AdcGroupTriggSrc	211
3.3.1.7.10	AdcHwExtGateSelect	212
3.3.1.7.11	AdcHwExtTrigSelect	213
3.3.1.7.12	AdcHwGateSignal	213
3.3.1.7.13	AdcHwTrigSignal	214
3.3.1.7.14	AdcHwTrigTimer	215
3.3.1.7.15	AdcNotification	215
3.3.1.7.16	AdcResRegDefinition	216
3.3.1.7.17	AdcStreamingBufferMode	216
3.3.1.7.18	AdcStreamingNumSamples	217
3.3.1.8	Container: AdcHwUnit	218
3.3.1.8.1	AdcAnalogClockSyncDelay	218
3.3.1.8.2	AdcCalibrationSampleTime	218
3.3.1.8.3	AdcClockSource	219
3.3.1.8.4	AdcHwUnitId	219
3.3.1.8.5	AdcIdlePrechargeEnable	220
3.3.1.8.6	AdcInputBufferEnable	221
3.3.1.8.7	AdcMSBDoubleClkEnable	221
3.3.1.8.8	AdcPostCalibrationDisable	222
3.3.1.8.9	AdcPrechargeReference	222
3.3.1.8.10	AdcPrescale	223
3.3.1.8.11	AdcReferencePrechargePhases	223
3.3.1.8.12	AdcRequestSource0ConvMode	224
3.3.1.8.13	AdcRequestSource1ConvMode	224
3.3.1.8.14	AdcRequestSource2ConvMode	225
3.3.1.8.15	AdcSampleSyncEnable	225
3.3.1.8.16	AdcSyncConvMode	226
3.3.1.9	Container: AdcHwUnitInputClass	226
3.3.1.9.1	AdcChConvMode	227
3.3.1.9.2	AdcChPreChargeClkCycles	227
3.3.1.9.3	AdcChSESPSEnable	228
3.3.1.9.4	AdcChSampleTime	228
3.3.1.10	Container: AdcPowerStateConfig	229
3.3.1.10.1	AdcPowerState	229
3.3.1.10.2	AdcPowerStateReadyCbkRef	229
3.3.1.11	Container: AdcPublishedInformation	230
3.3.1.11.1	AdcChannelValueSigned	230
3.3.1.11.2	AdcGroupFirstChannelFixed	231
3.3.1.11.3	AdcMaxChannelResolution	231
3.3.1.12	Container: CommonPublishedInformation	231
3.3.1.12.1	ArMajorVersion	232
3.3.1.12.2	ArMinorVersion	232

Table of contents

3.3.1.12.3	ArPatchVersion	232
3.3.1.12.4	ModuleId	233
3.3.1.12.5	Release	233
3.3.1.12.6	SwMajorVersion	234
3.3.1.12.7	SwMinorVersion	234
3.3.1.12.8	SwPatchVersion	235
3.3.1.12.9	VendordId	235
3.3.1.13	Container: EruGatingConfig	235
3.3.1.13.1	AdcEruErsInputPin	236
3.3.1.13.2	AdcEruErsRef	236
3.3.1.13.3	AdcEruOguRef	236
3.3.1.14	Container: EruTriggerConfig	237
3.3.1.14.1	AdcEruErsInputPin	237
3.3.1.14.2	AdcEruErsRef	238
3.3.1.14.3	AdcEruOguRef	238
3.3.1.15	Container: GtmGatingTimerConfig	238
3.3.1.15.1	GtmTimerCM0Ticks	239
3.3.1.15.2	GtmTimerClockSelect	239
3.3.1.15.3	GtmTimerTimePeriod	240
3.3.1.15.4	GtmTimerUsed	240
3.3.1.16	Container: GtmTriggerTimerConfig	241
3.3.1.16.1	GtmTimerCM0Ticks	241
3.3.1.16.2	GtmTimerClockSelect	242
3.3.1.16.3	GtmTimerTimePeriod	242
3.3.1.16.4	GtmTimerUsed	243
3.3.2	Functions - Type definitions	243
3.3.2.1	Adc_ChannelRangeSelectType	243
3.3.2.2	Adc_ChannelType	244
3.3.2.3	Adc_ConfigType	244
3.3.2.4	Adc_ConversionTimeType	245
3.3.2.5	Adc_GroupAccessModeType	245
3.3.2.6	Adc_GroupConvModeType	245
3.3.2.7	Adc_GroupDefType	246
3.3.2.8	Adc_GroupPriorityType	246
3.3.2.9	Adc_GroupReplacementType	247
3.3.2.10	Adc_GroupType	247
3.3.2.11	Adc_HwTriggerSignalType	248
3.3.2.12	Adc_HwTriggerTimerType	248
3.3.2.13	Adc_NotifyFnPtrType	248
3.3.2.14	Adc_PowerStateRequestResultType	249
3.3.2.15	Adc_PowerStateType	249
3.3.2.16	Adc_PrescaleType	250

Table of contents

3.3.2.17	Adc_PriorityImplementationType	250
3.3.2.18	Adc_ResolutionType	250
3.3.2.19	Adc_ResultAlignmentType	251
3.3.2.20	Adc_SamplingTimeType	251
3.3.2.21	Adc_StartupCalibStatusType	251
3.3.2.22	Adc_StatusType	252
3.3.2.23	Adc_StreamBufferModeType	252
3.3.2.24	Adc_StreamNumSampleType	253
3.3.2.25	Adc_SyncConvModeType	253
3.3.2.26	Adc_TriggerSourceType	254
3.3.2.27	Adc_ValueGroupType	254
3.3.3	Functions - APIs	254
3.3.3.1	Adc_Init	254
3.3.3.2	Adc_DelInit	256
3.3.3.3	Adc_SetupResultBuffer	257
3.3.3.4	Adc_EnableGroupNotification	258
3.3.3.5	Adc_DisableGroupNotification	259
3.3.3.6	Adc_StartGroupConversion	260
3.3.3.7	Adc_StopGroupConversion	261
3.3.3.8	Adc_EnableHardwareTrigger	262
3.3.3.9	Adc_DisableHardwareTrigger	264
3.3.3.10	Adc_GetGroupStatus	265
3.3.3.11	Adc_ReadGroup	266
3.3.3.12	Adc_GetStreamLastPointer	267
3.3.3.13	Adc_GetCurrentPowerState	268
3.3.3.14	Adc_GetTargetPowerState	269
3.3.3.15	Adc_PreparePowerState	270
3.3.3.16	Adc_SetPowerState	271
3.3.3.17	Adc_GetVersionInfo	273
3.3.3.18	Adc_TriggerStartupCal	274
3.3.3.19	Adc_GetStartupCalStatus	275
3.3.3.20	Adc_InitCheck	276
3.3.4	Notifications and Callbacks	277
3.3.5	Scheduled functions	277
3.3.6	Interrupt service routines	277
3.3.6.1	Adc_ChEventInterruptHandler	277
3.3.6.2	Adc_RS0EventInterruptHandler	278
3.3.6.3	Adc_RS1EventInterruptHandler	279
3.3.6.4	Adc_RS2EventInterruptHandler	280
3.3.7	Error codes classification	281
3.3.7.1	Development errors	281
3.3.7.2	Production errors	285

Table of contents

3.3.7.3	Safety errors	285
3.3.7.4	Runtime errors	286
3.3.8	Deviations and limitations	286
3.3.8.1	Deviations	286
3.3.8.2	Limitations	286
3.3.9	Unsupported hardware features	287
4	BFX driver	288
4.1	User information	288
4.1.1	Description	288
4.1.2	Hardware-software mapping	288
4.1.3	File structure	288
4.1.3.1	C file structure	288
4.1.3.2	Code generator plugin files	288
4.1.4	Integration hints	289
4.1.4.1	Integration with AUTOSAR stack	289
4.1.4.2	Multicore and Resource Manager	290
4.1.4.3	MCU support	291
4.1.4.4	Port support	291
4.1.4.5	DMA support	291
4.1.4.6	Interrupt connections	291
4.1.4.7	Example usage	292
4.1.5	Key architectural considerations	292
4.2	Assumptions of Use (AoUs)	293
4.3	Reference information	294
4.3.1	Configuration interfaces	294
4.3.2	Functions - Type definitions	294
4.3.3	Functions - APIs	294
4.3.3.1	Bfx_SetBit_u8u8	294
4.3.3.2	Bfx_SetBit_u16u8	295
4.3.3.3	Bfx_SetBit_u32u8	295
4.3.3.4	Bfx_ClrBit_u8u8	296
4.3.3.5	Bfx_ClrBit_u16u8	297
4.3.3.6	Bfx_ClrBit_u32u8	298
4.3.3.7	Bfx_GetBit_u8u8_u8	299
4.3.3.8	Bfx_GetBit_u16u8_u8	300
4.3.3.9	Bfx_GetBit_u32u8_u8	300
4.3.3.10	Bfx_SetBits_u8u8u8u8	301
4.3.3.11	Bfx_SetBits_u16u8u8u8	302
4.3.3.12	Bfx_SetBits_u32u8u8u8	303
4.3.3.13	Bfx_GetBits_u8u8u8_u8	304
4.3.3.14	Bfx_GetBits_u16u8u8_u16	305

Table of contents

4.3.3.15	Bfx_GetBits_u32u8u8_u32	306
4.3.3.16	Bfx_SetBitMask_u8u8	307
4.3.3.17	Bfx_SetBitMask_u16u16	308
4.3.3.18	Bfx_SetBitMask_u32u32	309
4.3.3.19	Bfx_ClrBitMask_u8u8	310
4.3.3.20	Bfx_ClrBitMask_u16u16	310
4.3.3.21	Bfx_ClrBitMask_u32u32	311
4.3.3.22	Bfx_TstBitMask_u8u8_u8	312
4.3.3.23	Bfx_TstBitMask_u16u16_u8	313
4.3.3.24	Bfx_TstBitMask_u32u32_u8	314
4.3.3.25	Bfx_TstBitLnMask_u8u8_u8	315
4.3.3.26	Bfx_TstBitLnMask_u16u16_u8	316
4.3.3.27	Bfx_TstBitLnMask_u32u32_u8	316
4.3.3.28	Bfx_TstParityEven_u8_u8	317
4.3.3.29	Bfx_TstParityEven_u16_u8	318
4.3.3.30	Bfx_TstParityEven_u32_u8	319
4.3.3.31	Bfx_ToggleBits_u8	320
4.3.3.32	Bfx_ToggleBits_u16	321
4.3.3.33	Bfx_ToggleBits_u32	321
4.3.3.34	Bfx_ToggleBitMask_u8u8	322
4.3.3.35	Bfx_ToggleBitMask_u16u16	323
4.3.3.36	Bfx_ToggleBitMask_u32u32	324
4.3.3.37	Bfx_ShiftBitRt_u8u8	325
4.3.3.38	Bfx_ShiftBitRt_u16u8	325
4.3.3.39	Bfx_ShiftBitRt_u32u8	326
4.3.3.40	Bfx_ShiftBitLt_u8u8	327
4.3.3.41	Bfx_ShiftBitLt_u16u8	328
4.3.3.42	Bfx_ShiftBitLt_u32u8	329
4.3.3.43	Bfx_RotBitRt_u8u8	330
4.3.3.44	Bfx_RotBitRt_u16u8	331
4.3.3.45	Bfx_RotBitRt_u32u8	331
4.3.3.46	Bfx_RotBitLt_u8u8	332
4.3.3.47	Bfx_RotBitLt_u16u8	333
4.3.3.48	Bfx_RotBitLt_u32u8	334
4.3.3.49	Bfx_CopyBit_u8u8u8u8	335
4.3.3.50	Bfx_CopyBit_u16u8u16u8	336
4.3.3.51	Bfx_CopyBit_u32u8u32u8	337
4.3.3.52	Bfx_PutBits_u8u8u8u8	338
4.3.3.53	Bfx_PutBits_u16u8u8u16	339
4.3.3.54	Bfx_PutBits_u32u8u8u32	340
4.3.3.55	Bfx_PutBitsMask_u8u8u8	341
4.3.3.56	Bfx_PutBitsMask_u16u16u16	342

Table of contents

4.3.3.57	Bfx_PutBitsMask_u32u32u32	343
4.3.3.58	Bfx_PutBit_u8u8u8	343
4.3.3.59	Bfx_PutBit_u16u8u8	344
4.3.3.60	Bfx_PutBit_u32u8u8	345
4.3.3.61	Bfx_GetVersionInfo	346
4.3.4	Notifications and Callbacks	347
4.3.5	Scheduled functions	347
4.3.6	Interrupt service routines	347
4.3.7	Error codes classification	347
4.3.7.1	Development errors	347
4.3.7.2	Production errors	347
4.3.7.3	Safety errors	347
4.3.7.4	Runtime errors	347
4.3.8	Deviations and limitations	347
4.3.8.1	Deviations	347
4.3.8.2	Limitations	347
4.3.9	Unsupported hardware features	348
5	Can_17_McmCan driver	349
5.1	User information	349
5.1.1	Description	349
5.1.2	Hardware-software mapping	349
5.1.2.1	SRC: dependent hardware peripheral	350
5.1.2.2	M_CAN: primary hardware peripheral	351
5.1.2.3	SCU: dependent hardware peripheral	351
5.1.2.4	Port: dependent hardware peripheral	352
5.1.3	File structure	352
5.1.3.1	C File Structure	352
5.1.3.2	Code Generator Plugin Files	354
5.1.4	Integration hints	355
5.1.4.1	Integration with AUTOSAR stack	356
5.1.4.2	Multicore and Resource Manager	359
5.1.4.3	MCU support	360
5.1.4.4	Port support	360
5.1.4.5	DMA support	360
5.1.4.6	Interrupt connections	360
5.1.4.7	Example usage	367
5.1.5	Key architectural considerations	372
5.1.5.1	CAN interrupt handling	372
5.1.5.2	Multi-period Tx and Rx	373
5.1.5.3	Mixed mode Rx processing	373
5.2	Assumptions of Use (AoUs)	374

Table of contents

5.3	Reference information	375
5.3.1	Configuration interfaces	375
5.3.1.1	Container: CanConfigSet	375
5.3.1.2	Container: CanController	375
5.3.1.2.1	CanBusoffProcessing	376
5.3.1.2.2	CanControllerActivation	376
5.3.1.2.3	CanControllerBaseAddress	377
5.3.1.2.4	CanControllerDefaultBaudrate	377
5.3.1.2.5	CanControllerId	378
5.3.1.2.6	CanControllerLoopbackEnable	378
5.3.1.2.7	CanCpuClockRef	379
5.3.1.2.8	CanPeripheralBusClockRef	379
5.3.1.2.9	CanRxInputSelection	380
5.3.1.2.10	CanRxProcessing	381
5.3.1.2.11	CanTxProcessing	381
5.3.1.2.12	CanWakeUpFunctionalityAPI	382
5.3.1.2.13	CanWakeUpProcessing	382
5.3.1.2.14	CanWakeUpSourceRef	383
5.3.1.2.15	CanWakeUpSupport	384
5.3.1.3	Container: CanControllerBaudrateConfig	384
5.3.1.3.1	CanControllerBaudRate	384
5.3.1.3.2	CanControllerBaudRateConfigID	385
5.3.1.3.3	CanControllerPropSeg	386
5.3.1.3.4	CanControllerSeg1	386
5.3.1.3.5	CanControllerSeg2	387
5.3.1.3.6	CanControllerSyncJumpWidth	387
5.3.1.4	Container: CanControllerFdBaudrateConfig	388
5.3.1.4.1	CanControllerFdBaudRate	388
5.3.1.4.2	CanControllerPropSeg	389
5.3.1.4.3	CanControllerSeg1	389
5.3.1.4.4	CanControllerSeg2	390
5.3.1.4.5	CanControllerSyncJumpWidth	390
5.3.1.4.6	CanControllerTrcvDelayCompensationOffset	391
5.3.1.4.7	CanControllerTxBitRateSwitch	392
5.3.1.5	Container: CanHwFilter	392
5.3.1.5.1	CanHwFilterCode	392
5.3.1.5.2	CanHwFilterMask	393
5.3.1.6	Container: CanIcom	393
5.3.1.7	Container: CanIcomConfig	394
5.3.1.7.1	CanIcomConfigId	394
5.3.1.7.2	CanIcomWakeOnBusOff	394
5.3.1.8	Container: CanIComGeneral	395

Table of contents

5.3.1.8.1	CanIcomLevel	395
5.3.1.8.2	CanIcomVariant	395
5.3.1.9	Container: CanIcomRxMessage	396
5.3.1.9.1	CanIcomCounterValue	396
5.3.1.9.2	CanIcomMessageId	397
5.3.1.9.3	CanIcomMessageIdMask	397
5.3.1.9.4	CanIcomMissingMessageTimerValue	398
5.3.1.9.5	CanIcomPayloadLength	398
5.3.1.9.6	CanIcomPayloadLengthError	399
5.3.1.10	Container: CanIcomRxMessageSignalConfig	399
5.3.1.10.1	CanIcomSignalMask	399
5.3.1.10.2	CanIcomSignalMaskLower32bits	400
5.3.1.10.3	CanIcomSignalMaskUpper32bits	401
5.3.1.10.4	CanIcomSignalOperation	401
5.3.1.10.5	CanIcomSignalRef	402
5.3.1.10.6	CanIcomSignalValue	402
5.3.1.10.7	CanIcomSignalValueLower32bits	403
5.3.1.10.8	CanIcomSignalValueUpper32bits	403
5.3.1.11	Container: CanIcomWakeupCauses	404
5.3.1.12	Container: CanTTController	404
5.3.1.12.1	CanTTControllerApplWatchdogLimit	404
5.3.1.12.2	CanTTControllerCycleCountMax	405
5.3.1.12.3	CanTTControllerExpectedTxTrigger	405
5.3.1.12.4	CanTTControllerExternalClockSynchronisation	406
5.3.1.12.5	CanTTControllerGlobalTimeFiltering	407
5.3.1.12.6	CanTTControllerInitialRefOffset	407
5.3.1.12.7	CanTTControllerInterruptEnable	408
5.3.1.12.8	CanTTControllerLevel2	409
5.3.1.12.9	CanTTControllerNTUConfig	409
5.3.1.12.10	CanTTControllerOperationMode	410
5.3.1.12.11	CanTTControllerSyncDeviation	410
5.3.1.12.12	CanTTControllerTURRestore	411
5.3.1.12.13	CanTTControllerTimeMaster	412
5.3.1.12.14	CanTTControllerTimeMasterPriority	412
5.3.1.12.15	CanTTControllerTxEnableWindowLength	413
5.3.1.12.16	CanTTControllerWatchTriggerGapTimeMark	413
5.3.1.12.17	CanTTControllerWatchTriggerTimeMark	414
5.3.1.12.18	CanTTIRQProcessing	414
5.3.1.13	Container: CanTTHardwareObjectTrigger	415
5.3.1.13.1	CanTTHardwareObjectBaseCycle	415
5.3.1.13.2	CanTTHardwareObjectCycleRepetition	416
5.3.1.13.3	CanTTHardwareObjectTimeMark	416

Table of contents

5.3.1.13.4	CanTTHardwareObjectTriggerId	417
5.3.1.13.5	CanTTHardwareObjectTriggerType	417
5.3.1.14	Container: CommonPublishedInformation	418
5.3.1.14.1	ArMajorVersion	418
5.3.1.14.2	ArMinorVersion	418
5.3.1.14.3	ArPatchVersion	419
5.3.1.14.4	ModuleId	419
5.3.1.14.5	Release	420
5.3.1.14.6	SwMajorVersion	420
5.3.1.14.7	SwMinorVersion	421
5.3.1.14.8	SwPatchVersion	421
5.3.1.14.9	VendorApiInfix	422
5.3.1.14.10	VendorId	422
5.3.1.15	Container: Can	423
5.3.1.16	Container: CanGeneral	423
5.3.1.16.1	CanDeInitApi	423
5.3.1.16.2	CanDevErrorDetection	423
5.3.1.16.3	CanIndex	424
5.3.1.16.4	CanInitDeInitApiMode	424
5.3.1.16.5	CanLPduReceiveCalloutFunction	425
5.3.1.16.6	CanMainFunctionBusoffPeriod	426
5.3.1.16.7	CanMainFunctionModePeriod	426
5.3.1.16.8	CanMainFunctionWakeupPeriod	427
5.3.1.16.9	CanMultiCoreErrorDetect	427
5.3.1.16.10	CanMultiplexedTransmission	428
5.3.1.16.11	CanOsCounterRef	428
5.3.1.16.12	CanPublicIcomSupport	429
5.3.1.16.13	CanSetBaudrateApi	429
5.3.1.16.14	CanSupportTTCANRef	430
5.3.1.16.15	CanTimeoutDuration	431
5.3.1.16.16	CanVersionInfoApi	431
5.3.1.17	Container: CanMainFunctionRWPeriods	432
5.3.1.17.1	CanMainFunctionPeriod	432
5.3.1.18	Container: CanHardwareObject	432
5.3.1.18.1	CanControllerRef	432
5.3.1.18.2	CanFdPaddingValue	433
5.3.1.18.3	CanHandleType	433
5.3.1.18.4	CanHwFIFOThreshold	434
5.3.1.18.5	CanHwObjectCount	434
5.3.1.18.6	CanIdType	435
5.3.1.18.7	CanMainFunctionRWPeriodRef	436
5.3.1.18.8	CanObjectId	436

Table of contents

5.3.1.18.9	CanObjectType	437
5.3.1.18.10	CanTriggerTransmitEnable	438
5.3.2	Functions - Type definitions	438
5.3.2.1	CanTrcv_TrcvModeType	438
5.3.2.2	CanTrcv_TrcvWakeupModeType	439
5.3.2.3	CanTrcv_TrcvWakeupReasonType	439
5.3.2.4	Can_HwHandleType	440
5.3.2.5	Can_HwType	440
5.3.2.6	Can_17_McmCan_ConfigType	441
5.3.2.7	Can_PduType	441
5.3.2.8	Can_IdType	441
5.3.2.9	Can_StateTransitionType	442
5.3.2.10	Can_ReturnType	442
5.3.3	Functions - APIs	443
5.3.3.1	Can_17_McmCan_CheckBaudrate	443
5.3.3.2	Can_17_McmCan_Init	444
5.3.3.3	Can_17_McmCan_GetVersionInfo	445
5.3.3.4	Can_17_McmCan_SetBaudrate	446
5.3.3.5	Can_17_McmCan_SetControllerMode	447
5.3.3.6	Can_17_McmCan_DisableControllerInterrupts	448
5.3.3.7	Can_17_McmCan_EnableControllerInterrupts	448
5.3.3.8	Can_17_McmCan_Write	449
5.3.3.9	Can_17_McmCan_SetIcomConfiguration	451
5.3.3.10	Can_17_McmCan_Delnit	452
5.3.4	Notifications and callbacks	453
5.3.5	Scheduled functions	453
5.3.5.1	Can_17_McmCan_MainFunction_Write	453
5.3.5.2	Can_17_McmCan_MainFunction_Write_(x)	454
5.3.5.3	Can_17_McmCan_MainFunction_Read	455
5.3.5.4	Can_17_McmCan_MainFunction_Read_(x)	456
5.3.5.5	Can_17_McmCan_MainFunction_BusOff	457
5.3.5.6	Can_17_McmCan_MainFunction_Wakeup	457
5.3.5.7	Can_17_McmCan_MainFunction_Mode	458
5.3.6	Interrupt service routines	459
5.3.6.1	Can_17_McmCan_IsrBusOffHandler	459
5.3.6.2	Can_17_McmCan_IsrReceiveHandler	460
5.3.6.3	Can_17_McmCan_IsrRxFIFOHandler	461
5.3.6.4	Can_17_McmCan_IsrTransmitHandler	462
5.3.7	Error codes classification	463
5.3.7.1	Development errors	463
5.3.7.2	Production errors	465
5.3.7.3	Safety errors	466

Table of contents

5.3.7.4	Runtime errors	466
5.3.8	Deviations and limitations	466
5.3.8.1	Deviations	466
5.3.8.2	Limitations	466
5.3.9	Unsupported hardware features	467
6	CanTrcv_17_V9251 driver	468
6.1	User information	468
6.1.1	Description	468
6.1.2	Hardware-software mapping	468
6.1.2.1	TLE9251V: primary hardware peripheral	470
6.1.2.2	SCU: dependent hardware peripheral	470
6.1.2.3	Port: dependent hardware peripheral	470
6.1.2.4	SRC: dependent hardware peripheral	470
6.1.3	File structure	471
6.1.3.1	C file structure	471
6.1.3.2	Code generator plugin files	472
6.1.4	Integration hints	473
6.1.4.1	Integration with AUTOSAR stack	473
6.1.4.2	Multicore and Resource Manager	476
6.1.4.3	MCU support	476
6.1.4.4	Port support	476
6.1.4.5	DMA support	477
6.1.4.6	Interrupt connections	477
6.1.4.7	Example usage	478
6.1.5	Key architectural considerations	479
6.1.5.1	CAN transceiver wake up: only Interrupt mode is supported	479
6.1.5.2	User mode is not supported	479
6.2	Assumptions of Use (AoUs)	480
6.3	Reference information	481
6.3.1	Configuration interfaces	481
6.3.1.1	Container: CommonPublished Information	482
6.3.1.1.1	ArMajorVersion	482
6.3.1.1.2	ArMinorVersion	482
6.3.1.1.3	ArPatchVersion	483
6.3.1.1.4	ModuleId	483
6.3.1.1.5	Release	483
6.3.1.1.6	SwMajorVersion	484
6.3.1.1.7	SwMinorVersion	484
6.3.1.1.8	SwPatchVersion	485
6.3.1.1.9	VendorApilInfix	485
6.3.1.1.10	VendorId	486

Table of contents

6.3.1.2	Container: CanTrcv	486
6.3.1.2.1	Config Variant	486
6.3.1.3	Container: CanTrcvConfigSet	487
6.3.1.3.1	CanTrcvSPICommRetries	487
6.3.1.3.2	CanTrcvSPICommTimeout	487
6.3.1.4	Container: CanTrcvChannel	488
6.3.1.4.1	CanTrcvAccess	488
6.3.1.4.2	CanTrcvChannelId	488
6.3.1.4.3	CanTrcvChannelUsed	489
6.3.1.4.4	CanTrcvControlsPowerSupply	489
6.3.1.4.5	CanTrcvHwPnSupport	490
6.3.1.4.6	CanTrcvIcuChannelRef	491
6.3.1.4.7	CanTrcvInitState	491
6.3.1.4.8	CanTrcvMaxBaudrate	492
6.3.1.4.9	CanTrcvPorWakeUpSourceRef	492
6.3.1.4.10	CanTrcvSyserrWakeUpSourceRef	493
6.3.1.4.11	CanTrcvWakeUpByBusUsed	494
6.3.1.4.12	CanTrcvWakeUpSourceRef	494
6.3.1.5	Container: CanTrcvDemEventParameterRefs	495
6.3.1.5.1	CANTRCV_E_BUS_ERROR	495
6.3.1.6	Container: CanTrcvDioAccess	496
6.3.1.7	Container: CanTrcvDioChannelAccess	496
6.3.1.7.1	CanTrcvDioSymNameRef	496
6.3.1.7.2	CanTrcvHardwareInterfaceName	496
6.3.1.8	Container: CanTrcvGeneral	497
6.3.1.8.1	CanTrcvDevErrorDetect	497
6.3.1.8.2	CanTrcvGetVersionInfo	497
6.3.1.8.3	CanTrcvIndex	498
6.3.1.8.4	CanTrcvMainFunctionDiagnosticsPeriod	498
6.3.1.8.5	CanTrcvMainFunctionPeriod	499
6.3.1.8.6	CanTrcvTimerType	499
6.3.1.8.7	CanTrcvWaitTime	500
6.3.1.8.8	CanTrcvWakeUpSupport	500
6.3.1.9	Container: CanTrcvPartialNetwork	501
6.3.1.9.1	CanTrcvBaudRate	501
6.3.1.9.2	CanTrcvBusErrFlag	502
6.3.1.9.3	CanTrcvPnCanIdIsExtended	502
6.3.1.9.4	CanTrcvPnEnabled	503
6.3.1.9.5	CanTrcvPnFrameCanId	503
6.3.1.9.6	CanTrcvPnFrameCanIdMask	504
6.3.1.9.7	CanTrcvPnFrameDlc	504
6.3.1.9.8	CanTrcvPowerOnFlag	505

Table of contents

6.3.1.10	Container: CanTrcvPnPnFrameDataMaskSpec	505
6.3.1.10.1	CanTrcvPnPnFrameDataMask	506
6.3.1.10.2	CanTrcvPnPnFrameDataMaskIndex	506
6.3.1.11	Container: CanTrcvSpiAccess	507
6.3.1.12	Container: CanTrcvSpiSequence	507
6.3.1.12.1	CanTrcvSpiAccessSynchronous	507
6.3.1.12.2	CanTrcvSpiSequenceName	508
6.3.2	Functions - Type definitions	508
6.3.2.1	CanTrcv_17_V9251_ConfigType	508
6.3.3	Functions - APIs	508
6.3.3.1	CanTrcv_17_V9251_Init	508
6.3.3.2	CanTrcv_17_V9251_SetOpMode	509
6.3.3.3	CanTrcv_17_V9251_GetOpMode	511
6.3.3.4	CanTrcv_17_V9251_GetBusWuReason	512
6.3.3.5	CanTrcv_17_V9251_GetVersionInfo	513
6.3.3.6	CanTrcv_17_V9251_SetWakeupMode	513
6.3.3.7	CanTrcv_17_V9251_CheckWakeup	515
6.3.4	Notifications and Callbacks	516
6.3.5	Scheduled functions	516
6.3.6	Interrupt service routines	516
6.3.7	Error codes classification	516
6.3.7.1	Development errors	516
6.3.7.2	Production errors	517
6.3.7.3	Safety errors	517
6.3.7.4	Runtime errors	517
6.3.8	Deviations and limitations	517
6.3.8.1	Deviations	517
6.3.8.2	Limitations	518
6.3.9	Unsupported hardware features	518
7	CanTrcv_17_W9255 driver	519
7.1	User information	519
7.1.1	Description	519
7.1.2	Hardware-software mapping	519
7.1.2.1	TLE9255W: primary hardware peripheral	521
7.1.2.2	SCU: dependent hardware peripheral	521
7.1.2.3	Port: dependent hardware peripheral	522
7.1.2.4	SRC: dependent hardware peripheral	522
7.1.3	File structure	522
7.1.3.1	C file structure	522
7.1.3.2	Code generator plugin files	524
7.1.4	Integration hints	525

Table of contents

7.1.4.1	Integration with AUTOSAR stack	525
7.1.4.2	Multicore and Resource Manager	528
7.1.4.3	MCU support	528
7.1.4.4	Port support	529
7.1.4.5	DMA support	529
7.1.4.6	Interrupt connections	529
7.1.4.7	Example usage	530
7.1.5	Key architectural considerations	532
7.1.5.1	Wake-up by interrupt mode	532
7.1.5.2	User mode support	532
7.1.5.3	CanTrcv_17_W9255_SetOpMode and CanTrcv_17_W9255_CheckWakeFlag APIs implemented as synchronous	533
7.2	Assumptions of Use (AoUs)	534
7.3	Reference information	535
7.3.1	Configuration interfaces	535
7.3.1.1	Container: CanTrcvDemEventParameterRefs	535
7.3.1.1.1	CANTRCV_E_BUS_ERROR	536
7.3.1.2	Container: CommonPublished Information	536
7.3.1.2.1	ArMajorVersion	536
7.3.1.2.2	ArMinorVersion	537
7.3.1.2.3	ArPatchVersion	537
7.3.1.2.4	ModuleId	538
7.3.1.2.5	Release	538
7.3.1.2.6	SwMajorVersion	539
7.3.1.2.7	SwMinorVersion	539
7.3.1.2.8	SwPatchVersion	540
7.3.1.2.9	VendorApiInfix	540
7.3.1.2.10	VendorId	540
7.3.1.3	Container: CanTrcv	541
7.3.1.3.1	Config Variant	541
7.3.1.4	Container: CanTrcvChannel	541
7.3.1.4.1	CanTrcvAccess	542
7.3.1.4.2	CanTrcvChannelId	542
7.3.1.4.3	CanTrcvChannelUsed	543
7.3.1.4.4	CanTrcvControlsPowerSupply	543
7.3.1.4.5	CanTrcvHwPnSupport	544
7.3.1.4.6	CanTrcvIcuChannelRef	544
7.3.1.4.7	CanTrcvInitState	545
7.3.1.4.8	CanTrcvMaxBaudrate	545
7.3.1.4.9	CanTrcvPorWakeUpSourceRef	546
7.3.1.4.10	CanTrcvSyserrWakeUpSourceRef	546
7.3.1.4.11	CanTrcvWakeUpByBusUsed	547

Table of contents

7.3.1.4.12	CanTrcvWakeupSourceRef	548
7.3.1.5	Container: CanTrcvConfigSet	548
7.3.1.5.1	CanTrcvSPICommRetries	548
7.3.1.5.2	CanTrcvSPICommTimeout	549
7.3.1.6	Container: CanTrcvDioAccess	549
7.3.1.7	Container: CanTrcvDioChannelAccess	550
7.3.1.7.1	CanTrcvDioSymNameRef	550
7.3.1.7.2	CanTrcvHardwareInterfaceName	550
7.3.1.8	Container: CanTrcvGeneral	551
7.3.1.8.1	CanTrcvDevErrorDetect	551
7.3.1.8.2	CanTrcvGetVersionInfo	551
7.3.1.8.3	CanTrcvIndex	552
7.3.1.8.4	CanTrcvMainFunctionDiagnosticsPeriod	552
7.3.1.8.5	CanTrcvMainFunctionPeriod	553
7.3.1.8.6	CanTrcvTimerType	553
7.3.1.8.7	CanTrcvWaitTime	554
7.3.1.8.8	CanTrcvWakeUpSupport	554
7.3.1.9	Container: CanTrcvPartialNetwork	555
7.3.1.9.1	CanTrcvBaudRate	555
7.3.1.9.2	CanTrcvBusErrFlag	555
7.3.1.9.3	CanTrcvPnCanIdIsExtended	556
7.3.1.9.4	CanTrcvPnEnabled	557
7.3.1.9.5	CanTrcvPnFrameCanId	557
7.3.1.9.6	CanTrcvPnFrameCanIdMask	558
7.3.1.9.7	CanTrcvPnFrameDlc	558
7.3.1.9.8	CanTrcvPowerOnFlag	559
7.3.1.10	Container: CanTrcvPnFrameDataMaskSpec	559
7.3.1.10.1	CanTrcvPnFrameDataMask	559
7.3.1.10.2	CanTrcvPnFrameDataMaskIndex	560
7.3.1.11	Container: CanTrcvSpiAccess	560
7.3.1.12	Container: CanTrcvSpiSequence	560
7.3.1.12.1	CanTrcvSpiAccessSynchronous	560
7.3.1.12.2	CanTrcvSpiSequenceName	561
7.3.2	Functions - Type definitions	562
7.3.2.1	CanTrcv_17_W9255_ConfigType	562
7.3.2.2	CanTrcv_17_W9255_PNActivationType	562
7.3.2.3	CanTrcv_17_W9255_TrsvFlagStateType	563
7.3.3	Functions - APIs	563
7.3.3.1	CanTrcv_17_W9255_Init	563
7.3.3.2	CanTrcv_17_W9255_SetOpMode	564
7.3.3.3	CanTrcv_17_W9255_GetOpMode	565
7.3.3.4	CanTrcv_17_W9255_GetBusWuReason	566

Table of contents

7.3.3.5	CanTrcv_17_W9255_GetVersionInfo	568
7.3.3.6	CanTrcv_17_W9255_SetWakeupMode	568
7.3.3.7	CanTrcv_17_W9255_CheckWakeup	570
7.3.3.8	CanTrcv_17_W9255_CheckWakeFlag	571
7.3.3.9	CanTrcv_17_W9255_ClearTrcvTimeoutFlag	571
7.3.3.10	CanTrcv_17_W9255_ClearTrcvWufFlag	572
7.3.3.11	CanTrcv_17_W9255_GetTrcvSystemData	573
7.3.3.12	CanTrcv_17_W9255_ReadTrcvSilenceFlag	574
7.3.3.13	CanTrcv_17_W9255_ReadTrcvTimeoutFlag	576
7.3.3.14	CanTrcv_17_W9255_SetPNAactivationState	577
7.3.4	Notifications and Callbacks	577
7.3.5	Scheduled functions	577
7.3.5.1	CanTrcv_17_W9255_MainFunction	578
7.3.6	Interrupt service routines	578
7.3.7	Error codes classification	578
7.3.7.1	Development errors	579
7.3.7.2	Production errors	581
7.3.7.3	Safety errors	581
7.3.7.4	Runtime errors	581
7.3.8	Deviations and limitations	581
7.3.8.1	Deviations	581
7.3.8.2	Limitations	581
7.3.9	Unsupported hardware features	582
8	CRC driver	583
8.1	User information	583
8.1.1	Description	583
8.1.2	Hardware-software mapping	583
8.1.2.1	SCU: dependent hardware peripheral	584
8.1.2.2	FCE: primary hardware peripheral	585
8.1.3	File structure	586
8.1.3.1	C file structure	586
8.1.3.2	Code generator plugin files	587
8.1.4	Integration hints	588
8.1.4.1	Integration with AUTOSAR stack	588
8.1.4.2	Multicore and Resource Manager	590
8.1.4.3	MCU support	591
8.1.4.4	Port support	591
8.1.4.5	DMA support	591
8.1.4.6	Interrupt connections	591
8.1.4.7	Example usage	592
8.1.5	Key architectural considerations	593

Table of contents

8.1.5.1	Dynamic channel handling	593
8.2	Assumptions of Use (AoUs)	594
8.3	Reference information	595
8.3.1	Configuration interfaces	595
8.3.1.1	Container: CommonPublishedInformation	595
8.3.1.1.1	ARMinorVersion	595
8.3.1.1.2	ARPatchVersion	596
8.3.1.1.3	ArMajorVersion	596
8.3.1.1.4	ModuleId	597
8.3.1.1.5	SwMajorVersion	597
8.3.1.1.6	SwMinorVersion	598
8.3.1.1.7	SwPatchVersion	598
8.3.1.1.8	VendorId	598
8.3.1.2	Container: Crc	599
8.3.1.3	Container: CrcGeneral	599
8.3.1.3.1	Crc16Mode	599
8.3.1.3.2	Crc16ReturnValue	600
8.3.1.3.3	Crc32Mode	600
8.3.1.3.4	Crc32P4Mode	601
8.3.1.3.5	Crc32P4ReturnValue	601
8.3.1.3.6	Crc32ReturnValue	602
8.3.1.3.7	Crc8H2FMode	602
8.3.1.3.8	Crc8H2FReturnValue	602
8.3.1.3.9	Crc8Mode	603
8.3.1.3.10	Crc8ReturnValue	603
8.3.1.3.11	CrcRuntimeApiMode	604
8.3.1.3.12	CrcSafetyEnable	604
8.3.1.3.13	CrcVersionInfoApi	605
8.3.1.4	Container: CrcPublishedInformation	605
8.3.1.4.1	CrcInitialValue16	606
8.3.1.4.2	CrcInitialValue32	606
8.3.1.4.3	CrcInitialValue32P4	606
8.3.1.4.4	CrcInitialValue8	607
8.3.1.4.5	CrcInitialValue8H2F	607
8.3.2	Functions - Type definitions	608
8.3.3	Functions - APIs	608
8.3.3.1	Crc_CalculateCRC32P4	608
8.3.3.2	Crc_CalculateCRC8H2F	609
8.3.3.3	Crc_GetVersionInfo	610
8.3.3.4	Crc_CalculateCRC8	611
8.3.3.5	Crc_CalculateCRC16	612
8.3.3.6	Crc_CalculateCRC32	613

Table of contents

8.3.4	Notifications and Callbacks	614
8.3.5	Scheduled functions	615
8.3.6	Interrupt service routines	615
8.3.7	Error codes classification	615
8.3.7.1	Development errors	615
8.3.7.2	Production errors	615
8.3.7.3	Safety errors	615
8.3.7.4	Runtime errors	615
8.3.8	Deviations and limitations	615
8.3.8.1	Deviations	616
8.3.8.2	Limitations	616
8.3.9	Unsupported hardware features	616
9	DIO driver	617
9.1	User information	617
9.1.1	Description	617
9.1.2	Hardware-software mapping	617
9.1.2.1	Port: primary hardware peripheral	618
9.1.2.2	SCU: dependent hardware peripheral	619
9.1.3	File structure	619
9.1.3.1	C file structure	619
9.1.3.2	Code generator plugin files	621
9.1.4	Integration hints	622
9.1.4.1	Integration with AUTOSAR stack	622
9.1.4.2	Multicore and Resource Manager	624
9.1.4.3	MCU support	624
9.1.4.4	Port support	624
9.1.4.5	DMA support	624
9.1.4.6	Interrupt connections	624
9.1.4.7	Example usage	625
9.1.5	Key architectural considerations	626
9.1.5.1	Implementation type	626
9.1.5.2	User mode support	626
9.2	Assumptions of Use (AoUs)	627
9.3	Reference information	628
9.3.1	Configuration interfaces	628
9.3.1.1	Container: CommonPublishedInformation	628
9.3.1.1.1	ArMajorVersion	628
9.3.1.1.2	ArMinorVersion	629
9.3.1.1.3	ArPatchVersion	629
9.3.1.1.4	ModuleId	630
9.3.1.1.5	Release	630

Table of contents

9.3.1.1.6	SwMajorVersion	630
9.3.1.1.7	SwMinorVersion	631
9.3.1.1.8	SwPatchVersion	631
9.3.1.1.9	VendorID	632
9.3.1.2	Container: Dio	632
9.3.1.2.1	Config Variant	632
9.3.1.3	Container: DioChannel	633
9.3.1.3.1	DioChannelId	633
9.3.1.4	Container: DioChannelGroup	633
9.3.1.4.1	DioChannelGroupIdentification	634
9.3.1.4.2	DioPortMask	634
9.3.1.4.3	DioPortOffset	635
9.3.1.5	Container: DioConfig	635
9.3.1.6	Container: DioGeneral	635
9.3.1.6.1	DioDevErrorDetect	635
9.3.1.6.2	DioFlipChannelApi	636
9.3.1.6.3	DioSafetyEnable	636
9.3.1.6.4	DioVersionInfoApi	637
9.3.1.7	Container: DioPort	637
9.3.1.7.1	DioPortId	637
9.3.2	Functions - Type definitions	638
9.3.2.1	Dio_ChannelGroupType	638
9.3.2.2	Dio_ChannelType	638
9.3.2.3	Dio_ConfigType	639
9.3.2.4	Dio_LevelType	639
9.3.2.5	Dio_PortType	639
9.3.2.6	Dio_PortLevelType	640
9.3.3	Functions - APIs	640
9.3.3.1	Dio_FlipChannel	640
9.3.3.2	Dio_GetVersionInfo	641
9.3.3.3	Dio_ReadChannel	642
9.3.3.4	Dio_ReadChannelGroup	642
9.3.3.5	Dio_ReadPort	643
9.3.3.6	Dio_WriteChannel	644
9.3.3.7	Dio_WriteChannelGroup	645
9.3.3.8	Dio_WritePort	646
9.3.4	Notifications and Callbacks	646
9.3.5	Scheduled functions	646
9.3.6	Interrupt service routines	646
9.3.7	Error codes classification	647
9.3.7.1	Development errors	647
9.3.7.2	Production errors	647

Table of contents

9.3.7.3	Safety errors	647
9.3.7.4	Runtime errors	647
9.3.8	Deviations and limitations	647
9.3.8.1	Deviations	648
9.3.8.2	Limitations	648
9.3.9	Unsupported hardware features	648
10	FEE driver	649
10.1	User information	649
10.1.1	Description	649
10.1.2	Hardware-software mapping	649
10.1.3	File structure	650
10.1.3.1	C File Structure	650
10.1.3.2	Code Generator Plugin Files	652
10.1.4	Integration hints	654
10.1.4.1	Integration with AUTOSAR stack	654
10.1.4.2	Multicore and Resource Manager	656
10.1.4.3	MCU support	656
10.1.4.4	Port support	656
10.1.4.5	DMA support	656
10.1.4.6	Interrupt connections	656
10.1.4.7	Example usage	657
10.1.5	Key architectural considerations	663
10.1.5.1	Double-sector algorithm is used	663
10.1.5.2	Quasi-static data algorithm is used	663
10.1.5.3	Post build variant	664
10.1.5.4	Callback notification when erase verify error occurs	664
10.1.5.5	Notifications to upper layer	664
10.1.5.6	Handling ECC errors during GC	664
10.1.5.7	Ongoing erase cannot be canceled	664
10.1.5.8	DET check for Fee_JobErrorNotification and Fee_JobEndNotification	664
10.1.5.9	Init check	664
10.1.5.10	Handling of invalid sectors	664
10.1.5.11	Behavior of Fee_17_EraseQuasiStatic data for QS blocks with multiple instances	665
10.1.5.12	Behavior of illegal state notifications	665
10.1.5.13	User mode support	665
10.2	Assumptions of Use (AoUs)	666
10.3	Reference information	669
10.3.1	Configuration interfaces	669
10.3.1.1	Container: CommonPublishedInformation	670
10.3.1.1.1	ArMajorVersion	671
10.3.1.1.2	ArMinorVersion	671

Table of contents

10.3.1.1.3	ArPatchVersion	672
10.3.1.1.4	ModuleId	672
10.3.1.1.5	Release	672
10.3.1.1.6	SwMajorVersion	673
10.3.1.1.7	SwMinorVersion	673
10.3.1.1.8	SwPatchVersion	674
10.3.1.1.9	VendorId	674
10.3.1.2	Container: Fee	675
10.3.1.3	Container: FeeBlockConfiguration	675
10.3.1.3.1	FeeBlockNumber	675
10.3.1.3.2	FeeBlockSize	676
10.3.1.3.3	FeeDeviceIndex	676
10.3.1.3.4	FeeImmediateData	677
10.3.1.3.5	FeeNumberOfWriteCycles	677
10.3.1.3.6	FeeQsBlockAddress	678
10.3.1.3.7	FeeQsBlockInstances	678
10.3.1.3.8	FeeQuasiStaticManager	679
10.3.1.4	Container: FeeDemEventParameterRefs	680
10.3.1.4.1	FEE_E_GC_ERASE	680
10.3.1.4.2	FEE_E_GC_INIT	680
10.3.1.4.3	FEE_E_GC_READ	681
10.3.1.4.4	FEE_E_GC_TRIG	681
10.3.1.4.5	FEE_E_GC_WRITE	682
10.3.1.4.6	FEE_E_INVALIDATE	683
10.3.1.4.7	FEE_E_READ	683
10.3.1.4.8	FEE_E_UNCONFIG_BLK_EXCEEDED	684
10.3.1.4.9	FEE_E_WRITE	684
10.3.1.4.10	FEE_E_WRITE_CYCLES_EXHAUSTED	685
10.3.1.5	Container: FeeGeneral	685
10.3.1.5.1	FeeBlockTypeConfigured	686
10.3.1.5.2	FeeDevErrorDetect	686
10.3.1.5.3	FeeInitCheckApi	687
10.3.1.5.4	FeeMainFunctionPeriod	687
10.3.1.5.5	FeeNvmJobEndNotification	688
10.3.1.5.6	FeeNvmJobErrorNotification	688
10.3.1.5.7	FeePollingMode	689
10.3.1.5.8	FeeQsJobEndNotification	689
10.3.1.5.9	FeeQsJobErrorNotification	690
10.3.1.5.10	FeeSafetyEnable	691
10.3.1.5.11	FeeSetModeSupported	691
10.3.1.5.12	FeeVersionInfoApi	692
10.3.1.5.13	FeeVirtualPageSize	693

Table of contents

10.3.1.6	Container: FeelfxSpecificConfig	693
10.3.1.6.1	FeeCancelAllApi	693
10.3.1.6.2	FeeEraseAllEnable	694
10.3.1.6.3	FeeGcRestart	694
10.3.1.6.4	FeeGetCycleCountApi	695
10.3.1.6.5	FeeGetPrevDataApi	696
10.3.1.6.6	FeeMaxBlockCount	696
10.3.1.6.7	FeeMaxBytesPerCycle	697
10.3.1.6.8	FeeNvmIllegalStateNotification	698
10.3.1.6.9	FeeQsHardenErrorNotification	698
10.3.1.6.10	FeeQsIllegalStateNotification	699
10.3.1.6.11	FeeStateVarStructure	700
10.3.1.6.12	FeeThresholdValue	700
10.3.1.6.13	FeeUnConfigBlkOverflowHandle	700
10.3.1.6.14	FeeUnConfigBlock	701
10.3.1.6.15	FeeUseEraseSuspend	702
10.3.1.6.16	FeeVirginFlashIllegalState	702
10.3.1.7	Container: FeePublishedInformation	703
10.3.1.7.1	FeeBlockOverhead	703
10.3.1.7.2	FeePageOverhead	704
10.3.2	Functions - Type definitions	704
10.3.2.1	Fee_BlockType	704
10.3.2.2	Fee_ConfigType	705
10.3.2.3	Fee_DataType	706
10.3.2.4	Fee_NotifyFunctionPtrType	706
10.3.2.5	Fee_PageType	706
10.3.2.6	Fee_QsBlock_StateType	706
10.3.2.7	Fee_QuasiStaticBlockInfoType	707
10.3.2.8	Fee_StateDataType	708
10.3.2.9	Fee_UserType	708
10.3.3	Functions - APIs	708
10.3.3.1	Fee_17_CancelAll	708
10.3.3.2	Fee_17_DisableGcStart	709
10.3.3.3	Fee_17_EnableGcStart	710
10.3.3.4	Fee_17_EraseQuasiStaticData	711
10.3.3.5	Fee_17_GetCycleCount	712
10.3.3.6	Fee_17_GetPrevData	713
10.3.3.7	Fee_17_GetQuasiStaticBlockInfo	714
10.3.3.8	Fee_17_GetQuasiStaticJobResult	716
10.3.3.9	Fee_17_InitCheck	717
10.3.3.10	Fee_Cancel	718
10.3.3.11	Fee_EraseImmediateBlock	719

Table of contents

10.3.3.12	Fee_GetJobResult	720
10.3.3.13	Fee_GetStatus	721
10.3.3.14	Fee_GetVersionInfo	722
10.3.3.15	Fee_Init	723
10.3.3.16	Fee_InvalidateBlock	724
10.3.3.17	Fee_Read	725
10.3.3.18	Fee_SetMode	726
10.3.3.19	Fee_Write	726
10.3.4	Notifications and Callbacks	727
10.3.4.1	Fee_17_IllegalStateNotification	727
10.3.4.2	Fee_17_JobEraseErrorNotification	728
10.3.4.3	Fee_17_JobProgErrorNotification	729
10.3.4.4	Fee_JobEndNotification	730
10.3.4.5	Fee_JobErrorNotification	731
10.3.5	Scheduled functions	732
10.3.5.1	Fee_MainFunction	732
10.3.6	Interrupt service routines	733
10.3.7	Error codes classification	733
10.3.7.1	Development errors	733
10.3.7.2	Production errors	733
10.3.7.3	Safety errors	734
10.3.7.4	Runtime errors	735
10.3.8	Deviations and limitations	735
10.3.8.1	Deviations	735
10.3.8.2	Limitations	736
10.3.9	Unsupported hardware features	737
11	Fls_17_Dmu driver	738
11.1	User information	738
11.1.1	Description	738
11.1.2	Hardware-software mapping	738
11.1.2.1	DMU-DFlash0: primary hardware peripheral	740
11.1.2.2	SCU: dependent hardware peripheral	740
11.1.2.3	SRC: dependent hardware peripherals	741
11.1.3	File structure	742
11.1.3.1	C File Structure	742
11.1.3.2	Code Generator Plugin Files	743
11.1.4	Integration hints	744
11.1.4.1	Integration with AUTOSAR stack	744
11.1.4.2	Multicore and Resource Manager	746
11.1.4.3	MCU support	746
11.1.4.4	Port support	746

Table of contents

11.1.4.5	DMA support	746
11.1.4.6	Interrupt connections	746
11.1.4.7	Example usage	748
11.1.5	Key architectural considerations	751
11.2	Assumptions of Use (AoUs)	752
11.3	Reference information	754
11.3.1	Configuration interfaces	754
11.3.1.1	Container: CommonPublishedInformation	754
11.3.1.1.1	ArMajorVersion	755
11.3.1.1.2	ArMinorVersion	755
11.3.1.1.3	ArPatchVersion	755
11.3.1.1.4	ModuleId	756
11.3.1.1.5	Release	756
11.3.1.1.6	SwMajorVersion	757
11.3.1.1.7	SwMinorVersion	757
11.3.1.1.8	SwPatchVersion	758
11.3.1.1.9	VendorApIInfix	758
11.3.1.1.10	VendorId	759
11.3.1.2	Container: Fls	759
11.3.1.3	Container: FlsConfigSet	759
11.3.1.3.1	FlsAcErase	759
11.3.1.3.2	FlsAcWrite	760
11.3.1.3.3	FlsCallCycle	761
11.3.1.3.4	FlsDefaultMode	761
11.3.1.3.5	FlsEraseVerifyErrNotif	762
11.3.1.3.6	FlsJobEndNotification	762
11.3.1.3.7	FlsJobErrorNotification	763
11.3.1.3.8	FlsMaxReadFastMode	763
11.3.1.3.9	FlsMaxReadNormalMode	764
11.3.1.3.10	FlsMaxWriteFastMode	764
11.3.1.3.11	FlsMaxWriteNormalMode	765
11.3.1.3.12	FlsProgVerifyErrNotif	766
11.3.1.3.13	FlsProtection	766
11.3.1.3.14	FlsWaitStateErrorCorrection	767
11.3.1.3.15	FlsWaitStateRead	767
11.3.1.4	Container: FlsExternalDriver	768
11.3.1.4.1	FlsSpiReference	768
11.3.1.5	Container: FlsGeneral	768
11.3.1.5.1	FlsAcLoadOnJobStart	769
11.3.1.5.2	FlsBaseAddress	769
11.3.1.5.3	FlsBlankCheckApi	770
11.3.1.5.4	FlsCancelApi	770

Table of contents

11.3.1.5.5	FlsCompareApi	771
11.3.1.5.6	FlsDevErrorDetect	771
11.3.1.5.7	FlsDriverIndex	772
11.3.1.5.8	FlsGetJobResultApi	772
11.3.1.5.9	FlsGetStatusApi	773
11.3.1.5.10	FlsIfxFeeUse	773
11.3.1.5.11	FlsInitApiMode	774
11.3.1.5.12	FlsInitCheckApi	774
11.3.1.5.13	FlsRunTimeErrorDetect	775
11.3.1.5.14	FlsRuntimeApiMode	775
11.3.1.5.15	FlsSafetyEnable	776
11.3.1.5.16	FlsSetModeApi	776
11.3.1.5.17	FlsTotalSize	777
11.3.1.5.18	FlsUseInterrupts	778
11.3.1.5.19	FlsVersionInfoApi	778
11.3.1.6	Container: FlsIfxSpecificConfig	778
11.3.1.6.1	FlsEraseSuspendTimeout	779
11.3.1.6.2	FlsIllegalStateNotification	779
11.3.1.6.3	FlsStateVarStruct	780
11.3.1.6.4	FlsUseEraseSuspend	780
11.3.1.7	Container: FlsPublishedInformation	780
11.3.1.7.1	FlsAcLocationErase	781
11.3.1.7.2	FlsAcLocationWrite	781
11.3.1.7.3	FlsAcSizeErase	782
11.3.1.7.4	FlsAcSizeWrite	782
11.3.1.7.5	FlsEraseTime	782
11.3.1.7.6	FlsErasedValue	783
11.3.1.7.7	FlsExpectedHwId	783
11.3.1.7.8	FlsSpecifiedEraseCycles	784
11.3.1.7.9	FlsWriteTime	784
11.3.1.8	Container: FlsSector	785
11.3.1.8.1	FlsNumberOfSectors	785
11.3.1.8.2	FlsPageSize	785
11.3.1.8.3	FlsSectorSize	786
11.3.1.8.4	FlsSectorStartaddress	786
11.3.1.9	Container: FlsSectorList	787
11.3.2	Functions - Type definitions	787
11.3.2.1	Fls_17_Dmu_AddressType	787
11.3.2.2	Fls_17_Dmu_ConfigType	787
11.3.2.3	Fls_17_Dmu_HardenType	788
11.3.2.4	Fls_17_Dmu_Job_Type	788
11.3.2.5	Fls_17_Dmu_LengthType	789

Table of contents

11.3.2.6	Fls_17_Dmu_NotifFunctionPtrType	789
11.3.3	Functions - APIs	789
11.3.3.1	Fls_17_Dmu_BankCheck	789
11.3.3.2	Fls_17_Dmu_Cancel	791
11.3.3.3	Fls_17_Dmu_CancelNonEraseJobs	791
11.3.3.4	Fls_17_Dmu_Compare	792
11.3.3.5	Fls_17_Dmu_CompareWordsSync	794
11.3.3.6	Fls_17_Dmu_ControlTimeoutDet	795
11.3.3.7	Fls_17_Dmu_Erase	796
11.3.3.8	Fls_17_Dmu_GetJobResult	797
11.3.3.9	Fls_17_Dmu_GetNotifCaller	798
11.3.3.10	Fls_17_Dmu_GetOperStatus	799
11.3.3.11	Fls_17_Dmu_GetStatus	799
11.3.3.12	Fls_17_Dmu_GetVersionInfo	800
11.3.3.13	Fls_17_Dmu_Init	801
11.3.3.14	Fls_17_Dmu_InitCheck	802
11.3.3.15	Fls_17_Dmu_IsHardeningRequired	803
11.3.3.16	Fls_17_Dmu_Read	804
11.3.3.17	Fls_17_Dmu_ReadWordsSync	805
11.3.3.18	Fls_17_Dmu_ResumeErase	807
11.3.3.19	Fls_17_Dmu_SetMode	808
11.3.3.20	Fls_17_Dmu_SuspendErase	808
11.3.3.21	Fls_17_Dmu_VerifyErase	809
11.3.3.22	Fls_17_Dmu_VerifySectorErase	811
11.3.3.23	Fls_17_Dmu_Write	812
11.3.4	Notifications and Callbacks	813
11.3.5	Scheduled functions	813
11.3.5.1	Fls_17_Dmu_MainFunction	813
11.3.6	Interrupt service routines	815
11.3.6.1	Fls_17_Dmu_Isr	815
11.3.7	Error codes classification	816
11.3.7.1	Development errors	816
11.3.7.2	Production errors	817
11.3.7.3	Safety errors	818
11.3.7.4	Runtime errors	819
11.3.8	Deviations and limitations	820
11.3.8.1	Deviations	820
11.3.8.2	Limitations	822
11.3.9	Unsupported hardware features	822
12	GPT driver	824
12.1	User information	824

Table of contents

12.1.1	Description	824
12.1.2	Hardware-software mapping	824
12.1.2.1	GTM: primary hardware peripheral	825
12.1.2.2	SCU: dependent hardware peripheral	826
12.1.2.3	GPT12: primary hardware peripheral	827
12.1.3	File structure	828
12.1.3.1	C file structure	828
12.1.3.2	Code generator plugin files	829
12.1.4	Integration hints	830
12.1.4.1	Integration with AUTOSAR stack	830
12.1.4.2	Multicore and resource manager	834
12.1.4.3	MCU support	835
12.1.4.4	Port support	835
12.1.4.5	DMA support	835
12.1.4.6	Interrupt connections	835
12.1.4.7	Example usage	837
12.1.5	Key architectural considerations	839
12.1.5.1	Hardware dependency	839
12.1.5.2	User mode support	839
12.2	Assumptions of Use (AoUs)	840
12.3	Reference information	841
12.3.1	Configuration interfaces	841
12.3.1.1	Container: CommonPublishedInformation	841
12.3.1.1.1	ArMajorVersion	841
12.3.1.1.2	ArMinorVersion	842
12.3.1.1.3	ArPatchVersion	842
12.3.1.1.4	ModuleId	843
12.3.1.1.5	Release	843
12.3.1.1.6	SwMajorVersion	844
12.3.1.1.7	SwMinorVersion	844
12.3.1.1.8	SwPatchVersion	844
12.3.1.1.9	VendorId	845
12.3.1.2	Container: Gpt12TimerOutputModuleConfiguration	845
12.3.1.2.1	Gpt12ChannelClockDivider	845
12.3.1.2.2	Gpt12TimerUsed	846
12.3.1.3	Container: GtmTimerOutputModuleConfiguration	846
12.3.1.3.1	GtmTimerClockSelect	847
12.3.1.3.2	GtmTimerUsed	847
12.3.1.4	Container: Gpt	848
12.3.1.4.1	Config Variant	848
12.3.1.5	Container: GptChannelConfigSet	848
12.3.1.6	Container: GptChannelConfiguration	849

Table of contents

12.3.1.6.1	GptAssignedHwUnit	849
12.3.1.6.2	GptChannelClkSrcRef	849
12.3.1.6.3	GptChannelId	850
12.3.1.6.4	GptChannelMode	850
12.3.1.6.5	GptChannelTickFrequency	851
12.3.1.6.6	GptChannelTickValueMax	851
12.3.1.6.7	GptEnableWakeup	852
12.3.1.6.8	GptNotification	852
12.3.1.6.9	GptTimerChannelUsage	853
12.3.1.7	Container: GptClockReferencePoint	853
12.3.1.7.1	GptClockReference	854
12.3.1.8	Container: GptConfigurationOfOptApiServices	854
12.3.1.8.1	GptDeinitApi	854
12.3.1.8.2	GptEnableDisableNotificationApi	855
12.3.1.8.3	GptInitCheckApi	855
12.3.1.8.4	GptTimeElapsedApi	856
12.3.1.8.5	GptTimeRemainingApi	856
12.3.1.8.6	GptVersionInfoApi	857
12.3.1.8.7	GptWakeupFunctionalityApi	857
12.3.1.9	Container: GptDriverConfiguration	858
12.3.1.9.1	GptDevErrorDetect	858
12.3.1.9.2	GptMultiCoreErrorDetect	858
12.3.1.9.3	GptPredefTimer100us32bitEnable	859
12.3.1.9.4	GptPredefTimer1usEnablingGrade	859
12.3.1.9.5	GptReportWakeupSource	860
12.3.1.9.6	GptSafetyEnable	861
12.3.1.10	Container: GptWakeupConfiguration	861
12.3.1.10.1	GptWakeupSourceRef	861
12.3.2	Functions - Type definitions	862
12.3.2.1	Gpt_ChannelType	862
12.3.2.2	Gpt_ModeType	862
12.3.2.3	Gpt_NotificationPtrType	862
12.3.2.4	Gpt_PredefTimerType	863
12.3.2.5	Gpt_ConfigType	863
12.3.2.6	Gpt_ValueType	863
12.3.3	Functions - APIs	864
12.3.3.1	Gpt_GetVersionInfo	864
12.3.3.2	Gpt_Init	865
12.3.3.3	Gpt_Deinit	866
12.3.3.4	Gpt_StartTimer	867
12.3.3.5	Gpt_StopTimer	868
12.3.3.6	Gpt_GetTimeElapsed	869

Table of contents

12.3.3.7	Gpt_GetTimeRemaining	869
12.3.3.8	Gpt_GetPredefTimerValue	870
12.3.3.9	Gpt_EnableNotification	871
12.3.3.10	Gpt_DisableNotification	872
12.3.3.11	Gpt_EnableWakeup	873
12.3.3.12	Gpt_DisableWakeup	874
12.3.3.13	Gpt_CheckWakeup	875
12.3.3.14	Gpt_SetMode	876
12.3.3.15	Gpt_InitCheck	876
12.3.4	Notifications and Callbacks	877
12.3.4.1	Gpt_Isr	877
12.3.5	Scheduled functions	878
12.3.6	Interrupt service routines	878
12.3.7	Error codes classification	878
12.3.7.1	Development errors	878
12.3.7.2	Production errors	880
12.3.7.3	Safety errors	880
12.3.7.4	Runtime errors	880
12.3.8	Deviations and limitations	880
12.3.8.1	Deviations	880
12.3.8.2	Limitations	881
12.3.9	Unsupported hardware features	881
13	Icu_17_TimerIp driver	882
13.1	User information	882
13.1.1	Description	882
13.1.2	Hardware-software mapping	882
13.1.2.1	GTM-TIM: primary hardware peripheral	882
13.1.2.2	CCU6: primary hardware peripheral	883
13.1.2.3	SCU	884
13.1.2.4	GPT12: primary hardware peripheral	884
13.1.2.5	Port: dependent hardware peripheral	885
13.1.3	File structure	885
13.1.3.1	C file structure	885
13.1.3.2	Code generator plugin files	887
13.1.4	Integration hints	888
13.1.4.1	Integration with AUTOSAR stack	889
13.1.4.2	Multicore and Resource Manager	893
13.1.4.3	MCU support	893
13.1.4.4	Port support	894
13.1.4.5	DMA support	894
13.1.4.6	Interrupt connections	894

Table of contents

13.1.4.7	Example usage	896
13.1.5	Key architectural considerations	905
13.1.5.1	Overflow handling for signal measurement	905
13.1.5.2	Accessing shared SFR	905
13.2	Assumptions of Use (AoUs)	906
13.3	Reference information	907
13.3.1	Configuration interfaces	907
13.3.1.1	Container: CCU6CC6Configuration	907
13.3.1.1.1	CCChannelInputSelection	908
13.3.1.1.2	CCU6KernelUsed	908
13.3.1.1.3	Cc6xChannel	909
13.3.1.2	Container: CCU6xParameters	909
13.3.1.2.1	CCU6InterruptNode	909
13.3.1.2.2	T12ClkSelection	910
13.3.1.2.3	T12PrescalerEnabled	910
13.3.1.3	Container: CommonPublishedInformation	911
13.3.1.3.1	ArMajorVersion	911
13.3.1.3.2	ArMinorVersion	911
13.3.1.3.3	ArPatchVersion	912
13.3.1.3.4	ModuleId	912
13.3.1.3.5	Release	913
13.3.1.3.6	SwMajorVersion	913
13.3.1.3.7	SwMinorVersion	913
13.3.1.3.8	SwPatchVersion	914
13.3.1.3.9	VendorApilInfix	914
13.3.1.3.10	VendorId	915
13.3.1.4	Container: ERUIInputConfiguration	915
13.3.1.4.1	EruErsReference	915
13.3.1.4.2	EruInputPin	916
13.3.1.4.3	EruOguReference	916
13.3.1.5	Container: GPT12Configuration	917
13.3.1.5.1	GPT12BlockReference	917
13.3.1.5.2	GPT12CounterType	918
13.3.1.5.3	GPT12DirPortSelection	918
13.3.1.5.4	GPT12InputPortSelection	919
13.3.1.6	Container: GtmTimerInputConfiguration	919
13.3.1.6.1	GtmTimerUsed	919
13.3.1.7	Container: IcuTimestampMeasurement	920
13.3.1.7.1	IcuTimestampMeasurementProperty	920
13.3.1.7.2	IcuTimestampNotification	920
13.3.1.8	Container: TimChannelFilterConfig	921
13.3.1.8.1	TimChFilterCounterFreqSelect	921

Table of contents

13.3.1.8.2	TimChFilterModeForFallingEdge	922
13.3.1.8.3	TimChFilterModeForRisingEdge	922
13.3.1.8.4	TimChFilterTimeForFallingEdge	923
13.3.1.8.5	TimChFilterTimeForRisingEdge	923
13.3.1.8.6	TimChannelFilterEnable	924
13.3.1.9	Container: TimChannelGeneral	924
13.3.1.9.1	OverflowISRThreshold	924
13.3.1.9.2	TimChannelClockSelect	925
13.3.1.9.3	TimChannelGpr0InputSelect	926
13.3.1.9.4	TimChannelInputSelect	926
13.3.1.9.5	TimChannelPortPinSelect	927
13.3.1.9.6	TimlInterruptMode	928
13.3.1.10	Container: Icu	928
13.3.1.10.1	Config Variant	928
13.3.1.11	Container: IcuChannel	929
13.3.1.11.1	IcuAssignedHwUnit	929
13.3.1.11.2	IcuChannelId	929
13.3.1.11.3	IcuDefaultStartEdge	930
13.3.1.11.4	IcuMeasurementMode	931
13.3.1.11.5	IcuWakeupCapability	932
13.3.1.12	Container: IcuConfigSet	933
13.3.1.12.1	IcuMaxChannel	933
13.3.1.13	Container: IcuGeneral	934
13.3.1.13.1	IcuDevErrorDetect	934
13.3.1.13.2	IcuIndex	934
13.3.1.13.3	IcuInitDeInitApiMode	935
13.3.1.13.4	IcuMultiCoreErrorDetect	935
13.3.1.13.5	IcuReportWakeupSource	936
13.3.1.13.6	IcuRuntimeApiMode	936
13.3.1.13.7	IcuSafetyEnable	937
13.3.1.14	Container: IcuOptionalApis	937
13.3.1.14.1	IcuDeInitApi	937
13.3.1.14.2	IcuDisableWakeupApi	938
13.3.1.14.3	IcuEdgeCountApi	938
13.3.1.14.4	IcuEdgeDetectApi	939
13.3.1.14.5	IcuEnableWakeupApi	940
13.3.1.14.6	IcuGetDutyCycleValuesApi	940
13.3.1.14.7	IcuGetInputStateApi	941
13.3.1.14.8	IcuGetTimeElapsedApi	941
13.3.1.14.9	IcuGetVersionInfoApi	942
13.3.1.14.10	IcuIncrementalInterfaceApi	942
13.3.1.14.11	IcuInitCheckApi	943

Table of contents

13.3.1.14.12	IcuSetModeApi	944
13.3.1.14.13	IcuSignalMeasurementApi	944
13.3.1.14.14	IcuTimestampApi	945
13.3.1.14.15	IcuWakeupFunctionalityApi	946
13.3.1.15	Container: IcuSignalEdgeDetection	946
13.3.1.15.1	IcuSignalNotification	946
13.3.1.16	Container: IcuSignalMeasurement	947
13.3.1.16.1	IcuSignalMeasurementProperty	947
13.3.1.17	Container: IcuWakeup	948
13.3.1.17.1	IcuChannelWakeUpInfo	948
13.3.2	Functions - Type definitions	948
13.3.2.1	Icu_17_TimerIp_NotifiPtrType	948
13.3.2.2	Icu_17_TimerIp_ModeType	949
13.3.2.3	Icu_17_TimerIp_ChannelType	949
13.3.2.4	Icu_17_TimerIp_EncCountDirType	950
13.3.2.5	Icu_17_TimerIp_InputStateType	950
13.3.2.6	Icu_17_TimerIp_ConfigType	950
13.3.2.7	Icu_17_TimerIp_ActivationType	950
13.3.2.8	Icu_17_TimerIp_ValueType	951
13.3.2.9	Icu_17_TimerIp_DutyCycleType	951
13.3.2.10	Icu_17_TimerIp_IndexType	952
13.3.2.11	Icu_17_TimerIp_EdgeNumberType	952
13.3.2.12	Icu_17_TimerIp_MeasurementModeType	952
13.3.2.13	Icu_17_TimerIp_SignalMeasurementPropertyType	953
13.3.2.14	Icu_17_TimerIp_TimestampBufferType	953
13.3.3	Functions - APIs	954
13.3.3.1	Icu_17_TimerIp_CalibratePos	954
13.3.3.2	Icu_17_TimerIp_CheckWakeup	955
13.3.3.3	Icu_17_TimerIp_DelInit	955
13.3.3.4	Icu_17_TimerIp_DisableEdgeCount	956
13.3.3.5	Icu_17_TimerIp_DisableNotification	957
13.3.3.6	Icu_17_TimerIp_DisableWakeup	958
13.3.3.7	Icu_17_TimerIp_EnableEdgeCount	959
13.3.3.8	Icu_17_TimerIp_EnableMultiEdgeDetection	960
13.3.3.9	Icu_17_TimerIp_EnableNotification	961
13.3.3.10	Icu_17_TimerIp_EnableWakeup	962
13.3.3.11	Icu_17_TimerIp_GetDutyCycleValues	963
13.3.3.12	Icu_17_TimerIp_GetEdgeNumbers	964
13.3.3.13	Icu_17_TimerIp_GetInputState	965
13.3.3.14	Icu_17_TimerIp_GetTimeElapsed	967
13.3.3.15	Icu_17_TimerIp_GetTimestampIndex	968
13.3.3.16	Icu_17_TimerIp_GetVersionInfo	969

Table of contents

13.3.3.17	Icu_17_TimerIp_InitCheck	969
13.3.3.18	Icu_17_TimerIp_ReadEncCount	970
13.3.3.19	Icu_17_TimerIp_ReadEncCountDir	971
13.3.3.20	Icu_17_TimerIp_ResetEdgeCount	972
13.3.3.21	Icu_17_TimerIp_SetActivationCondition	973
13.3.3.22	Icu_17_TimerIp_StartInclInterface	975
13.3.3.23	Icu_17_TimerIp_StartSignalMeasurement	976
13.3.3.24	Icu_17_TimerIp_StartTimestamp	977
13.3.3.25	Icu_17_TimerIp_StopInclInterface	978
13.3.3.26	Icu_17_TimerIp_StopSignalMeasurement	979
13.3.3.27	Icu_17_TimerIp_StopTimestamp	980
13.3.3.28	Icu_17_TimerIp_EnableEdgeDetection	981
13.3.3.29	Icu_17_TimerIp_DisableEdgeDetection	982
13.3.3.30	Icu_17_TimerIp_Init	983
13.3.3.31	Icu_17_TimerIp_SetMode	984
13.3.4	Notifications and Callbacks	985
13.3.4.1	Icu_17_TimerIp_Timer_Isr	985
13.3.5	Scheduled functions	986
13.3.6	Interrupt service routines	986
13.3.7	Error codes classification	986
13.3.7.1	Development errors	986
13.3.7.2	Production errors	990
13.3.7.3	Safety errors	990
13.3.7.4	Runtime errors	992
13.3.8	Deviations and limitations	992
13.3.8.1	Deviations	992
13.3.8.2	Limitations	992
13.3.9	Unsupported hardware features	992
14	IRQ driver	993
14.1	User information	993
14.1.1	Description	993
14.1.2	Hardware-software mapping	993
14.1.3	File structure	994
14.1.3.1	C file structure	994
14.1.3.2	Code generator plugin files	995
14.1.4	Integration hints	996
14.1.4.1	Integration with AUTOSAR stack	996
14.1.4.2	Multicore and Resource Manager	997
14.1.4.3	MCU support	997
14.1.4.4	PORT support	998
14.1.4.5	DMA support	998

Table of contents

14.1.4.6	Interrupt connections	998
14.1.4.7	Example usage	998
14.1.4.7.1	Configuring the driver	998
14.1.4.7.2	Initializing interrupts	999
14.1.4.7.3	Clearing interrupts	999
14.1.5	Key architectural considerations	1000
14.2	Assumptions of Use (AoUs)	1000
14.3	Reference information	1000
14.3.1	Configuration interfaces	1000
14.3.1.1	Container: IrqGeneral	1000
14.3.1.1.1	IrqOSekEnable	1000
14.3.1.2	Container: Irq<ModuleName>Config	1001
14.3.1.2.1	Container: Irq<ModuleName>CatConfig	1001
14.3.1.2.2	Container: Irq<ModuleName>PrioConfig	1002
14.3.1.2.3	Container: Irq<ModuleName>TosConfig	1002
14.3.1.3	Container: CommonPublishedInformation	1003
14.3.1.3.1	ArMajorVersion	1003
14.3.1.3.2	ArMinorVersion	1003
14.3.1.3.3	ArPatchVersion	1004
14.3.1.3.4	SwMajorVersion	1004
14.3.1.3.5	SwMinorVersion	1005
14.3.1.3.6	SwPatchVersion	1005
14.3.1.3.7	ModuleId	1005
14.3.1.3.8	VendorId	1006
14.3.2	Functions – Type definitions	1006
14.3.3	Functions - APIs	1006
14.3.3.1	Interrupt init functions	1006
14.3.3.2	Irq_ClearAllInterruptFlags	1007
14.3.4	Notifications and callbacks	1007
14.3.5	Scheduled functions	1007
14.3.6	Interrupt service routines	1007
14.3.7	Error codes classification	1007
14.3.7.1	Production errors	1008
14.3.7.2	Development errors	1008
14.3.7.3	Runtime errors	1008
14.3.8	Deviations and limitations	1008
14.3.8.1	Deviations	1008
14.3.8.2	Limitations	1008
14.3.9	Unsupported hardware features	1008
15	Lin_17_AscLin driver	1009
15.1	User information	1009

Table of contents

15.1.1	Description	1009
15.1.2	Hardware-software mapping	1009
15.1.2.1	ASCLIN: primary hardware peripheral	1010
15.1.2.2	Port: dependent hardware peripheral	1011
15.1.2.3	SCU: dependent hardware peripheral	1011
15.1.2.4	SRC: dependent hardware peripheral	1012
15.1.3	File structure	1012
15.1.3.1	C file structure	1012
15.1.3.2	Code generator plugin files	1014
15.1.4	Integration hints	1015
15.1.4.1	Integration with AUTOSAR stack	1015
15.1.4.2	Multicore and Resource Manager	1017
15.1.4.3	MCU support	1017
15.1.4.4	Port support	1019
15.1.4.5	DMA support	1021
15.1.4.6	Interrupt connections	1021
15.1.4.7	Example usage	1024
15.1.5	Key architectural considerations	1027
15.1.5.1	ASCLIN hardware: used for LIN feature	1027
15.1.5.2	Modes of operation - TxFIFO and RxFIFO modes	1027
15.1.5.3	Addition of LinInterruptDisabled configuration parameter	1028
15.2	Assumptions of Use (AoUs)	1029
15.3	Reference information	1030
15.3.1	Configuration interfaces	1030
15.3.1.1	Container: CommonPublishedInformation	1030
15.3.1.1.1	ArMajorVersion	1030
15.3.1.1.2	ArMinorVersion	1031
15.3.1.1.3	ArPatchVersion	1031
15.3.1.1.4	ModuleId	1032
15.3.1.1.5	Release	1032
15.3.1.1.6	SwMajorVersion	1033
15.3.1.1.7	SwMinorVersion	1033
15.3.1.1.8	SwPatchVersion	1033
15.3.1.1.9	VendorApiInfix	1034
15.3.1.1.10	VendorId	1034
15.3.1.2	Container: Lin	1035
15.3.1.2.1	Config Variant	1035
15.3.1.3	Container: LinChannel	1035
15.3.1.3.1	LinAutoCalcBaudParams	1035
15.3.1.3.2	LinChanAssignedHw	1036
15.3.1.3.3	LinChannelBaudDenominator	1037
15.3.1.3.4	LinChannelBaudNumerator	1037

Table of contents

15.3.1.3.5	LinChannelBaudPreScalar	1038
15.3.1.3.6	LinChannelBaudRate	1038
15.3.1.3.7	LinChannelEcuMWakeupSource	1039
15.3.1.3.8	LinChannelId	1039
15.3.1.3.9	LinChannelWakeupSupport	1040
15.3.1.3.10	LinClockRef	1040
15.3.1.3.11	LinInterByteSpace	1041
15.3.1.3.12	LinRxAlternateInputSignal	1041
15.3.1.4	Container: LinDemEventParameterRefs	1042
15.3.1.4.1	LIN_E_TIMEOUT	1042
15.3.1.5	Container: LinGeneral	1042
15.3.1.5.1	LinCsrClksel	1042
15.3.1.5.2	LinDevErrorDetect	1043
15.3.1.5.3	LinHwMcuTrigSleepEnable	1043
15.3.1.5.4	LinIndex	1044
15.3.1.5.5	LinInitDeInitApiMode	1044
15.3.1.5.6	LinInterruptDisabled	1045
15.3.1.5.7	LinSysClockRef	1046
15.3.1.5.8	LinTimeoutDuration	1046
15.3.1.5.9	LinVersionInfoApi	1047
15.3.1.6	Container: LinGlobalConfig	1047
15.3.2	Functions - Type definitions	1047
15.3.2.1	Lin_17_AscLin_ConfigType	1047
15.3.2.2	Lin_FramePidType	1048
15.3.2.3	Lin_FrameCsModelType	1048
15.3.2.4	Lin_FrameResponseType	1048
15.3.2.5	Lin_FrameDlType	1049
15.3.2.6	Lin_PduType	1049
15.3.2.7	Lin_StatusType	1049
15.3.3	Functions - APIs	1051
15.3.3.1	Lin_17_AscLin_CheckWakeup	1051
15.3.3.2	Lin_17_AscLin_GetStatus	1052
15.3.3.3	Lin_17_AscLin_GoToSleep	1053
15.3.3.4	Lin_17_AscLin_GoToSleepInternal	1054
15.3.3.5	Lin_17_AscLin_Init	1055
15.3.3.6	Lin_17_AscLin_SendFrame	1056
15.3.3.7	Lin_17_AscLin_Wakeup	1057
15.3.3.8	Lin_17_AscLin_WakeupInternal	1057
15.3.3.9	Lin_17_AscLin_GetVersionInfo	1058
15.3.4	Notifications and Callbacks	1059
15.3.5	Scheduled functions	1059
15.3.6	Interrupt service routines	1059

Table of contents

15.3.6.1	Lin_17_AscLin_IsrError	1059
15.3.6.2	Lin_17_AscLin_IsrReceive	1060
15.3.6.3	Lin_17_AscLin_IsrTransmit	1061
15.3.7	Error codes classification	1062
15.3.7.1	Development errors	1062
15.3.7.2	Production errors	1063
15.3.7.3	Safety errors	1063
15.3.7.4	Runtime errors	1063
15.3.8	Deviations and limitations	1063
15.3.8.1	Deviations	1063
15.3.8.2	Limitations	1064
15.3.9	Unsupported hardware features	1064
16	MCALLIB driver	1065
16.1	User information	1065
16.1.1	Description	1065
16.1.2	Hardware-software mapping	1065
16.1.2.1	STM: primary hardware peripheral	1066
16.1.2.2	CORE_ID-CPU: primary hardware peripheral	1066
16.1.2.3	SCU: dependent hardware peripheral	1066
16.1.3	File structure	1067
16.1.3.1	C file structure	1067
16.1.3.2	Code generator plugin files	1068
16.1.4	Integration hints	1069
16.1.4.1	Integration with AUTOSAR stack	1069
16.1.4.2	Multicore and Resource Manager	1073
16.1.4.3	MCU support	1073
16.1.4.4	Port support	1073
16.1.4.5	DMA support	1073
16.1.4.6	Interrupt connections	1074
16.1.4.7	Example usage	1075
16.1.5	Key architectural considerations	1075
16.1.5.1	User mode	1075
16.1.5.2	Spinlock	1075
16.2	Assumptions of Use (AoUs)	1076
16.3	Reference information	1078
16.3.1	Configuration interfaces	1078
16.3.1.1	Container: McalLibGeneral	1079
16.3.1.1.1	McalLibSafetyEnable	1079
16.3.1.2	Container: CommonPublishedInformation	1079
16.3.1.2.1	ArMajorVersion	1079
16.3.1.2.2	ArMinorVersion	1080

Table of contents

16.3.1.2.3	ArPatchVersion	1080
16.3.1.2.4	ModuleId	1081
16.3.1.2.5	Release	1081
16.3.1.2.6	SwMajorVersion	1082
16.3.1.2.7	SwMinorVersion	1082
16.3.1.2.8	SwPatchVersion	1082
16.3.1.2.9	VendorId	1083
16.3.1.3	Container: McalLib	1083
16.3.1.4	Container: McalLibPublishedInformation	1083
16.3.1.4.1	McalLibBackUpClockFrequency	1084
16.3.1.4.2	McalLibDsprCore0EndAddr	1084
16.3.1.4.3	McalLibDsprCore0StartAddr	1084
16.3.1.4.4	McalLibDsprCore1EndAddr	1085
16.3.1.4.5	McalLibDsprCore1StartAddr	1085
16.3.1.4.6	McalLibDsprCore2EndAddr	1086
16.3.1.4.7	McalLibDsprCore2StartAddr	1086
16.3.1.4.8	McalLibDsprCore3EndAddr	1087
16.3.1.4.9	McalLibDsprCore3StartAddr	1087
16.3.1.4.10	McalLibDsprCore4EndAddr	1088
16.3.1.4.11	McalLibDsprCore4StartAddr	1088
16.3.1.4.12	McalLibDsprCore5EndAddr	1088
16.3.1.4.13	McalLibDsprCore5StartAddr	1089
16.3.1.4.14	McalLibMcalAvailableCores	1089
16.3.1.4.15	McalLibPsprCore0EndAddr	1090
16.3.1.4.16	McalLibPsprCore0StartAddr	1090
16.3.1.4.17	McalLibPsprCore1EndAddr	1091
16.3.1.4.18	McalLibPsprCore1StartAddr	1091
16.3.1.4.19	McalLibPsprCore2EndAddr	1092
16.3.1.4.20	McalLibPsprCore2StartAddr	1092
16.3.1.4.21	McalLibPsprCore3EndAddr	1092
16.3.1.4.22	McalLibPsprCore3StartAddr	1093
16.3.1.4.23	McalLibPsprCore4EndAddr	1093
16.3.1.4.24	McalLibPsprCore4StartAddr	1094
16.3.1.4.25	McalLibPsprCore5EndAddr	1094
16.3.1.4.26	McalLibPsprCore5StartAddr	1095
16.3.2	Functions - Type definitions	1095
16.3.2.1	unsigned_int	1095
16.3.3	Functions - APIs	1096
16.3.3.1	Mcal_WriteSafetyEndInitProtReg16	1096
16.3.3.2	Mcal_WriteSafetyEndInitProtRegMask	1097
16.3.3.3	McalLib_GetVersionInfo	1098
16.3.3.4	Mcal_GetCpuIndex	1098

Table of contents

16.3.3.5	Mcal_GetCpuPhysicalId	1099
16.3.3.6	Mcal_DelayGetTick	1100
16.3.3.7	Mcal_DelayResetTickCalibration	1101
16.3.3.8	Mcal_DelayTickResolution	1102
16.3.3.9	Mcal_GetBitAtomic	1103
16.3.3.10	Mcal_SetBitAtomic	1103
16.3.3.11	Mcal_GetGlobalDsprAddress	1104
16.3.3.12	Mcal_GetGlobalPsprAddress	1105
16.3.3.13	Mcal_GetLocalDsprAddress	1106
16.3.3.14	Mcal_GetLocalPsprAddress	1107
16.3.3.15	Mcal_GetPeripheralEndInitPassword	1108
16.3.3.16	Mcal_GetCpuWdgPassword	1109
16.3.3.17	Mcal_GetSafetyEndInitPassword	1110
16.3.3.18	Mcal_WriteSafetyEndInitProtReg	1110
16.3.3.19	Mcal_SetCpuWdgPassword	1111
16.3.3.20	Mcal_SetPeripheralEndInitPassword	1112
16.3.3.21	Mcal_SetSafetyEndInitPassword	1113
16.3.3.22	Mcal_GetSpinlock	1114
16.3.3.23	Mcal_ReleaseSpinlock	1115
16.3.3.24	Mcal_WriteCpuEndInitProtReg	1115
16.3.3.25	Mcal_WritePeripEndInitProtReg	1116
16.3.4	Notifications and Callbacks	1117
16.3.5	Scheduled functions	1117
16.3.6	Interrupt service routines	1117
16.3.7	Error codes classification	1117
16.3.7.1	Development errors	1118
16.3.7.2	Production errors	1118
16.3.7.3	Safety errors	1118
16.3.7.4	Runtime errors	1118
16.3.8	Deviations and limitations	1118
16.3.8.1	Deviations and Assumptions	1118
16.3.8.2	Limitations	1119
16.3.9	Unsupported hardware features	1119
17	MCU driver	1120
17.1	User information	1120
17.1.1	Description	1120
17.1.2	Hardware-software mapping	1120
17.1.2.1	GTM: primary hardware peripheral	1121
17.1.2.2	SCU: primary hardware peripheral	1122
17.1.2.3	CONVERTER: primary hardware peripheral	1122
17.1.2.4	CCU6: primary hardware peripheral	1122

Table of contents

17.1.2.5	GPT12: primary hardware peripheral	1123
17.1.2.6	PMS: primary hardware peripheral	1123
17.1.2.7	STM: primary hardware peripheral	1123
17.1.3	File structure	1124
17.1.3.1	C file structure	1124
17.1.3.2	Code generator plugin files	1128
17.1.4	Integration hints	1130
17.1.4.1	Integration with AUTOSAR stack	1130
17.1.4.2	Multicore and Resource Manager	1133
17.1.4.3	MCU support	1134
17.1.4.4	Port support	1134
17.1.4.5	DMA support	1134
17.1.4.6	Interrupt connections	1134
17.1.4.7	Example usage	1135
17.1.5	Key architectural considerations	1136
17.1.5.1	GTM: usage with complex drivers	1136
17.1.5.2	Multicore support for MCU	1136
17.1.5.3	Usage of Mcu_DelInit API	1136
17.1.5.4	Error handling for Timer IP APIs	1136
17.1.5.5	User mode support	1136
17.1.5.6	Reset reason due to HSM	1137
17.1.5.7	Reset reason due to multiple resets	1137
17.1.5.8	Power modes entry	1137
17.1.5.9	Generic AoUs to users of MCU	1137
17.1.5.10	Timer channel reservation in MCU hardware resource allocation	1137
17.1.5.11	Usage of Mcu_SetMode API	1137
17.1.5.12	Cluster 0 clock should not be disabled if GTM is to be used	1137
17.1.5.13	CCU6 and GPT12 initialization is performed only for the kernel/timers reserved by the user	1137
17.1.5.14	Approximation of frequency to divider calculation	1138
17.1.5.15	Timer APIs in the driver	1138
17.2	Assumptions of Use (AoUs)	1139
17.3	Reference information	1141
17.3.1	Configuration interfaces	1141
17.3.1.1	Container: McuClockMonitorConfig	1146
17.3.1.1.1	McuBackupClockMonEnable	1147
17.3.1.1.2	McuBackupClockRangeMonEnable	1147
17.3.1.1.3	McuPll0ClockMonEnable	1148
17.3.1.1.4	McuPll1ClockMonEnable	1148
17.3.1.1.5	McuPll2ClockMonEnable	1149
17.3.1.1.6	McuSpbClockMonEnable	1149
17.3.1.2	Container: McuGpt12PrescalerConf	1150

Table of contents

17.3.1.2.1	Gpt1BlockPrescalerSel	1150
17.3.1.2.2	Gpt2BlockPrescalerSel	1150
17.3.1.3	Container: McuStmAllocationConf	1151
17.3.1.3.1	McuStmCmp0RegAllocationConf	1151
17.3.1.3.2	McuStmCmp1RegAllocationConf	1151
17.3.1.4	Container: CommonPublishedInformation	1152
17.3.1.4.1	ArMajorVersion	1152
17.3.1.4.2	ArMinorVersion	1153
17.3.1.4.3	ArPatchVersion	1153
17.3.1.4.4	ModuleId	1153
17.3.1.4.5	Release	1154
17.3.1.4.6	SwMajorVersion	1154
17.3.1.4.7	SwMinorVersion	1155
17.3.1.4.8	SwPatchVersion	1155
17.3.1.4.9	VendorId	1156
17.3.1.5	Container: GtmAtomActionTimeBaseUnitConf	1156
17.3.1.5.1	GtmAtomActionTimeBaseSelection	1156
17.3.1.5.2	GtmAtomActionTimeBaseValue	1157
17.3.1.6	Container: GtmAtomChannelConf	1157
17.3.1.6.1	GtmAtomChInternalTriggerEnable	1157
17.3.1.6.2	GtmAtomChResetCn0OnTriggerEnable	1158
17.3.1.7	Container: GtmAtomGlobalConf	1159
17.3.1.8	Container: GtmAtomGroupConf	1159
17.3.1.9	Container: GtmClusterConf	1159
17.3.1.9.1	GtmCmuClusterInputClockDividerEnable	1159
17.3.1.10	Container: GtmClusterConfClockSetting	1160
17.3.1.10.1	GtmClusterConfClock0Src	1160
17.3.1.10.2	GtmClusterConfClock1Src	1160
17.3.1.10.3	GtmClusterConfClock2Src	1161
17.3.1.10.4	GtmClusterConfClock3Src	1162
17.3.1.10.5	GtmClusterConfClock4Src	1162
17.3.1.10.6	GtmClusterConfClock5Src	1163
17.3.1.10.7	GtmClusterConfClock6Src	1163
17.3.1.10.8	GtmClusterConfClock7Src	1164
17.3.1.11	Container: GtmClusterFixedClockSetting	1164
17.3.1.11.1	GtmClusterFixedClockSrc	1164
17.3.1.12	Container: GtmConfigClockSetting	1165
17.3.1.12.1	GtmCmuConfigClock0Div	1165
17.3.1.12.2	GtmCmuConfigClock0Enable	1165
17.3.1.12.3	GtmCmuConfigClock1Div	1166
17.3.1.12.4	GtmCmuConfigClock1Enable	1167
17.3.1.12.5	GtmCmuConfigClock2Div	1167

Table of contents

17.3.1.12.6	GtmCmuConfigClock2Enable	1168
17.3.1.12.7	GtmCmuConfigClock3Div	1168
17.3.1.12.8	GtmCmuConfigClock3Enable	1169
17.3.1.12.9	GtmCmuConfigClock4Div	1169
17.3.1.12.10	GtmCmuConfigClock4Enable	1170
17.3.1.12.11	GtmCmuConfigClock5Div	1170
17.3.1.12.12	GtmCmuConfigClock5Enable	1171
17.3.1.12.13	GtmCmuConfigClock6Div	1172
17.3.1.12.14	GtmCmuConfigClock6Enable	1172
17.3.1.12.15	GtmCmuConfigClock7Div	1173
17.3.1.12.16	GtmCmuConfigClock7Enable	1173
17.3.1.13	Container: GtmExtClockSetting	1174
17.3.1.13.1	GtmCmuExtClock0Denominator	1174
17.3.1.13.2	GtmCmuExtClock0Enable	1174
17.3.1.13.3	GtmCmuExtClock0Numerator	1175
17.3.1.13.4	GtmCmuExtClock1Denominator	1175
17.3.1.13.5	GtmCmuExtClock1Enable	1176
17.3.1.13.6	GtmCmuExtClock1Numerator	1176
17.3.1.13.7	GtmCmuExtClock2Denominator	1177
17.3.1.13.8	GtmCmuExtClock2Enable	1177
17.3.1.13.9	GtmCmuExtClock2Numerator	1178
17.3.1.14	Container: GtmFixedClockSetting	1178
17.3.1.14.1	GtmCmuFixedClockEnable	1179
17.3.1.14.2	GtmCmuFixedClockSel	1179
17.3.1.15	Container: GtmGlobalConfiguration	1180
17.3.1.16	Container: GtmTBUChannelConf	1180
17.3.1.16.1	GtmTbuChClockSourceSelection	1180
17.3.1.16.2	GtmTbuChMode	1181
17.3.1.16.3	GtmTbuChModuloCntrSel	1181
17.3.1.16.4	GtmTbuChResolutionSel	1182
17.3.1.16.5	GtmTbuChannelEnable	1182
17.3.1.17	Container: GtmTomActionTimeBaseUnitConf	1183
17.3.1.17.1	GtmTomActionTimeBaseSelection	1183
17.3.1.17.2	GtmTomActionTimeBaseValue	1184
17.3.1.18	Container: GtmTomChannelConf	1184
17.3.1.18.1	GtmTomChInternalTriggerEnable	1184
17.3.1.18.2	GtmTomChResetCn0OnTriggerEnable	1185
17.3.1.19	Container: GtmTomGroupConf	1186
17.3.1.20	Container: GtmTriggerForAdc	1186
17.3.1.20.1	GtmAdcTrigger0Select	1186
17.3.1.20.2	GtmAdcTrigger1Select	1187
17.3.1.20.3	GtmAdcTrigger2Select	1188

Table of contents

17.3.1.20.4	GtmAdcTrigger3Select	1189
17.3.1.20.5	GtmAdcTrigger4Select	1189
17.3.1.21	Container: GtmTriggerForDsadc	1190
17.3.1.21.1	GtmDsadcTrigger0Select	1191
17.3.1.21.2	GtmDsadcTrigger1Select	1192
17.3.1.21.3	GtmDsadcTrigger2Select	1193
17.3.1.21.4	GtmDsadcTrigger3Select	1194
17.3.1.22	Container: Mcu	1194
17.3.1.22.1	Config Variant	1194
17.3.1.23	Container: McuAscLinChannelAllocationConf	1195
17.3.1.23.1	McuAscLinChannelAllocationConf	1195
17.3.1.24	Container: McuAscLinAllocationConf	1195
17.3.1.25	Container: McuCcu6ModuleAllocationConf	1196
17.3.1.25.1	McuCcu6ModuleAllocationConf	1196
17.3.1.26	Container: McuClockReferencePoint	1196
17.3.1.26.1	McuClockRefSelection	1196
17.3.1.26.2	McuClockReferencePointFrequency	1198
17.3.1.27	Container: McuClockReferencePointConfig	1198
17.3.1.28	Container: McuClockSettingConfig	1198
17.3.1.28.1	McuClockSettingId	1198
17.3.1.29	Container: McuDemEventParameterRefs	1199
17.3.1.29.1	MCU_E_CLOCK_FAILURE	1199
17.3.1.30	Container: McuDemEventParameterRefsConf	1200
17.3.1.30.1	MCU_E_CCU6_CLC_DISABLE_ERR	1200
17.3.1.30.2	MCU_E_CCU6_CLC_ENABLE_ERR	1200
17.3.1.30.3	MCU_E_CCUCON_UPDATE_ERR	1201
17.3.1.30.4	MCU_E_CONVCTRL_CLC_DISABLE_ERR	1201
17.3.1.30.5	MCU_E_CONVCTRL_CLC_ENABLE_ERR	1202
17.3.1.30.6	MCU_E_GPT12_CLC_DISABLE_ERR	1202
17.3.1.30.7	MCU_E_GPT12_CLC_ENABLE_ERR	1202
17.3.1.30.8	MCU_E_GTM_CLC_DISABLE_ERR	1203
17.3.1.30.9	MCU_E_GTM_CLC_ENABLE_ERR	1203
17.3.1.30.10	MCU_E_OSC_FAILURE	1204
17.3.1.30.11	MCU_E_PERIPHERAL_PLL_LOCK_LOSS	1204
17.3.1.30.12	MCU_E_PERIPHERAL_PLL_TIMEOUT_ERR	1205
17.3.1.30.13	MCU_E_PMSWCR_UPDATE_ERR	1205
17.3.1.30.14	MCU_E_SYSTEM_PLL_LOCK_LOSS	1206
17.3.1.30.15	MCU_E_SYSTEM_PLL_TIMEOUT_ERR	1206
17.3.1.31	Container: McuEruAllocationConf	1207
17.3.1.32	Container: McuEruChannelInputLineConf	1207
17.3.1.32.1	McuEruChannelInputLineConf	1207
17.3.1.33	Container: McuEruChannelOutputUnitConf	1208

Table of contents

17.3.1.33.1	McuEruChannelOutputUnitConf	1208
17.3.1.34	Container: McuEruGlobalConf	1208
17.3.1.34.1	McuEruInputFilterRegVal	1208
17.3.1.35	Container: McuExternalClockOutputConfig	1209
17.3.1.35.1	McuExtClock0Enable	1209
17.3.1.35.2	McuExtClock1Enable	1210
17.3.1.35.3	McuExtClock1Inverted	1210
17.3.1.35.4	McuExtClockOutSel0	1211
17.3.1.35.5	McuExtClockOutSel1	1211
17.3.1.35.6	McuFoutClockDiv	1212
17.3.1.36	Container: McuGeneralConfiguration	1213
17.3.1.36.1	McuCCU61SleepModeEnabled	1213
17.3.1.36.2	McuCcu60SleepModeEnabled	1214
17.3.1.36.3	McuClearColdResetStatusApi	1214
17.3.1.36.4	McuClockSourceFailureNotification	1215
17.3.1.36.5	McuDevErrorDetect	1215
17.3.1.36.6	McuGetRamStateApi	1216
17.3.1.36.7	McuGpt12SleepModeEnabled	1216
17.3.1.36.8	McuGtmSleepModeEnabled	1217
17.3.1.36.9	McudleModeCpuCore	1217
17.3.1.36.10	McufxCpuCcuconApi	1218
17.3.1.36.11	McufxDeInitApi	1218
17.3.1.36.12	McufxLpmApi	1219
17.3.1.36.13	McufxTrapApi	1220
17.3.1.36.14	McuInitCheckApi	1220
17.3.1.36.15	McuInitClock	1221
17.3.1.36.16	McuInitDeInitApiMode	1221
17.3.1.36.17	McuMainOscillatorFrequency	1222
17.3.1.36.18	McuMultiCoreErrorDetect	1222
17.3.1.36.19	McuNoPll	1223
17.3.1.36.20	McuOscAmpRegulationEnable	1223
17.3.1.36.21	McuOscCapacitance0Enable	1224
17.3.1.36.22	McuOscCapacitance1Enable	1224
17.3.1.36.23	McuOscCapacitance2Enable	1225
17.3.1.36.24	McuOscCapacitance3Enable	1226
17.3.1.36.25	McuOscillatorMode	1226
17.3.1.36.26	McuPerformResetApi	1227
17.3.1.36.27	McuRuntimeApiMode	1227
17.3.1.36.28	McuSafetyEnable	1228
17.3.1.36.29	McuStandbyEntryMode	1228
17.3.1.36.30	McuSystemModeCpuCore	1229
17.3.1.36.31	McuVersionInfoApi	1229

Table of contents

17.3.1.36.32	McuSysClkFrequency	1230
17.3.1.37	Container: GtmTomGlobalConf	1230
17.3.1.38	Container: McuGpt12ModuleAllocationConf	1230
17.3.1.38.1	McuGpt12ModuleAllocationConf	1231
17.3.1.38.2	McuGpt12TimerAllocation	1231
17.3.1.39	Container: McuGtmAllocationConf	1232
17.3.1.40	Container: McuGtmAtomAllocationConf	1232
17.3.1.41	Container: McuGtmAtomChannelAllocationConf	1232
17.3.1.41.1	McuAtomChannelEventHandledByDsadc	1232
17.3.1.41.2	McuGtmAtomChannelAllocationConf	1233
17.3.1.42	Container: McuGtmClockManagementConf	1233
17.3.1.42.1	GtmCmuGlobalClockDenominator	1233
17.3.1.42.2	GtmCmuGlobalClockNumerator	1234
17.3.1.43	Container: McuGtmTimAllocationConf	1234
17.3.1.44	Container: McuGtmTimChannelAllocationConf	1234
17.3.1.44.1	McuGtmTimChannelAllocationConf	1235
17.3.1.45	Container: McuGtmTomAllocationConf	1235
17.3.1.46	Container: McuGtmTomChannelAllocationConf	1235
17.3.1.46.1	McuGtmTomChannelAllocationConf	1235
17.3.1.46.2	McuTomChannelEventHandledByDsadc	1236
17.3.1.47	Container: McuHardwareResourceAllocationConf	1236
17.3.1.48	Container: McuModeSettingConf	1237
17.3.1.48.1	McuEvrcLPMOnSleepReqEnable	1237
17.3.1.48.2	McuMode	1237
17.3.1.49	Container: McuModuleConfiguration	1238
17.3.1.49.1	McuClockSrcFailureNotification	1238
17.3.1.49.2	McuNumberOfMcuModes	1238
17.3.1.49.3	McuRamSectors	1239
17.3.1.49.4	McuResetSetting	1239
17.3.1.50	Container: McuPeripheralPllSettingConfig	1240
17.3.1.50.1	McuClockReferencePointFrequency1	1240
17.3.1.50.2	McuClockReferencePointFrequency2	1241
17.3.1.50.3	McuFreqSource1ClockDivSelect	1241
17.3.1.50.4	McuPerPllK2DivStepDownChangeDelay	1242
17.3.1.50.5	McuPerPllK2DivStepUpChangeDelay	1242
17.3.1.50.6	McuPerPllK3DivStepDownChangeDelay	1243
17.3.1.50.7	McuPerPllK3DivStepUpChangeDelay	1243
17.3.1.50.8	McuPeripheralPllK2Divider	1244
17.3.1.50.9	McuPeripheralPllK3Divider	1244
17.3.1.50.10	McuPeripheralPllNDivider	1245
17.3.1.50.11	McuPeripheralPllPDivide	1245
17.3.1.50.12	McuPll2DivSelect	1246

Table of contents

17.3.1.51	Container: McuPlldistributionSettingConfig	1246
17.3.1.51.1	McuAdasFrequency	1246
17.3.1.51.2	McuAdcFrequency	1247
17.3.1.51.3	McuAscLinFastFrequency	1248
17.3.1.51.4	McuAscLinSlowClockSourceSelection	1248
17.3.1.51.5	McuAscLinSlowFrequency	1249
17.3.1.51.6	McuBBBFrequency	1249
17.3.1.51.7	McuCPU0Frequency	1250
17.3.1.51.8	McuCPU1Frequency	1250
17.3.1.51.9	McuCPU2Frequency	1251
17.3.1.51.10	McuCPU3Frequency	1251
17.3.1.51.11	McuCPU4Frequency	1252
17.3.1.51.12	McuCPU5Frequency	1252
17.3.1.51.13	McuClockDistributionInpClockSel	1253
17.3.1.51.14	McuConvCtrlPhaseSynchConf	1254
17.3.1.51.15	McuEbuClkEnable	1255
17.3.1.51.16	McuEbuFrequency	1255
17.3.1.51.17	McuErayClkEnable	1256
17.3.1.51.18	McuErayFrequency	1256
17.3.1.51.19	McuFSI2Frequency	1257
17.3.1.51.20	McuFSIFrequency	1257
17.3.1.51.21	McuGEthFrequency	1258
17.3.1.51.22	McuGTMFrequency	1258
17.3.1.51.23	McuHsctFrequency	1259
17.3.1.51.24	McuHspd160Frequency	1260
17.3.1.51.25	McuHspd320Frequency	1260
17.3.1.51.26	McuHspdClkEnable	1261
17.3.1.51.27	McuI2CFrequency	1261
17.3.1.51.28	McuLowPowerDivValue	1262
17.3.1.51.29	McuMCanClockSourceSelection	1262
17.3.1.51.30	McuMCanFrequency	1263
17.3.1.51.31	McuMcanHFrequency	1264
17.3.1.51.32	McuMscClockSourceSelection	1264
17.3.1.51.33	McuMscFrequency	1265
17.3.1.51.34	McuQspiClockSourceSelection	1265
17.3.1.51.35	McuQspiFrequency	1266
17.3.1.51.36	McuReferenceFrequency1	1266
17.3.1.51.37	McuReferenceFrequency2	1267
17.3.1.51.38	McuSPBFrequency	1267
17.3.1.51.39	McuSRIFrequency	1268
17.3.1.51.40	McuSTMFrequency	1268
17.3.1.52	Container: McuPublishedInformation	1269

Table of contents

17.3.1.53	Container: McuRamSectorSettingConf	1269
17.3.1.53.1	McuRamDefaultValue	1269
17.3.1.53.2	McuRamSectionBaseAddress	1270
17.3.1.53.3	McuRamSectionSize	1270
17.3.1.53.4	McuRamSectorSettingId	1271
17.3.1.54	Container: McuResetReasonConf	1271
17.3.1.54.1	McuResetReason	1271
17.3.1.55	Container: McuStdByModeESR0Conf	1272
17.3.1.55.1	McuStdbyModeESR0EdgeDetection	1272
17.3.1.55.2	McuStdbyModeESR0FltEnable	1272
17.3.1.55.3	McuStdbyModeESR0WakeupEnable	1273
17.3.1.56	Container: McuStdByModeESR1Conf	1274
17.3.1.56.1	McuStdbyModeESR1EdgeDetection	1274
17.3.1.56.2	McuStdbyModeESR1FltEnable	1274
17.3.1.56.3	McuStdbyModeESR1WakeupEnable	1275
17.3.1.57	Container: McuStdByModePinAConf	1275
17.3.1.57.1	McuStdbyModePinAEdgeDetection	1276
17.3.1.57.2	McuStdbyModePinAFltEnable	1276
17.3.1.57.3	McuStdbyModePinAWakeupEnable	1277
17.3.1.58	Container: McuStdByModePinBConf	1277
17.3.1.58.1	McuStdbyModePinBEdgeDetection	1278
17.3.1.58.2	McuStdbyModePinBFltEnable	1278
17.3.1.58.3	McuStdbyModePinBWakeupEnable	1279
17.3.1.59	Container: McuStdByModeWakeUpTimerConf	1279
17.3.1.59.1	McuStdbyModeWakeUpTimerClkDiv	1279
17.3.1.59.2	McuStdbyModeWakeUpTimerEnable	1280
17.3.1.59.3	McuStdbyModeWakeUpTimerMode	1281
17.3.1.59.4	McuStdbyModeWakeUpTimerValue	1281
17.3.1.60	Container: McuStdbyModeSettingConf	1282
17.3.1.60.1	McuStdbyModeBlankingFilterDelay	1282
17.3.1.60.2	McuStdbyModeClkSelection	1283
17.3.1.60.3	McuStdbyModeESR0TriStateEnable	1283
17.3.1.60.4	McuStdbyModePORSTFilterEnable	1284
17.3.1.60.5	McuStdbyModePortTriStateEnable	1284
17.3.1.60.6	McuStdbyModeRamEnable	1285
17.3.1.60.7	McuStdbyModeWakeUpFromEVR	1286
17.3.1.60.8	McuStdbyModeWakeUpFromPORST	1286
17.3.1.60.9	McuStdbyModeWakeUpFromSCR	1287
17.3.1.61	Container: McuSystemPllSettingConfig	1287
17.3.1.61.1	McuClockReferencePointFrequency0	1287
17.3.1.61.2	McuFMPllModAmp	1288
17.3.1.61.3	McuFmPllEnable	1289

Table of contents

17.3.1.61.4	McuPllInputSrcSelection	1289
17.3.1.61.5	McuSysPllK2DivStepDownChangeDelay	1290
17.3.1.61.6	McuSysPllK2DivStepUpChangeDelay	1290
17.3.1.61.7	McuSystemPllK2Divider	1291
17.3.1.61.8	McuSystemPllNDivider	1291
17.3.1.61.9	McuSystemPllPDivider	1292
17.3.1.62	Container: McuResetSettingConf	1292
17.3.1.62.1	McuESR0ResetConf	1292
17.3.1.62.2	McuESR1ResetConf	1293
17.3.1.62.3	McuSMUResetConf	1293
17.3.1.62.4	McuSTM0ResetConf	1294
17.3.1.62.5	McuSTM0ResetOnApplResetEnable	1294
17.3.1.62.6	McuSTM1ResetConf	1295
17.3.1.62.7	McuSTM1ResetOnApplResetEnable	1295
17.3.1.62.8	McuSTM2ResetConf	1296
17.3.1.62.9	McuSTM2ResetOnApplResetEnable	1297
17.3.1.62.10	McuSTM3ResetConf	1297
17.3.1.62.11	McuSTM3ResetOnApplResetEnable	1298
17.3.1.62.12	McuSTM4ResetConf	1298
17.3.1.62.13	McuSTM4ResetOnApplResetEnable	1299
17.3.1.62.14	McuSTM5ResetConf	1299
17.3.1.62.15	McuSTM5ResetOnApplResetEnable	1300
17.3.1.62.16	McuSWResetConf	1300
17.3.1.63	Container: McuTrapSettingConf	1301
17.3.1.63.1	McuCPU0ESR0TrapEnable	1301
17.3.1.63.2	McuCPU0ESR1TrapEnable	1301
17.3.1.63.3	McuCPU0SMUTrapEnable	1302
17.3.1.63.4	McuCPU0Trap2Enable	1302
17.3.1.63.5	McuCPU1ESR0TrapEnable	1303
17.3.1.63.6	McuCPU1ESR1TrapEnable	1303
17.3.1.63.7	McuCPU1SMUTrapEnable	1304
17.3.1.63.8	McuCPU1Trap2Enable	1304
17.3.1.63.9	McuCPU2ESR0TrapEnable	1305
17.3.1.63.10	McuCPU2ESR1TrapEnable	1306
17.3.1.63.11	McuCPU2SMUTrapEnable	1306
17.3.1.63.12	McuCPU2Trap2Enable	1307
17.3.1.63.13	McuCPU3ESR0TrapEnable	1307
17.3.1.63.14	McuCPU3ESR1TrapEnable	1308
17.3.1.63.15	McuCPU3SMUTrapEnable	1308
17.3.1.63.16	McuCPU3Trap2Enable	1309
17.3.1.63.17	McuCPU4ESR0TrapEnable	1309
17.3.1.63.18	McuCPU4ESR1TrapEnable	1310

Table of contents

17.3.1.63.19	McuCPU4SMUTrapEnable	1311
17.3.1.63.20	McuCPU4Trap2Enable	1311
17.3.1.63.21	McuCPU5ESR0TrapEnable	1312
17.3.1.63.22	McuCPU5ESR1TrapEnable	1312
17.3.1.63.23	McuCPU5SMUTrapEnable	1313
17.3.1.63.24	McuCPU5Trap2Enable	1313
17.3.2	Functions - Type definitions	1314
17.3.2.1	Mcu_17_Ccu6_TimerChIntType	1314
17.3.2.2	Mcu_17_Eru_SrcIdentifierType	1314
17.3.2.3	Mcu_17_Gpt12_ClkPrescalarType	1315
17.3.2.4	Mcu_17_Gpt12_TimerBlockType	1315
17.3.2.5	Mcu_17_Gtm_AtomCh	1315
17.3.2.6	Mcu_17_Gtm_AtomChArray	1316
17.3.2.7	Mcu_17_Gtm_MappedPortTimerOutType	1316
17.3.2.8	Mcu_17_Gtm_TimCh	1317
17.3.2.9	Mcu_17_Gtm_TimChArray	1317
17.3.2.10	Mcu_17_Gtm_TimerEnableType	1317
17.3.2.11	Mcu_17_Gtm_TimerEnTriggerType	1317
17.3.2.12	Mcu_17_Gtm_TimerOutputEnableType	1318
17.3.2.13	Mcu_17_Gtm_TimerOutputEnTriggerType	1318
17.3.2.14	Mcu_17_Gtm_TimerUpdateEnableType	1318
17.3.2.15	Mcu_17_Gtm_TomCh	1319
17.3.2.16	Mcu_17_Gtm_TomChArray	1319
17.3.2.17	Mcu_17_Gtm_TomTgc	1319
17.3.2.18	Mcu_17_Gtm_TomTgcArray	1320
17.3.2.19	Mcu_17_Stm_ComIntEnableType	1320
17.3.2.20	Mcu_17_Stm_StmCmpIdentifierType	1321
17.3.2.21	Mcu_17_Stm_StmIdentifierType	1321
17.3.2.22	Mcu_17_Stm_TimerConfigType	1321
17.3.2.23	Mcu_17_Timer_CallbackFuncPtrType	1321
17.3.2.24	Mcu_RamStateType	1322
17.3.2.25	Mcu_CpuldType	1322
17.3.2.26	Mcu_CpuModeType	1323
17.3.2.27	Mcu_TrapRequestType	1323
17.3.2.28	Mcu_ConfigType	1323
17.3.2.29	Mcu_PlStatusType	1324
17.3.2.30	Mcu_ClockType	1324
17.3.2.31	Mcu_ResetType	1324
17.3.2.32	Mcu_RawResetType	1325
17.3.2.33	Mcu_RamSectionType	1326
17.3.2.34	Mcu_ModeType	1326
17.3.2.35	Mcu_17_Gtm_TimChConfigType	1326

Table of contents

17.3.2.36	Mcu_17_Gtm_TimerChIdentifierType	1327
17.3.2.37	Mcu_17_Gtm_TimerOutType	1327
17.3.2.38	Mcu_17_Gtm_TomAtomChConfigType	1328
17.3.2.39	Mcu_17_Gtm_TimerStatusType	1328
17.3.2.40	Mcu_17_Ccu6_ComparatorType	1329
17.3.2.41	Mcu_17_Ccu6_KernelIdentifierType	1329
17.3.2.42	Mcu_17_Ccu6_TimerChIdentifierType	1329
17.3.2.43	Mcu_17_Ccu6_TimerConfigType	1330
17.3.2.44	Mcu_17_Ccu6_TimerType	1331
17.3.2.45	Mcu_17_Gpt12_TimerChIdentifierType	1331
17.3.2.46	Mcu_17_Gpt12_TimerConfigType	1331
17.3.3	Functions - APIs	1332
17.3.3.1	Mcu_GetRamState	1332
17.3.3.2	Mcu_Init	1333
17.3.3.3	Mcu_InitRamSection	1334
17.3.3.4	Mcu_InitClock	1335
17.3.3.5	Mcu_DistributePllClock	1336
17.3.3.6	Mcu_GetPllStatus	1337
17.3.3.7	Mcu_GetResetReason	1338
17.3.3.8	Mcu_GetResetRawValue	1338
17.3.3.9	Mcu_PerformReset	1339
17.3.3.10	Mcu_SetMode	1340
17.3.3.11	Mcu_GetVersionInfo	1341
17.3.3.12	Mcu_ClearColdResetStatus	1342
17.3.3.13	Mcu_DeInit	1343
17.3.3.14	Mcu_GetCpudleModelInitiator	1344
17.3.3.15	Mcu_GetCpuState	1345
17.3.3.16	Mcu_GetWakeupCause	1346
17.3.3.17	Mcu_ClearWakeupCause	1346
17.3.3.18	Mcu_GetTrapCause	1347
17.3.3.19	Mcu_SetTrapRequest	1348
17.3.3.20	Mcu_ClearTrapRequest	1349
17.3.3.21	Mcu_UpdateCpuCcuconReg	1350
17.3.3.22	Mcu_InitCheck	1351
17.3.3.23	Mcu_17_Gtm_AtomChannelInit	1351
17.3.3.24	Mcu_17_Gtm_AtomChInitCheck	1352
17.3.3.25	Mcu_17_Gtm_AtomChannelDeInit	1353
17.3.3.26	Mcu_17_Gtm_AtomChannelEnable	1354
17.3.3.27	Mcu_17_Gtm_AtomChannelDisable	1355
17.3.3.28	Mcu_17_Gtm_IsAtomChannelEnabled	1355
17.3.3.29	Mcu_17_Gtm_AtomChannelShadowTransfer	1356
17.3.3.30	Mcu_17_Gtm_AtomChUpdateEnDis	1357

Table of contents

17.3.3.31	Mcu_17_Gtm_AtomChEndisStatUpdate	1358
17.3.3.32	Mcu_17_Gtm_AtomChEndisCtrlUpdate	1359
17.3.3.33	Mcu_17_Gtm_AtomChOutEnStatUpdate	1360
17.3.3.34	Mcu_17_Gtm_AtomChOutEnCtrlUpdate	1360
17.3.3.35	Mcu_17_Gtm_AtomTriggerRequest	1361
17.3.3.36	Mcu_17_Gtm_TomChannelInit	1362
17.3.3.37	Mcu_17_Gtm_TomChInitCheck	1363
17.3.3.38	Mcu_17_Gtm_TomChannelDeInit	1364
17.3.3.39	Mcu_17_Gtm_TomChannelEnable	1364
17.3.3.40	Mcu_17_Gtm_TomChannelDisable	1365
17.3.3.41	Mcu_17_Gtm_IsTomChannelEnabled	1366
17.3.3.42	Mcu_17_Gtm_TomChannelShadowTransfer	1367
17.3.3.43	Mcu_17_Gtm_TomChUpdateEnDis	1368
17.3.3.44	Mcu_17_Gtm_TomChOutEnStatUpdate	1368
17.3.3.45	Mcu_17_Gtm_TomChOutEnCtrlUpdate	1369
17.3.3.46	Mcu_17_Gtm_TomChEndisStatUpdate	1370
17.3.3.47	Mcu_17_Gtm_TomChEndisCtrlUpdate	1371
17.3.3.48	Mcu_17_Gtm_TomTriggerRequest	1372
17.3.3.49	Mcu_17_Gtm_TimChannelInit	1373
17.3.3.50	Mcu_17_Gtm_TimChInitCheck	1373
17.3.3.51	Mcu_17_Gtm_TimChannelDeInit	1374
17.3.3.52	Mcu_17_Gtm_TimChannelEnable	1375
17.3.3.53	Mcu_17_Gtm_TimChannelDisable	1376
17.3.3.54	Mcu_17_Gtm_IsTimChannelEnabled	1376
17.3.3.55	Mcu_17_Gtm_ConnectPortPinToTim	1377
17.3.3.56	Mcu_17_Gtm_ConnectTimerOutToPortPin	1378
17.3.3.57	Mcu_17_Ccu6_TimerInit	1379
17.3.3.58	Mcu_17_Ccu6_TimerInitCheck	1380
17.3.3.59	Mcu_17_Ccu6_TimerDeInit	1381
17.3.3.60	Mcu_17_Ccu6_TimerStart	1381
17.3.3.61	Mcu_17_Ccu6_TimerStop	1382
17.3.3.62	Mcu_17_Ccu6_TimerIntEnDis	1383
17.3.3.63	Mcu_17_Ccu6_TimerShadowTransfer	1384
17.3.3.64	Mcu_17_Gpt12_TimerInit	1384
17.3.3.65	Mcu_17_Gpt12_TimerInitCheck	1385
17.3.3.66	Mcu_17_Gpt12_TimerDeInit	1386
17.3.3.67	Mcu_17_Gpt12_TimerStart	1387
17.3.3.68	Mcu_17_Gpt12_TimerStop	1387
17.3.3.69	Mcu_17_Stm_SetupComparator	1388
17.3.3.70	Mcu_17_Stm_CheckComparator	1389
17.3.3.71	Mcu_17_Stm_ComparatorIntDisable	1390
17.3.4	Notifications and callbacks	1390

Table of contents

17.3.4.1	Mcu_ClockFailureNotification	1391
17.3.5	Scheduled functions	1391
17.3.6	Interrupt service routines	1392
17.3.6.1	Mcu_17_Ccu6_ChannelIsr	1392
17.3.6.2	Mcu_17_Eru_GatingIsr	1393
17.3.6.3	Mcu_17_Gpt12_ChannelIsr	1393
17.3.6.4	Mcu_17_Gtm_AtomChannelIsr	1394
17.3.6.5	Mcu_17_Gtm_TimChannelIsr	1395
17.3.6.6	Mcu_17_Gtm_TomChannelIsr	1396
17.3.6.7	Mcu_17_Stm_CompareMatchIsr	1397
17.3.7	Error codes classification	1398
17.3.7.1	Development errors	1398
17.3.7.2	Production errors	1399
17.3.7.3	Safety errors	1401
17.3.7.4	Runtime errors	1401
17.3.8	Deviations and limitations	1401
17.3.8.1	Deviations	1401
17.3.8.2	Limitations	1402
17.3.9	Unsupported hardware features	1402
18	OCU driver	1403
18.1	User information	1403
18.1.1	Description	1403
18.1.2	Hardware-software mapping	1403
18.1.2.1	GTM: primary hardware peripheral	1404
18.1.2.2	SCU: dependent hardware peripheral	1404
18.1.2.3	Port: dependent hardware peripheral	1405
18.1.3	File structure	1405
18.1.3.1	C file structure	1405
18.1.3.2	Code generator plugin files	1407
18.1.4	Integration hints	1408
18.1.4.1	Integration with AUTOSAR stack	1408
18.1.4.2	Multicore and Resource Manager	1413
18.1.4.3	MCU support	1413
18.1.4.4	Port support	1413
18.1.4.5	DMA support	1414
18.1.4.6	Interrupt connections	1414
18.1.4.7	Example usage	1415
18.1.5	Key architectural considerations	1418
18.1.5.1	DMA support	1418
18.1.5.2	Default pin action	1418
18.1.5.3	Default pin level	1419

Table of contents

18.1.5.4	Threshold values at Start channel	1419
18.2	Assumptions of Use (AoUs)	1420
18.3	Reference information	1422
18.3.1	Configuration interfaces	1422
18.3.1.1	Container: GtmTimerOutputModuleConfiguration	1425
18.3.1.1.1	GtmTimerClockSelect	1425
18.3.1.1.2	GtmTimerPortPinSelect	1425
18.3.1.1.3	GtmTimerUsed	1426
18.3.1.2	Container: Ocu	1427
18.3.1.2.1	Config Variant	1427
18.3.1.3	Container: OcuChannel	1427
18.3.1.3.1	OcuAssignedHardwareChannel	1427
18.3.1.3.2	OcuChannelId	1428
18.3.1.3.3	OcuChannelTickDuration	1428
18.3.1.3.4	OcuDefaultThreshold	1429
18.3.1.3.5	OcuHardwareTriggeredAdc	1429
18.3.1.3.6	OcuHardwareTriggeredDMA	1430
18.3.1.3.7	OcuMaxCounterValue	1430
18.3.1.3.8	OcuNotification	1431
18.3.1.3.9	OcuOutputPinUsed	1431
18.3.1.3.10	OcuOutputPinDefaultState	1432
18.3.1.4	Container: OcuConfigSet	1433
18.3.1.4.1	OcuCountdirection	1433
18.3.1.5	Container: OcuConfigurationOfOptionalApis	1433
18.3.1.5.1	OcuDeInitApi	1433
18.3.1.5.2	OcuGetCounterApi	1434
18.3.1.5.3	OcuInitCheckApi	1434
18.3.1.5.4	OcuNotificationSupported	1435
18.3.1.5.5	OcuSetAbsoluteThresholdApi	1435
18.3.1.5.6	OcuSetPinActionApi	1436
18.3.1.5.7	OcuSetPinStateApi	1436
18.3.1.5.8	OcuSetRelativeThresholdApi	1437
18.3.1.5.9	OcuVersionInfoApi	1437
18.3.1.6	Container: OcuGeneral	1438
18.3.1.6.1	OcuDevErrorDetect	1438
18.3.1.6.2	OcuMultiCoreErrorDetect	1438
18.3.1.6.3	OcuSafetyEnable	1439
18.3.1.7	Container: OcuGroup	1439
18.3.1.7.1	OcuGroupDefinition	1439
18.3.1.7.2	OcuGroupId	1440
18.3.1.8	Container: OcuHWSpecificSettings	1440
18.3.1.8.1	OcuClockSource	1440

Table of contents

18.3.1.8.2	OcuPrescale	1441
18.3.1.9	Container: CommonPublishedInformation	1441
18.3.1.9.1	ArMajorVersion	1442
18.3.1.9.2	ArMinorVersion	1442
18.3.1.9.3	ArPatchVersion	1442
18.3.1.9.4	ModuleId	1443
18.3.1.9.5	Release	1443
18.3.1.9.6	SwMajorVersion	1444
18.3.1.9.7	SwMinorVersion	1444
18.3.1.9.8	SwPatchVersion	1445
18.3.1.9.9	Vendor ID	1445
18.3.2	Functions - Type definitions	1445
18.3.2.1	Ocu_ChannelType	1445
18.3.2.2	Ocu_ValueType	1446
18.3.2.3	Ocu_PinStateType	1446
18.3.2.4	Ocu_PinActionType	1446
18.3.2.5	Ocu_ConfigType	1447
18.3.2.6	Ocu_ReturnType	1447
18.3.2.7	Ocu_NotifiPtrType	1448
18.3.3	Functions - APIs	1448
18.3.3.1	Ocu_Init	1448
18.3.3.2	Ocu_Delnit	1449
18.3.3.3	Ocu_EnableNotification	1450
18.3.3.4	Ocu_DisableNotification	1451
18.3.3.5	Ocu_GetCounter	1452
18.3.3.6	Ocu_SetAbsoluteThreshold	1453
18.3.3.7	Ocu_SetPinAction	1454
18.3.3.8	Ocu_SetPinState	1455
18.3.3.9	Ocu_SetRelativeThreshold	1456
18.3.3.10	Ocu_StartChannel	1457
18.3.3.11	Ocu_StopChannel	1458
18.3.3.12	Ocu_InitCheck	1459
18.3.3.13	Ocu_GetVersionInfo	1460
18.3.4	Notifications and Callbacks	1461
18.3.4.1	Ocu_Timer_Isr	1461
18.3.5	Scheduled functions	1462
18.3.6	Interrupt service routines	1462
18.3.7	Error codes classification	1462
18.3.7.1	Development errors	1462
18.3.7.2	Production errors	1464
18.3.7.3	Safety errors	1465
18.3.7.4	Runtime errors	1465

Table of contents

18.3.8	Deviations and limitations	1466
18.3.8.1	Deviations	1466
18.3.8.2	Limitations	1466
18.3.9	Unsupported hardware features	1466
19	PORT driver	1467
19.1	User information	1467
19.1.1	Description	1467
19.1.2	Hardware-software mapping	1467
19.1.2.1	Port: primary hardware peripheral	1468
19.1.2.2	SCU: dependent hardware peripheral	1469
19.1.3	File structure	1470
19.1.3.1	C file structure	1470
19.1.3.2	Code generator plugin file	1471
19.1.4	Integration hints	1472
19.1.4.1	Integration with AUTOSAR stack	1472
19.1.4.2	Multicore and Resource Manager	1474
19.1.4.3	MCU support	1474
19.1.4.4	Port support	1474
19.1.4.5	DMA support	1474
19.1.4.6	Interrupt connections	1474
19.1.4.7	Example usage	1475
19.1.5	Key architectural considerations	1476
19.1.5.1	Pin support for Ethernet	1476
19.1.5.2	LCK bit not checked before writing to PCSR register by Port_Init API	1476
19.2	Assumptions of Use (AoUs)	1477
19.3	Reference information	1478
19.3.1	Configuration interfaces	1478
19.3.1.1	Container: CommonPublishedInformation	1478
19.3.1.1.1	ArMajorVersion	1478
19.3.1.1.2	ArMinorVersion	1479
19.3.1.1.3	ArPatchVersion	1479
19.3.1.1.4	ModuleId	1480
19.3.1.1.5	Release	1480
19.3.1.1.6	SwMajorVersion	1481
19.3.1.1.7	SwMinorVersion	1481
19.3.1.1.8	SwPatchVersion	1481
19.3.1.1.9	VendorId	1482
19.3.1.2	Container: PortConfigSet	1482
19.3.1.3	Container: PortContainer	1482
19.3.1.3.1	PortNumber	1483
19.3.1.3.2	PortNumberOfPortPins	1483

Table of contents

19.3.1.4	Container: PortGeneral	1483
19.3.1.4.1	PortDevErrorDetect	1484
19.3.1.4.2	PortInitApiMode	1484
19.3.1.4.3	PortInitCheckApi	1485
19.3.1.4.4	PortSafetyEnable	1485
19.3.1.4.5	PortSetPinDirectionApi	1486
19.3.1.4.6	PortSetPinModeApi	1486
19.3.1.4.7	PortVersionInfoApi	1487
19.3.1.5	Container: PortLVDS	1487
19.3.1.5.1	PortLVDSMode	1487
19.3.1.5.2	PortLVDSPadSupply	1488
19.3.1.5.3	PortLVDSPinPair	1488
19.3.1.5.4	PortLVDSRxEnController	1489
19.3.1.5.5	PortLVDSRxPathEnable	1490
19.3.1.5.6	PortLVDSRxTerminationMode	1490
19.3.1.5.7	PortLVDSTxEnController	1491
19.3.1.5.8	PortLVDSTxPathEnable	1491
19.3.1.5.9	PortLVDSTxPowerDownPullDown	1492
19.3.1.6	Container: PortPin	1492
19.3.1.6.1	PortPinControllerSelect	1493
19.3.1.6.2	PortPinDirection	1493
19.3.1.6.3	PortPinDirectionChangeable	1494
19.3.1.6.4	PortPinEmergencyStop	1494
19.3.1.6.5	PortPinEnableAnalogInputOnly	1495
19.3.1.6.6	PortPinId	1495
19.3.1.6.7	PortPinInitialMode	1496
19.3.1.6.8	PortPinInputPadLevel	1496
19.3.1.6.9	PortPinInputPullResistor	1497
19.3.1.6.10	PortPinLevelValue	1497
19.3.1.6.11	PortPinMode	1498
19.3.1.6.12	PortPinModeChangeable	1499
19.3.1.6.13	PortPinOutputPadDriveStrength	1499
19.3.1.6.14	PortPinOutputPinDriveMode	1500
19.3.1.6.15	PortPinSymbolicName	1501
19.3.2	Functions - Type definitions	1501
19.3.2.1	Port_ConfigType	1501
19.3.2.2	Port_PinDirectionType	1501
19.3.2.3	Port_PinModeType	1502
19.3.2.4	Port_PinType	1502
19.3.3	Functions - APIs	1502
19.3.3.1	Port_GetVersionInfo	1502
19.3.3.2	Port_Init	1503

Table of contents

19.3.3.3	Port_InitCheck	1504
19.3.3.4	Port_RefreshPortDirection	1505
19.3.3.5	Port_SetPinDirection	1506
19.3.3.6	Port_SetPinMode	1507
19.3.4	Notifications and callbacks	1507
19.3.5	Scheduled functions	1507
19.3.6	Interrupt service routines	1508
19.3.7	Error codes classification	1508
19.3.7.1	Development errors	1508
19.3.7.2	Production errors	1508
19.3.7.3	Safety errors	1508
19.3.7.4	Run time errors	1509
19.3.8	Deviations and limitations	1509
19.3.8.1	Deviations	1509
19.3.8.2	Limitations	1509
19.3.9	Unsupported hardware features	1509
20	Pwm_17_GtmCcu6 driver	1510
20.1	User information	1510
20.1.1	Description	1510
20.1.2	Hardware-software mapping	1510
20.1.2.1	GTM: primary hardware peripheral	1511
20.1.2.2	SCU: dependent hardware peripheral	1512
20.1.2.3	Port: dependent hardware peripheral	1512
20.1.2.4	CCU6: primary hardware peripheral	1513
20.1.3	File structure	1513
20.1.3.1	C file structure	1513
20.1.3.2	Code generator plugin files	1515
20.1.4	Integration hints	1517
20.1.4.1	Integration with AUTOSAR stack	1517
20.1.4.2	Multicore and Resource Manager	1520
20.1.4.3	MCU support	1520
20.1.4.4	Port support	1521
20.1.4.5	DMA support	1521
20.1.4.6	Interrupt connections	1521
20.1.4.7	Example usage	1523
20.1.5	Key architectural considerations	1524
20.1.5.1	User mode support	1524
20.2	Assumptions of Use (AoUs)	1525
20.3	Reference information	1526
20.3.1	Configuration interfaces	1526
20.3.1.1	Container: CCU6CC6Configuration	1526

Table of contents

20.3.1.1.1	CCU6KernelUsed	1526
20.3.1.1.2	CCU6TimerClockSelect	1527
20.3.1.1.3	CCU6TimerPrescalarEnabled	1528
20.3.1.1.4	CCU6TimerUsed	1528
20.3.1.1.5	Cc6xChannel	1529
20.3.1.2	Container: CommonPublishedInformation	1529
20.3.1.2.1	ArMajorVersion	1529
20.3.1.2.2	ArMinorVersion	1530
20.3.1.2.3	ArPatchVersion	1530
20.3.1.2.4	ModuleId	1530
20.3.1.2.5	Release	1531
20.3.1.2.6	SwMajorVersion	1531
20.3.1.2.7	SwMinorVersion	1532
20.3.1.2.8	SwPatchVersion	1532
20.3.1.2.9	VendorApiInfix	1533
20.3.1.2.10	VendorId	1533
20.3.1.3	Container: GtmTimerOutputModuleConfiguration	1533
20.3.1.3.1	GtmTimerClockSelect	1534
20.3.1.3.2	GtmTimerPortPinSelect	1534
20.3.1.3.3	GtmTimerUsed	1535
20.3.1.4	Container: Pwm	1536
20.3.1.4.1	Config Variant	1536
20.3.1.5	Container: PwmChannel	1536
20.3.1.5.1	PwmAssignedHwUnit	1536
20.3.1.5.2	PwmChannelClass	1537
20.3.1.5.3	PwmChannelId	1538
20.3.1.5.4	PwmCoherentUpdate	1538
20.3.1.5.5	PwmDutycycleDefault	1539
20.3.1.5.6	PwmIdleState	1539
20.3.1.5.7	PwmMcuClockReferencePoint	1540
20.3.1.5.8	PwmNotification	1540
20.3.1.5.9	PwmPeriodDefault	1541
20.3.1.5.10	PwmPolarity	1542
20.3.1.5.11	PwmReferenceChannel	1542
20.3.1.5.12	PwmShiftValue	1543
20.3.1.6	Container: PwmChannelConfigSet	1543
20.3.1.7	Container: PwmConfigurationOfOptApiServices	1544
20.3.1.7.1	PwmDeInitApi	1544
20.3.1.7.2	PwmGetOutputState	1544
20.3.1.7.3	PwmSetDutyCycle	1545
20.3.1.7.4	PwmSetOutputToldle	1545
20.3.1.7.5	PwmSetPeriodAndDuty	1546

Table of contents

20.3.1.7.6	PwmVersionInfoApi	1547
20.3.1.8	Container: PwmGeneral	1547
20.3.1.8.1	PwmChannelCoherentSelection	1547
20.3.1.8.2	PwmDevErrorDetect	1548
20.3.1.8.3	PwmDutyShiftInTicks	1548
20.3.1.8.4	PwmDutycycleUpdatedEndperiod	1549
20.3.1.8.5	PwmEnable0Or100DutyNotification	1550
20.3.1.8.6	PwmHandleShiftByOffset	1550
20.3.1.8.7	PwmIndex	1551
20.3.1.8.8	PwmInitCheckApi	1551
20.3.1.8.9	PwmLowPowerStatesSupport	1552
20.3.1.8.10	PwmMultiCoreErrorDetect	1552
20.3.1.8.11	PwmNotificationSupported	1553
20.3.1.8.12	PwmPeriodUpdatedEndperiod	1553
20.3.1.8.13	PwmPowerStateAsynchTransitionMode	1554
20.3.1.8.14	PwmSafetyEnable	1554
20.3.1.9	Container: PwmPowerStateConfig	1555
20.3.1.9.1	PwmPowerState	1555
20.3.1.9.2	PwmPowerStateReadyCbkRef	1555
20.3.2	Functions - Type definitions	1556
20.3.2.1	Pwm_17_GtmCcu6_ChannelType	1556
20.3.2.2	Pwm_17_GtmCcu6_NotifiPtrType	1556
20.3.2.3	Pwm_17_GtmCcu6_PeriodType	1557
20.3.2.4	Pwm_17_GtmCcu6_OutputStateType	1557
20.3.2.5	Pwm_17_GtmCcu6_EdgeNotificationType	1557
20.3.2.6	Pwm_17_GtmCcu6_ChannelClassType	1558
20.3.2.7	Pwm_17_GtmCcu6_ConfigType	1558
20.3.3	Functions - APIs	1559
20.3.3.1	Pwm_17_GtmCcu6_DelInit	1559
20.3.3.2	Pwm_17_GtmCcu6_DisableNotification	1559
20.3.3.3	Pwm_17_GtmCcu6_EnableNotification	1560
20.3.3.4	Pwm_17_GtmCcu6_GetOutputState	1562
20.3.3.5	Pwm_17_GtmCcu6_GetVersionInfo	1562
20.3.3.6	Pwm_17_GtmCcu6_Init	1563
20.3.3.7	Pwm_17_GtmCcu6_InitCheck	1564
20.3.3.8	Pwm_17_GtmCcu6_SetDutyCycle	1565
20.3.3.9	Pwm_17_GtmCcu6_SetOutputTidle	1566
20.3.3.10	Pwm_17_GtmCcu6_SetPeriodAndDuty	1567
20.3.4	Notifications and Callbacks	1569
20.3.4.1	Pwm_17_GtmCcu6_Isr	1569
20.3.5	Scheduled functions	1570
20.3.6	Interrupt service routines	1570

Table of contents

20.3.7	Error codes classification	1570
20.3.7.1	Development errors	1570
20.3.7.2	Production errors	1572
20.3.7.3	Safety errors	1572
20.3.7.4	Runtime errors	1573
20.3.8	Deviations and limitations	1573
20.3.8.1	Deviations	1573
20.3.8.2	Limitations	1573
20.3.9	Unsupported hardware features	1580
21	SPI driver	1581
21.1	User information	1581
21.1.1	Description	1581
21.1.2	Hardware-software mapping	1581
21.1.2.1	QSPI: primary hardware peripheral	1582
21.1.2.2	SCU: dependent hardware peripheral	1583
21.1.2.3	SRC: dependent hardware peripheral	1583
21.1.2.4	Port: dependent hardware peripheral	1584
21.1.2.5	DMA: dependent hardware peripheral	1584
21.1.3	File structure	1584
21.1.3.1	C file structure	1584
21.1.3.2	Code generator plugin files	1586
21.1.4	Integration hints	1587
21.1.4.1	Integration with AUTOSAR stack	1587
21.1.4.2	Multicore and Resource Manager	1591
21.1.4.3	MCU support	1592
21.1.4.4	Port support	1592
21.1.4.5	DMA support	1594
21.1.4.6	Interrupt connections	1596
21.1.4.7	Example usage	1600
21.1.5	Key architectural considerations	1607
21.1.5.1	Transmission modes supported	1607
21.1.5.2	General configuration	1608
21.1.5.3	Multicore decision	1608
21.1.5.4	Sequence, jobs and channels	1609
21.1.5.5	Lookup tables	1609
21.1.5.6	Interruptible sequence behavior	1609
21.1.5.7	External demultiplexer feature	1610
21.2	Assumptions of Use (AoUs)	1611
21.3	Reference information	1613
21.3.1	Configuration interfaces	1613
21.3.1.1	Container: CommonPublishedInformation	1614

Table of contents

21.3.1.1.1	ArMajorVersion	1615
21.3.1.1.2	ArMinorVersion	1615
21.3.1.1.3	ArPatchVersion	1615
21.3.1.1.4	Module ID	1616
21.3.1.1.5	Release	1616
21.3.1.1.6	SwMajorVersion	1617
21.3.1.1.7	SwMinorVersion	1617
21.3.1.1.8	SwPatchVersion	1618
21.3.1.1.9	Vendor ID	1618
21.3.1.2	Container: Spi	1618
21.3.1.2.1	Config Variant	1619
21.3.1.3	Container: SpiBaudrateParams	1619
21.3.1.3.1	SpiBaudParamA	1619
21.3.1.3.2	SpiBaudParamB	1620
21.3.1.3.3	SpiBaudParamC	1621
21.3.1.3.4	SpiBaudParamQ	1621
21.3.1.3.5	SpiBaudParamTQ	1622
21.3.1.4	Container: SpiChannel	1623
21.3.1.4.1	SpiChannelId	1623
21.3.1.4.2	SpiChannelType	1624
21.3.1.4.3	SpiDataWidth	1624
21.3.1.4.4	SpiDefaultData	1625
21.3.1.4.5	SpiEbMaxLength	1625
21.3.1.4.6	SpilbNBuffers	1626
21.3.1.4.7	SpiTransferStart	1627
21.3.1.5	Container: SpiChannelList	1627
21.3.1.5.1	SpiChannelAssignment	1627
21.3.1.5.2	SpiChannelIndex	1628
21.3.1.6	Container: SpiCsGpio	1628
21.3.1.6.1	SpiCsGpioPinSelection	1628
21.3.1.6.2	SpiCsGpioPortSelection	1629
21.3.1.7	Container: SpiDelayParams	1629
21.3.1.7.1	SpiDelayParamIdleLength	1629
21.3.1.7.2	SpiDelayParamIdlePre	1630
21.3.1.7.3	SpiDelayParamLeadLength	1631
21.3.1.7.4	SpiDelayParamLeadPre	1631
21.3.1.7.5	SpiDelayParamTrailLength	1632
21.3.1.7.6	SpiDelayParamTrailPre	1633
21.3.1.8	Container: SpiDemEventParameterRefs	1633
21.3.1.8.1	SPI_E_HARDWARE_ERROR	1633
21.3.1.9	Container: SpiDriver	1634
21.3.1.9.1	SpiMaxChannel	1634

Table of contents

21.3.1.9.2	SpiMaxJob	1634
21.3.1.9.3	SpiMaxSequence	1635
21.3.1.9.4	SpiSystemclock	1636
21.3.1.10	Container: SpiExternalDevice	1636
21.3.1.10.1	SpiAutoCalcBaudParams	1636
21.3.1.10.2	SpiAutoCalcDelayParams	1637
21.3.1.10.3	SpiBaudrate	1637
21.3.1.10.4	SpiCsIdentifier	1638
21.3.1.10.5	SpiCsPolarity	1639
21.3.1.10.6	SpiCsSelection	1639
21.3.1.10.7	SpiDataShiftEdge	1640
21.3.1.10.8	SpiEnableCs	1640
21.3.1.10.9	SpiHwUnit	1641
21.3.1.10.10	SpidleTime	1642
21.3.1.10.11	SpiInternalLoopBackSupport	1642
21.3.1.10.12	SpiParitySupport	1643
21.3.1.10.13	SpiShiftClockIdleLevel	1643
21.3.1.10.14	SpiTimeClk2Cs	1644
21.3.1.10.15	SpiTrailingTime	1645
21.3.1.11	Container: SpiGeneral	1645
21.3.1.11.1	SpiCancelApi	1645
21.3.1.11.2	SpiChannelBuffersAllowed	1646
21.3.1.11.3	SpiDevErrorDetect	1647
21.3.1.11.4	SpiHwStatusApi	1647
21.3.1.11.5	SpiInitCheckApi	1648
21.3.1.11.6	SpiInitDeInitApiMode	1648
21.3.1.11.7	SpiInterruptibleSeqAllowed	1649
21.3.1.11.8	SpiLevelDelivered	1649
21.3.1.11.9	SpiMainFunctionPeriod	1650
21.3.1.11.10	SpiMulticoreCheckEnable	1651
21.3.1.11.11	SpiRuntimeApiMode	1651
21.3.1.11.12	SpiSafetyCheckEnable	1652
21.3.1.11.13	SpiSupportConcurrentSyncTransmit	1652
21.3.1.11.14	SpiSyncTransmittimeoutDuration	1653
21.3.1.11.15	SpiUserCallbackHeaderFile	1653
21.3.1.11.16	SpiVersionInfoApi	1654
21.3.1.12	Container: SpiHwConfigurationQspix	1654
21.3.1.12.1	SpiExternalDemux	1655
21.3.1.12.2	SpiHWPinMRSTQspix	1655
21.3.1.12.3	SpiHwConfigKernel	1656
21.3.1.12.4	SpiJobQueueLengthQspix	1656
21.3.1.12.5	SpiSLSO0StrobeDelay	1657

Table of contents

21.3.1.12.6	SpiSleepEnableQspix	1658
21.3.1.13	Container: SpiHwDmaConfigurationQspix	1658
21.3.1.13.1	SpiHwDmaChannelReceptionRef	1658
21.3.1.13.2	SpiHwDmaChannelTransmissionRef	1659
21.3.1.14	Container: SpiJob	1659
21.3.1.14.1	SpiDeviceAssignment	1659
21.3.1.14.2	SpiFrameBasedCS	1660
21.3.1.14.3	SpiHwUnitSynchronous	1660
21.3.1.14.4	SpiJobEndNotification	1661
21.3.1.14.5	SpiJobId	1662
21.3.1.14.6	SpiJobPriority	1662
21.3.1.15	Container: SpiPublishedInformation	1663
21.3.1.15.1	SpiMaxHwUnit	1663
21.3.1.16	Container: SpiSequence	1663
21.3.1.16.1	SpiInterruptibleSequence	1664
21.3.1.16.2	SpiJobAssignment	1664
21.3.1.16.3	SpiSeqEndNotification	1665
21.3.1.16.4	SpiSeqenceld	1665
21.3.2	Functions - Type definitions	1666
21.3.2.1	Spi_AsyncModeType	1666
21.3.2.2	Spi_ConfigType	1667
21.3.2.3	Spi_JobEndNotification	1667
21.3.2.4	Spi_JobResultType	1667
21.3.2.5	Spi_LoopBackType	1668
21.3.2.6	Spi_SeqEndNotification	1668
21.3.2.7	Spi_SeqResultType	1668
21.3.2.8	Spi_StatusType	1669
21.3.2.9	Spi_DataBufferType	1669
21.3.2.10	Spi_NumberOfDataType	1670
21.3.2.11	Spi_ChannelType	1670
21.3.2.12	Spi_JobType	1670
21.3.2.13	Spi_SequenceType	1671
21.3.2.14	Spi_HWUnitType	1671
21.3.3	Functions - APIs	1671
21.3.3.1	Spi_AsyncTransmit	1671
21.3.3.2	Spi_Cancel	1673
21.3.3.3	Spi_ControlLoopBack	1674
21.3.3.4	Spi_DelInit	1675
21.3.3.5	Spi_GetHWUnitStatus	1675
21.3.3.6	Spi_GetJobResult	1677
21.3.3.7	Spi_GetSequenceResult	1678
21.3.3.8	Spi_GetStatus	1679

Table of contents

21.3.3.9	Spi_GetVersionInfo	1680
21.3.3.10	Spi_Init	1680
21.3.3.11	Spi_InitCheck	1681
21.3.3.12	Spi_ReadIB	1682
21.3.3.13	Spi_SetAsyncMode	1684
21.3.3.14	Spi_SetupEB	1685
21.3.3.15	Spi_SyncTransmit	1686
21.3.3.16	Spi_WritelB	1687
21.3.4	Notifications and Callbacks	1688
21.3.4.1	Spi_QspiDmaCallOut	1688
21.3.4.2	Spi_QspiDmaErrCallout	1689
21.3.5	Scheduled functions	1690
21.3.5.1	Spi_MainFunction_Handling	1690
21.3.6	Interrupt service routines	1691
21.3.6.1	Spi_IsrQspiError	1691
21.3.6.2	Spi_IsrQspiPT2	1692
21.3.7	Error codes classification	1693
21.3.7.1	Development errors	1693
21.3.7.2	Production errors	1695
21.3.7.3	Safety errors	1695
21.3.7.4	Runtime errors	1696
21.3.8	Deviations and limitations	1696
21.3.8.1	Deviations	1696
21.3.8.2	Limitations	1697
21.3.9	Unsupported hardware features	1698
22	Wdg_17_Scu driver	1699
22.1	User information	1699
22.1.1	Description	1699
22.1.2	Hardware-software mapping	1699
22.1.2.1	STM: dependent hardware peripheral	1700
22.1.2.2	GTM: dependent hardware peripheral	1701
22.1.2.3	SCU: primary hardware peripheral	1701
22.1.2.4	SMU: dependent hardware peripheral	1702
22.1.3	File structure	1702
22.1.3.1	C file structure	1702
22.1.3.2	Code generator plugin files	1704
22.1.4	Integration hints	1705
22.1.4.1	Integration with AUTOSAR stack	1705
22.1.4.2	Multicore and Resource Manager	1709
22.1.4.3	MCU support	1710
22.1.4.4	Port support	1712

Table of contents

22.1.4.5	DMA support	1712
22.1.4.6	Interrupt connections	1712
22.1.4.7	Example usage	1715
22.1.5	Key architectural considerations	1721
22.1.5.1	WDG driver states	1721
22.1.5.2	WDG driver critical section	1721
22.1.5.3	WDG driver timer configuration and services	1721
22.1.5.4	WDG driver timer dependency	1721
22.1.5.5	WDG driver multicore error reporting	1721
22.1.5.6	WDG driver password access	1721
22.1.5.7	WDG driver trigger	1722
22.1.5.8	WDG driver code generator dependency	1722
22.1.5.9	WDG driver variation point	1722
22.1.5.10	STM timer plausibility check	1722
22.2	Assumptions of Use (AoUs)	1723
22.3	Reference information	1724
22.3.1	Configuration interfaces	1724
22.3.1.1	Container: GtmTimerConfiguration	1726
22.3.1.2	Container: WdgSettingsConfig	1726
22.3.1.3	Container: CommonPublishedInformation	1726
22.3.1.3.1	ArMajorVersion	1726
22.3.1.3.2	ArMinorVersion	1726
22.3.1.3.3	ArPatchVersion	1727
22.3.1.3.4	ModuleId	1727
22.3.1.3.5	Release	1728
22.3.1.3.6	SwMajorVersion	1728
22.3.1.3.7	SwMinorVersion	1729
22.3.1.3.8	SwPatchVersion	1729
22.3.1.3.9	VendorApiInfix	1729
22.3.1.3.10	VendorId	1730
22.3.1.4	Container: GtmTimerOutputModuleConfiguration	1730
22.3.1.4.1	GtmTimerClockSelect	1731
22.3.1.4.2	GtmTimerUsed	1732
22.3.1.5	Container: Wdg	1732
22.3.1.5.1	IMPLEMENTATION_CONFIG_VARIANT	1733
22.3.1.6	Container: WdgDemEventParameterRefs	1733
22.3.1.6.1	WDG_E_DISABLE_REJECTED	1733
22.3.1.6.2	WDG_E_MODE_FAILED	1734
22.3.1.7	Container: WdgExternalConfiguration	1735
22.3.1.8	Container: WdgGeneral	1735
22.3.1.8.1	WdgDevErrorDetect	1735
22.3.1.8.2	WdgDisableAllowed	1735

Table of contents

22.3.1.8.3	WdgIndex	1736
22.3.1.8.4	WdgInitApiMode	1736
22.3.1.8.5	WdgInitCheckApi	1737
22.3.1.8.6	WdgInitialTimeout	1738
22.3.1.8.7	WdgMaxTimeout	1738
22.3.1.8.8	WdgRunArea	1739
22.3.1.8.9	WdgRuntimeApiMode	1739
22.3.1.8.10	WdgSafetyEnable	1740
22.3.1.8.11	WdgTriggerLocation	1740
22.3.1.8.12	WdgTriggerTimerSelection	1741
22.3.1.8.13	WdgVersionInfoApi	1741
22.3.1.9	Container: WdgPublishedInformation	1742
22.3.1.9.1	WdgTriggerMode	1742
22.3.1.10	Container: WdgSettingsConfig	1743
22.3.1.10.1	WdgCPUDisableAllowed	1743
22.3.1.10.2	WdgCPUInitialPassowrd	1743
22.3.1.10.3	WdgCPUInitialTimeout	1744
22.3.1.10.4	WdgCPUMaxTimeout	1744
22.3.1.10.5	WdgCoreId	1745
22.3.1.10.6	WdgDefaultMode	1746
22.3.1.10.7	WdgSystemClockRef	1746
22.3.1.11	Container: WdgSettingsFast	1747
22.3.1.11.1	WdgFastModeTimeoutValue	1747
22.3.1.12	Container: WdgSettingsOff	1748
22.3.1.13	Container: WdgSettingsSlow	1748
22.3.1.13.1	WdgSlowModeTimeoutValue	1748
22.3.1.14	Container: WdgTriggerTimerSetting	1749
22.3.1.14.1	WdgFastRefreshTime	1749
22.3.1.14.2	WdgSlowRefreshTime	1749
22.3.2	Functions - Type definitions	1750
22.3.2.1	Wdg_17_Scu_ConfigType	1750
22.3.3	Functions - APIs	1750
22.3.3.1	Wdg_17_Scu_Init	1750
22.3.3.2	Wdg_17_Scu_InitCheck	1752
22.3.3.3	Wdg_17_Scu_SetMode	1753
22.3.3.4	Wdg_17_Scu_SetTriggerCondition	1754
22.3.3.5	Wdg_17_Scu_GetVersionInfo	1755
22.3.4	Notifications and Callbacks	1755
22.3.4.1	Wdg_17_Scu_Lsr	1755
22.3.5	Scheduled functions	1757
22.3.6	Interrupt service routines	1757
22.3.7	Error codes classification	1757

Table of contents

22.3.7.1	Development errors	1757
22.3.7.2	Production errors	1757
22.3.7.3	Safety errors	1758
22.3.7.4	Runtime errors	1759
22.3.8	Deviations and limitations	1759
22.3.8.1	Deviations	1759
22.3.8.2	Limitations	1759
22.3.9	Unsupported hardware features	1759
	Revision history	1760
	Disclaimer	1763

Generic information

1 Generic information

1.1 Application integration

This section describes the typical workflow required to integrate MCAL modules with user environment.

Note: This Generic information chapter is applicable to CD, COM-E and DEMO packages as well.

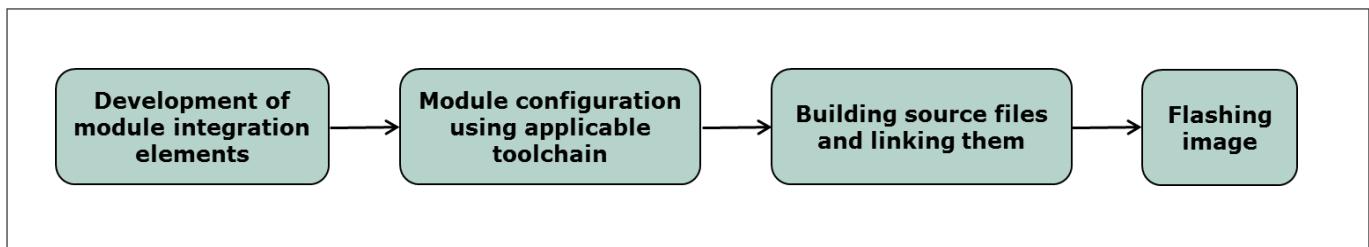


Figure 1 Application integration flow

- Being module specific, details of integration elements are provided in sections from *Integration hints* found in module chapters.
- Rest of the workflow topics being generic are described in the subsequent sections.

1.1.1 Package installation and project creation

This section describes how the modules may be installed and projects(s) created.

1.1.1.1 Installing MCAL package

Note: In this chapter, the options shown in the images are for illustration purpose only. The actual list/names should be referred in the tool.

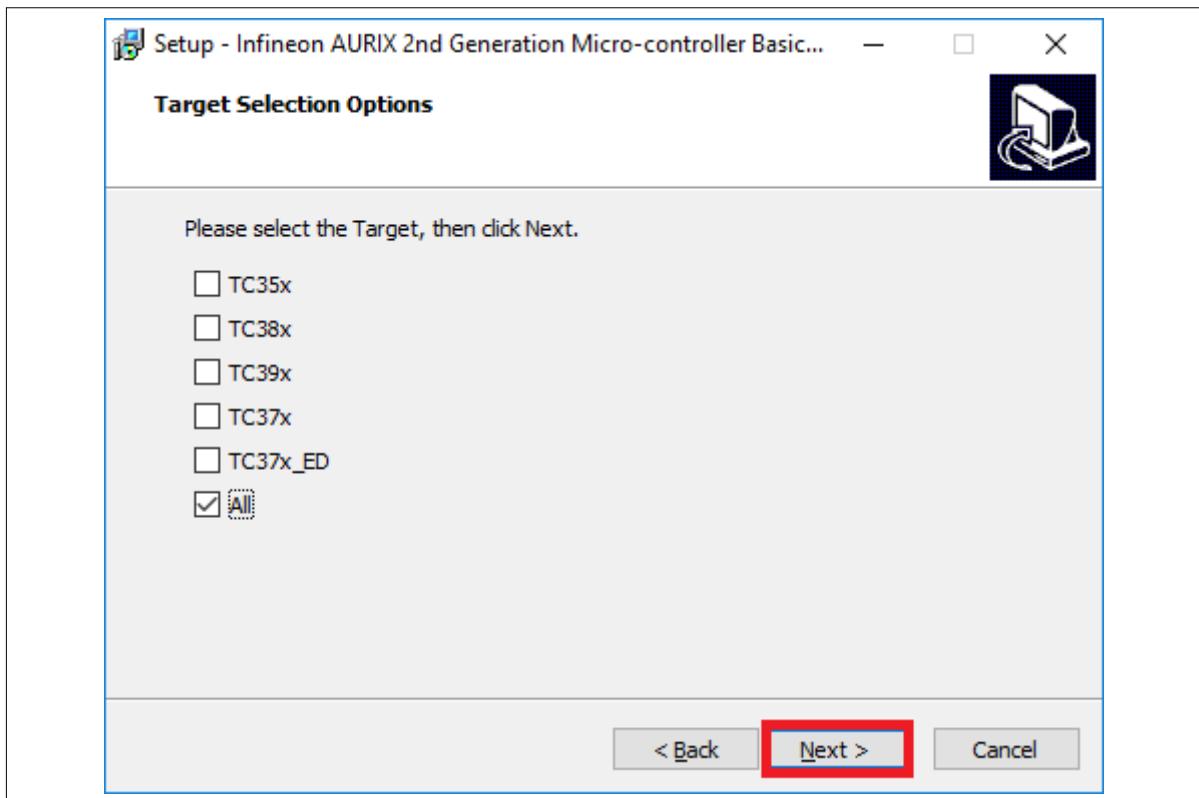
The MCAL code may be extracted from the archive to any directory. The procedure for installing the software is as follows:

Generic information**Figure 2** **Setup window**

1. Run the `.exe` file provided with the package.

Note: Similarly, you can select CD, COM-E and/or DEMO executable files from the respective package folders.

2. Click **Next**. The **Setup** window displays.

Generic information**Figure 3 Target selection options**

3. Select a target to install. Multiple devices can be selected at a time.
4. Click **Next**.

Note: *Selecting the All option installs the listed Targets.*

Note: *The list of devices shown in the above image are for sample representation, actual list may vary based on the devices supported in the package.*

Generic information

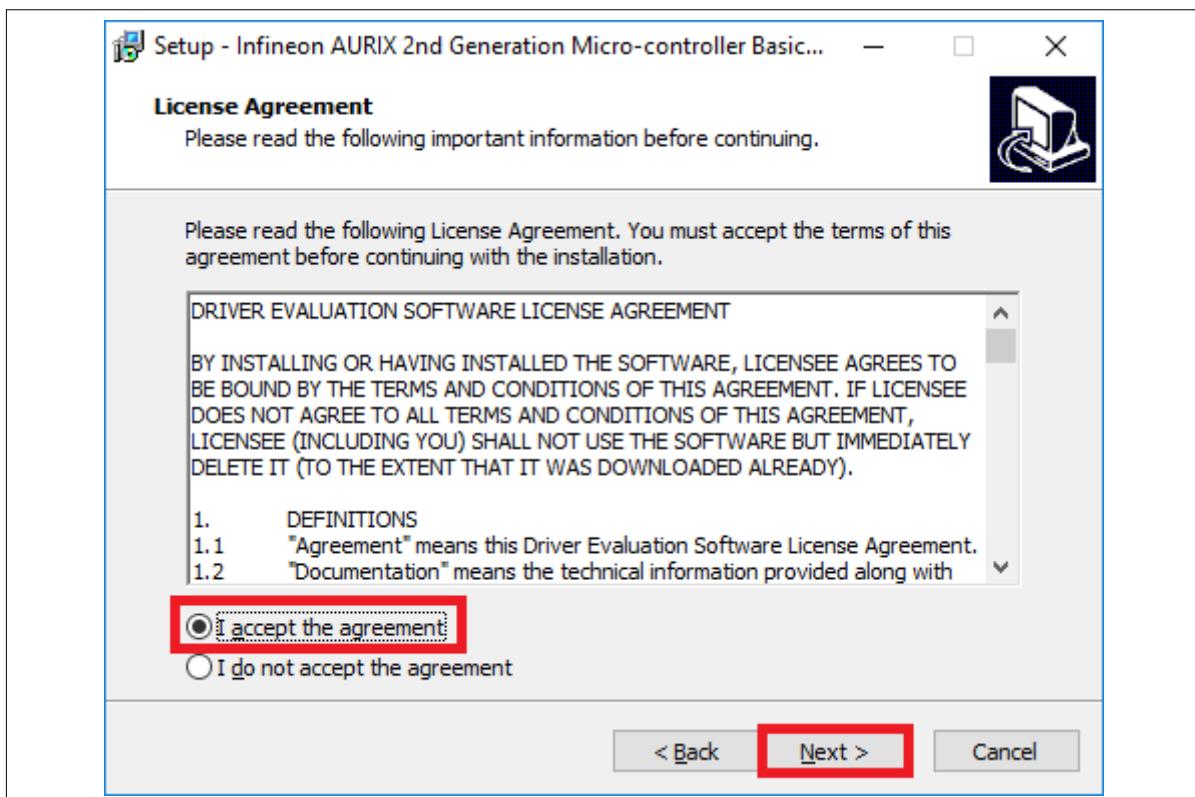


Figure 4 License agreement

5. Select the **I accept the agreement** option to accept the terms and conditions. Note that the installation will not proceed unless this option is selected.
6. Click **Next**.

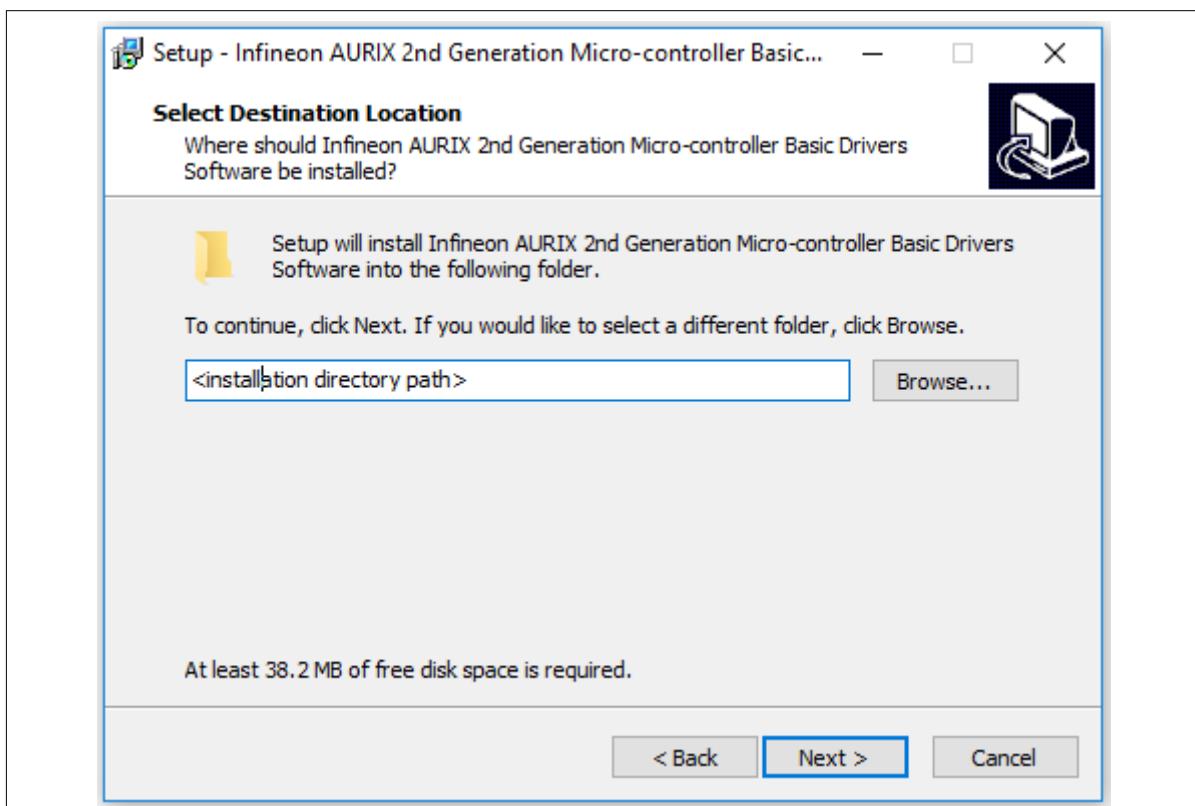


Figure 5 Select destination location

Generic information

7. Click **Browse** to navigate to a folder to install the drivers and the plug-in files.
8. Click **Next**.

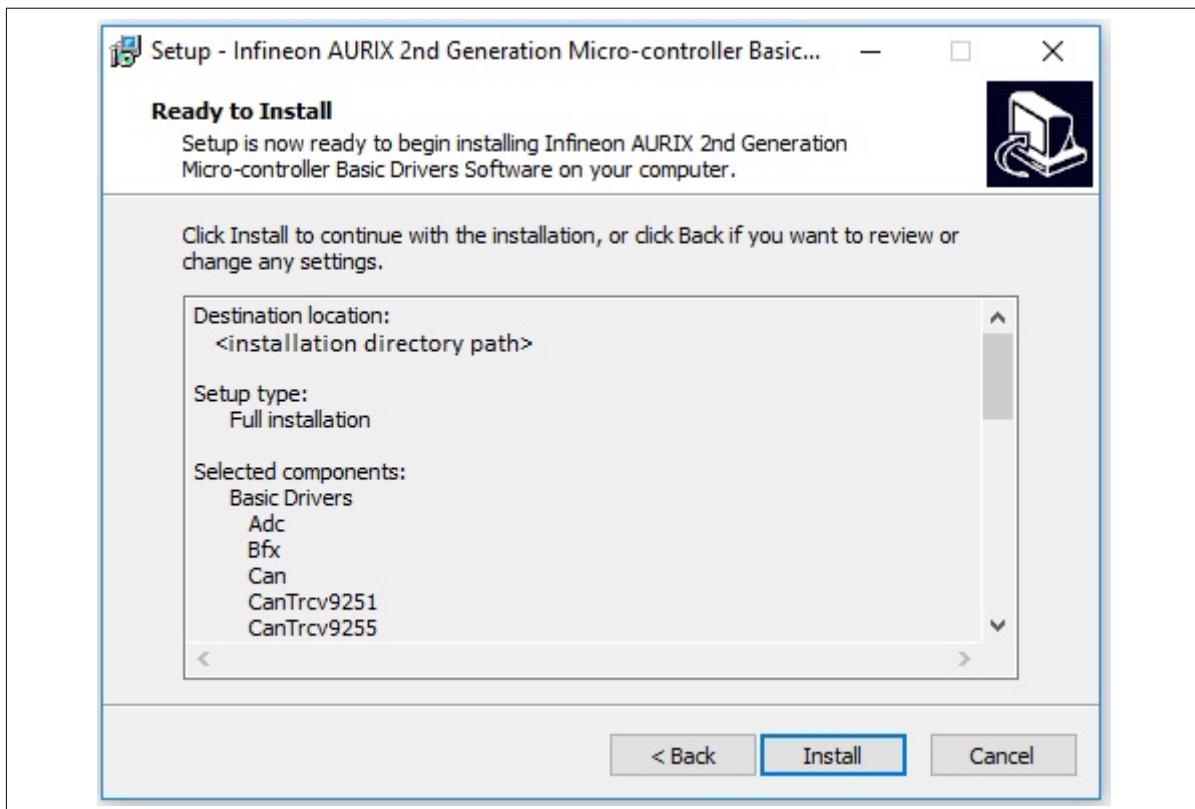


Figure 6 Ready to install

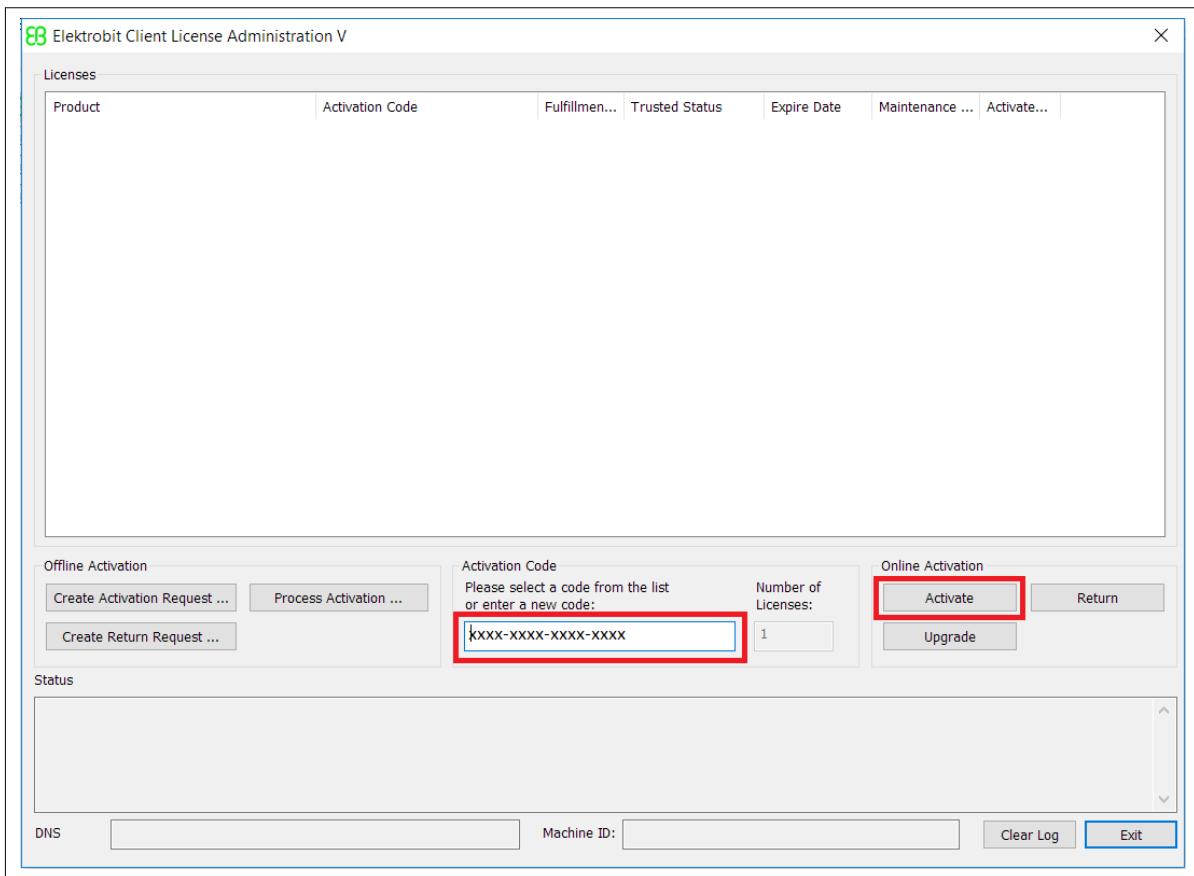
9. Click **Install**. This will initiate the installation process.
10. Copy all the plug-ins located at `<Installed-Package>\McIsar\PluginsTresos\eclipse\plugins` to the EB tresos™ plug-in path.

Note: *Previous installation should be uninstalled before a new package is installed, unless you intend to overwrite the existing installation.*

1.1.1.2 EB tresos™: license handling

The following procedure must be followed to activate the EB tresos™ license:

1. Install the **EB_Client_License_Administrator** tool.
2. Open the `<Install Path>\EB_Client_License_Administrator\bin\ClientLicenseAdministrator.exe`.
3. Enter the activation code. Click **Activate**.

Generic information

Figure 7 Tresos license activation

4. Status message **SUCCESSFULLY SEND ACTIVATION REQUEST** indicates the license has been activated successfully.

1.1.1.3 EB tresos: Eclipse handling

Due to a bug in the Eclipse tool, all directories named `org.eclipse.*` underneath the configuration directory of the EB tresos™ Studio installation have to be removed.

The `org.eclipse.*` directories contain cached information about installed plugins. Not deleting them can cause several undefined effects.

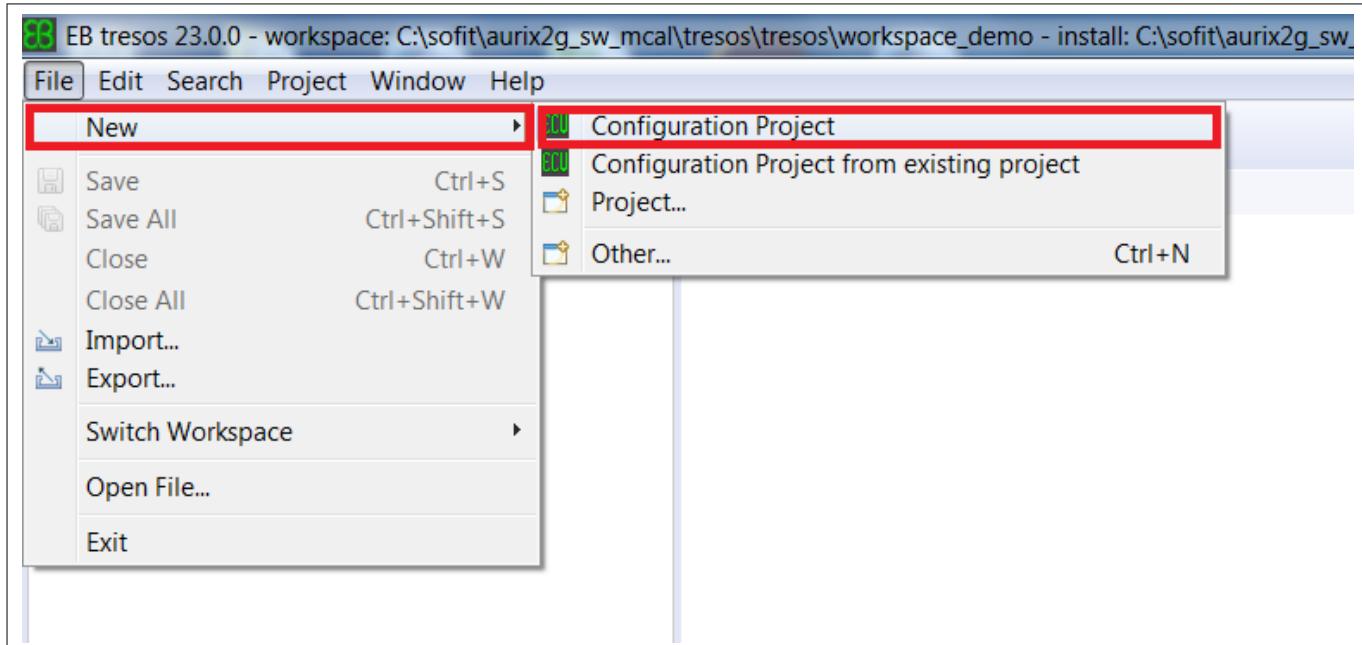
The new plugin will be available after restarting the EB tresos™ Studio.

If you have purchased EB tresos™ AutoCore together with EB tresos™ Studio, you can also integrate the pluginin with the user build environment. You will find the instructions on how to do this in the EB tresos™ AutoCore documentation: [EB tresos AutoCore documentation/EB tresos AutoCore user's guide/Build environment](#).

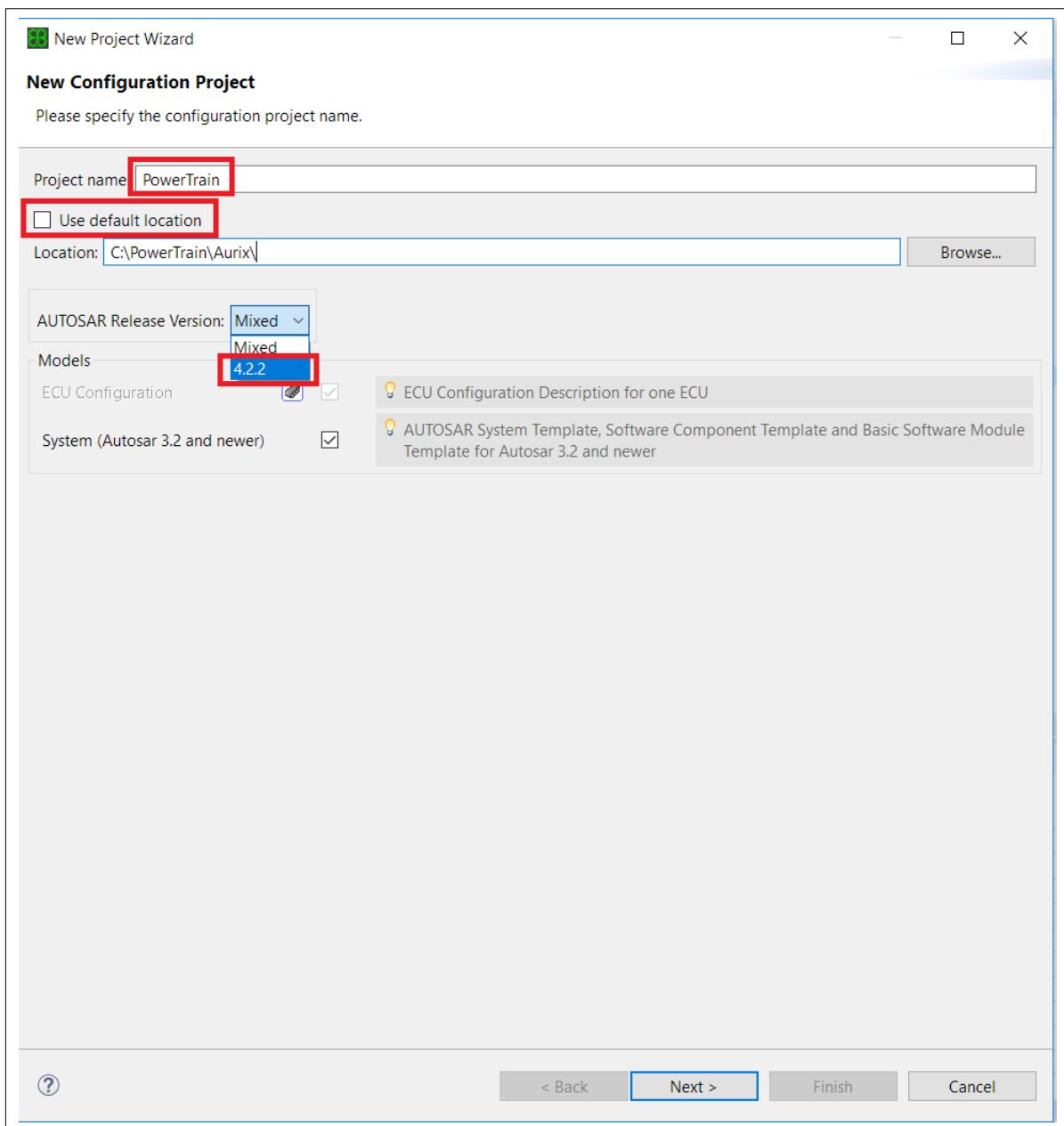
Generic information**1.1.1.4 Creating an EB tresos project****Prerequisites**

- The EB tresos tool works only in the 64-bit Windows version.
- User must have a license/key for the EB tresos tool.

The procedure for creating an EB tresos project is as follows:

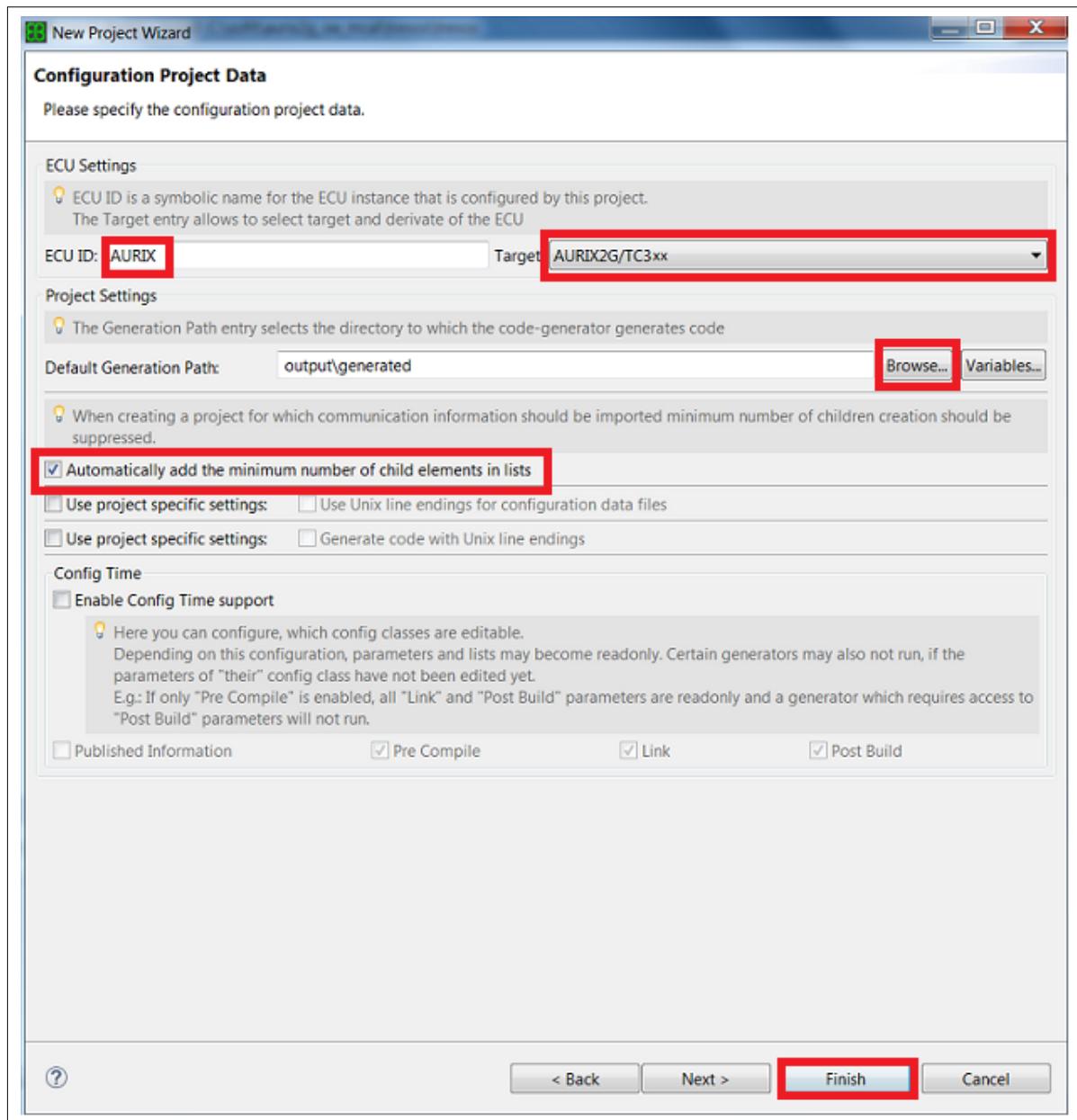
**Figure 8 Configuration Project option**

1. Click **File > New > Configuration Project**. The **New Project Wizard** displays.

Generic information**Figure 9 New Project Wizard**

2. In the **New Project Wizard**, do the following:

- In the **Project name** field, specify the project name.
- The project that you create will map to a directory structure in the file system. The default file system location is displayed in the **Location** field. If you want to create the project and its contained resources in a different location, uncheck the **Use default location** field and browse to a location where you want to save the project.
- From the **AUTOSAR Release Version** drop-down list, select a version.
- Click the **Next** button.

Generic information

Figure 10 Configuration Project Data
3. Do the following:

- Specify a name for the ECU instance in the **ECU ID** field.
- From the **Target** drop-down list, select a target.
- In the **Default Generation Path** field, browse to a location in your computer where you want to store the output.
- Check the **Automatically add the minimum number of child elements in lists** option.
- Click the **Finish** button.

Generic information

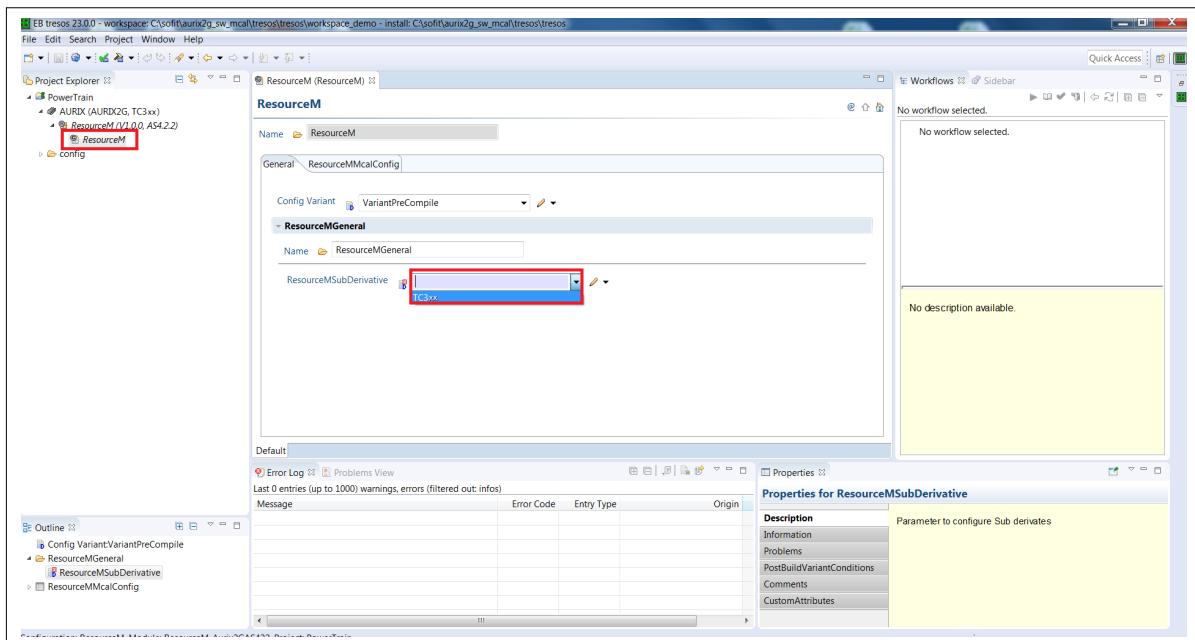


Figure 11 Sub-derivate selection

4. In the left pane, click the arrow next to the project name (PowerTrain in this case) to expand the elements.
5. Double-click **ResourceM**.
6. In the **ResourceMSubDerivative** drop-down list, select the exact sub-derivate.
7. Click the **Save** icon. Restart the EB tresos tool or switch workspace.

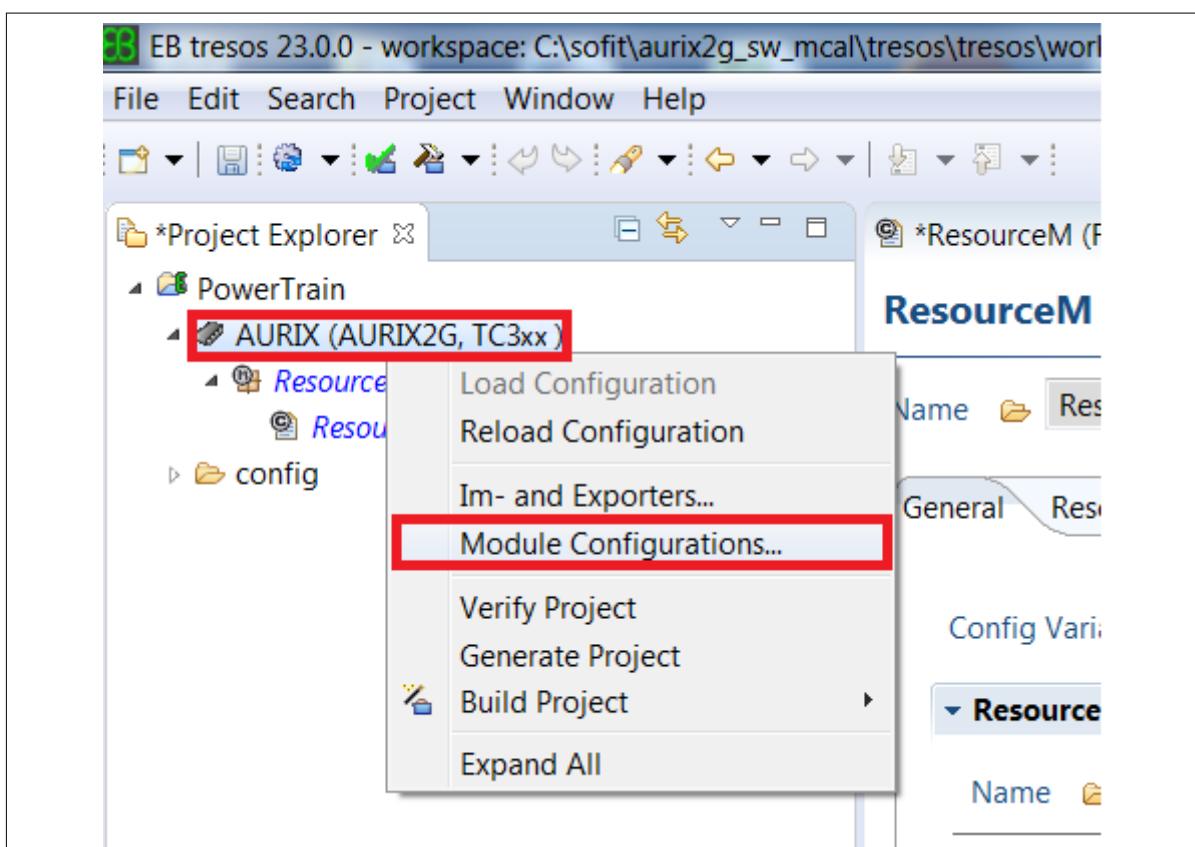


Figure 12 Module Configurations selection

8. Right-click the **ECU ID** and select **Module Configurations....**

Generic information

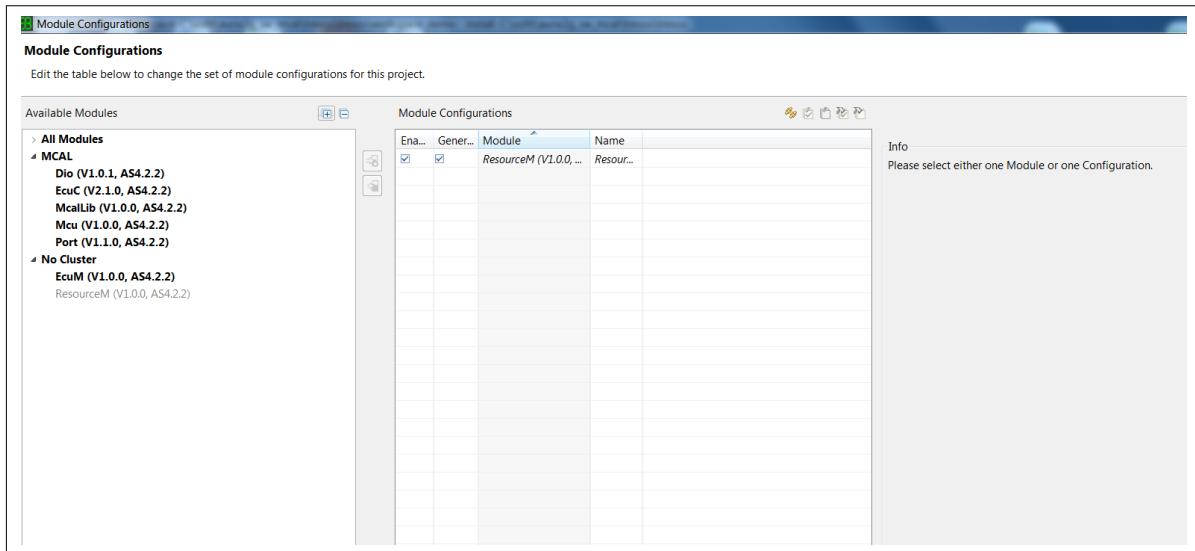


Figure 13 Module Configurations window

9. The **Module Configurations** window displays.

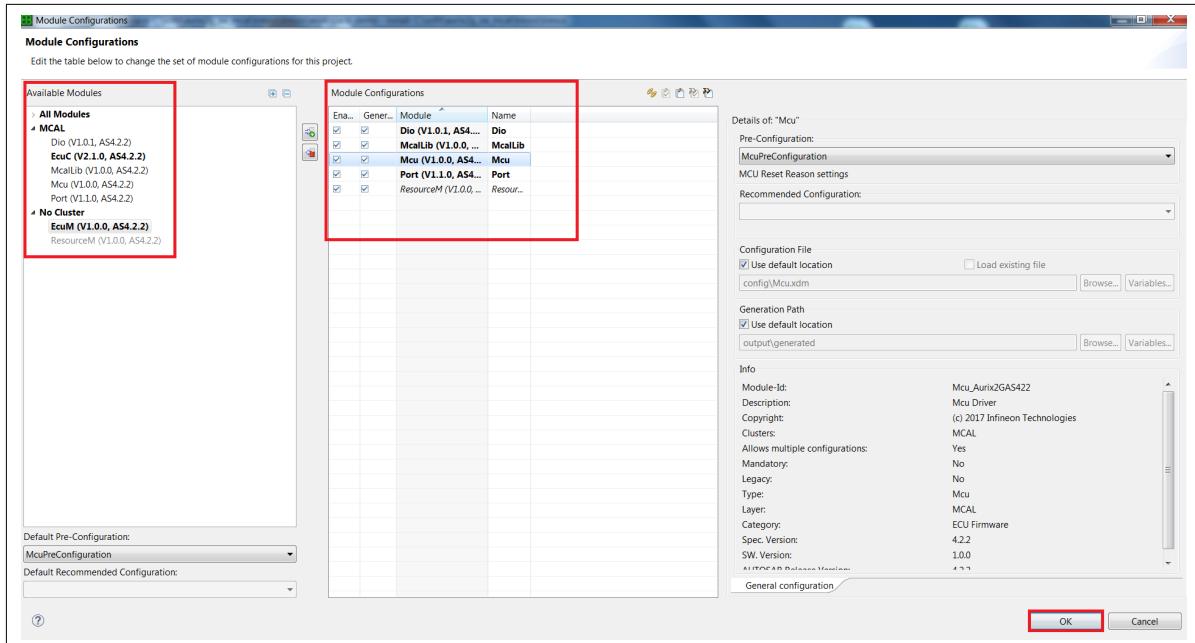


Figure 14 Modules populated

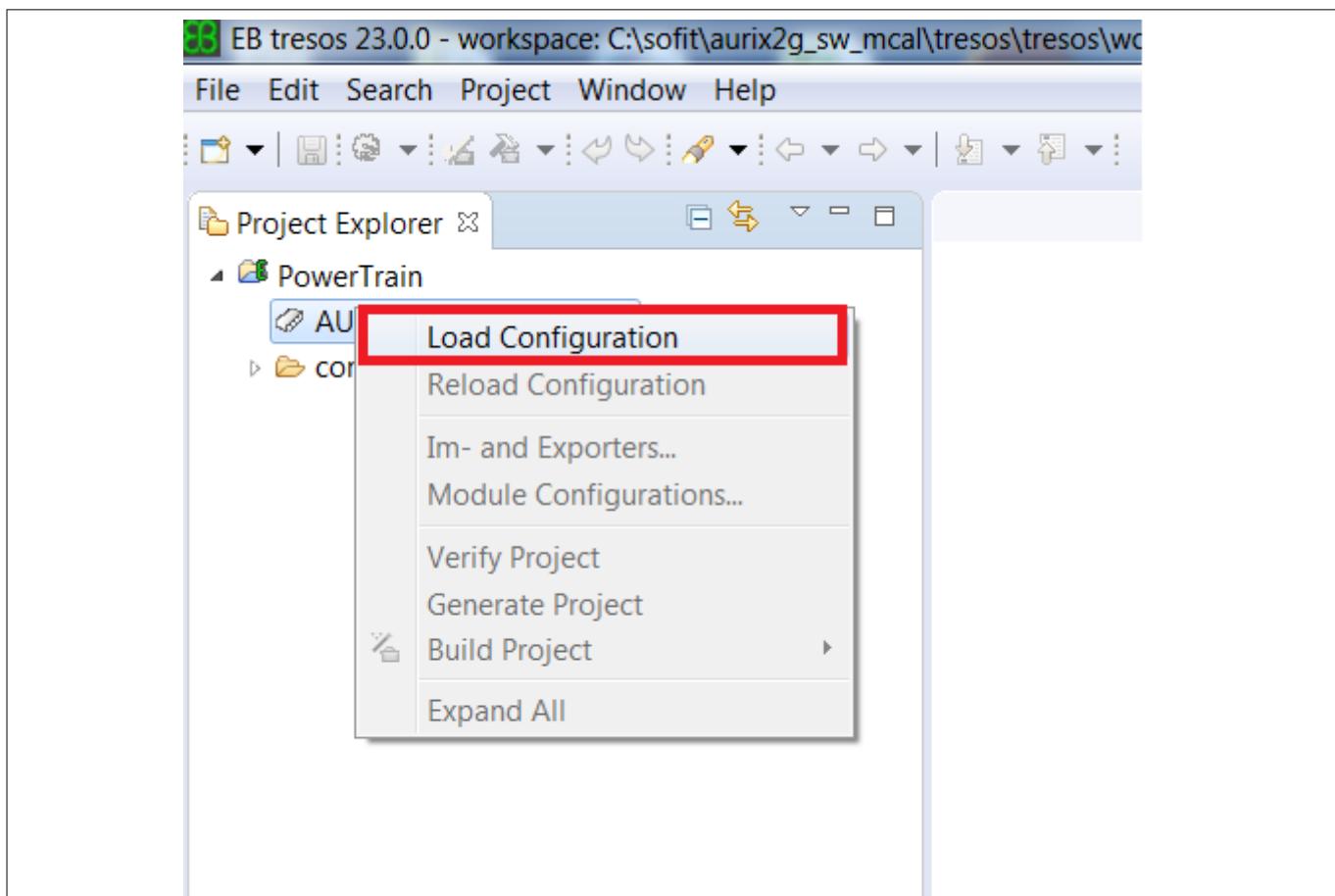
10. In the **Module Configurations** window, in the **Available Modules** section, double-click a module to move it to the **Module Configurations** section.
 11. Click the **OK** button to finish the EB tresos project creation.

Generic information**1.1.2 Typical module parameter configuration**

This section describes how a user may configure a typical parameter and generate configuration files.

1.1.2.1 Configuring parameters

Once the EB tresos project is created, one needs to configure the module parameters.

**Figure 15 Loading configuration**

1. Right-click the **ECU ID** and select the **Load Configuration** option. Alternatively, you can double-click the **ECU ID** to display the editor.

Generic information

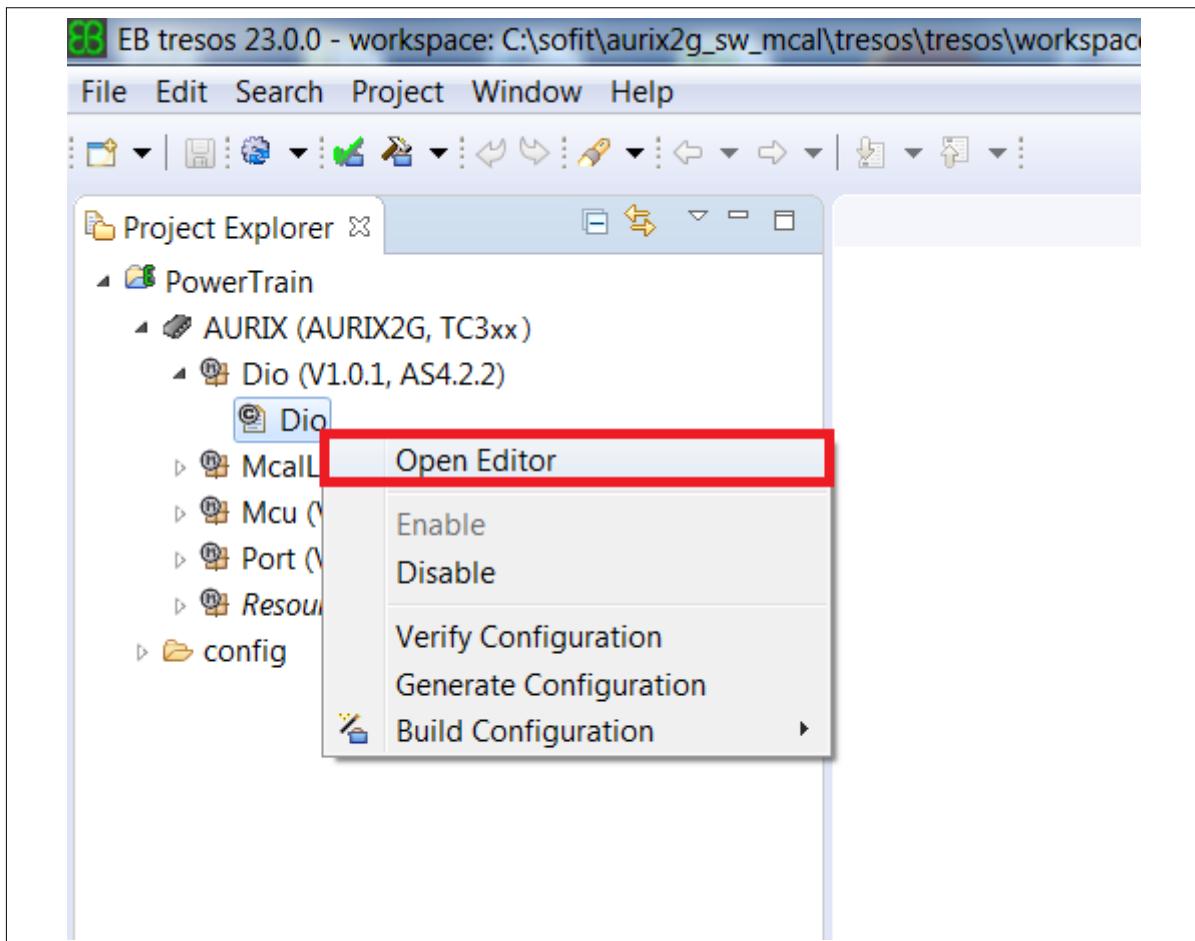


Figure 16 Open Editor option

2. After loading the configuration, right-click a module that you want to configure and select the **Open Editor** option. Alternatively, you can double-click the module to display the editor.

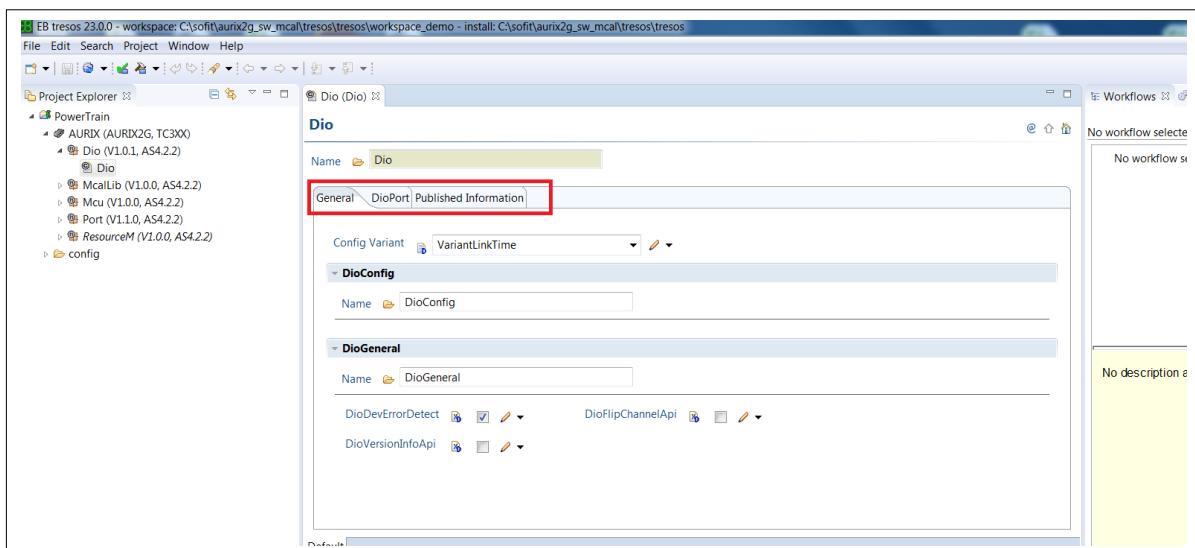


Figure 17 Parameter selection

3. In the editor section, click the respective parameter tab you want to configure.

Generic information

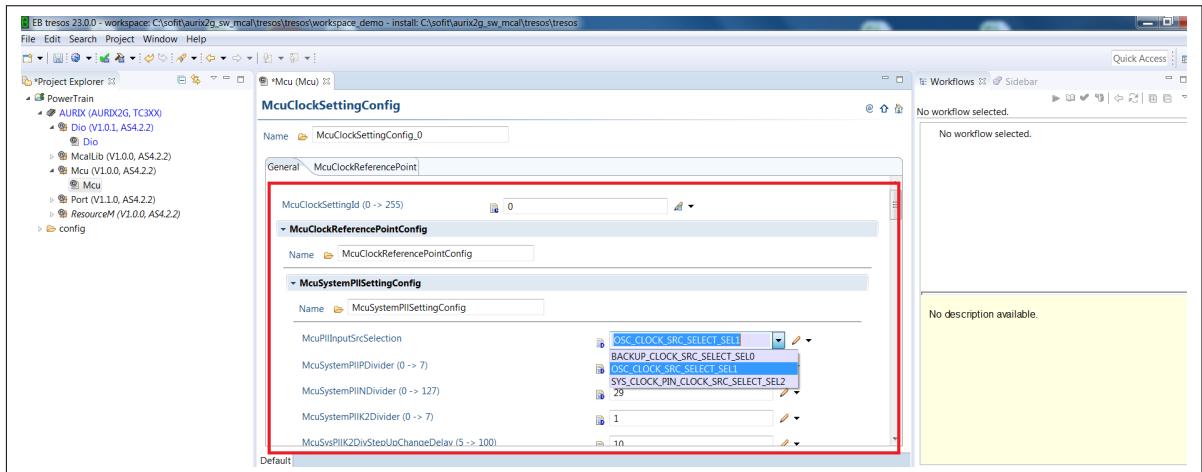


Figure 18 Configuring parameter

4. Configure/edit the required parameters as applicable.
5. Click the **Save** icon on the top left of the screen.

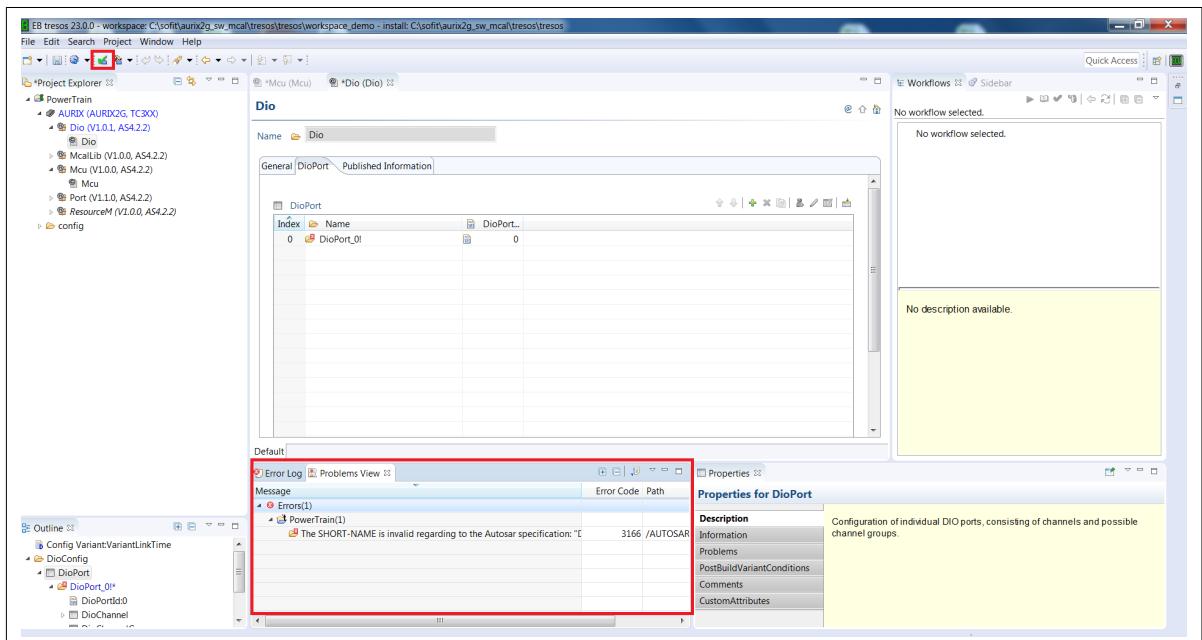
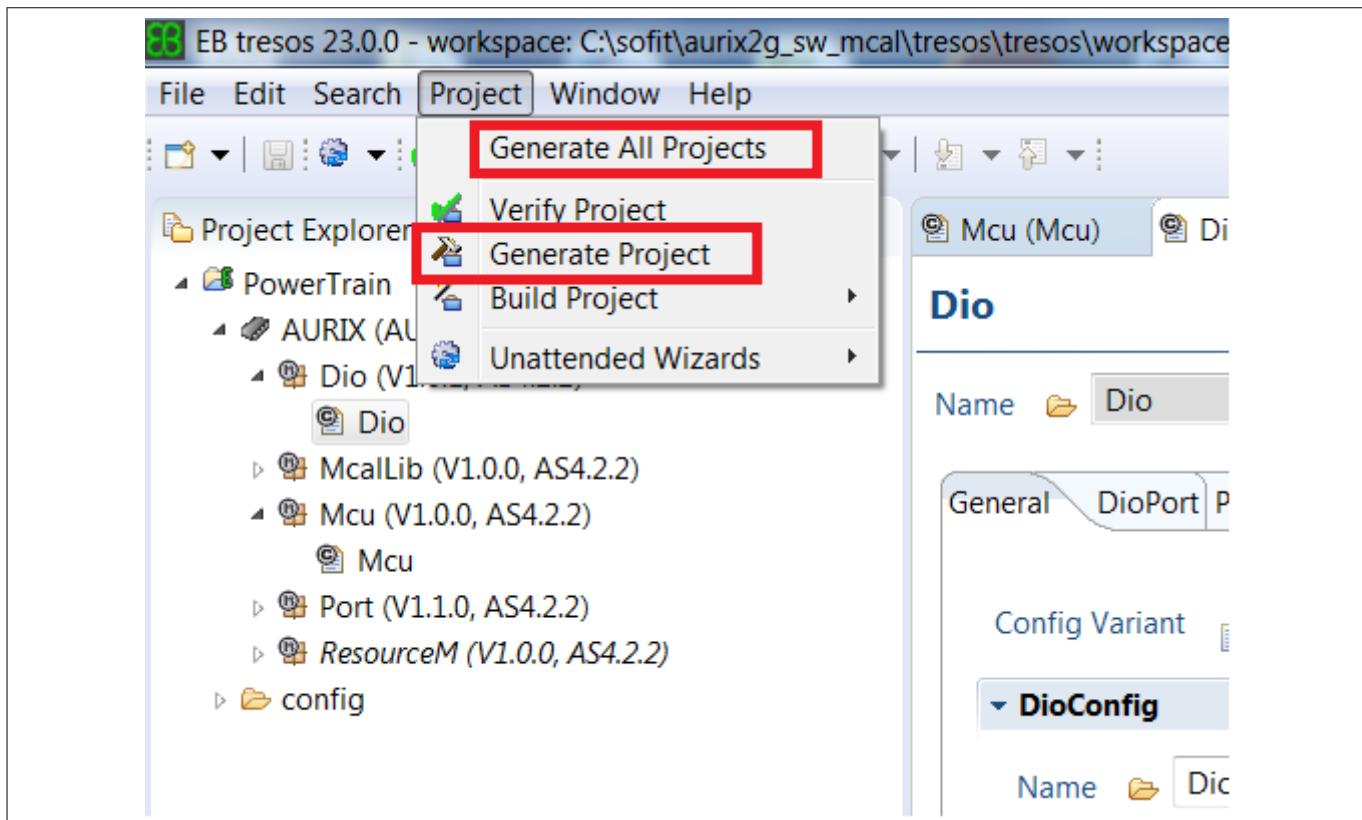


Figure 19 Parameter validation

6. Click the **Verify** icon on the top of the screen. If there are any configuration errors, they are displayed at the bottom of the screen. Click the **Error Log** and **Problems View** tabs to view and resolve the error(s).

1.1.2.2 Generating configuration files

The steps to generate configuration files are as follows:

Generic information

Figure 20 Generating project

1. From the menu options, select **Project**.
2. Select either **Generate All Projects** or **Generate Project** option.
 - **Generate All Projects:** selecting this option will generate the configured XDM, <module>_PBcfg.c, <module>_PBcfg.h, <module>_Cfg.h and <module>_Bswmd.arxml files for all the projects.
 - **Generate Project:** selecting this option will generate the configured XDM, <module>_PBcfg.c, <module>_PBcfg.h, <module>_Cfg.h and <module>_Bswmd.arxml files for the selected project.

You can locate the generated files in the following location in your computer: \tresos\tresos\workspace \PowerTrain\output\generated.

1.1.3 Building and linking to generate final application binary

This section describes how to build all source files and link them together to generate the ELF/HEX file(s).

This package leverages BIFACES, the standard build framework developed by Infineon. Any other package(s), supplied by Infineon, supporting BIFACES can be integrated with this package, if relevant.

1.1.3.1 Building package

For the list of derivatives supported, refer the release notes provided with this package.

Note: Before proceeding with the following steps, to build a variant, make sure to copy all the plug-ins located at <Installed-Package>|McSar|PluginsTresos|eclipse|plugins to the EB tresos plug-in path.

The steps to build a variant are as follows:

Generic information

1. Open the command line shell.
 - To build <device>, go to C:\<package>\DemoWorkspace\McalDemo\<device>.
2. In the shell, invoke the DemoAppBuild.bat with different parameters. The user is provided with the following options.
 - Parameter 1 is compiler.
 - The compiler can be *TASKING*, *GNU* or *DCC* or *GHS*. If no parameter is passed, *TASKING* is selected by default.
 - Note:* *The GHS compiler support is provided only for TC35x and TC38x devices.*
 - Parameter 2 is the option to build with or without TRESOS generation.
 - User can pass *WITHOUT_TRESOS* or *WITH_TRESOS*. If no parameter is passed, the build system takes *WITH_TRESOS* as the default parameter.
 - Parameter 3 is the option to set the TRESOS path.
 - User has to pass the full TRESOS path including the Drive. If no parameter is passed, the build system takes C:\sofit\aurix2g_sw_mcal\tresos\23.0.0\tresos\tresos\bin as the default path.
 - Parameter 4 is the option to set the COMPILER path.
 - User has to pass the full COMPILER path including the Drive. If no parameter is passed, the build system takes the default COMPILER path (that is, *TASKING*, *GNU*, *GHS* or *DCC*) from SOFIT.

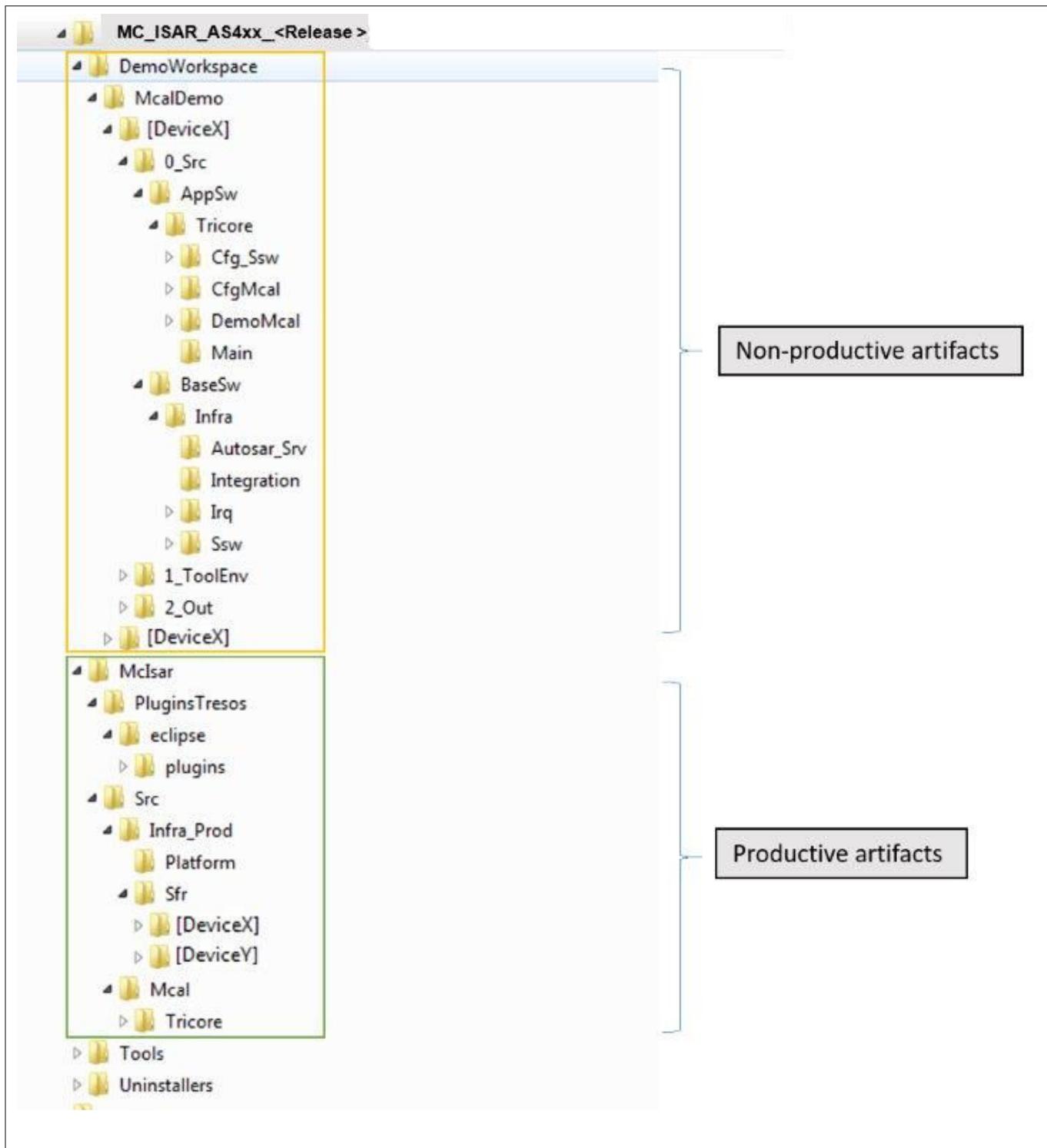
Example commands to build DemoApp

- Double-click the DemoAppBuild.bat file.
 - This will build for the *TASKING* compiler and will generate the TRESOS code as well.
- To build for the *GNU/DCC/GHS* compiler without generating TRESOS code:
 - DemoAppBuild.bat GNU/DCC/GHS
WITHOUT_TRESOS
- To build for the WindRiver compiler with the TRESOS code generated and with different TRESOS and Compiler paths:
 - DemoAppBuild.bat DCC WITH_TRESOS C:/EB/Tresos/bin
C:/WindRiver/Compiler/diab-5.9.6.4/WIN32

1.1.3.2 Release folder structure

The following diagram depicts the organization of the release folders along with Makefile distribution.

Note: *Similar folder structure will be created for all the supported derivatives.*

Generic information

Figure 21 Release folder structure

Note: *The 2_Out folder is generated after compilation.*

Note: *The availability of the modules for the released product must be referred from the Release Notes.*

Generic information

1.1.3.3 Makefile adaptations

To make the process easier, Infineon have developed a wrapper over the Makefile in the form of a batch file. When this batch file is run, it automatically modifies the Makefile.

The following Makefiles are updated:

- ConfigPrjExtn.mk
- Tresos.mk

To modify Makefile(s) to suit the user environment, one needs to edit Makefile(s) manually.

The procedure to manually edit Makefile is as follows:

1. Open the Tresos.mk file (located at <package>\DemoWorkspace\McalDemo\<derivative>\1_ToolEnv\0_Build\1_Config\Extensions) in a text editor. By default, the DemoApp basic build is selected. The Tresos.mk file allows to configure the following parameter(s):
 - EB tresos bin path
 - EB tresos command
 - DemoApp configuration path
2. Note: *By default, all modules are selected for build.*
2. The XML files are placed in the <package>/Tools>. Here, there are several files based on the compiler/derivative/FR_rebuild. These files can be modified for:
 - additional external paths for extra code to be built
 - discard file patterns
3. In the Config_<compiler>.mk file (located at <package>\DemoWorkspace\McalDemo\<derivative>\1_ToolEnv\0_Build\1_Config\Config_Tricore_<compiler>), edit the following parameter(s):
 - Compiler path
 - Compiler options
 - Assembler options
 - User includes

Editing batch file(s)

The procedure to manually edit a batch file to adapt to user environment is as follows. However, Infineon recommends using only the batch file (located at <package>\DemoWorkspace\McalDemo\<derivative>) instead of editing the Makefiles.

1. Open the DemoAppBuild.bat file in a text editor. The user is allowed to build the code for the supported compiler(s).
2. The following parameter(s) can be edited in the DemoAppBuild.bat file.
 - Parameter 1: compiler (for example, GNU)
 - Parameter 2: can have three possible combinations as follows:
 - with Tresos generation for both FR and Normal builds (this option is selected by default)
 - Values: empty, WITH_TRESOS
 - without Tresos generation for both FR and Normal build
 - Value: WITHOUT_TRESOS
 - without Tresos generation for Normal build and no FR build
 - Value: WITHOUT_TRESOS_WITHOUT_FR
 - Parameter 3: EB tresos bin path
 - Parameter 4: Compiler path

Generic information

1.1.3.4 **Linker script adaptations**

Linker is a program that takes one or more object files generated by a compiler and combines them into a single executable file. The linker program uses a linker script. The main purpose of the linker script is to describe how the sections in the input files should be mapped into the output file, and to control the memory layout of the output file.

The following linker scripts are provided by Infineon for the linking purpose:

- `Lcf_Tasking_Tricore.lsl` for the Tasking compiler
- `Lcf_Gnuc_Tricore.lsl` for the GNU compiler
- `Lcf_Dcc_Tricore.lsl` for the WindRiver compiler
- `Lcf_Ghs_Tricore.lsl` for the GreenHills compiler

Note: The GHS compiler support is provided only for TC35x and TC38x devices.

Each linker script can be modified by user as per their environment requirements.

MCAL modules use `<Module>_MemMap.h` file for defining compiler pragma, which controls the output sections like placing the code and global variables. It is entirely up to the integrator to define a linking scheme that places the output sections at desired addresses in the processor memory map.

Note: The linker script provided is compiler dependent.

For example, the ADC driver ensures that global variables and text segments are allocated to appropriate output sections.

Generic information

A sample for using compiler pragma for a global variable from `Adc.c` is as follows:

```
#define ADC_START_SEC_VAR_CLEARED_ASIL_B_CORE2_UNSPECIFIED
#include "Adc_MemMap.h"
    static Adc_GlobalDataType Adc_KernelData_Core2[ADC_KERNEL_USED_COUNT_CORE2];
#define ADC_STOP_SEC_VAR_CLEARED_ASIL_B_CORE2_UNSPECIFIED
#include "Adc_MemMap.h"
```

The user may define the compiler pragmas in `Adc_MemMap.h` as below:

```
#elif defined ADC_START_SEC_VAR_CLEARED_ASIL_B_CORE2_UNSPECIFIED
    #ifdef _TASKING_C_TRICORE_
        #pragma section_name_with_symbol
        #pragma section all "ClearedData_Cpu2.Unspecified"
        #pragma align 4
    #elif defined _GNU_C_TRICORE_
        #pragma section "ClearedData_Cpu2.Unspecified" aw 4
    #endif
    #undef ADC_START_SEC_VAR_CLEARED_ASIL_B_CORE2_UNSPECIFIED
    #undef MEMMAP_ERROR
#elif defined ADC_STOP_SEC_VAR_CLEARED_ASIL_B_CORE2_UNSPECIFIED
    #ifdef _TASKING_C_TRICORE_
        #pragma align restore
        #pragma section all
    #elif defined _GNU_C_TRICORE_
        #pragma section
    #endif
    #undef ADC_STOP_SEC_VAR_CLEARED_ASIL_B_CORE2_UNSPECIFIED
    #undef MEMMAP_ERROR
```

1.1.3.5 Initiation of compiling and linking

The DemoApp build is launched by invoking `DemoAppBuild.bat` present in the <Installed Package>\DemoWorkspace\McalDemo folder. The application (DemoApp files in this case), source and configuration files are compiled and linked together. The output files, *.elf/*.*.hex, are generated after completion of the build process in the <Installed-Package>\DemoWorkspace\McalDemo\2_Out\<Tricore_Tasking\Tricore_Gnuc\Tricore_Dcc\Tricore_Ghs> folder.

The output `DemoApp_Node0.elf/DemoApp_Node0.hex` and `DemoApp_Node1.elf/DemoApp_Node1.hex` file are generated if the FR module is selected for installation. The `DemoApp_Node0.elf/DemoApp_Node0.hex` file is used for executing the DemoApp of all modules other than FR. The `DemoApp_Node1.elf/DemoApp_Node1.hex` file is only used for executing the FR DemoApp. The output `DemoApp.elf/DemoApp.hex` file is generated if the FR module is not selected for installation. The output files are generated in the <Installed-Package>\DemoWorkspace\McalDemo\2_Out\<Tricore_Tasking\Tricore_Gnuc\Tricore_Dcc\Tricore_Ghs> folder.

Note: All packages (BASIC, COM-E, CD, DEMO) with all provided drivers should be installed in the same directory in order to build the DemoApp provided for MCAL drivers. Though it is possible to install individual packages in a separate directory for use but the DemoApp cannot be built individually selectively.

Generic information

Note: *In case of selective installation (for example, if only ETH is to be installed from the COM-E package), remove the modules and their respective .xdm files from the EB tresos workspace and also remove the driver source code, demo code, IRQ and integration files.*

Command to merge multiple configured XDM to single EPC file

Following is an example of merging XDM file to generate an EPC file:

```
C:/sofit/aurix2g_sw_mcal/tresos/tresos/bin/tresos_cmd.bat -DMapOptionalAsList=false
-Dtarget=AURIX2G -Dderivate=<device> legacy convert Port.xdm
Pwm_17_Gtm.xdm Spi.xdm Adc.xdm Dem.xdm Dio.xdm EcuM.xdm Gpt.xdm
Icu_17_TimerIp.xdm
Irq.xdm Mcu.xdm ResourceM.xdm Combined.epc@asc:4.2.2
```

Note: *-Dderivate should be changed as per the desired derivate to run the DemoApp.*

Command to generate code from EPC file

Following is an example command to generate code from the EPC file:

```
C:/sofit/aurix2g_sw_mcal/tresos/tresos/bin/tresos_cmd.bat -Dttarget=AURIX2G
-Dderivate=<device> legacy generate Combined.epc@asc:4.2.2 -o
outputdir -g Irq_Aurix2GAS422 -g Mcu_Aurix2GAS422 -g
Port_Aurix2GAS422 -g
Dem_Aurix2GAS422 -g EcuM_Aurix2GAS422 -g Adc_Aurix2GAS422 -g
Dio_Aurix2GAS422 -g
Gpt_Aurix2GAS422 -g Icu_Aurix2GAS422 -g Pwm_Aurix2GAS422 -g
Spi_Aurix2GAS422 -g
ResourceM_Aurix2GAS422
```

Utilities for building DemoApp

The following utilities are used for the build process:

Description	Location
The Makefile used for building the application contains rules for generating the executable	The file is located at <Installed-Package>\DemoWorkspace\<Device>\McalDemo\1_ToolEnv\0_Build

Note: *Any error in compilation and linking displays in the DOS prompt.*

Generic information

1.1.3.6 Flashing the image

Image is flashed using the UDE Memtool. The steps for flashing image using the UDE Memtool are as follows:

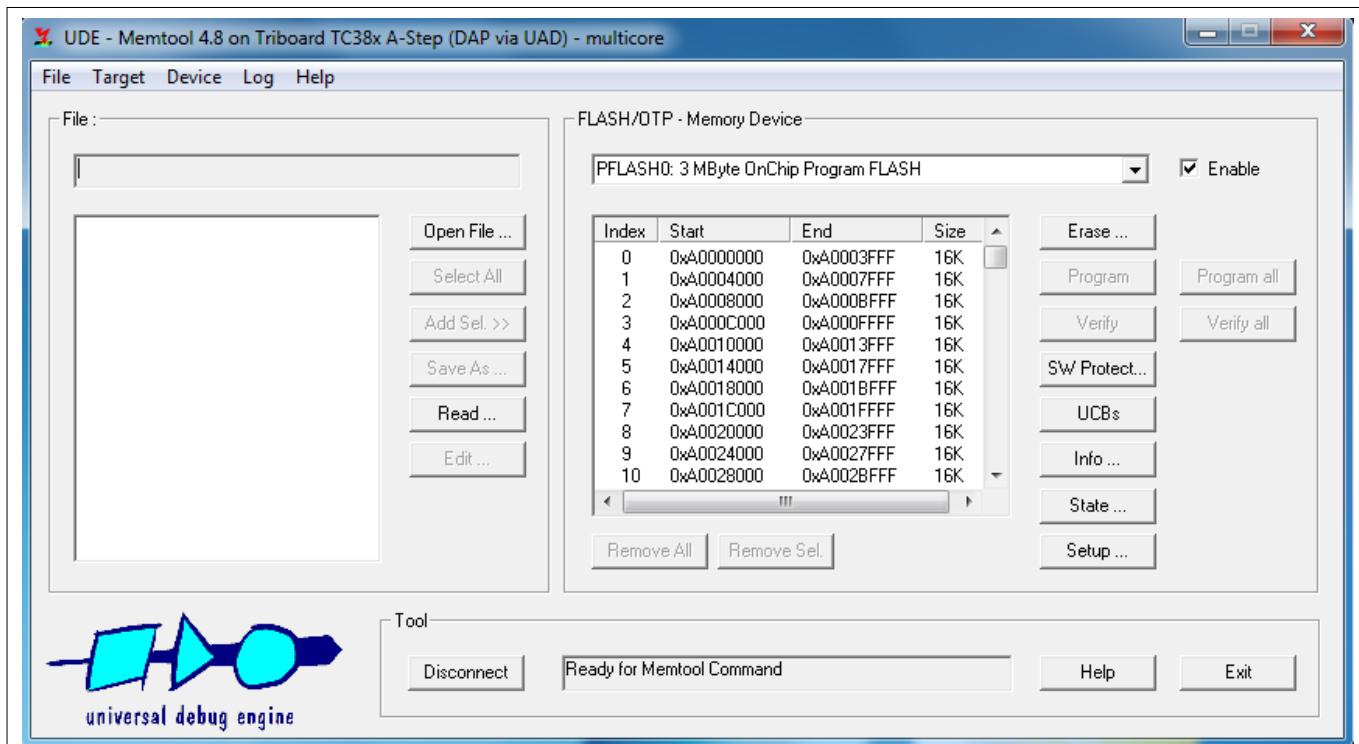
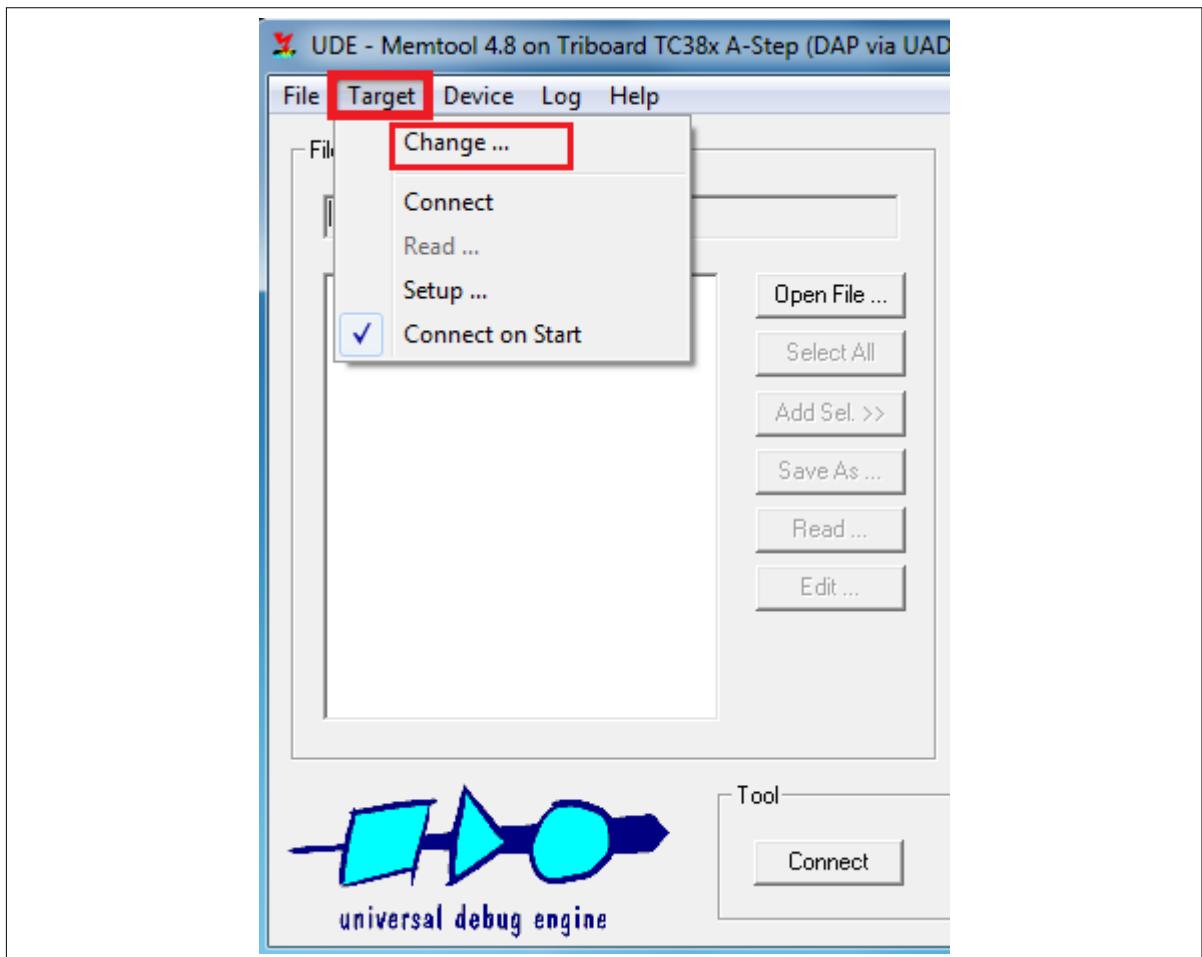
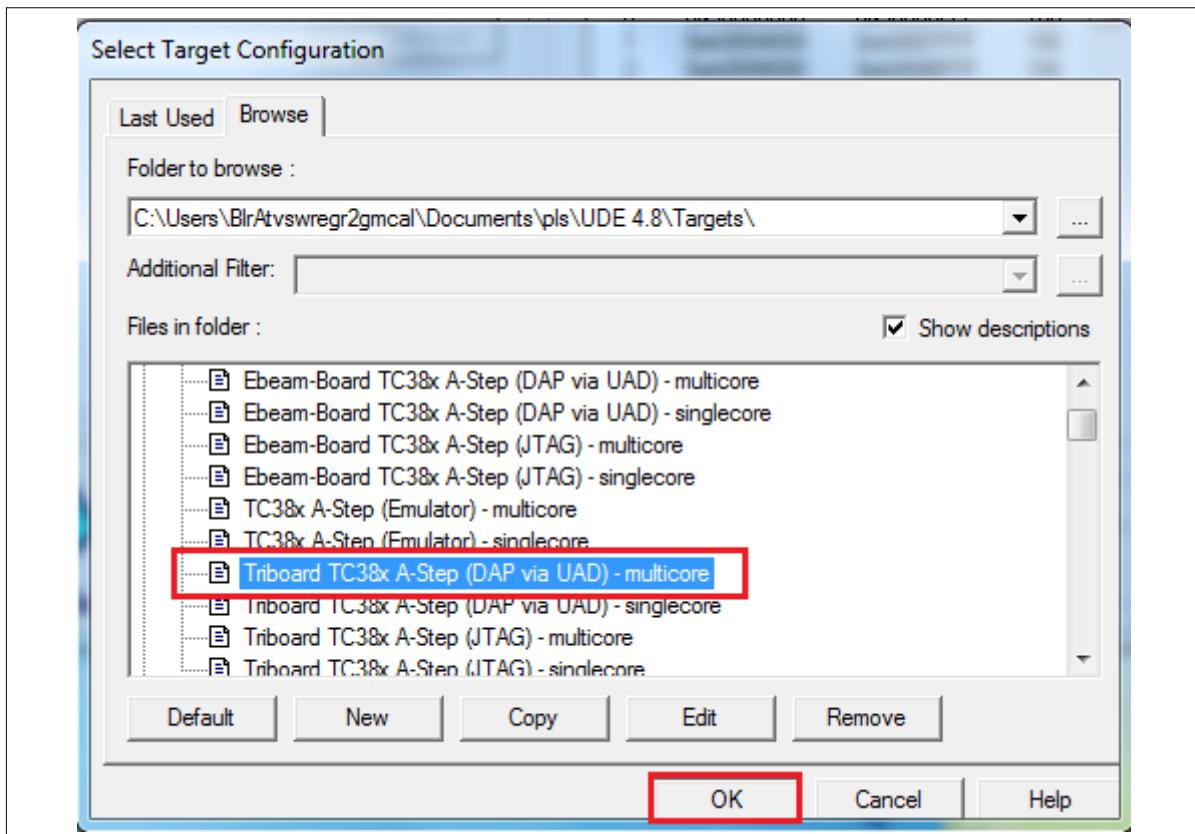


Figure 22 UDE Memtool interface

1. Double-click the `Memtool.exe` file to open the UDE Memtool.

Generic information**Figure 23 Configuring target file**

2. Click **Target > Change** to configure the target file for the respective device.

Generic information**Figure 24 Selecting target configuration**

3. Select a configuration file and click the **OK** button.

Generic information

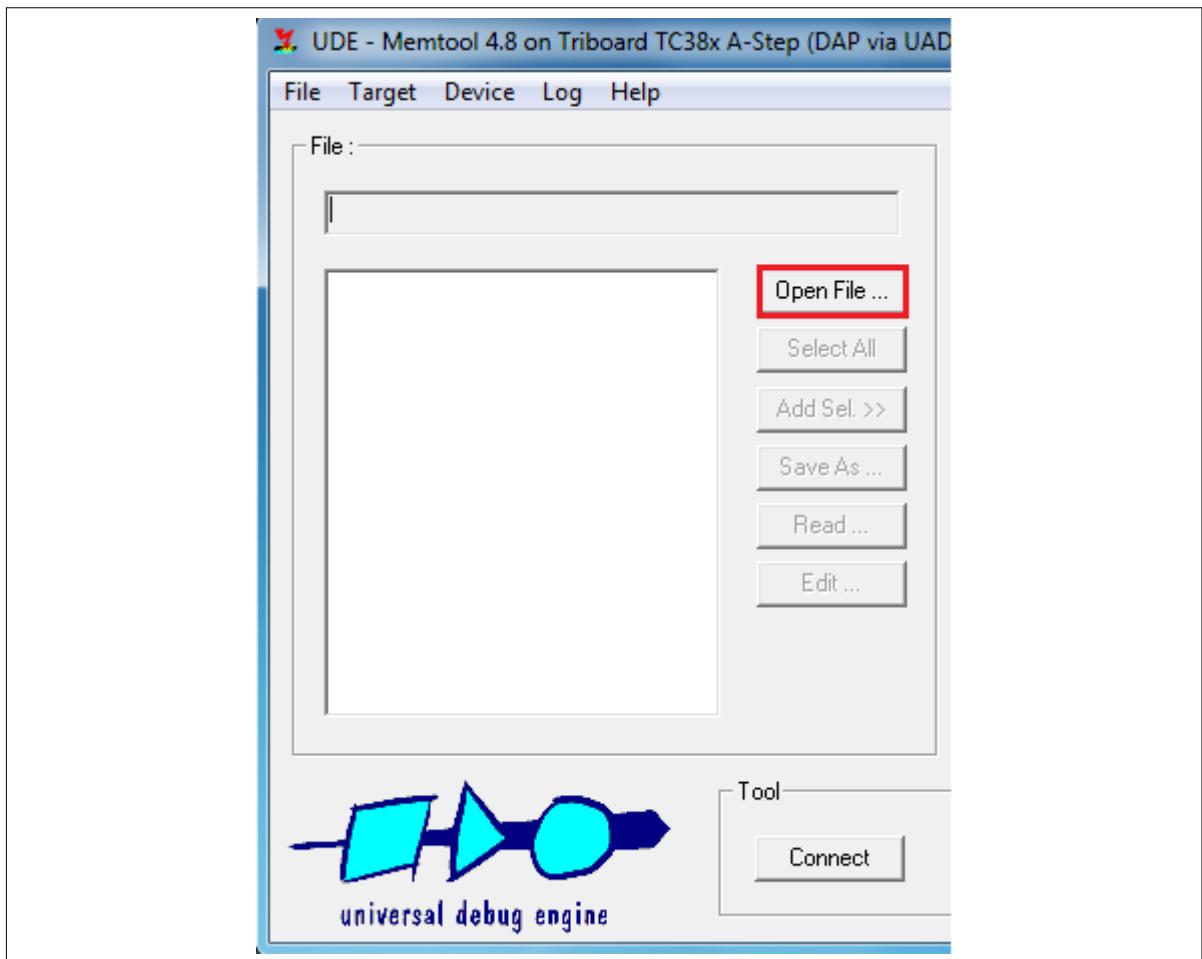


Figure 25 Opening a file

4. Click the **Open File** button.

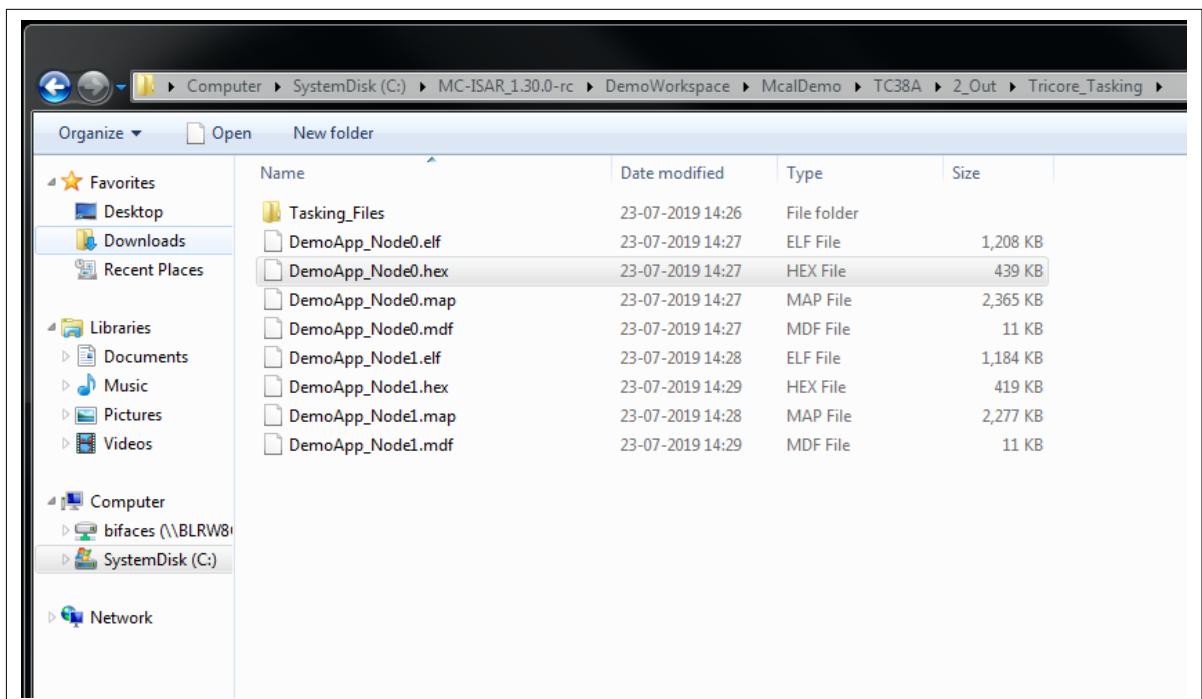
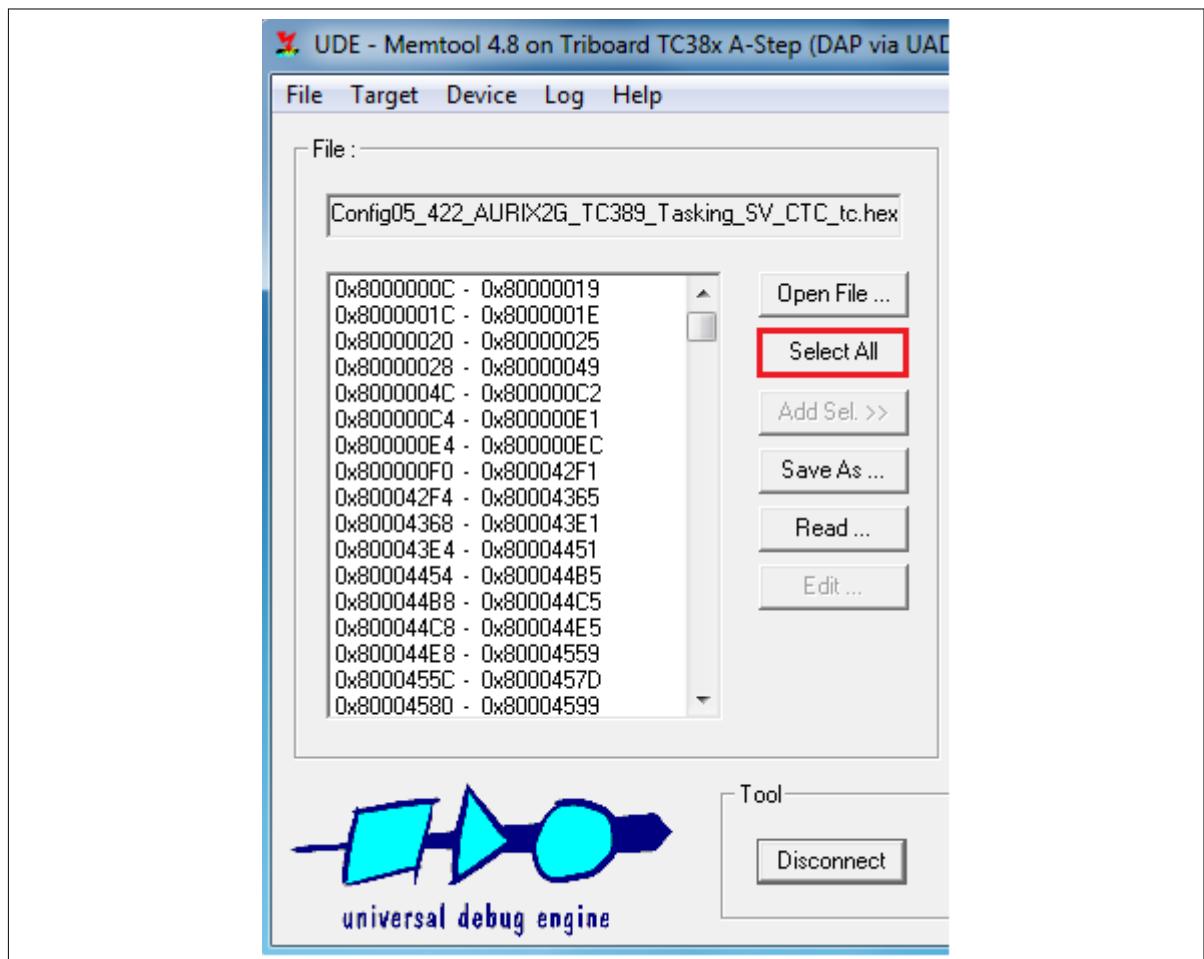


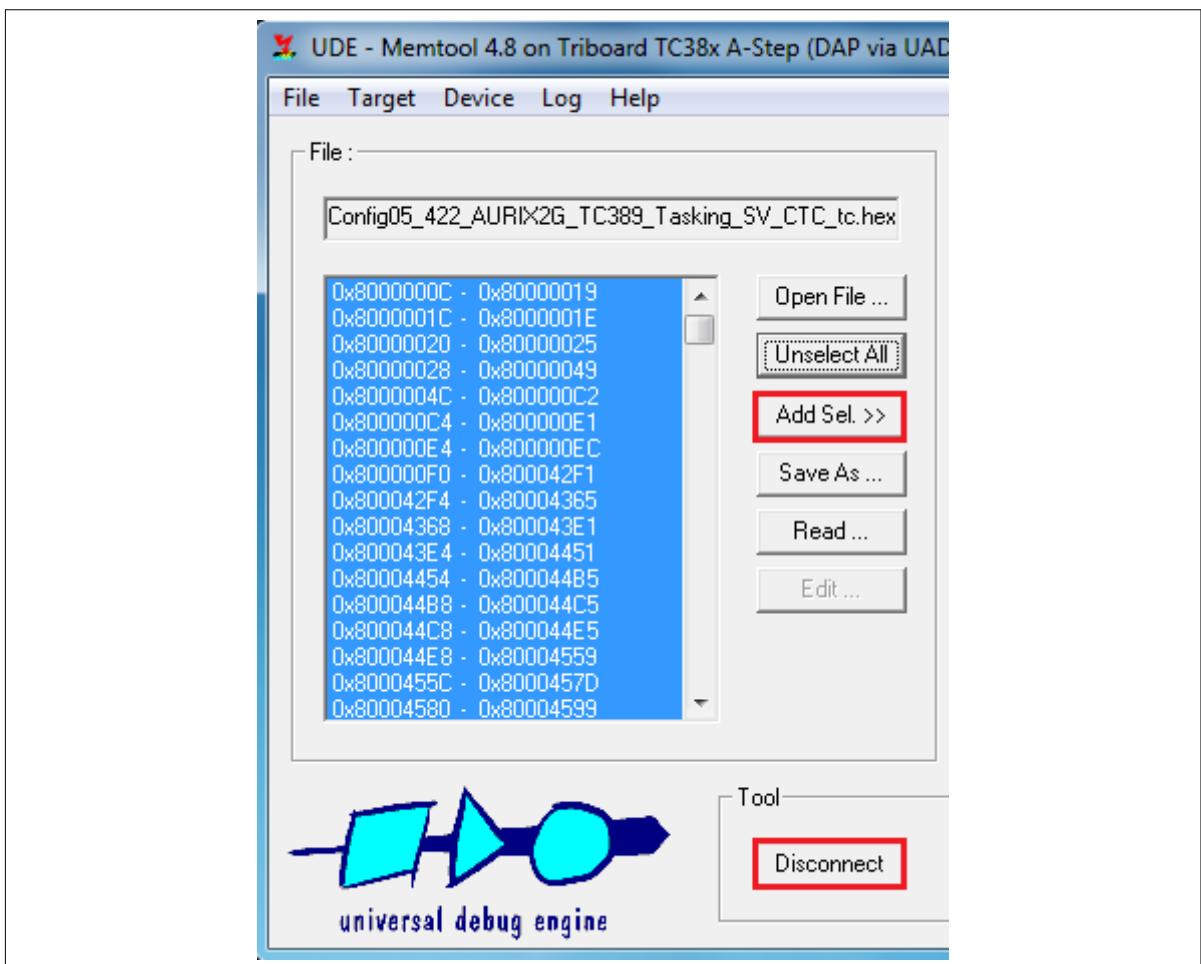
Figure 26 Selecting hex file

Generic information

5. Select the hex file.

**Figure 27 Selecting flashing sectors**

6. Click the **Select All** button to select flashing all sectors.

Generic information**Figure 28 Flashing memory**

7. Click the **Connect** button. Once connected successfully, the button turns to **Disconnect**.
8. Click the **Add Sel.** button to flash the memory.

Generic information

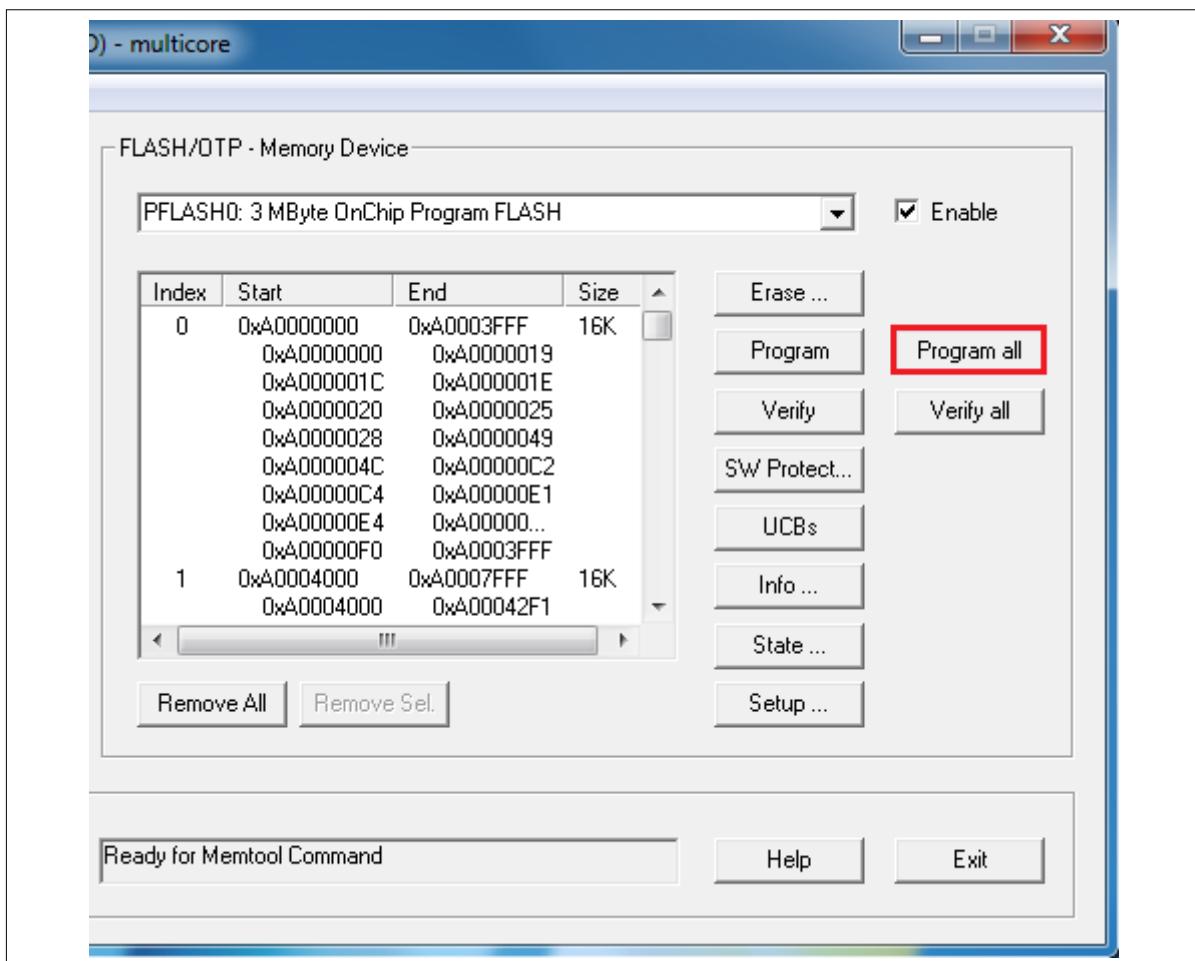


Figure 29 Selecting program all option

- Click the **Program all** button to flash the hex image.

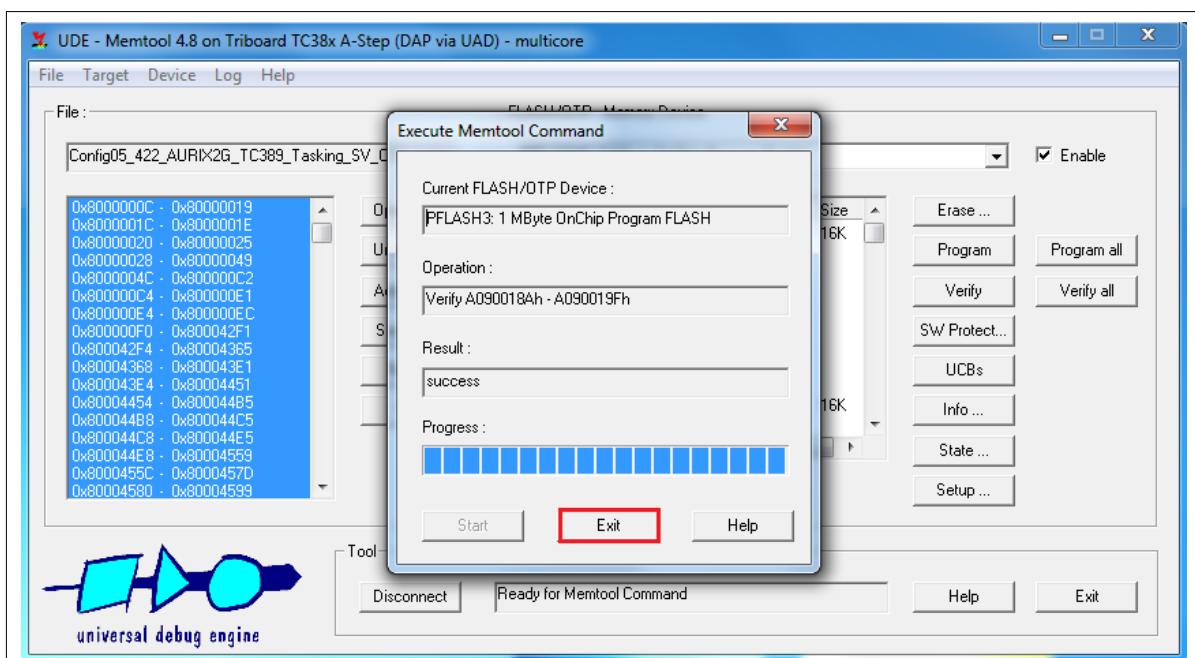


Figure 30 Executing command

- The selected image is flashed. Click the **Exit** button to finish the process.

Generic information

1.2 Multicore support

The MCAL software is capable of supporting invocation of APIs from multiple CPUs simultaneously. Outcome of this requires partitioning of MCAL configuration and runtime data.

A typical TC3xx device has multiple CPUs. The software stack hosted on a device of the TC3xx family can be partitioned and executed from multiple CPUs simultaneously.

For more information, refer to module-specific multicore section in this document.

Note: *The memory sections marked as global should be relocated to the non-cached LMU region. In devices with no LMU, non-cached DSPR/DLMU can be used.*

Table 3 **Module details**

Module	Multicore partitioning
ADC	EVADC hardware group
BFX	Supports multicore without resource partition. Invocation from all core in parallel is allowed.
CAN	MCMCAN node
CanTrcv_17_V9251	Not supported
CanTrcv_17_W9255	Not supported
CRC	Supports multicore without resource partition. Invocation from all cores in parallel is allowed.
DIO	Supports multicore without resource partition. Invocation from all cores in parallel is allowed.
DMA	DMA channels
DSADC	Not supported
ETH	Ethernet controller
FEE	Not supported
FLS	Not supported
FLSLOADER	Not supported
FR	Not supported
GPT	Logical GPT channels
HSSL	Not supported
I2C	Not supported
ICU	Logical ICU channels
IOM	Not supported
LIN	Not supported
MCALLIB	Supports multicore without resource partition. Invocation from all cores in parallel is allowed.
MCU	Supports multicore without resource partition. Invocation from all cores in parallel is allowed.

Generic information
Table 3 Module details (continued)

Module	Multicore partitioning
OCU	Logical OCU channels
PORT	Supports multicore without resource partition. Invocation from all cores in parallel is allowed.
PWM	Logical PWM channels
SENT	Sent channel
SMU	Supports multicore without resource partition. Invocation from all cores in parallel is allowed.
SPI	QSPI hardware Kernels
STM	STM timer
UART	Not supported
WDG	WDT of each core

Attention: *For detailed multicore capability of each driver, refer to the Integration hints sections.*

1.2.1 Multicore initialization

A module is initialized from the context of core, that is, each core must call the initialization function independently.

The following DETs are reported from initialization:

- <MOD>_E_MASTER_CORE_UNINIT: This error is reported if a slave core initialization is invoked prior to initialization of the master core.
- <MOD>_E_CORE_NOT_CONFIGURED: This error is reported if the API is invoked from a core which has no hardware resource allocated.
- <MOD>_E_CORE_MISMATCH: This error is reported if initialization of a module is invoked from slave core instead of master core (applicable to modules where initialization should be invoked by master core only).

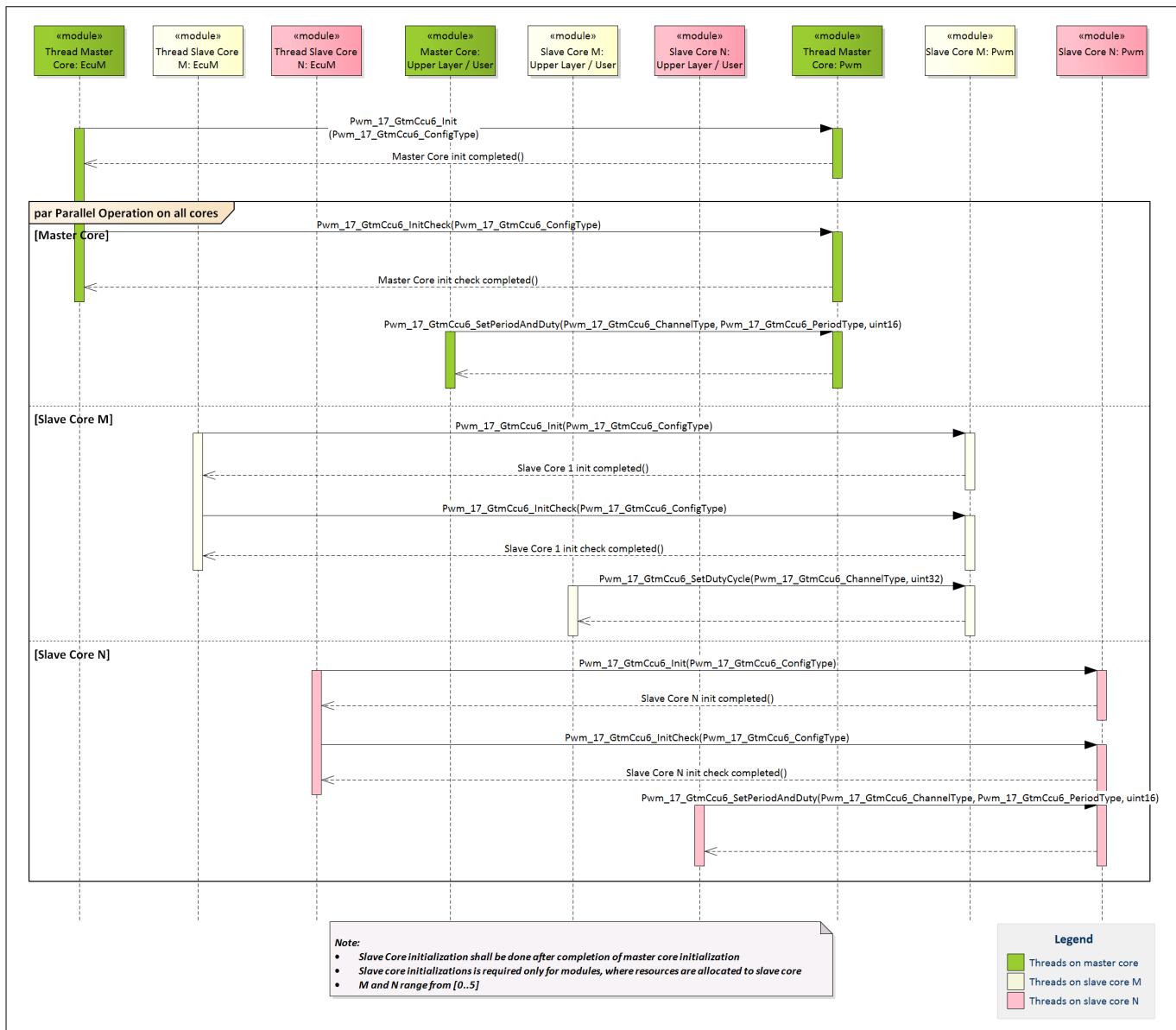
Hardware resources are partitioned between cores. Therefore, initialization should be performed by the respective CPU.

For example, if PWM initialization is called from CORE1, all the PWM channels that are allocated to CORE1 in the Resource Manager module will be initialized. If initialization is called from master core, it initializes common resources of the module for example, CLC, global status, global registers and so on along with resources allocated to it.

However, initialization from slave is required only when resources are allocated to any of the slave core.

Note: *Same configuration pointer must be passed from master and slaves cores while invoking initialization.*

The following diagram shows sample invocation of PWM driver initialization from different cores.

Generic information

Figure 31 Multicore initialization

1.2.2 Multicore de-initialization

De-initialization should be done from the context of core; each core should call the de-initialization function independently.

The following DETs are reported from de-initialization:

- <MOD>_E_SLAVE_CORE_INIT: This error is reported if de-initialization from master core is invoked while any of the slave core is still in the initialized state.
- <MOD>_E_CORE_MISMATCH: This error is reported if de-initialization of a module is invoked from the slave core instead of the master core (applicable to modules where de-initialization should be invoked by the master core only).

Hardware resources are partitioned between cores. Therefore, de-initialization should be performed by the respective CPU.

For example, if PWM de-init is called from CORE1, all the PWM channels that are allocated to CORE1 in the Resource Manager module will be de-initialized. If de-init is called from the master core, it de-initializes

Generic information

common resources of the module for example, CLC, global RAM status, global registers and so on along with resources allocated to it. Slave cores can be initialized and de-initialized indefinitely as long as the master core is in the initialized state.

However, de-initialization from slave is required only when resources are allocated to any of the slave core.

The following diagram shows sample invocation of the ADC driver de-Initialization from different cores.

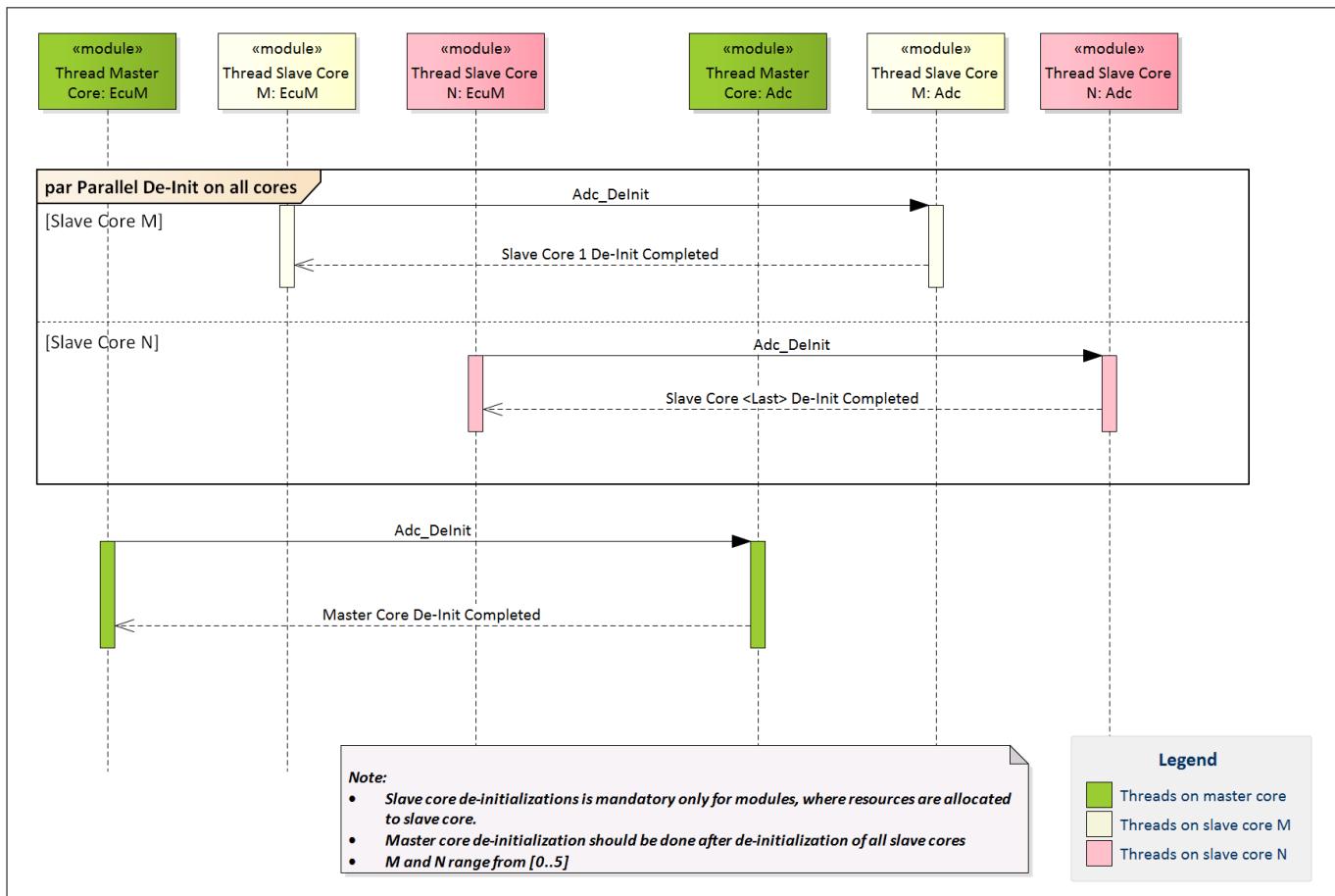


Figure 32 Multicore de-initialization

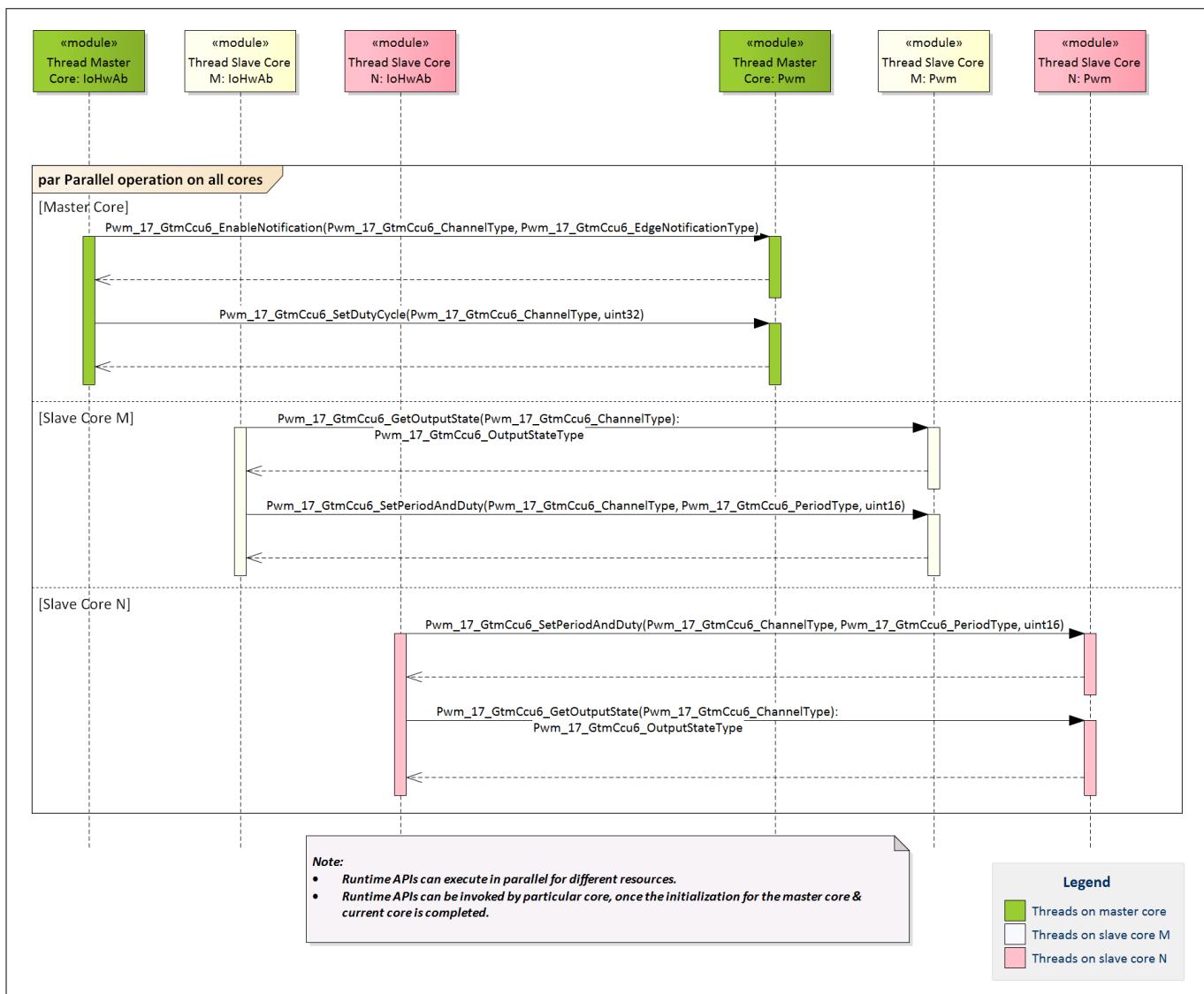
1.2.3 Multicore runtime

After initialization, runtime APIs can be called for valid channels. Error is reported if the channel passed is not configured for the core. The following diagram shows sample invocation of PWM runtime APIs from different cores.

The following DET is reported from runtime APIs:

- <MOD>_E_CORE_<RESOURCE>_MISMATCH: This error is reported if the API is invoked for an hardware resource that is not configured to be used by the current core.

During development phase such errors can be easily identified by DETs.

Generic information

Figure 33 Multicore runtime

1.2.4 Multicore interrupts

Hardware resources assigned to a CPU must have their interrupts routed to the same CPU. This must be ensured by MCAL users.

Interrupts of resources allocated to a core if serviced by different cores will require complex implementation of multiple spin locks and possibly reduce the efficiency of the software.

The interrupt service routines provided by modules are capable to be executed on several CPU cores in parallel.

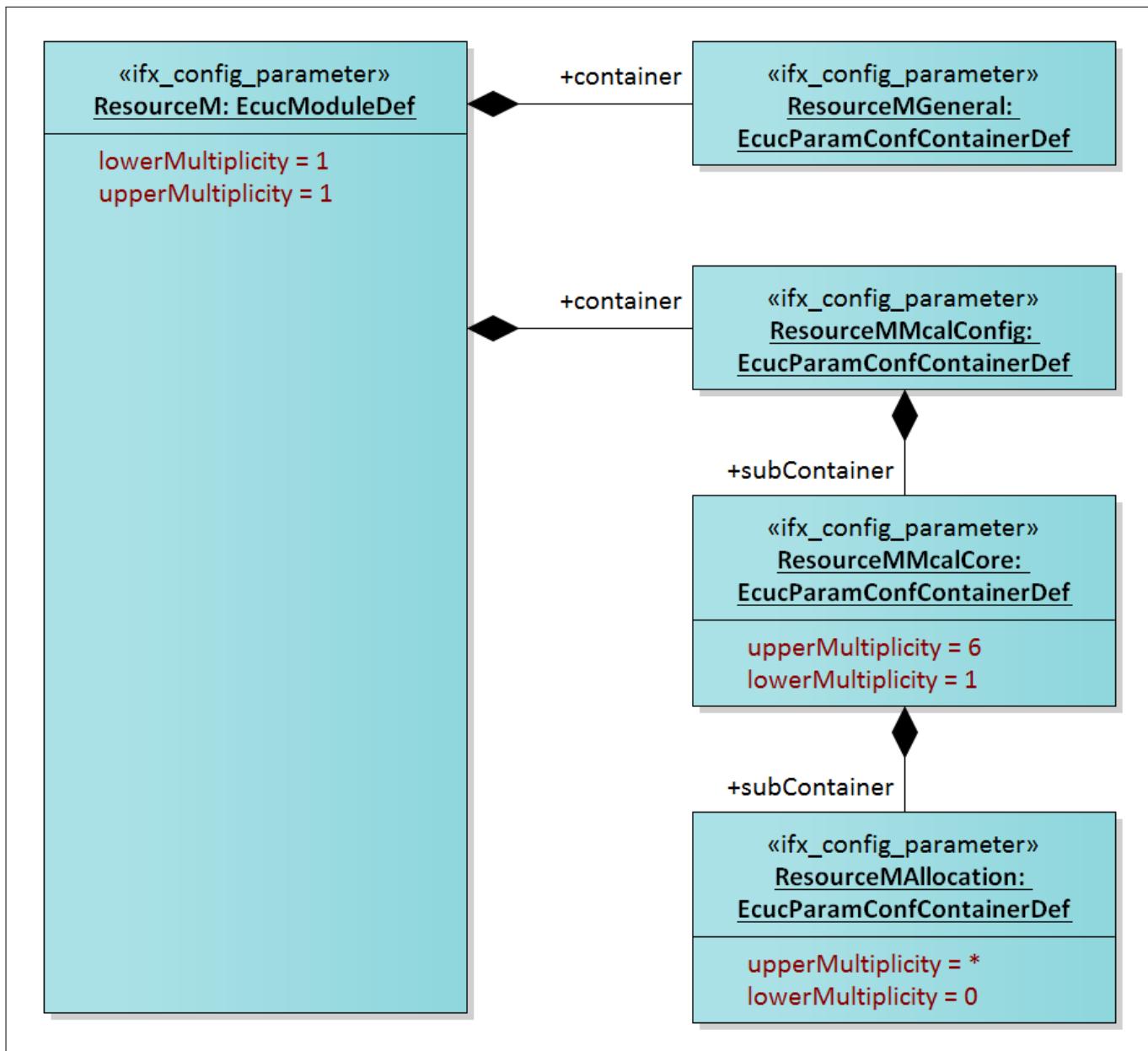
Generic information**1.2.5 Resource Manager (ResourceM)****1.2.5.1 Description**

The Resource Manager (ResourceM) module is responsible for allocating resources to cores. The unit of resource that is allocated to cores varies across modules. For example, logical PWM channels are allocated to cores for PWM module, QSPI kernels are allocated to cores in case of SPI module.

It is the responsibility of the integrator to restrict allocation of resource to single core.

The master core selection is done in the Resource Manager module. There is one designated master core in which the boot loader starts the master EcuM through EcuM_Init.

Note: *In case a hardware resource is not allocated to any core, it is by default allocated to the master core.*

Generic information
1.2.5.2 Configuration containers and parameters

Figure 34 Configuration container hierarchy
1.2.5.2.1 Container: ResourceM

The Resource Manager module is used for allocating hardware resources to the core.

1.2.5.2.2 Container: ResourceMAssignment

Container to configure resource allocations of each module assigned in the core.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

Generic information
ResourceMModuleName
Table 4 Specification for ResourceMModuleName

Name	ResourceMModuleName		
Description	This parameter is used to select module name.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC: ADC module selected CAN: CAN module selected DMA: DMA module selected ETH: ETH module selected GPT: GPT module selected ICU: ICU module selected PWM: PWM module selected OCU: OCU module selected SPI: SPI module selected STM: STM module selected SENT: SENT module selected Note: Availability of modules is based on the Release Notes.		
Default value	ADC		
Post-Build Variant Value	FALSE	Post-build variant multiplicity	-
Value Configuration Class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ResourceMResourceRef
Table 5 Specification for ResourceMResourceRef

Name	ResourceMResourceRef		
Description	This parameter contains reference to resource of the selected module.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to node		
Default value	NULL		
Post-Build Variant Value	FALSE	Post-build variant multiplicity	-
Value Configuration Class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

Generic information
Table 5 Specification for ResourceMResourceRef (continued)

Dependency	ResourceMModuleName
-------------------	---------------------

1.2.5.2.3 Container: ResourceMGeneral

General container to select sub-derivative.

ResourceMSubDerivative
Table 6 Specification for ResourceMSubDerivate

Name	ResourceMSubDerivative		
Description	Parameter to configure sub-derivates. Refer to the Hardware Manual for sub-derivate information.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NONE TC3xx: User can select any variant of TC3xx		
Default value	NONE		
Post-Build Variant Value	FALSE	Post-build variant multiplicity	-
Value Configuration Class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.2.5.2.4 Container: ResourceMMcalConfig

General container to select the master core and allocate resources.

ResourceMMasterCore
Table 7 Specification for ResourceMMasterCore

Name	ResourceMMasterCore		
Description	This parameter configures the master core.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CORE0: CORE0 is selected as master core. CORE[x]: CORE[x] is selected as master core, where [x] depends on the device.		
Default value	CORE0		
Post-Build Variant Value	FALSE	Post-build variant multiplicity	-

Generic information
Table 7 Specification for ResourceMMasterCore (continued)

Value Configuration Class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.2.5.2.5 Container: ResourceMMcalCore

Container to configure the MCAL core.

Post-Build Variant Multiplicity: FALSE

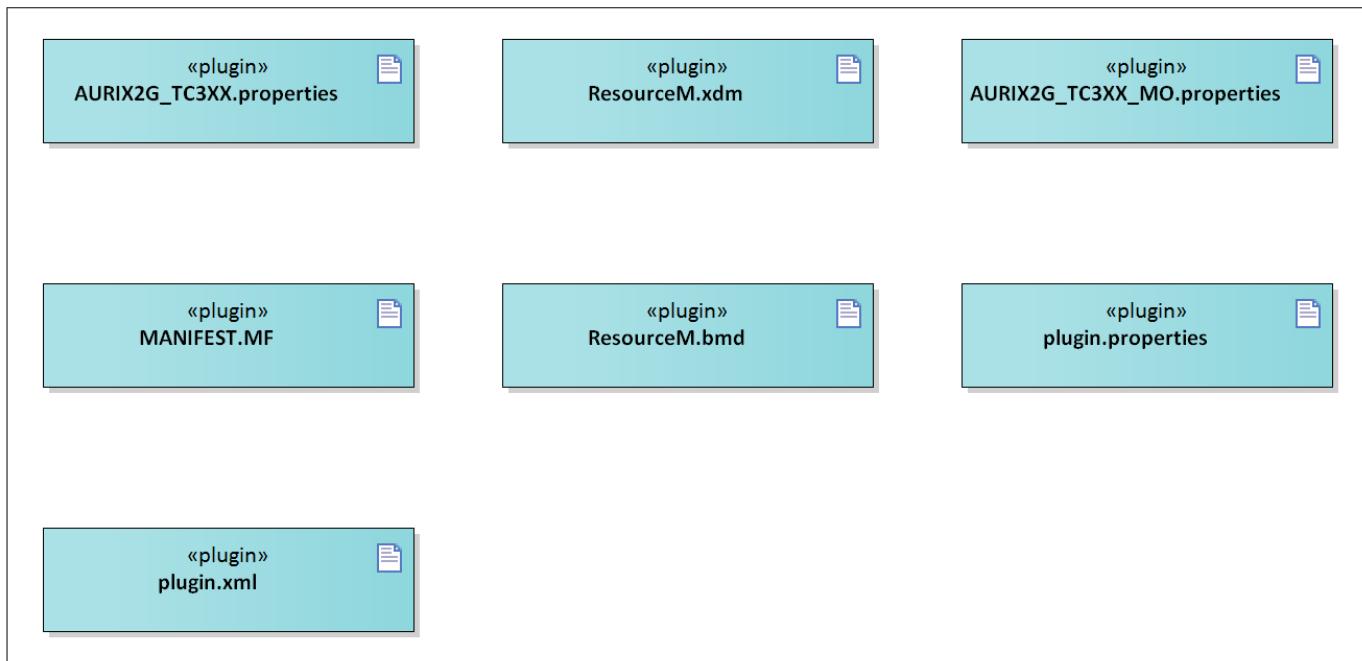
Multiplicity Configuration Class: Pre-Compile

ResourceMCoreID
Table 8 Specification for ResourceMCoreID

Name	ResourceMCoreID		
Description	Establishes a physical core ID mapping for each allocation.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CORE0: CORE0 is selected as master core. CORE[x]: CORE[x] is selected as master core, where [x] depends on the device.		
Default value	CORE0		
Post-Build Variant Value	FALSE	Post-build variant multiplicity	-
Value Configuration Class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.2.5.3 Code generator plugin files

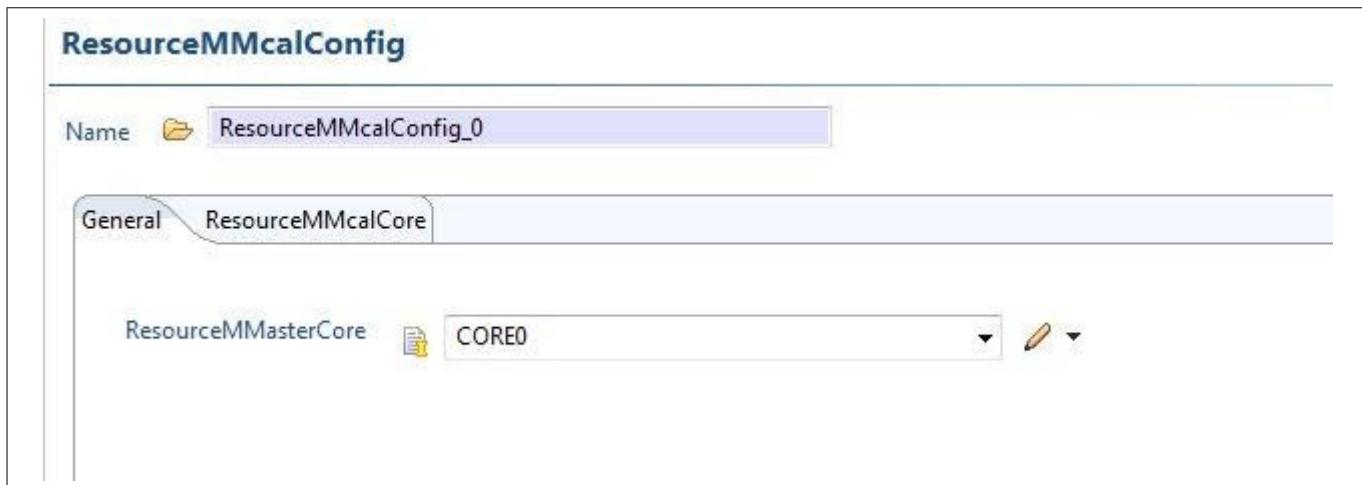
This section provides details of the code generator plugin files for the Resource Manager.

Generic information

Figure 35 **Code generator plug-in files**
Table 9 **Code generator plug-in files**

File name	Description
AURIX2G_TC3XX.properties	This file contains the variables defined for handling hardware derivates. Here, TC3XX = Derivatives supported as per the release notes.
AURIX2G_TC3XX_MO.properties	This file contains the variables defined for handling hardware sub-derivates. Here, TC3XX = Derivatives supported as per the release notes. Marking Option (MO) = based on defined marking options
MANIFEST.MF	Tresos plugin support file containing the metadata for the Resource Manager
ResourceM.bmd	AUTOSAR format XML data model schema file (for each device)
ResourceM.xdm	Tresos format XML data model schema file
plugin.properties	Tresos plugin support file for the Resource Manager
plugin.xml	Tresos plugin support file for the Resource Manager

1.2.5.4 Example usage
1.2.5.4.1 Configuring master core

CORE0 is selected as master core.

Generic information

Figure 36 **Selecting master core**
1.2.5.4.2 Configuring cores

CORE0 and CORE1 configured for resource allocation.

ResourceMMcalConfig												
General ResourceMMcalCore												
<table border="1"> <thead> <tr> <th>Index</th><th>Name</th><th>Resource...</th></tr> </thead> <tbody> <tr> <td>0</td><td>ResourceMMcalCore_0</td><td>CORE0</td></tr> <tr> <td>1</td><td>ResourceMMcalCore_1</td><td>CORE1</td></tr> </tbody> </table>				Index	Name	Resource...	0	ResourceMMcalCore_0	CORE0	1	ResourceMMcalCore_1	CORE1
Index	Name	Resource...										
0	ResourceMMcalCore_0	CORE0										
1	ResourceMMcalCore_1	CORE1										

Figure 37 **Resource allocation**
1.2.5.4.3 Configuring resources to core

In the following figure Pwm_Channel0, Pwm_Channel1 and Pwm_Channel2 are allocated to CORE0.

Generic information

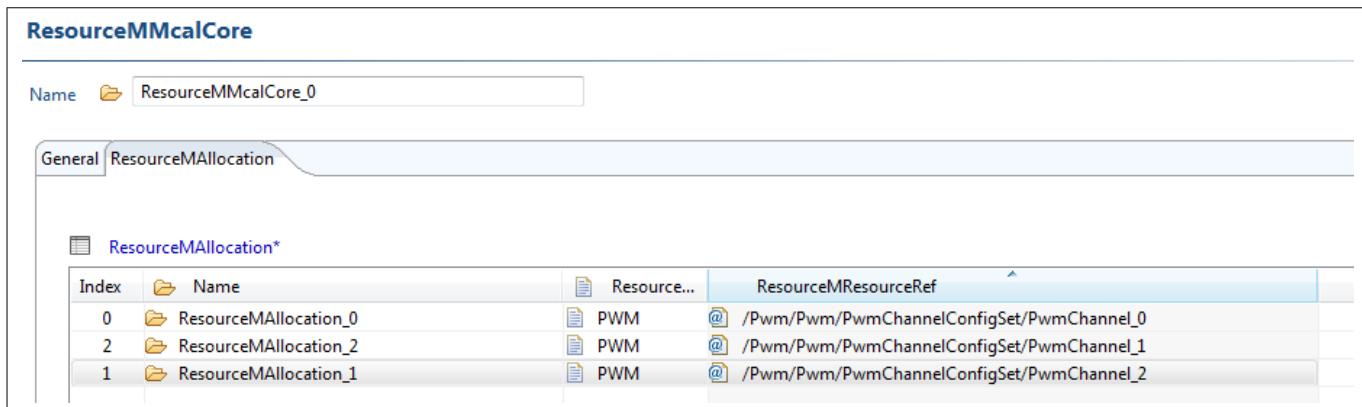


Figure 38 Configuring resources

Similarly Pwm_Channel3 and Pwm_Channel4 are allocated to CORE1.

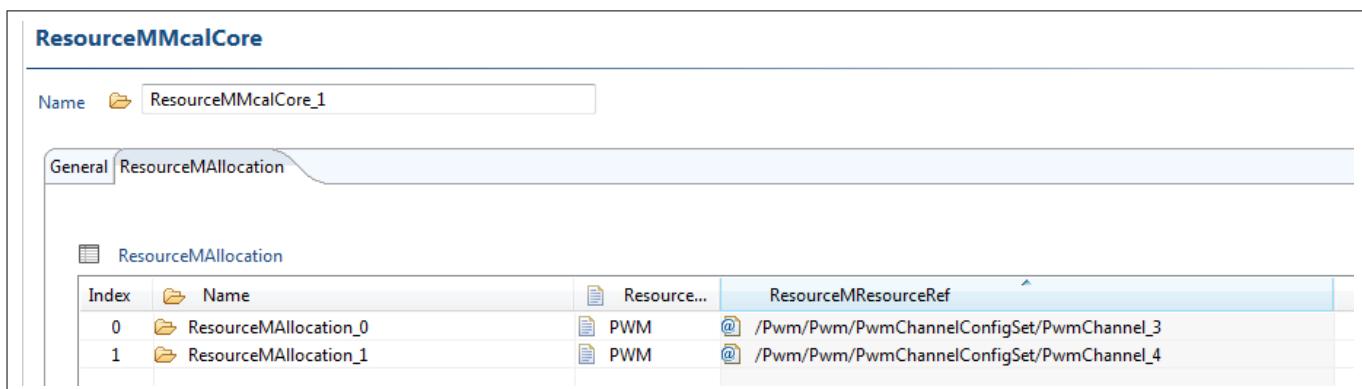


Figure 39 PWM allocation

1.3 User mode execution

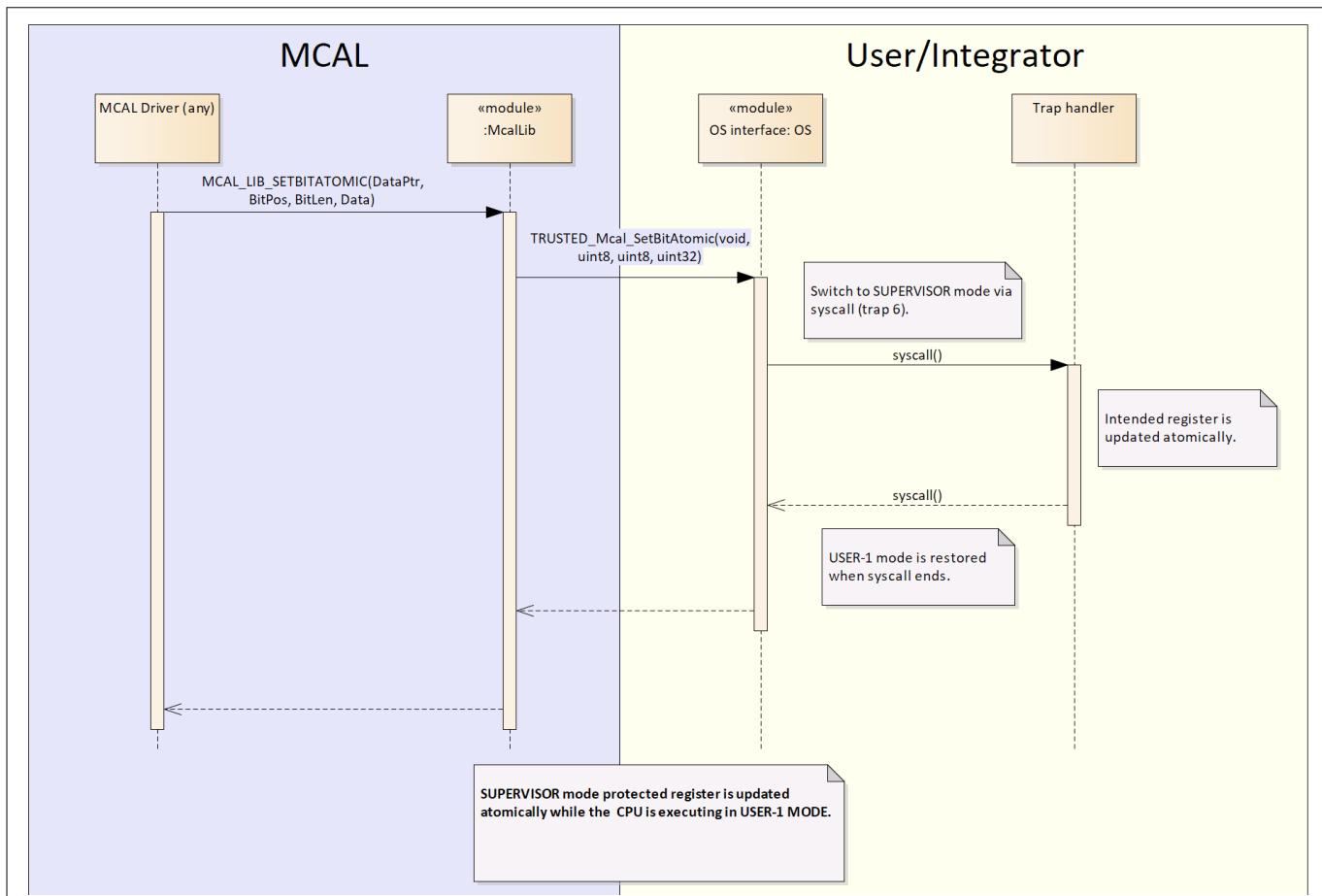
The TriCore™ CPU can execute in three privilege modes: User-0, User-1 and Supervisor modes. As SFR access is not possible in the User-0 mode, the MCAL drivers can be executed in the User-1 or Supervisor modes only. Each MCAL driver provides pre-compile configuration parameters to configure the execution mode of the CPU while executing Init/De-Init and runtime APIs independently.

Note: If a MCAL driver is not accessing any SFR which can be updated in the Supervisor mode only, then the configuration of execution mode for that MCAL driver may not be available

Examples of updating registers when CPU is in the User-1 and Supervisor modes are depicted in the following sub-sections.

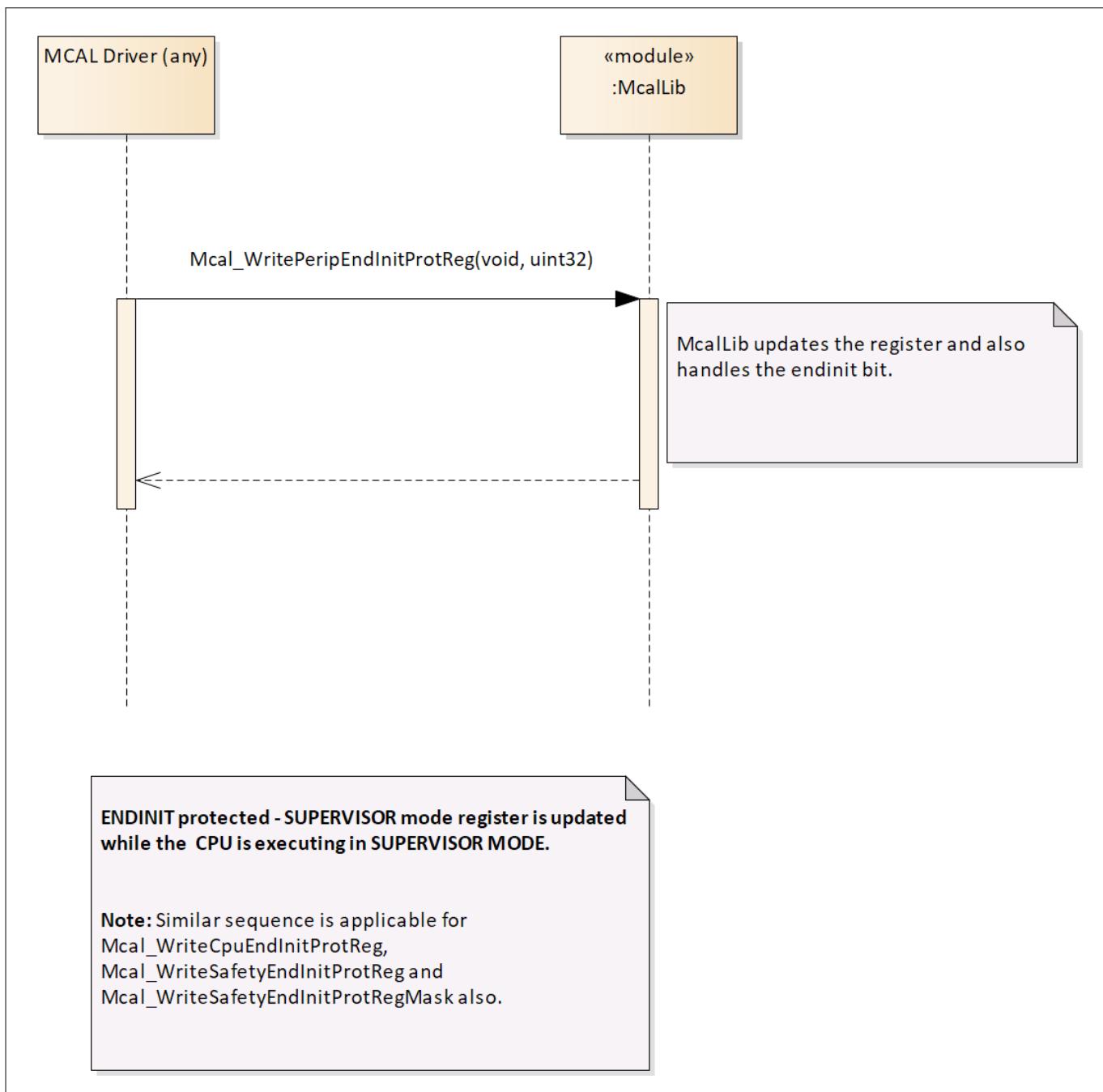
1.3.1 Atomic update of the Supervisor mode register when CPU is in the User-1 mode

The following sequence diagram depicts the use case of updating a Supervisor mode protected register atomically while the CPU is executing in the User-1 mode. The application software or OS is responsible to implement the redirected function call from MCALLIB.

Generic information

Figure 40 Atomic update of register when CPU is in the User-1 mode

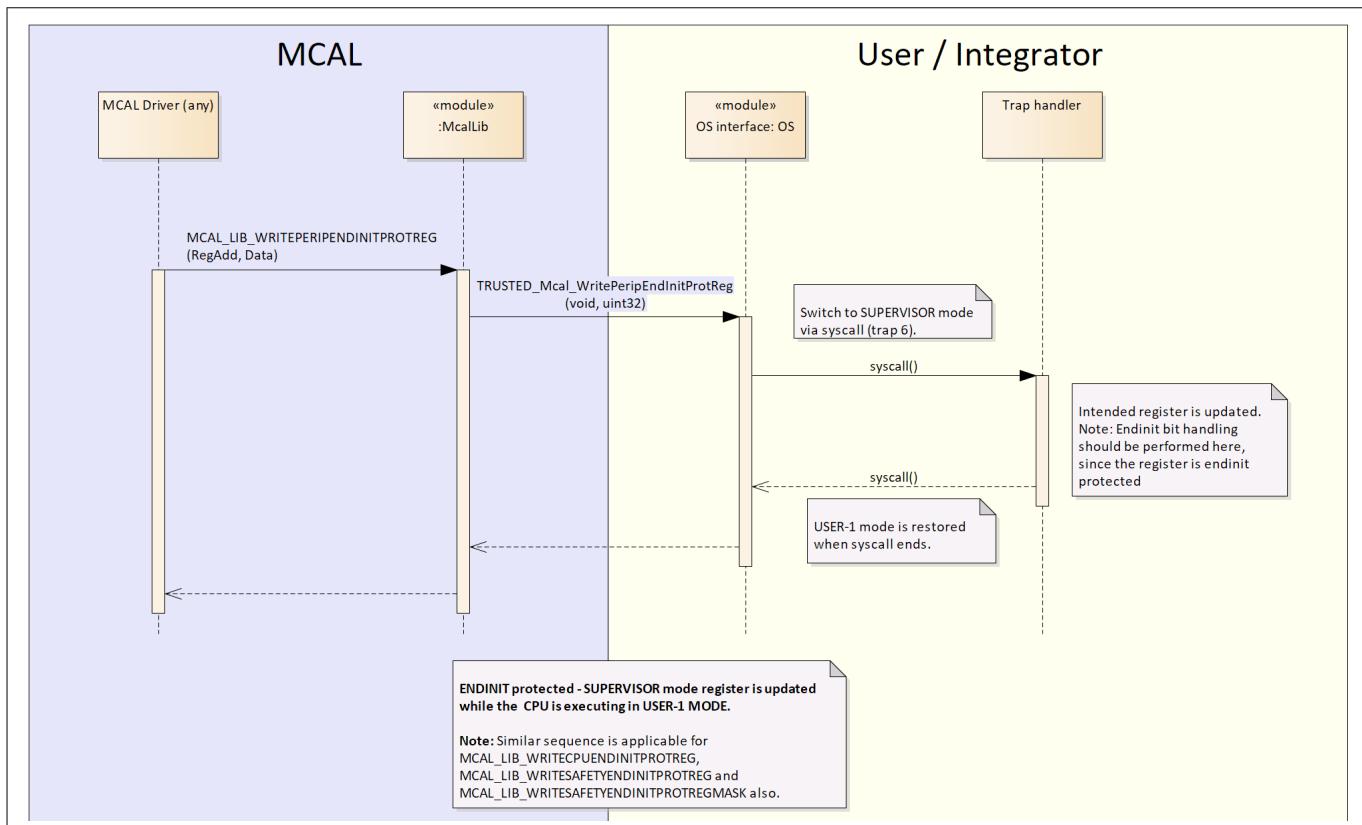
1.3.2 Protected Supervisor mode register update when CPU is in the Supervisor mode

The following sequence diagram depicts the use case of updating an EndInit protected Supervisor mode register while the CPU is executing in the Supervisor mode. In this case, the driver API calls the `Mcal_WritePeripEndInitProtReg()` MCALLIB driver function. This API handles the EndInit protection and updates the protected register as well.

Generic information

Figure 41 EndInit protected register update when CPU is in the Supervisor mode

1.3.3 Protected Supervisor mode register update when CPU is in the User-1 mode

The following sequence diagram depicts the use case of updating an ENDINIT protected Supervisor mode register while the CPU is executing in the User-1 mode. The application software or OS is responsible to implement the redirected function call from MCALLIB.

Generic information

Figure 42 **EndInit protected register update when CPU is in the User-1 mode**

1.3.4 Read Supervisor mode register when CPU is in the User-1 mode

The following sequence diagram depicts the use case of reading a Supervisor mode protected register while the CPU is executing in the User-1 mode. The application software or OS is responsible to implement the redirected function call from MCALLIB.

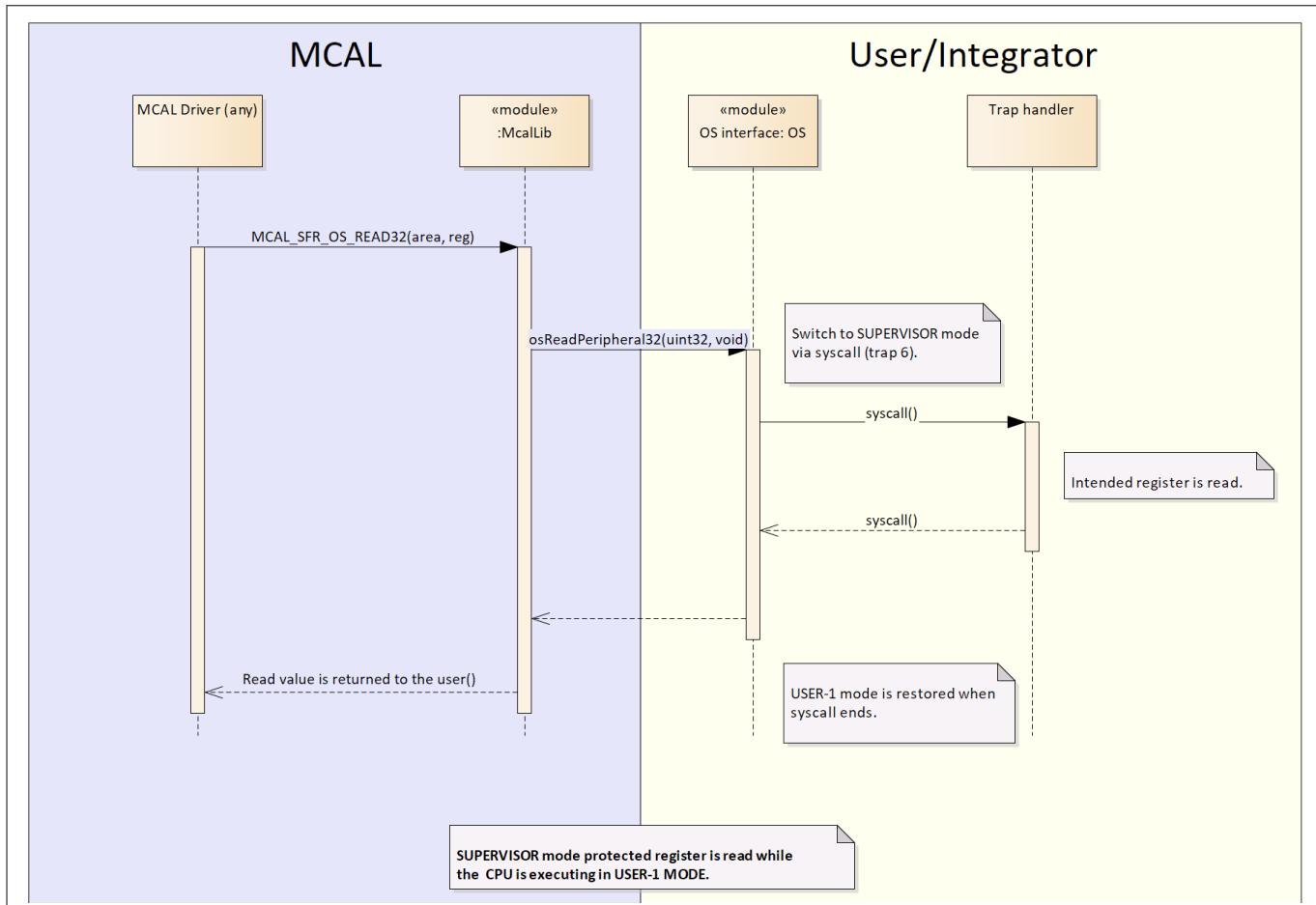
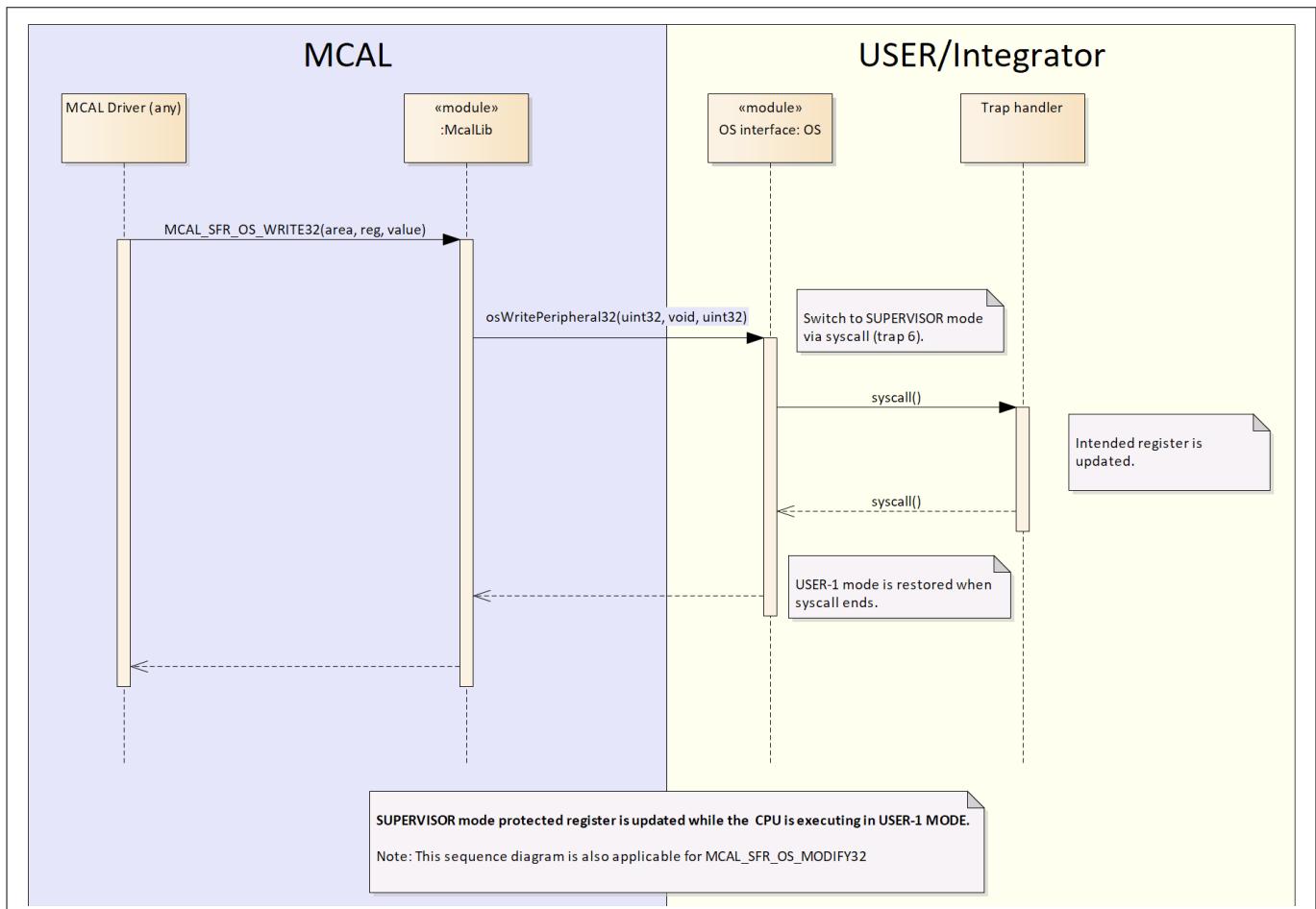
Generic information


Figure 43 Register read when CPU is in the User-1 mode

1.3.5 Write to Supervisor mode register when CPU is in the User-1 mode

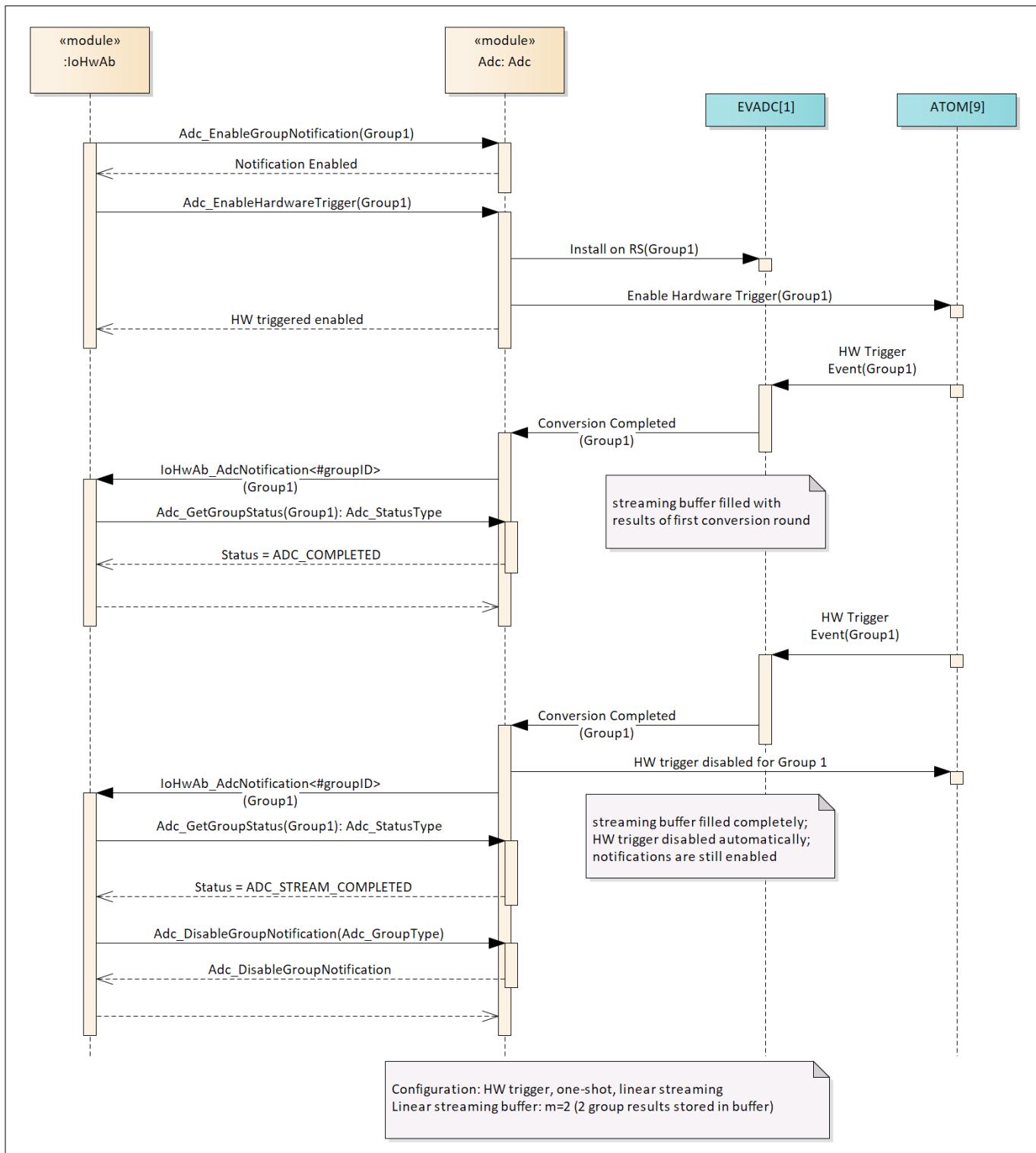
The following sequence diagram depicts the use case of updating a Supervisor mode protected register while the CPU is executing in the User-1 mode. The application software or OS is responsible to implement the redirected function call from MCALLIB.

Generic information

Figure 44 Register write when CPU is in the User-1 mode

1.4 Execution context of functions

The MCAL drivers provide both API functions and interrupt handlers. In general, the execution context for APIs shall be task and for interrupt handlers shall be interrupt. However, it is possible to invoke API functions in the context of the notification function invoked by the interrupt handler. An example to depict such a use-case is shown in the following diagram.

Note: The execution context for each API is available in the `Mod_Bswmd.arxml` file.

Generic information

Figure 45 Execution of API in interrupt context
1.5
Memory mapping

All the memory sections belong to the MemMap header file of the respective modules. The memory mapping section name for each module is derived as described in the subsequent sections. The user must use appropriate compiler pragmas in the `Mod_MemMap.h` header file to ensure correct allocation in memory space.

Generic information

Note: The naming convention for all memory sections is as per AUTOSAR, any deviation is captured separately in the module sections. All the memory sections used by the module are also present in the `<Module>_Bswmd.arxml` file under the element MEMORY-SECTIONS. User must refer to the EB tresos generated `<Module>_Bswmd.arxml` file for a complete list of memory sections per module.

1.5.1 Memory mapping for code

The memory mapping for code is as follows:

- Start Section: [SNP]_<VI>_<AI>_START_SEC<_username>_CODE<_mode>[_safety][_coreScope]
- Stop Section: [SNP]_<VI>_<AI>_STOP_SEC<_username>_CODE<_mode>[_safety][_coreScope]

Here,

[] = Indicates a mandatory field

<> = Indicates an optional field

SNP = Module abbreviation

VI = vendorId of the BSW module (applicable only for modules with upper multiplicity greater than 1). For software provided by Infineon, this value is fixed to 17.

AI = vendorApiInfix of the BSW module (applicable only for modules with upper multiplicity greater than 1)

safety = QM, ASIL_B

coreScope = GLOBAL, CORE0, CORE1, CORE2, CORE3, CORE4, CORE5, LOCAL (used for modules not supporting multicore)

username = this is an optional field and may be used for functions in code which are required to be placed in specific memory regions

mode = FAST, SLOW

Note: The memory sections of this category should be mapped to .text in linker command file.

1.5.2 Memory mapping for callout code

The memory mapping for callout code is as follows:

- Start Section: [SNP]_<VI>_<AI>_START_SEC[_username]_CALLOUT_CODE<_mode>[_safety][_coreScope]
- Stop Section: [SNP]_<VI>_<AI>_STOP_SEC[_username]_CALLOUT_CODE<_mode>[_safety][_coreScope]

Here,

[] = Indicates a mandatory field

<> = Indicates an optional field

SNP = Module abbreviation

VI = vendorId of the BSW module (applicable only for modules with upper multiplicity greater than 1). For software provided by Infineon, this value is fixed to 17.

AI = vendorApiInfix of the BSW module (applicable only for modules with upper multiplicity greater than 1)

username = The specific username for each callout function must be specified.

safety = QM, ASIL_B

coreScope = GLOBAL, CORE0, CORE1, CORE2, CORE3, CORE4, CORE5, LOCAL (used for modules not supporting multicore)

mode = FAST, SLOW

Note: The memory sections of this category should be mapped to .text in linker command file.

Generic information

1.5.3 Memory mapping for configuration data

The memory mapping for configuration data is as follows:

- Start Section: [SNP]_<VI>_<AI>_START_SEC_CONFIG_DATA[_safety][_coreScope][_alignment]
- Stop Section: [SNP]_<VI>_<AI>_STOP_SEC_CONFIG_DATA[_safety][_coreScope][_alignment]

Here,

[] = Indicates a mandatory field

<> = Indicates an optional field

SNP = Module abbreviation

VI = vendorId of the BSW module (applicable only for modules with upper multiplicity greater than 1). For software provided by Infineon, this value is fixed to 17.

AI = vendorApiInfix of the BSW module (applicable only for modules with upper multiplicity greater than 1)

safety = QM, ASIL_B

coreScope = GLOBAL, CORE0, CORE1, CORE2, CORE3, CORE4, CORE5, LOCAL (used for modules not supporting multicore)

alignment = UNSPECIFIED, 8, 16, 32 (in bits). In cases, where a higher order memory alignment is required it should be specified at the bit level. For example, 256 is to be used for a 32-byte alignment.

Note: *The memory sections of this category should be mapped to .rodata in linker command file.*

1.5.4 Memory mapping for constants

The memory mapping for constant data is as follows:

- Start Section: [SNP]_<VI>_<AI>_START_SEC_CONST[_safety][_coreScope][_alignment]
- Stop Section: [SNP]_<VI>_<AI>_STOP_SEC_CONST[_safety][_coreScope][_alignment]

Here,

[] = Indicates a mandatory field

<> = Indicates an optional field

SNP = Module abbreviation

VI = vendorId of the BSW module (applicable only for modules with upper multiplicity greater than 1). For software provided by Infineon, this value is fixed to 17.

AI = vendorApiInfix of the BSW module (applicable only for modules with upper multiplicity greater than 1)

safety = QM, ASIL_B

coreScope = GLOBAL, CORE0, CORE1, CORE2, CORE3, CORE4, CORE5, LOCAL (used for modules not supporting multicore)

alignment = UNSPECIFIED, 8, 16, 32 (in bits). In cases, where a higher order memory alignment is required it should be specified at the bit level. For example, 256 is to be used for a 32-byte alignment.

Note: *The memory sections of this category should be mapped to .rodata in linker command file.*

1.5.5 Memory mapping for variables

The memory mapping for variable is as follows:

- Start Section: [SNP]_<VI>_<AI>_START_SEC_VAR<_mode>[_initPolicy][_safety][_coreScope][_alignment]
- Stop Section: [SNP]_<VI>_<AI>_STOP_SEC_VAR<_mode>[_initPolicy][_safety][_coreScope][_alignment]

Generic information

Here,

[] = Indicates a mandatory field

<> = Indicates an optional field

SNP = Module abbreviation

VI = vendorId of the BSW module (applicable only for modules with upper multiplicity greater than 1). For software provided by Infineon, this value is fixed to 17.

AI = vendorApiInfix of the BSW module (applicable only for modules with upper multiplicity greater than 1)

initPolicy = CLEARED, INIT

safety = QM, ASIL_B

coreScope = GLOBAL, CORE0, CORE1, CORE2, CORE3, CORE4, CORE5, LOCAL (used for modules not supporting multicore)

alignment = UNSPECIFIED, 8, 16, 32 (in bits). In cases, where a higher order memory alignment is required it should be specified at the bit level. For example, 256 is to be used for a 32-byte alignment.

mode = FAST, SLOW

Note: *The memory sections of this category with initPolicy = CLEARED should be mapped to .bss in linker command file. If the mode is explicitly set to FAST in the memory section then, it shall be mapped to .zbss.*

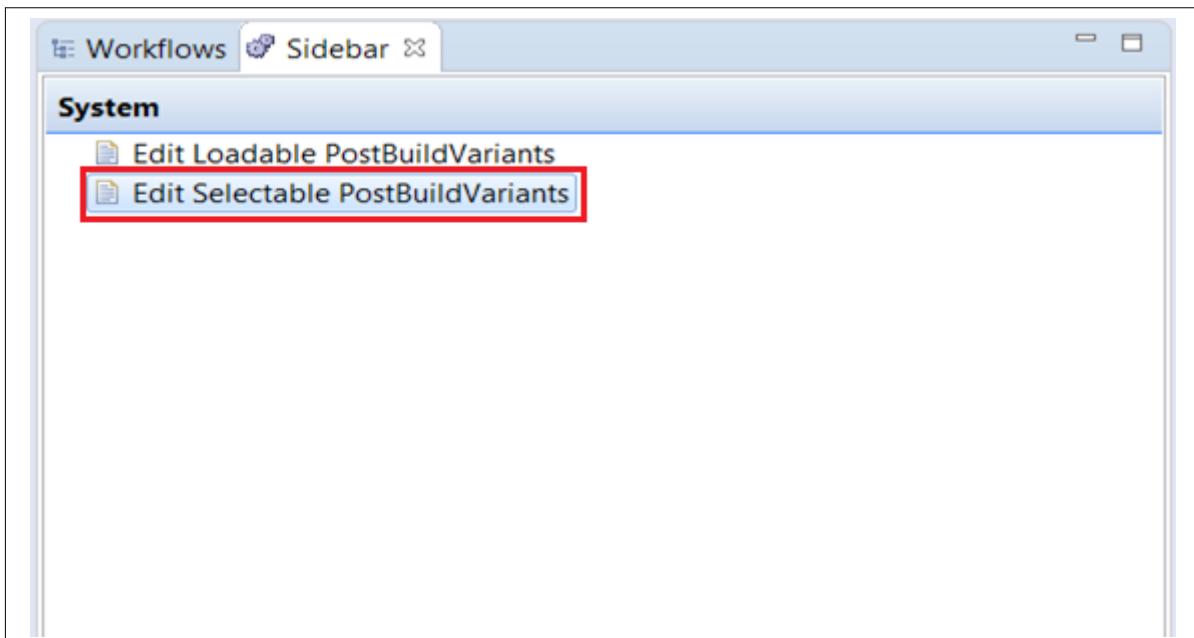
Note: *The memory sections of this category with initPolicy = INIT should be mapped to .data in linker command file. If the mode is explicitly set to FAST in the memory section then, it shall be mapped to .zdata.*

1.6 Variation point support

1.6.1 Configuring the Sidebar view

The procedure to prepare variation point supported EB tresos configuration project is as follows:

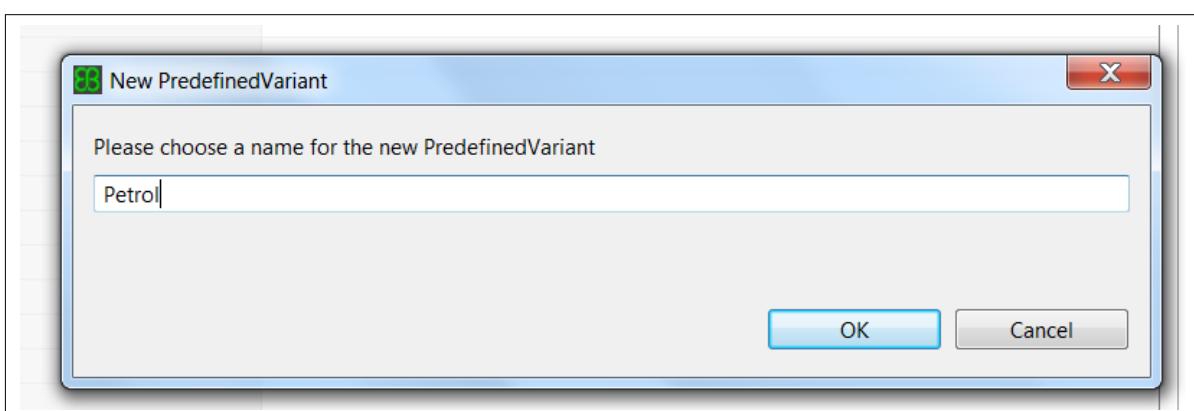
1. Start the EB tresos GUI and create EB tresos configuration project with the required modules along with that add the EcuC module in the project.
2. In the EB tresos GUI menu, select **Window > Show View > Sidebar**.

Generic information**Figure 46 PostBuildVariants configuration selection**

3. Double-click the **Edit Selectable PostBuildVariants** option.

**Figure 47 Adding predefined variants**

4. In the **Predefined Variants** tab, add a new predefined variants by clicking the plus (+) icon.

**Figure 48 New Predefined variant**

5. Specify a name for the new PredefinedVariant. For example, Petrol.

Generic information

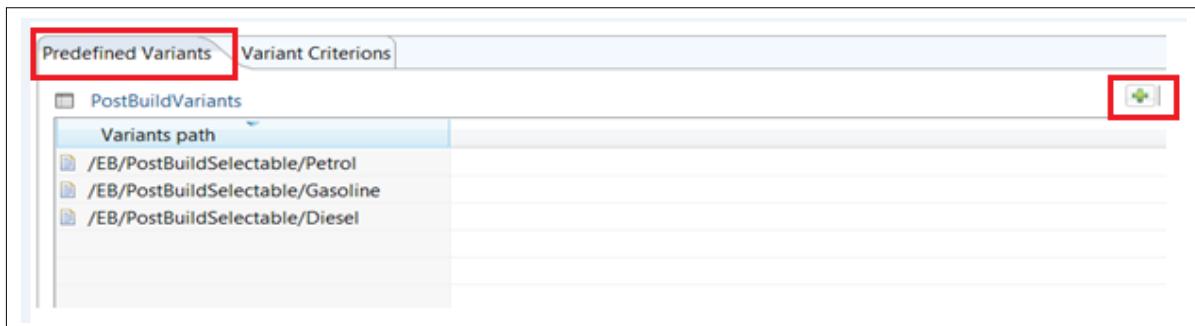


Figure 49 Predefined variants addition

6. Add the required number of variants in the **Predefined Variants** tab. For example, in the above figure, the following predefined variants are added: Petrol, Gasoline and Diesel.

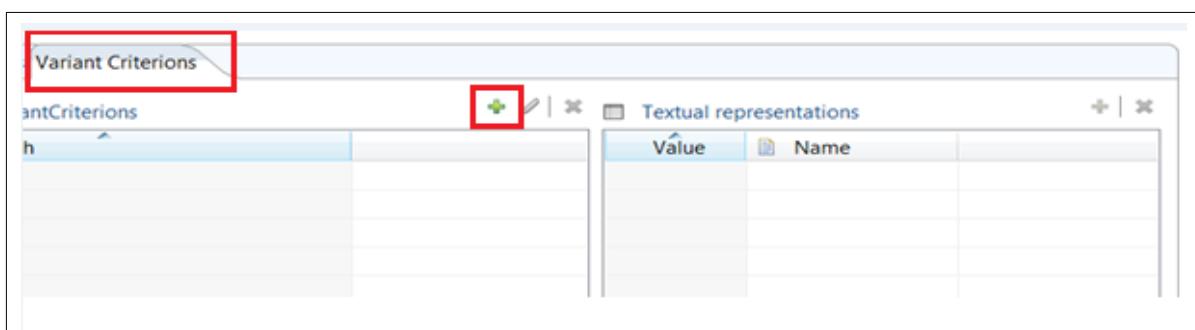


Figure 50 Variant criterions addition

7. In the **Variant Criterions** tab, click the plus (+) icon to add a new criterion.

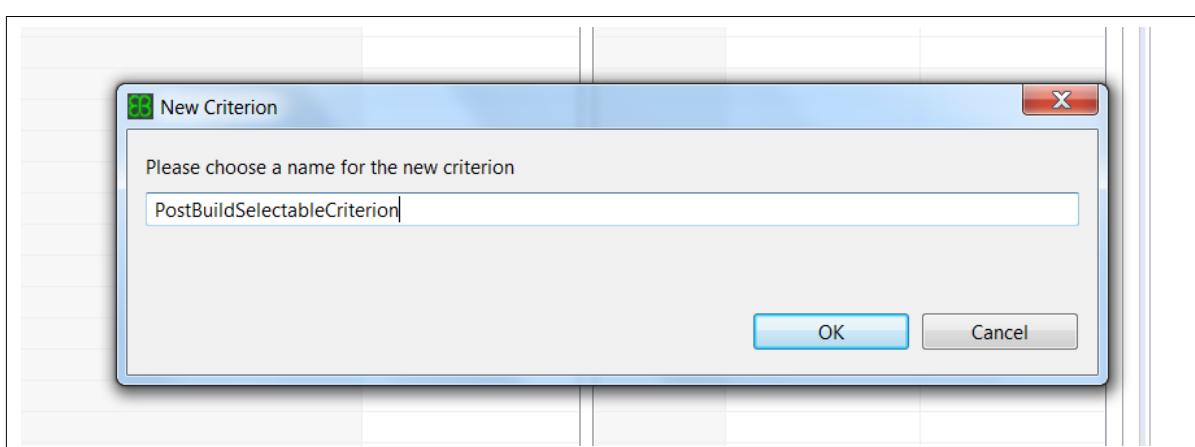


Figure 51 New criterion

8. In the **New Criterion** pop-up, specify a name for the new criterion. For example, PostBuildSelectableCriterion.

Generic information

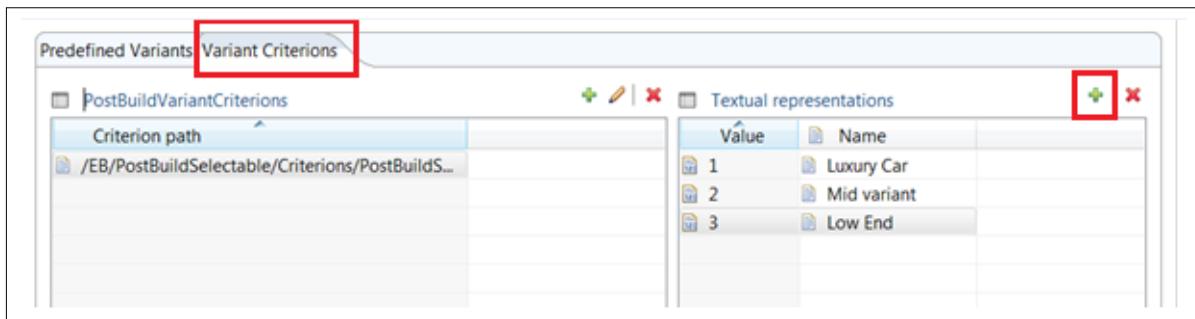


Figure 52 Variant criterions

9. In the **Variant Criterions** tab, select the Criterion path. This will enable the **Textual representations** options.
10. In the **Textual representations** pane, click the plus (+) icon to add and edit the *Value* and *Name* fields.
Value: <Userdefined_numericvalue>. Specify a value, for example, 1, 2, 3
Name: <Userdefined_Name>. Specify a name, for example, Luxury Car, Mid variant, Low End

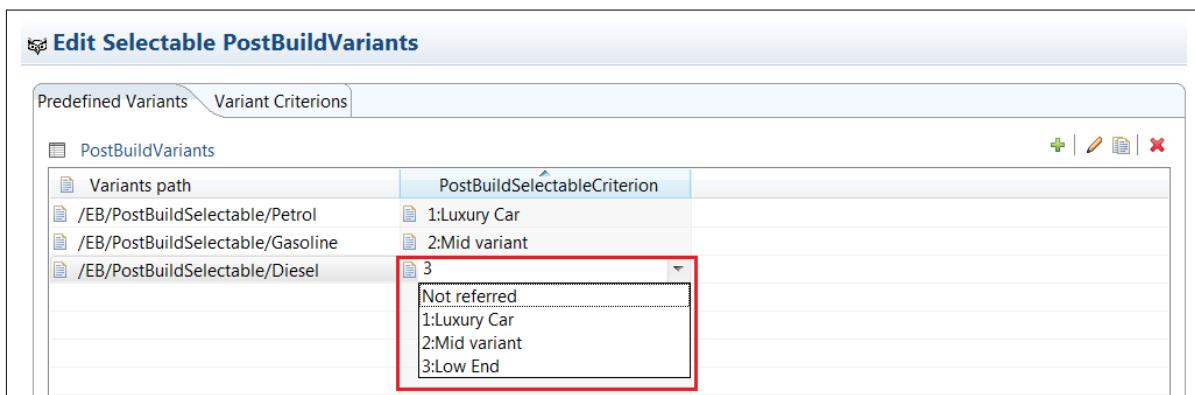


Figure 53 Edit variants

11. In the **Predefined Variants** tab, assign the selectable criterion for each PostBuildVariants by clicking the field and selecting a value from the drop-down list.
12. Close the **Edit Selectable PostBuild Variants** tab. If you keep the **Edit Selectable PostBuild Variants** tab opened, other module configurations cannot be done in the editor window.

1.6.2 Configuring EcuC module

The procedure to set up the EcuC module configuration is as follows:

1. In the **Project Explorer** window, double-click the EcuC module for the selected project.

Generic information

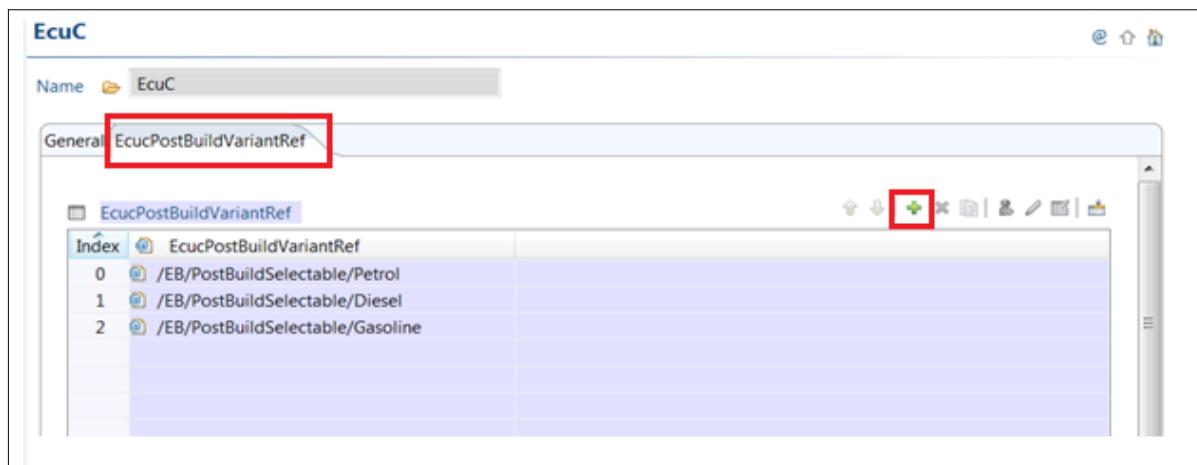


Figure 54 Selecting predefined variants

2. Add post-build variant references from the in the EcucPostBuildVariants/**EcucPostBuildVariantsRef** container.



Figure 55 Enable parameter

3. To enable the **EcuCSelectedPostBuildVariantRef** optional parameter, click the Red icon.

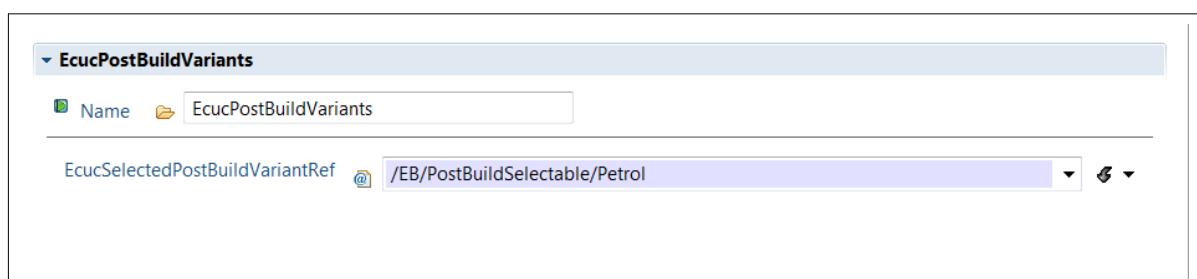


Figure 56 Selecting currently activated variant

4. Select a variant, you want to configure, from the drop-down list.
 - *Option (i):* Select the currently working activated variant Petrol from the **EcucSelectedPostBuildVariantRef** reference parameter.

Generic information

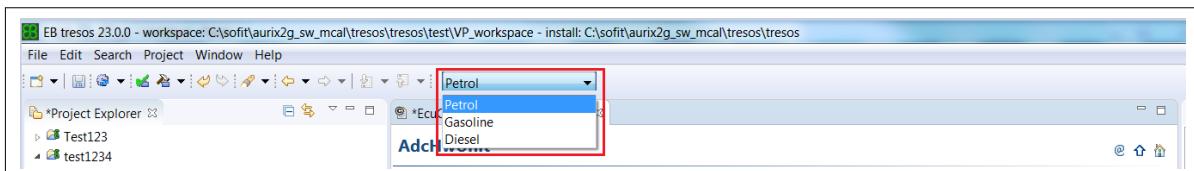


Figure 57 Selecting variant

- *Option (ii):* Select the currently working activated variant Petrol from the EB tresos menu bar. You can choose either of the options.

1.6.3 Configuring modules

The following module configuration procedure is for the Adc module, specifically for the AdcCalibrationSampleTime parameter. For other modules, follow the similar steps to configure the variation point supported containers/parameters.

The procedure to configure the variation point supported modules is as follows:

1. In the **Project Explorer** window, double-click a post-build module, for example, Adc module.
2. In the Adc module, select the AdcCalibrationSampleTime parameter (full path: AdcConfigSet/AdcHwUnit/AdcCalibrationSampleTime) and configure values for different variants.

Variant 1: Petrol variant configuration



Figure 58 Parameter configuration

3. Select the AdcCalibrationSampleTime parameter value from the drop-down list. For example, ADC_CAL_TIME_2_TIMES_TADCI.
4. Right-click the AdcCalibrationSampleTime parameter by clicking the icon.

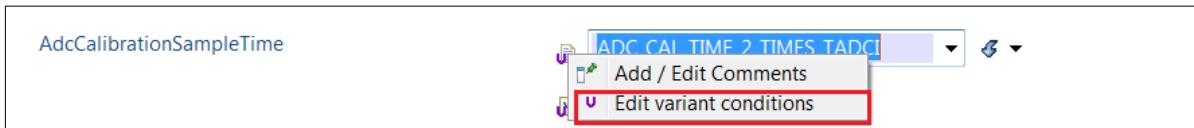


Figure 59 Parameter variant condition settings

5. Select the Edit variant conditions option.

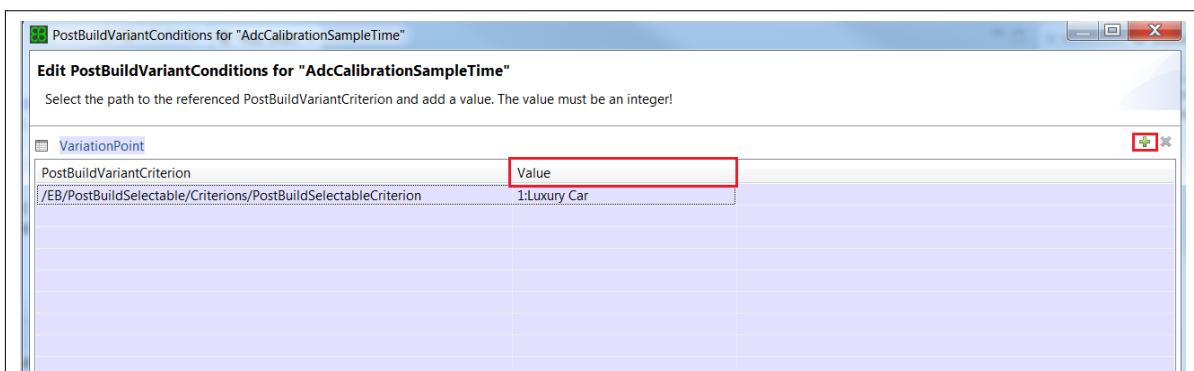


Figure 60 Selecting variant criterions

6. Click the plus (+) icon.
7. Select the PostBuildVariantCriterion from the drop-down list.

Generic information

8. Select a value from the drop-down list for the selected variant (which is already created and assigned, refer to *Section 1.5.1*).

Repeat all the steps 1 through 8 to configure other post-build variant parameters (with POSTBUILDVARIANTVALUE = true).

Variant 2: Diesel variant configuration

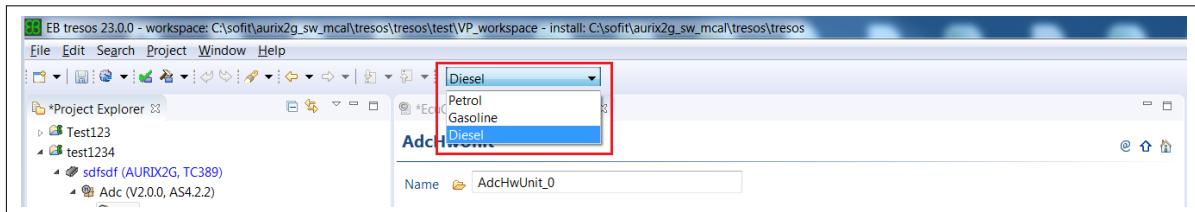


Figure 61 Selecting variants

9. Select the **Diesel** option in the EcuC/EcucPostBuildVariants/**EcucPostBuildVariantRef** parameter or from the EB tresos tool menu.

Note: When you are configuring for the next variant Diesel configuration, the AdcCalibrationSampleTime parameter value can be changed to a different value (for example, ADC_CAL_TIME_4_TIMES_TADC1).

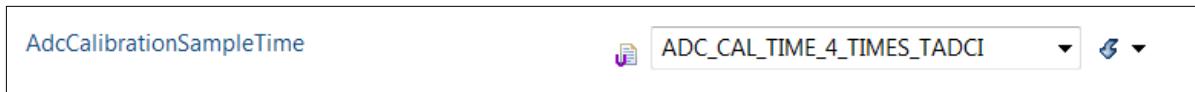


Figure 62 Parameter configuration

10. Right-click the AdcCalibrationSampleTime parameter **v** icon.

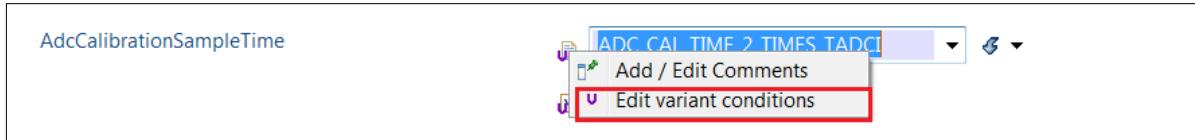


Figure 63 Parameter variant condition settings

11. Select the **Edit variant conditions** from the drop-down list.

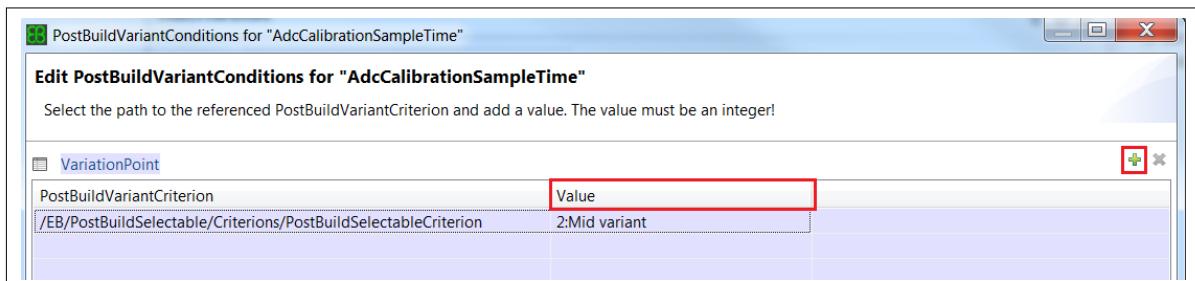


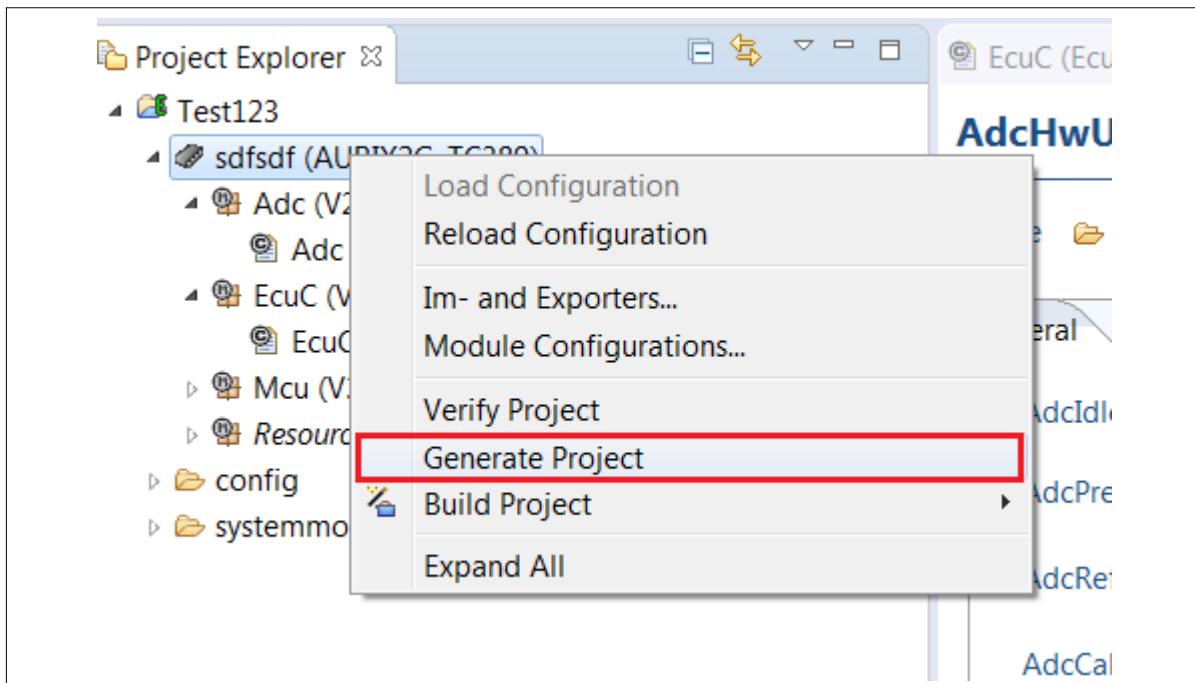
Figure 64 Selecting variant criterions

12. Select a value from the drop-down list for the selected variant (which is already created and assigned, refer to *Section 1.5.1*).

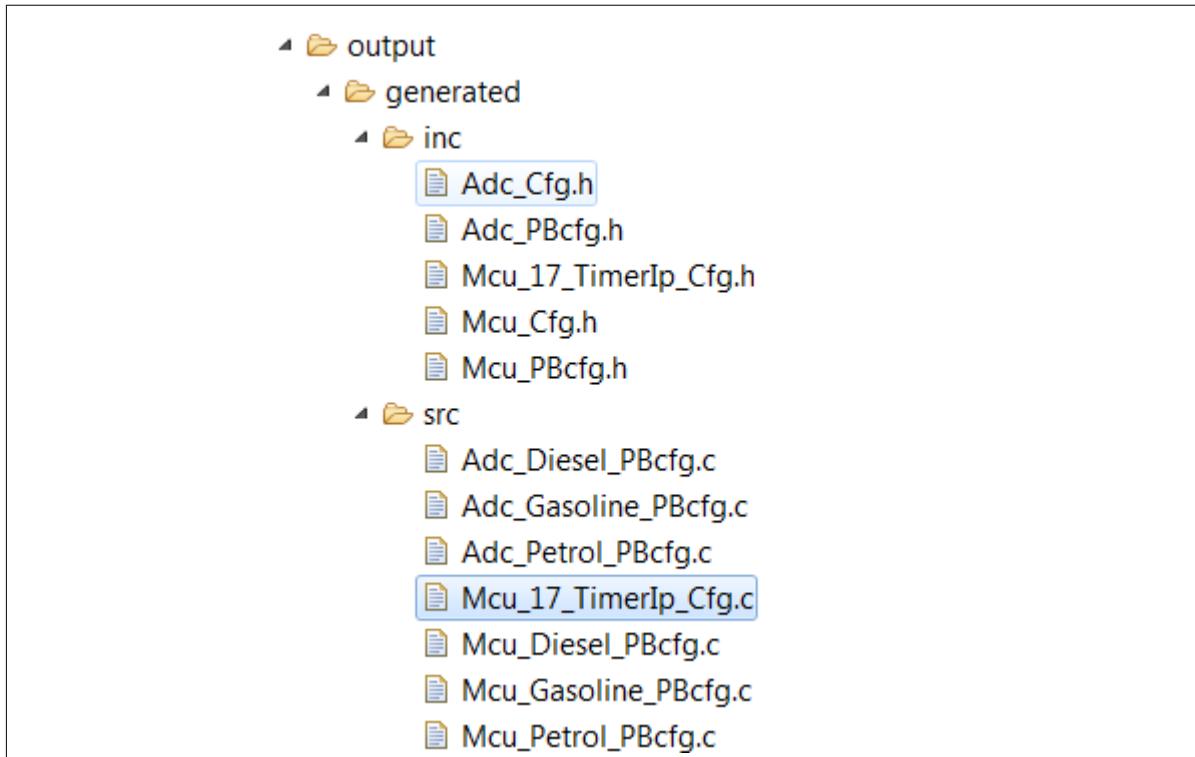
Similarly, you can configure parameter values for the Gasoline variant.

Note: Ensure to map the correct PostBuildVariantCriterion value for the selected variant.

Repeat all the steps 9 through 12 to configure other post-build variant parameters (with POSTBUILDVARIANTVALUE = true).

Generic information

Figure 65 Generate project

13. To generate the output files for all the variants, click the **Generate Project** option. The generated files are displayed in the **Project Explorer** window.


Figure 66 Generated output

1.6.4 Post-build variant multiplicity and Multiplicity configuration class

In general, the drivers provide configuration for multiple channels, groups, blocks, and so on (for example, ADC driver provides AdcGroup container, SPI driver provides SpiJob container). While configuring such containers

Generic information

the user must take into account the post-build variant multiplicity and multiplicity configuration class of such containers.

The user must consider the following points:

- If the post-build variant multiplicity is set to FALSE for a container, then the number of instances of such container cannot change across the post-build variants.
- If the multiplicity configuration class is set to Pre-compile, then the number of instances of such container cannot change across post-build configurations.

For details on the post-build variant multiplicity and multiplicity configuration class of such containers, refer to the *Configuration interfaces* section for each driver in this User Manual.

Safety view

2 Safety view

2.1 Safety objectives

The 2nd generation of AURIX™ MCAL software is developed as a software safety element out of context (MCAL SEooC). The following safety objectives will be realized by the MCAL SEooC.

2.1.1 Safety objective 1

[ASIL B] The MCAL SEooC shall ensure ASIL B functionality with respect to systematic software faults for the following drivers:

- Microcontroller drivers (McDrivers):
 - Microcontroller Unit (MCU) driver including GTM initialization
 - Watchdog (WDG) driver for the internal CPU watchdogs
 - General Purpose Timer (GPT) driver
- Memory, Memory hardware abstraction drivers (MemDrivers)
 - Internal Flash (FLS) driver
 - Flash EEPROM Emulation (FEE) driver
- I/O drivers
 - Port (PORT) driver
 - Digital I/O (DIO) driver
 - Analog-to-Digital Converter (ADC) driver
 - Delta Sigma Analog-to-Digital Converter (DSADC) driver
 - Pulse Width Modulation (PWM) driver
 - Input Capture Unit (ICU) driver
 - Output Capture Unit (OCU) driver
- Communication drivers, Communication hardware abstraction drivers (ComDrivers)
 - Serial Peripheral Interface (SPI) Handler driver
- AUTOSAR libraries
 - CRC library
 - BFX library
- Complex drivers
 - UART driver
 - DMA driver
 - SMU driver
 - MCAL Library

Note: *It is assumed that customer shall check and ensure the correctness of generated configuration data (from EB tresos using code templates).*

2.1.1.1 Safety realization

The safety objective is realized for different MCAL drivers as indicated in the subsequent sections:

Safety view

2.1.1.1.1 McDrivers, I/O drivers, ComDrivers, Complex drivers, AUTOSAR libraries

All the functional requirements for the above category of MCAL drivers are treated as safety requirements with an ASIL rating **ASIL B**. The MCU driver is extended with safety requirements to provide initialization of the GTM peripheral to allow its usage in other ASIL B drivers, that is, PWM, ICU, GPT, ADC, DSADC and OCU.

The safety measures as recommended for ASIL B against systematic software failures identified in the safety analysis are applied for the MCAL drivers. The safety measures which need to be fulfilled by the integrator will be provided as Assumptions of Use (AoU) in order to enable the usage of these drivers in safety relevant applications up to ASIL B and to facilitate flexibility in customers' safety concepts.

2.1.1.1.2 QM ComDrivers

The ComDrivers such as CAN, CANTRCV, LIN, FR, ETH drivers are developed as QM drivers assuming usage of black channel concept such as the E2E protection at the system level.

All the functional requirements from AUTOSAR for the above set of MCAL drivers are de-rated to QM assuming that the integrator performs ASIL decomposition at the system level using the black channel concept.

The black-channel concept allows communicating safety-relevant data between the sender and receiver using a potentially faulty channel (QM) referred to as Black channel. This channel does not have any data integrity measures implemented and has the potential to introduce both systematic and random failures. However, the sender and the receiver modules incorporate safety layer which can perform safety checks to detect any potential faults that has an impact on the integrity of the safety data.

It is assumed that the integrator performs ASIL decomposition (ISO 26262, Part 9, clause 5.4.7) between the intended functionality, in this case the transmission of data through the communication stack (QM) and an associated safety mechanism, for example, the AUTOSAR E2E protection (with an appropriate ASIL based on the integrator system safety concept). This then enables these drivers to be used in a safety relevant application.

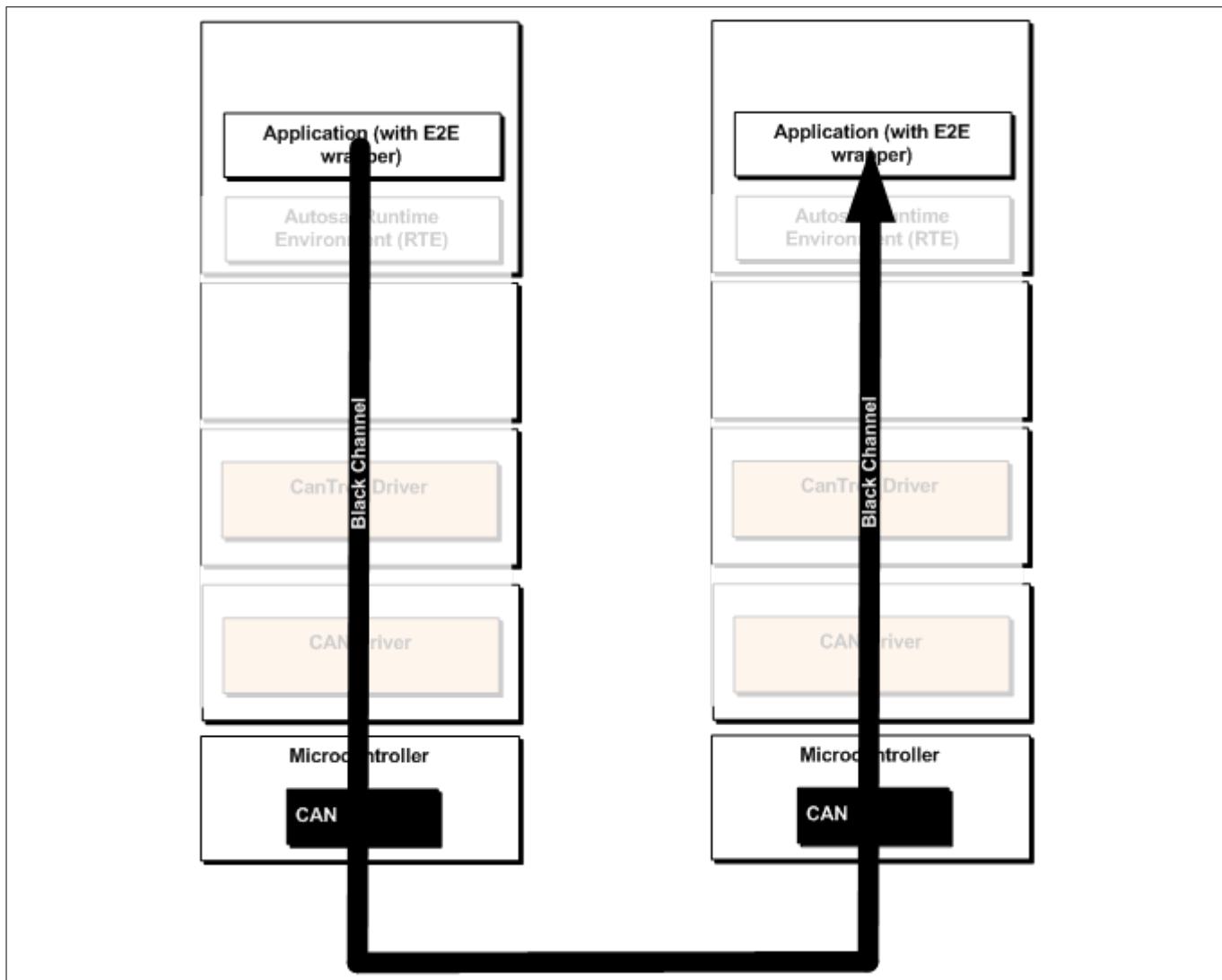
Safety view


Figure 67 Black channel - CAN as an example

2.1.1.3 MemDrivers

All the functional requirements for the above category of MCAL drivers are treated as safety requirements with an ASIL rating **ASIL B**. Additionally, it is assumed that the integrator uses system-level safety mechanism for example, CRC checks of the data by NVRAM Manager in order to detect potential violations of the intended functionality, that is, storage and retrieval of data using MemDrivers.

Nonetheless, safety analysis is carried out for the MemDrivers and additional safety measures as recommended for ASIL B are applied to these drivers to detect/avoid systematic software failures.

2.1.2 Safety objective 2

[ASIL D] The MCAL SEooC shall ensure no interference to other software units outside MCAL SEooC.

MCAL SEooC shall implement safety measures to ensure that there is no interference to other software units outside MCAL SEooC, therefore, enabling the MCAL SEooC to be used in an ASIL D partition at the system level.

Safety view

2.1.2.1 Safety realization

The ISO 26262-6 7.4.10 requires that “If the embedded software has to implement software components of different ASILs, or safety-related and non-safety-related software components, then all of the embedded software shall be treated in accordance with the highest ASIL, unless the software components meet the criteria for coexistence in accordance with ISO 26262-9:2011, Clause 6”.

The MCAL SEooC drivers will be integrated with other software units by the system integrator to realize the intended safety application. Hence, in the context of the safety application, the MCAL drivers will be treated as software components and therefore will need to be treated with the highest ASIL. Even though the MCAL SEooC drivers do not have any ASIL D functional safety requirements, it may co-exist with other ASIL D software units and hence necessary (tailored for MCAL) ASIL D process methods are used for the development of the MCAL drivers (all productive drivers including QM ComDrivers).

Note: *The ADC, SPI, DS-ADC MCAL drivers rely on the DMA for the realizing the required functionality. Due to limitations of the DMA hardware, the FFI cannot be satisfied. Hence, customer would need to implement additional measures for example, protection through Bus MPU to prevent interference. Refer to the Hardware User Manual for details on how to configure this BUS MPU (Safety Memory Protection).*

2.2 Common Assumptions of Use (AoUs)

The AoUs that are common to all the productive MCAL drivers are described in this section.

- **Activate hardware mechanisms for safe SFR access**

The integrated system shall activate all hardware safety mechanisms, which are related to SFR access.

[cover parentID=A2GT-REQ_AoU_SW-7]

Note: *All hardware safety mechanisms are documented in the Hardware Safety Manual (example for Bus Access Protection: SM[HW]:PORT:CFG_AS_AP).*

- **Enable hardware safety mechanisms for memory (Flash, RAM)**

The integrated system shall enable all hardware safety mechanisms for RAM and Flash.

[cover parentID=A2GT-REQ_AoU_SW-6]

Note: *All hardware safety mechanisms are documented in the Hardware Safety Manual (example for NVM: SM[HW]:NVM).*

- **Implement hardware AoUs described in the Hardware Safety Manual**

The integrated system shall implement the hardware AoUs as described in the Hardware Safety Manual.

[cover parentID=A2GT-REQ_AoU_SW-12]

- **Code and configuration data correctly flashed**

The integrated system shall ensure that the code and configuration data of MCAL are correctly flashed.

[cover parentID=A2GT-REQ_AoU_SW-11]

- **E2E protection for communication**

The integrated system shall provide an end-to-end protection for all types of safe communication with the expected ASIL level of the data to be transferred.

[cover parentID=A2GT-REQ_AoU_SW-1]

Safety view

Note: *This is affecting communication through CAN, FR and ETH.*

- **Ensure safe data storage by mechanism outside SEooC**

The integrated system shall implement safety mechanisms for safe data storage in software layers outside the SEooC MCAL memory drivers with the expected ASIL level. Refer to [MemDrivers](#).

[cover parentID=A2GT-REQ_AoU_SW-2]

Note: *CRC check for example, handled by NVRAM manager or SW-C is a measure to fulfill the safe data storage.*

Note: *MCAL memory drivers contain the following drivers: Internal Flash (FLS) driver, Flash EEPROM Emulation (FEE) driver.*

- **FFI in memory space for external services**

External services invoked by MCAL SEooC shall implement safety measures to ensure that there is no interference to memory space outside the specified one.

[cover parentID=A2GT-REQ_AoU_SW-9]

Note: *The external services accessed by MCAL software modules are documented in the module-specific sections of this document.*

- **Generation of configuration data**

The integrator shall check and ensure the correctness of generated configuration data.

[cover parentID=A2GT-REQ_AoU_SW-10]

Note: *Code generation in the EB tresos tool is based on code templates.*

Note: *Infineon shall provide the documentation, with production release, to enable the customer to verify generated configuration data.*

- **Hardware resource access**

The integrated system shall not access the hardware resources (documented in the module-specific sections of this document), which are controlled and configured by the MCAL SEooC.

[cover parentID=A2GT-REQ_AoU_SW-4]

Note: *SFRs and memories are typical hardware resources.*

Note: *SMU SFRs are locked after initialization. Therefore, they cannot be modified during the runtime phase.*

- **Interference from software outside MCAL SEooC**

The system integrator shall implement measures to ensure no interference from the environment to MCAL SEooC data (software variables) and hardware resources (for example, SFRs) controlled by MCAL SEooC.

[cover parentID=A2GT-REQ_AoU_SW-5]

Note: *The measure could be that the environment is developed according to the same ASIL as MCAL SEooC.*

Note: *This means that for example, MCAL SEooC will NOT implement mechanisms like redundant and diverse storage for safety-related global variables.*

Safety view

Note: *Potential integrator measures in this context are to define MPU partitions to protect RAM variables or SFR areas.*

- **Invoked external services**

The integrated system shall ensure that all external services invoked by MCAL SEooC (for example, OS change user/supervisor mode) shall be ISO26262 compliant with minimum ASIL B.

[cover parentID=A2GT-REQ_AoU_SW-3]

Note: *The external services accessed by MCAL software modules are documented in the module-specific sections of this document.*

- **Program flow monitoring at system level**

The integrated system shall implement program flow monitoring at system level.

[cover parentID=A2GT-REQ_AoU_SW-8]

- **STM is enabled**

The user of MCAL shall ensure that STM is enabled and not in sleep mode before invoking any MCAL APIs.

[cover parentID MCU={944C58EE-586A-49f6-8036-C206C63762E1}]

- **Multicore reset handling**

User shall ensure slave core are reset before resetting master core.

- **Interrupt mapping**

User shall ensure that hardware resources assigned to a core must have their interrupts routed to the same core.

- **ASIL partitioning**

ASIL partitioning is not supported in MCAL. The applications using a particular MCAL driver shall have the same ASIL levels, that is, applications having different ASILs shall not access the same MCAL driver.

- **Detect missing and unintended interrupts**

The integrated system shall implement mechanisms to detect the missing the CPU interrupts. The system shall also implement mechanisms to detect the missing and unexpected interrupts serviced by the DMA.

Note: *Recommendations for implementations are given in the hardware safety manual under ESM[SW]:IR:ISR_MONITOR.*

Note: *The MCAL drivers monitor the interrupt service requests routed to the CPU. The drivers report any unintended service requests to the application though safety error.*

[cover parentID=A2GT-REQ_AoU_SW-13]

- **SMU: Register monitoring test**

The integrated system shall take appropriate actions depending on the result of the register monitor test service provided by the SMU driver.

Note: *Refer to the hardware safety manual for recommendations on appropriate actions.*

[cover parentID=A2GT-REQ_AoU_SW-14]

- **SMU: Alive alarm test**

The integrated system shall take appropriate actions depending on the result of the SMU alive test provided by the SMU driver.

Note: *Refer to the hardware safety manual for recommendations on appropriate actions.*

Safety view

[cover parentID=A2GT-REQ_AoU_SW-15]

- **FCE: CRC value verification**

The integrated system shall verify the CRC value provided by the MCAL SEooC.

Note: *The comparison between the calculated CRC value and the expected CRC value is not provisioned by MCAL.*

[cover parentID=A2GT-REQ_AoU_SW-16]

- **DMA: Data CRC**

The integrated system shall compare the expected and the calculated DMA data checksum.

Note: *The DMA driver provides the checksum to the integrated system.*

Note: *MCAL drivers using the DMA driver do not use the data CRC feature.*

[cover parentID=A2GT-REQ_AoU_SW-18]

- **DMA: Error handling**

The integrated system shall define appropriate actions in case of safety-related DMA errors.

Note: *The DMA driver detects the DMA error interrupt for transactions initiated through the DMA driver and notifies the user through the callback function.*

Note: *Safety-related DMA errors are the ones which have the potential to impact the safety goal.*

Note: *Refer to the hardware safety manual for recommendations on appropriate actions.*

[cover parentID=A2GT-REQ_AoU_SW-19]

- **DMA: Address CRC**

The integrated system shall compare the expected and the calculated DMA address checksum.

Note: *The DMA driver passes the checksum to the integrated system.*

Note: *MCAL drivers using the DMA driver do not support the address CRC feature.*

[cover parentID=A2GT-REQ_AoU_SW-20]

- **DMA: Timestamp**

The integrated system shall compare the expected and the reported DMA timestamp.

Note: *The DMA driver passes the timestamp to the integrated system.*

Note: *MCAL drivers using the DMA do not use the timestamp feature.*

[cover parentID=A2GT-REQ_AoU_SW-21]

- **CONVCTRL: Fault injection**

The integrated system shall verify PHSDIV and counter using the fault injection mechanisms for CONVCTRL.

Note: *The fault injection is controlled by the PHSSFTY SFR. The MCAL SEooC is not accessing the PHSSFTY SFR. Refer the hardware user manual for detailed register description.*

Safety view

Note: Two separate fault injections need to be invoked, one for PHSDIV and one for the counter. Fault injection mechanisms should be executed after first invocation of `Mcu_InitClock()`.

[cover parentID=A2GT-REQ_AoU_SW-22]

Indirect SFR updates

MCAL addresses indirect updates to SFRs, which impact the functioning of the driver. The indirect SFR updates that are not affecting the MCAL functionality shall be addressed by the user.

2.3 Mapping of MCAL to external safety mechanism (ESM)

The following table lists the mapping of ESMs as provided in the *AURIX TC3xx Safety Manual* to the MCAL software.

ESM ID	Comments/implementation hints from MCAL
ESM[SW]:DMA:ERROR_HANDLING	The DMA driver detects the DMA error interrupt for transactions initiated via the DMA driver and notifies the user via call back function.
SMC[SW]:SCU:ERU_CONFIG	Programming of ERU SFRs are done in accordance with the target specification by MCAL drivers for ERU channels used by MCAL.
SMC[SW]:SMU:CONFIG	The SMU driver configures the SMU peripheral in core domain and standby domain.
ESM[SW]:SMU:APPLICATION_SW_ALARM	The SMU driver provides the <code>Smu_SetAlarmStatus</code> and <code>Smu_ClearAlarmStatus</code> APIs to set and clear alarms.
ESM[SW]:SMU:REG_MONITOR_TEST	The SMU driver provides the <code>Smu_RegisterMonitor</code> API to execute the register monitor test.
ESM[SW]:SMU:ALIVE_ALARM_TEST	The SMU driver provides the <code>Smu_CoreAliveTest</code> API to execute the core alive test.
SMC[SW]:FCE:CRC_CFG	The CRC driver configures the FCE channel and calculates the CRC for the data provided by the user. The comparison between calculated CRC value and the expected CRC value is not provisioned by MCAL.
ESM[SW]:IR:ISR_MONITOR	<p>The MCAL drivers monitor the interrupt service requests routed to CPU. The drivers report any unintended service requests to the application through safety error.</p> <p>Refer to Reporting of unintended service requests.</p> <p>Note: Unintended service requests routed to DMA is not monitored by MCAL drivers.</p>
ESM[SW]:DMA:TIMESTAMP	<p>The time stamp feature of DMA is supported in the DMA driver. The application software shall use this feature while using the DMA driver.</p> <p>Note: MCAL drivers using the DMA do not use the timestamp feature.</p>
ESM[SW]:DMA:DATA_CRC	The data CRC feature of DMA is supported in the DMA driver. The application software shall use this feature while using the DMA driver.

Safety view

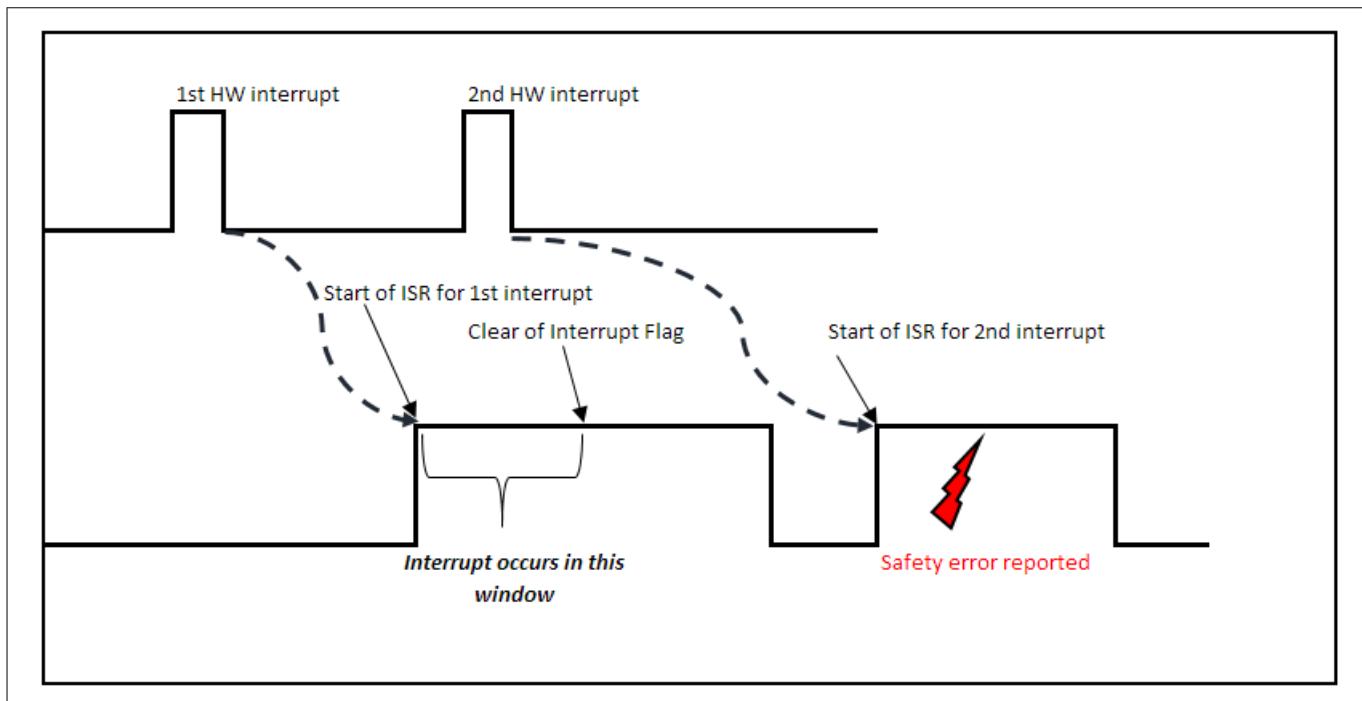
ESM ID	Comments/implementation hints from MCAL
	Note: MCAL drivers using the DMA driver do not use the data CRC feature.
ESM[SW]:DMA:ADDRESS_CRC	<p>The address CRC feature of DMA is supported in the DMA driver. The application software shall use this feature while using the DMA driver.</p> <p>Note: MCAL drivers using the DMA driver do not support the address CRC feature.</p>
ESM[SW]:CONVCTRL:CONFIG_CHECK	<p>The MCU driver performs a read back of CONVCTRL_PHSCFG register after programming it in the API Mcu_InitClock(). If the read value differs from the expected value then a safety error is reported. The reported safety error shall be addressed by the user software.</p>
ESM[SW]:EVADC:CONFIG_CHECK	<p>The ADC driver provides initialization and initialization check APIs. The initialization API programs the GxANCFG and GxSYNCTR SFRs. The initialization check reads back and verifies the content of GxANCFG and GxSYNCTR for correctness. User shall use these services to detect a failure in programming of GxANCFG or GxSYNCTR. A return value of E_NOT_OK from initialization check API indicates an error and shall be addressed by the user software.</p>

Note: *AoUs from AURIX TC3xx Safety Manual, other than the ones mentioned in the above table are not provisioned by MCAL and hence, requires evaluation of the integrator.*

2.3.1 Reporting of unintended service requests

Scenario for the ADC, DSADC and MCU drivers

The ADC, DSADC and MCU drivers may report an unintended service request safety error when a second hardware interrupt occurs in between the start of an ISR and clearing of the interrupt flag. **The safety error reported in such a scenario must be ignored by the user.** The following figure describes the scenario in detail.

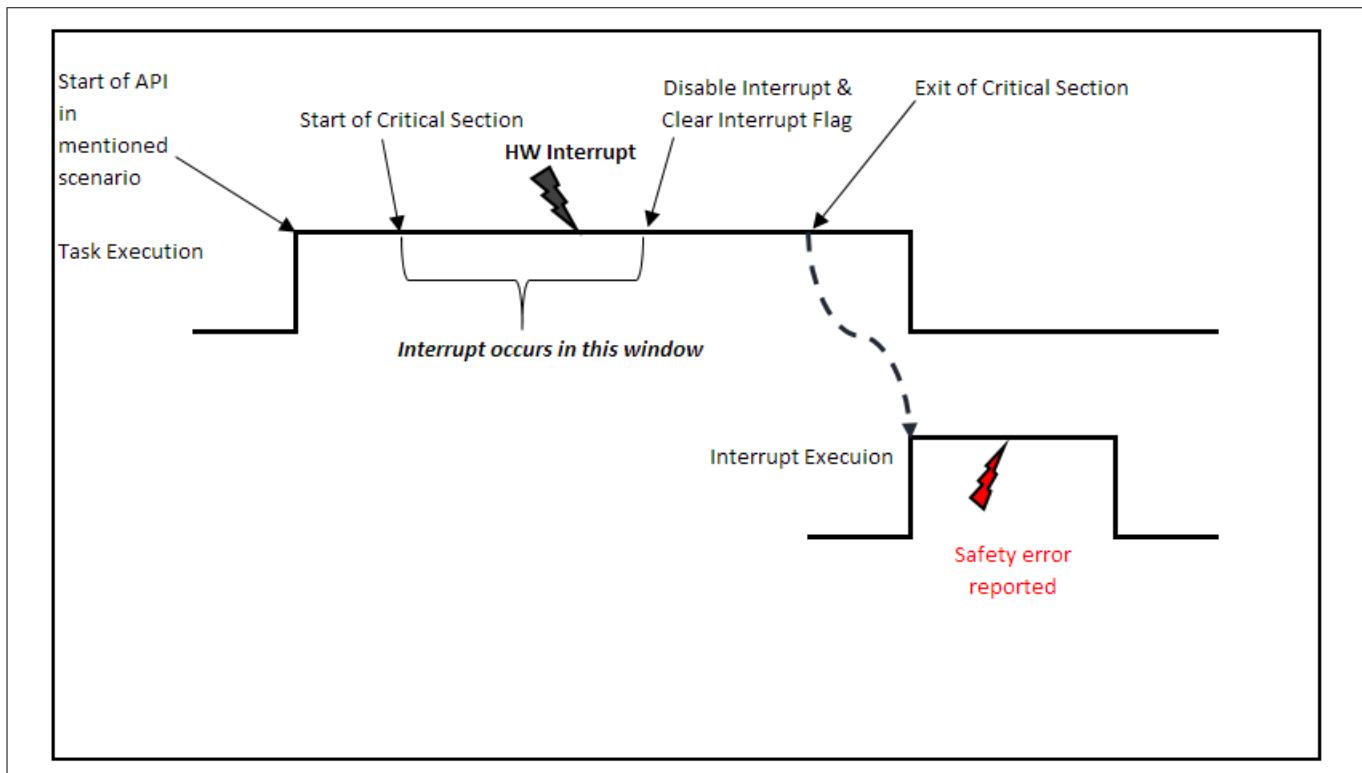
Safety view

Figure 68 Incorrect reporting of unintended service request - 1

Note: *This scenario typically happens when the interrupts occur at a rate higher than the service rate.*

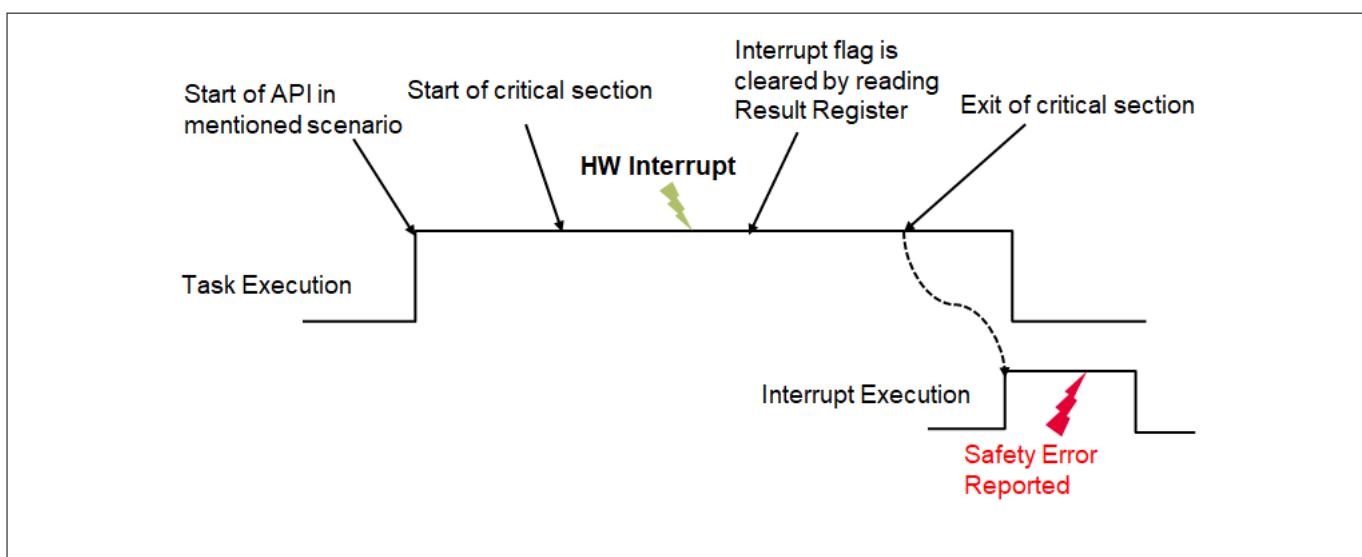
Scenario for the PWM and MCU drivers

The MCU driver may report an unintended service request safety error when the PWM APIs are invoked in any of the following scenarios and an interrupt for the same channel occurs in a small window as shown in the following figure:

- `Pwm_17_GtmCcu6_SetDutyCycle()` is invoked to set the duty cycle as 0% or 100% for a channel
- `Pwm_17_GtmCcu6_SetPeriodAndDuty()` is invoked to set period as 0 or duty as 0% or duty as 100% for a channel
- `Pwm_17_GtmCcu6_EnableNotification()` is invoked for a variable period channel for which the period is currently set to 0
- `Pwm_17_GtmCcu6_DisableNotification()` is invoked

Safety view

Figure 69 Incorrect reporting of unintended service request - 2
Scenario for the DSADC driver

The DSADC driver may report an unintended service request safety error when the DSADC channel is configured as DSADC_SINGLE_READ and the Dsadc_ReadResult API is invoked and an interrupt for the same channel occurs in a small window as shown in the following figure:


Figure 70 Incorrect reporting of unintended service request - 3

The safety error reported in the above scenarios must be ignored by the user.

ADC driver**3 ADC driver****3.1 User information****3.1.1 Description**

The ADC driver is responsible for providing standard analog-to-digital conversion services specified by AUTOSAR. The ADC driver provides user interface options to configure the driver parameters as described in the AUTOSAR ADC specification (Version AS 4.2.2). It also provides additional parameters to configure various functional blocks of EVADC. The EVADC module is the underlying analog converter. The APIs provided by the driver are multicore capable, that is, they may be invoked from several cores simultaneously. The ADC driver supports only the Primary and Secondary clusters of the EVADC, however, the fast-compare channels are not supported.

3.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

ADC driver

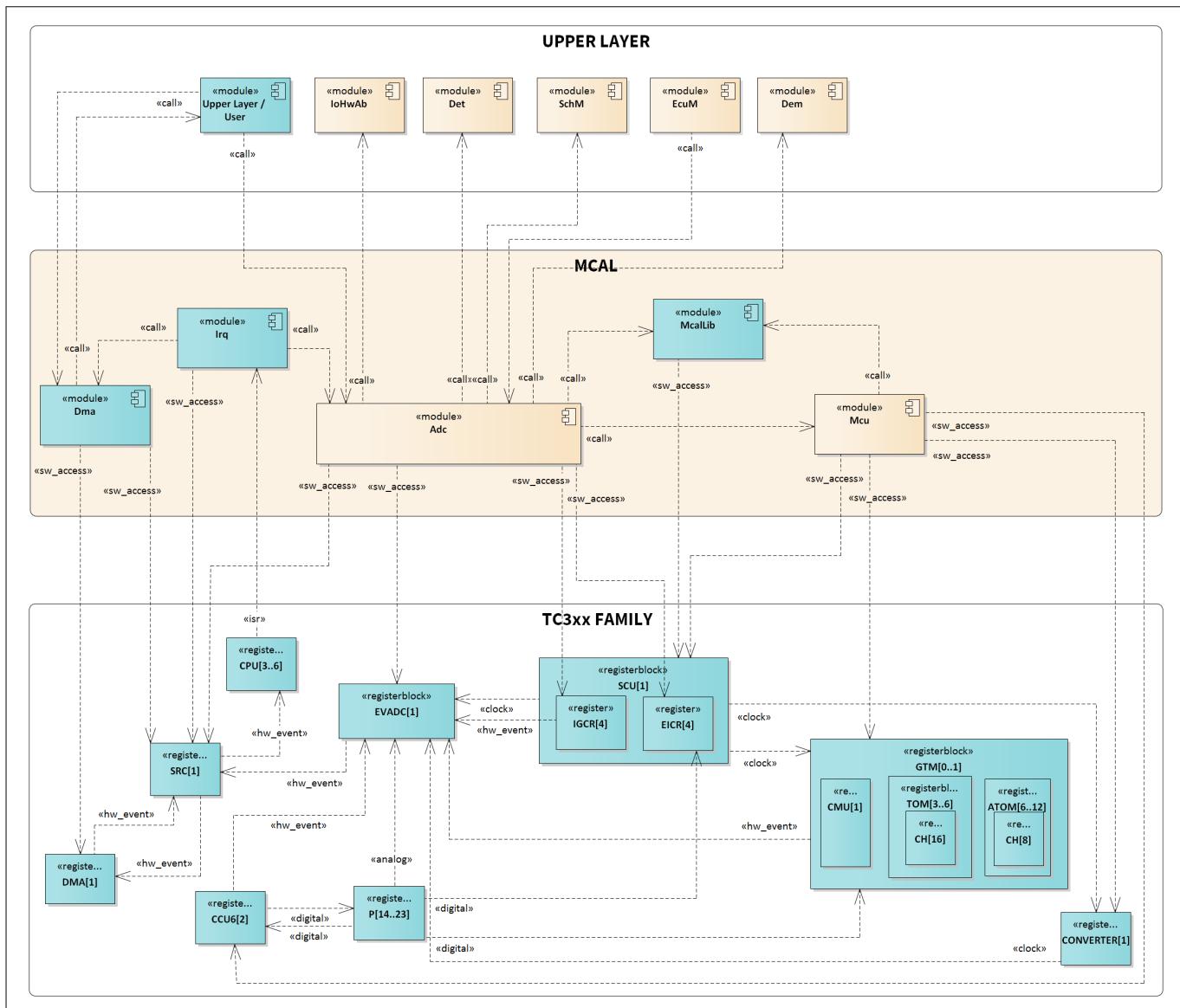


Figure 71 Mapping of hardware-software interfaces

3.1.2.1 EVADC: primary hardware peripheral

Hardware functional features

The ADC driver uses the EVADC IP for converting the analog signals to 12-bit digital values.

The key hardware functional features used by the driver are:

- Software-based trigger
- Timer- and event-based triggers
- Limit checking
- Priority mechanism of the arbiter
- Power saving modes
- Start-up calibration
- Synchronous conversion across hardware groups

The unsupported features of the EVADC IP are:

- Wait-for-Read mode

ADC driver

- Conversion result modification modes (Standard Data Reduction mode, Filtering mode and Difference mode)
- External multiplexer
- Fast compare channels
- FIFO mode of result registers

Users of the hardware

The ADC driver exclusively utilizes the EVADC IP.

Hardware diagnostic features

The supported diagnostic features of the EVADC IP are:

- Broken wire detection
- On-chip supervision signal (through the alias feature)

The unsupported diagnostic features of the EVADC IP are:

- Pull-down diagnostics
- Multiplexer diagnostics
- Converter diagnostics

Hardware events

The ADC driver uses the following hardware events from the EVADC IP:

- Result event: to trigger transfer of conversion results through DMA IP when the DMA mode of result handling is configured
- Request source event: to trigger transfer of conversion results and update group status when the interrupt mode of result handling is configured
- Channel event: for limit checking enabled AdcChannel groups

3.1.2.2 SRC: dependent hardware peripheral

Hardware functional features

The ADC driver depends on the interrupt router for raising an interrupt to the CPU based on the request source event or the channel event, which indicates the end of conversion of a sequence of channels. The ADC driver copies the conversion results from the result registers to the application buffers in the interrupt service routine of these events.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the application software. The ADC driver clears the pending interrupt requests in the SRC register.

Hardware diagnostic features

The SMU alarms configured for the interrupt router are not monitored by the ADC driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU and the DMA. The ADC driver provides interrupt handlers as software interfaces, which must be invoked from the ISR.

3.1.2.3 Port: dependent hardware peripheral

Hardware functional features

The analog signals are routed to the EVADC converter through the analog port pads. The external trigger events for the converter are routed through the digital port pad. These are configured and enabled through the PORT driver.

Users of the hardware

ADC driver

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from port pads are not used by the ADC driver.

3.1.2.4 SCU: dependent hardware peripheral

Hardware functional features

The ADC driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB and fADC clock signals for functioning.

Users of the hardware

The SCU IP supplies the clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the ADC driver.

Hardware events

Hardware events from the SCU IP are not used by the ADC driver.

3.1.2.5 EICR/IGCR: primary hardware peripheral

Hardware functional features

The ADC driver depends on the ERU IP for realizing the hardware-event-based triggers and gating features. The driver uses the trigger event derived from the event trigger logic block of the ERU IP.

The unsupported features of the ERU IP are:

- Pattern detection
- Generation of interrupts based on the trigger events

Users of the hardware

The ERU IP is used by the ADC, DSADC and ICU drivers. The EICR and IGCR channels used by each driver are reserved through the configuration interfaces of the MCU driver. The channel specific SFRs are programmed by the driver. Since multiple channels share common SFRs and to avoid corruption of data for other channels, the driver programs these SFRs atomically with a channel-specific mask. Glitch filter configuration for digital ports is done by the MCU driver.

Hardware diagnostic features

The SMU alarms configured for the ERU IP are not monitored by the ADC driver.

Hardware events

The ADC driver uses the following hardware events from the ERU IP:

- Rising edge event
- Falling edge event

3.1.2.6 GTM: dependent hardware peripheral

Hardware functional features

The ADC driver depends on the GTM IP for realizing the time-based trigger and gating features. The driver uses the compare-match event and the channel output signal for starting and stopping the conversions of an ADC channel group.

ADC driver

Users of the hardware

The GTM IP is used by the ADC, DSADC, PWM, OCU, GPT and ICU drivers. The GTM resources used by each driver are reserved through the configuration interface of MCU driver to avoid resource conflict. The ADC driver uses the TOM or ATOM channels for generating the trigger and gating signals. The driver invokes the APIs of the MCU driver to configure all the SFRs related to TOM or ATOM for generating the trigger and gating signals. In case an external PWM signal is used for triggering or gating an AdcChannel group then the user must configure the GTM channels.

Hardware diagnostic features

The SMU alarms configured for the GTM IP are not monitored by the ADC driver.

Hardware events

The ADC driver uses the following hardware events from the GTM:

- Compare-match event: to start the conversion
- Channel output level: to start/stop the conversion based on the timer output

3.1.2.7 CONVERTER: primary hardware peripheral

Hardware functional features

The ADC driver depends on the CONVERTER IP for providing the clock synchronization signal for the EVADC IP. The clock synchronization signal synchronizes the analog clocks of all the EVADC hardware groups.

Users of the hardware

The synchronization signal frequency is used by the ADC and DSADC drivers, however the configuration for generating the signals is done by the MCU driver.

Hardware diagnostic features

The SMU alarms configured for the CONVERTER IP are not monitored by the ADC driver.

Hardware events

Not applicable.

3.1.2.8 CCU6: dependent hardware peripheral

Hardware functional features

The ADC driver depends on the CCU6 events to trigger the conversion of an AdcChannel group. The user software may require an AdcChannel group to convert CCU6 events. In such a case, the user software must configure the CCU6 channel directly to generate events. The EVADC trigger line must be configured to the CCU6 event for the ADC channel group through configuration parameters.

Users of the hardware

The T12 or T13 channel of the CCU6 is exclusively used by the CCU6 user. While users of the T12 or T13 channels are many, a channel is allocated to and used by exactly one driver.

Hardware diagnostic features

The SMU alarms configured for the CCU6 are not monitored by the ADC driver.

Hardware events

The ADC driver uses the event generated by the timer channel. The event is used as a trigger for converting the analog channels.

3.1.2.9 DMA: dependent hardware peripheral

Hardware functional features

ADC driver

The ADC driver depends on the DMA IP for transferring the conversion results in the DMA mode of result handling.

Users of the hardware

The DMA channels are exclusively used by the DMA user. The DMA channels used for the ADC must be reserved and configured by the application through interfaces provided by the DMA.

Hardware diagnostic features

- Move engine error enabled during data transmission
- SMU alarms configured for the DMA are not monitored by the ADC driver

Hardware events

Hardware events from the DMA are not used by the ADC driver.

3.1.3 File structure

3.1.3.1 C file structure

This section provides details of the C files of the ADC driver.

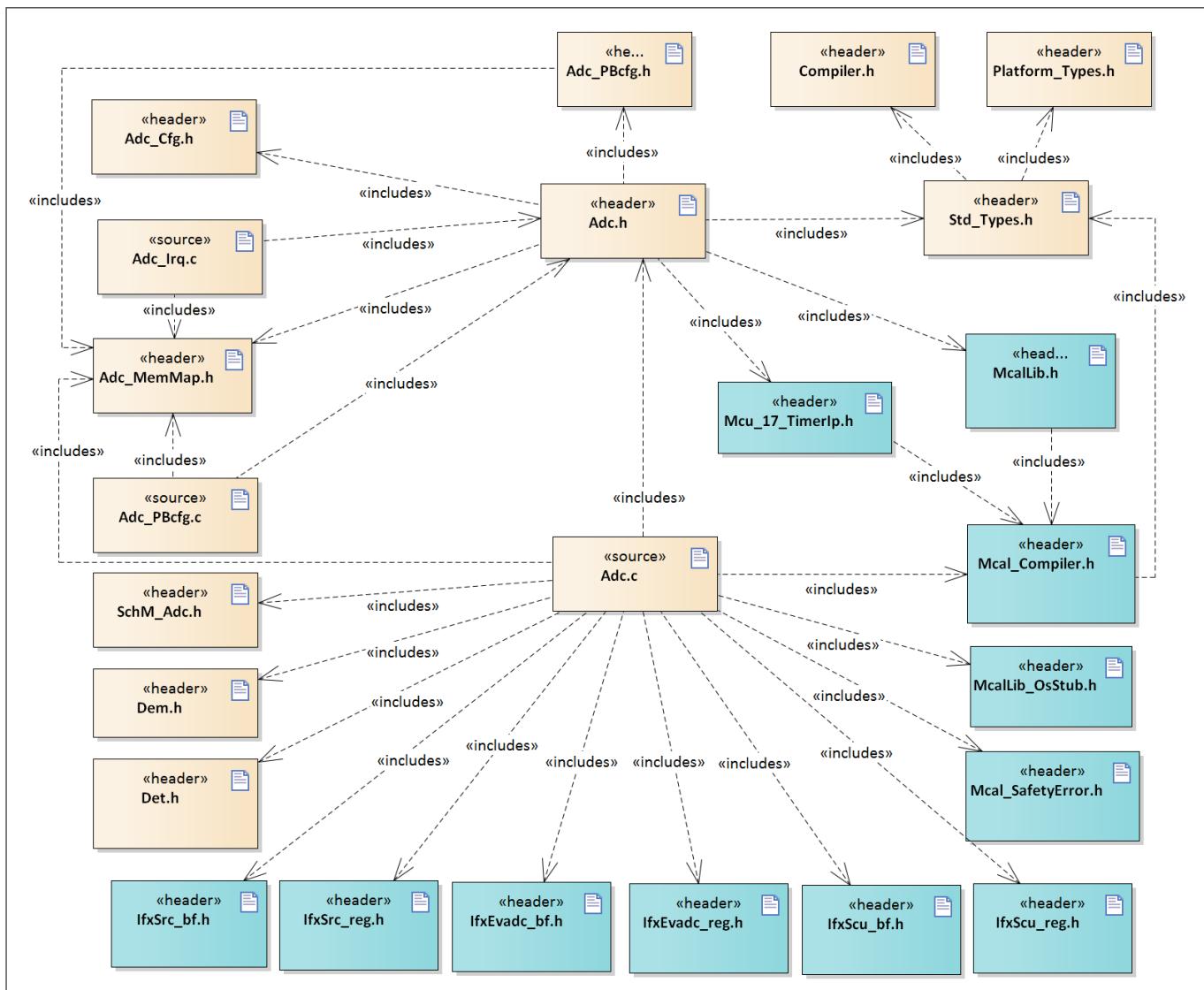


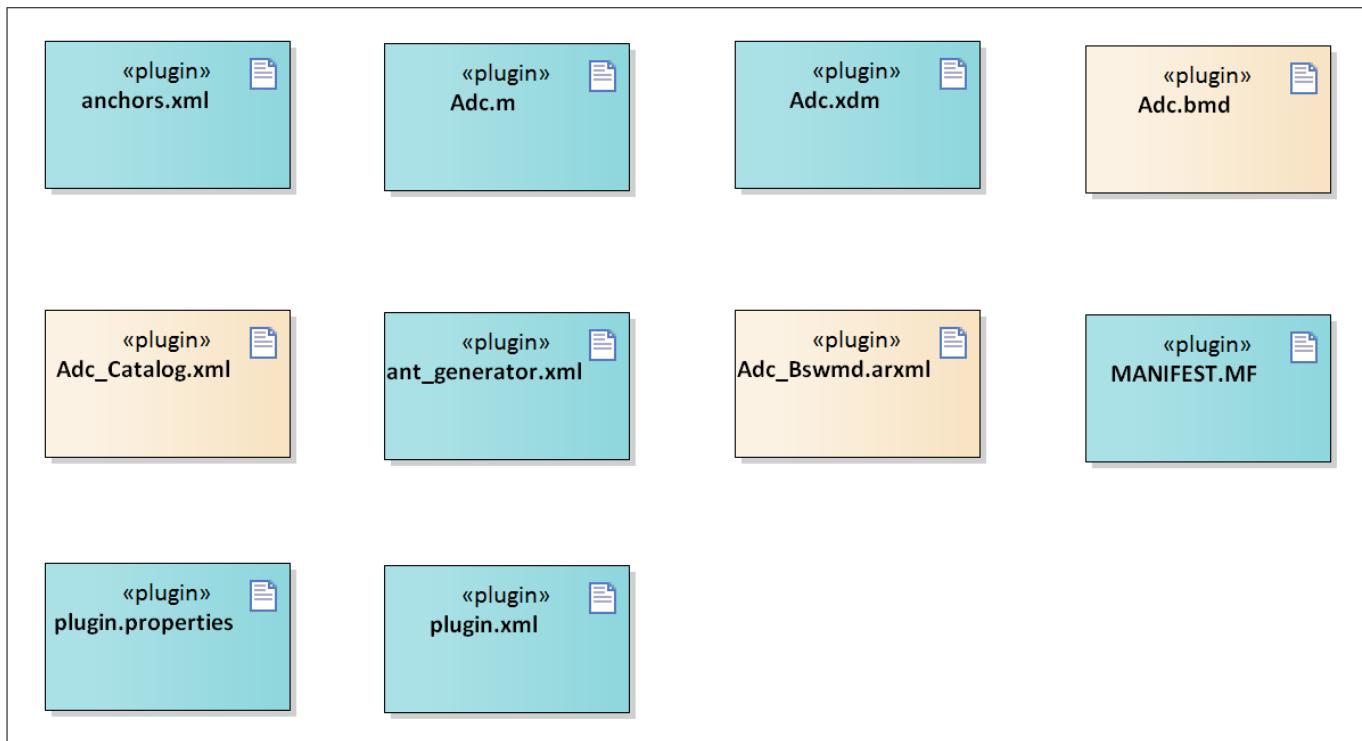
Figure 72 C file structure

ADC driver**Table 10 C file structure**

File name	Description
Adc.c	File (Static) containing implementation of APIs
Adc.h	Header file (Static) defining prototypes of data structures, APIs and interrupt handlers
Adc_Cfg.h	Header file (Generated) containing constants and pre-processor macros as #defines
Adc_Irq.c	Interrupt handler file for the ADC
Adc_MemMap.h	File (Static) containing the memory section definitions used by the ADC driver
Adc_PBcfg.c	File (Generated) containing definition of the configuration data structures
Adc_PBcfg.h	File (Generated) containing declaration of the post-build configuration data structures
Compiler.h	Provides abstraction from compiler-specific keywords
Dem.h	Provides the exported interfaces of Diagnostic Event Manager
Det.h	Provides the exported interfaces of Development Error Tracer
IfxEvadc_bf.h	SFR header file for EVADC
IfxEvadc_reg.h	SFR header file for EVADC
IfxScu_bf.h	SFR header file for SCU
IfxScu_reg.h	SFR header file for SCU
IfxSrc_bf.h	SFR header file for Interrupt Controller
IfxSrc_reg.h	SFR header file for Interrupt Controller
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore™. This shall be included by other drivers to call OS APIs.
Mcal_Compiler.h	Provides abstraction for TriCore™-intrinsic instruction
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Adc.h	Export Header for SchM functions of ADC
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

3.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the ADC driver.

ADC driver

Figure 73 **Code generator plugin files**
Table 11 **Code generator plugin files**

File name	Description
Adc.bmd	AUTOSAR format XML data model schema file
Adc.m	Code template macro file for the ADC driver
Adc.xdm	Tresos format XML data model schema file
Adc_Bswmd.arxml	AUTOSAR format module description file
Adc_Catalog.xml	AUTOSAR format catalog file as per catalog_V3_0_0.ml.xsd
MANIFEST.MF	Tresos plugin support file containing the metadata for the ADC driver
anchors.xml	Tresos anchors support file for the ADC driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configurations when using variation point
plugin.properties	Tresos plugin support file for the ADC driver
plugin.xml	Tresos plugin support file for the ADC driver

3.1.4 **Integration hints**

This section lists the key points that an integrator or user of the ADC driver must consider.

3.1.4.1 **Integration with AUTOSAR stack**

This section lists the modules, which are not part of the MCAL, but are required to integrate the ADC driver.

- **EcuM**

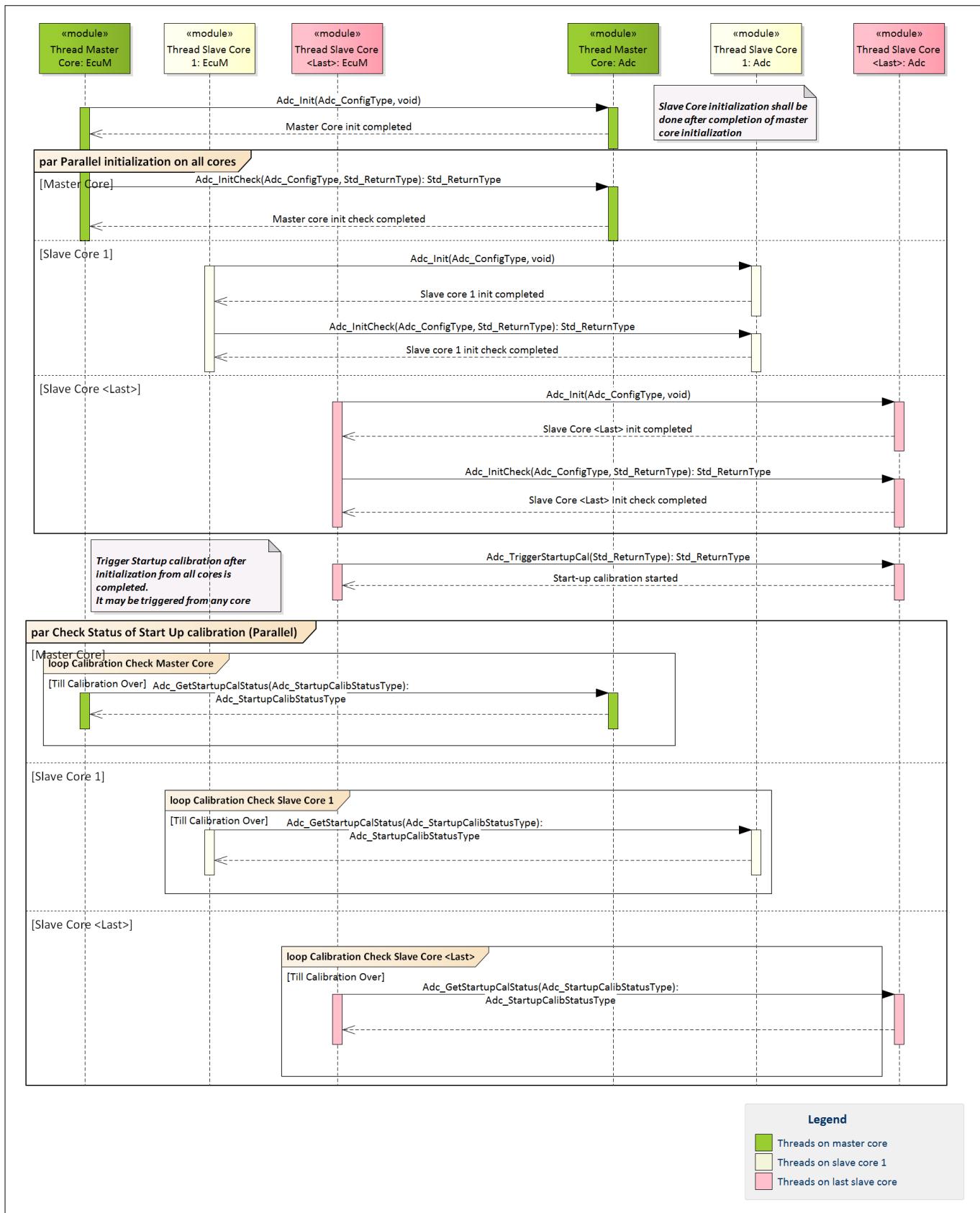
ADC driver

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of the ECU. Specifically, in the context of the MCAL, the EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

Initialization of ADC:

The initialization of the ADC driver should be invoked from each CPU core, which intends to use the services of the ADC driver. Initialization from the logical master core must be completed prior to invoking the initialization from the slave cores. All the slave cores can execute initialization in parallel. In case all the ADC resources are allocated to the master core, invoking the initialization from the master core alone is sufficient.

EVADC also requires a start-up calibration post-initialization, which must be invoked after initialization on cores is completed. A sample initialization sequence with start-up calibration is depicted in the following figure.

ADC driver**Figure 74 Sample initialization of ADC driver with start-up calibration****De-initialization of ADC:**

ADC driver

The de-initialization of the ADC driver should be invoked from each CPU core that used the services of the ADC driver. De-initialization from all the slave cores must be completed prior to invoking the de-initialization from the master core. The slave cores can execute de-initialization in parallel. In case all the ADC resources are allocated to the master core, invoking the de-initialization from the master core alone is sufficient. A sample de-initialization sequence is depicted in the following figure.

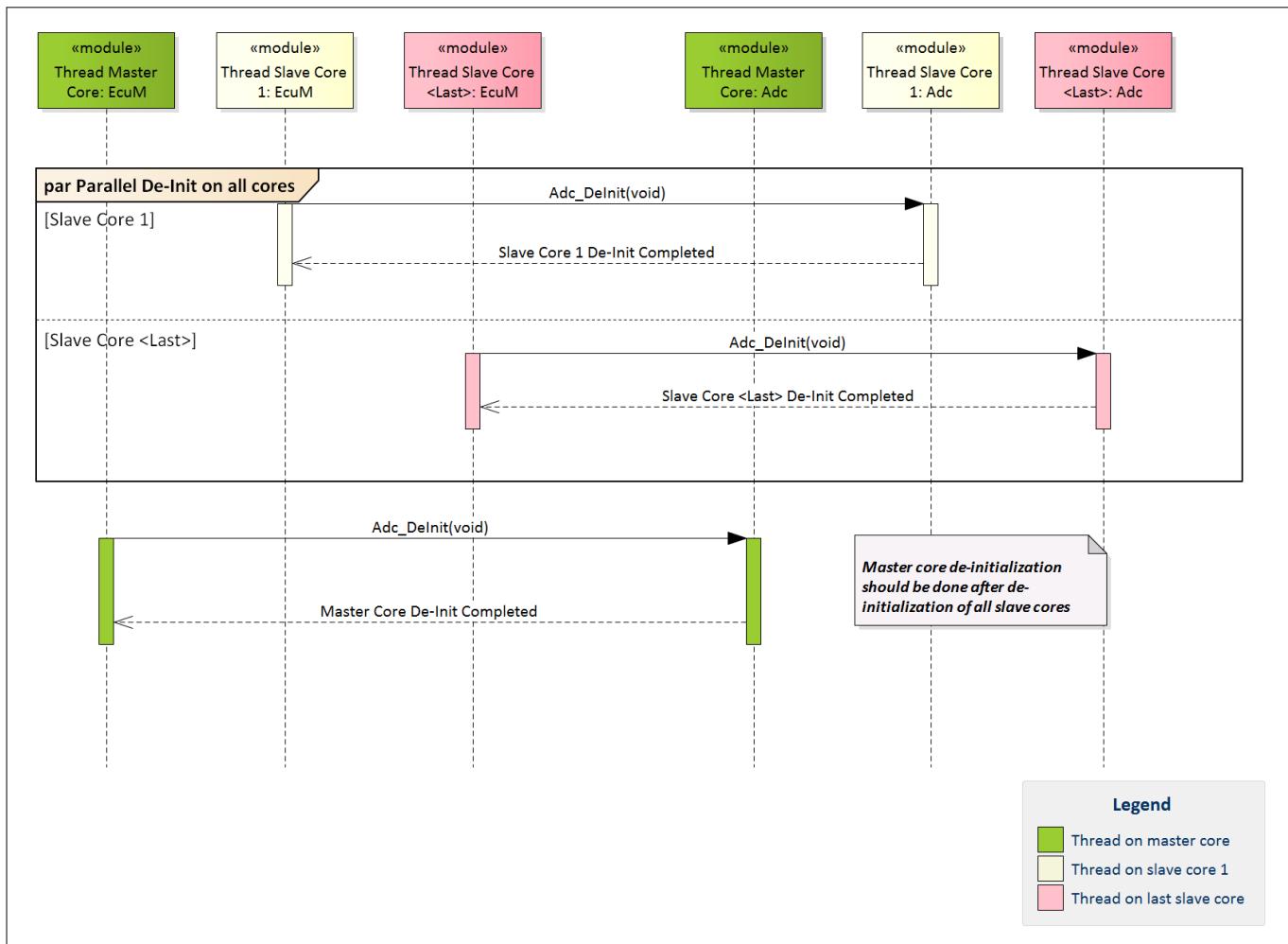


Figure 75 Sample de-initialization of ADC driver

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Adc_MemMap.h` file.

The `Adc_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place the appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements

ADC driver

are relocated to the correct memory region. A sample implementation listing the memory-section macros is as follows.

```
***** GLOBAL RAM DATA -- NON-CACHED LMU *****
if defined ADC_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
    /*User pragmas here for Non-cached LMU****/
#define ADC_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined ADC_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#define _TASKING_C_TRICORE_
    /*User pragmas here for Non-cached LMU****/
#define ADC_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR

***** CORE[x] CONFIG DATA -- PF[x] ****/ /*[x] = 0..5*/
#elif defined ADC_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
    /*User pragmas here for PF[x]****/
#define ADC_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined ADC_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
    /*User pragmas here for PF[x]****/
#define ADC_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#define MEMMAP_ERROR

***** CODE -- PF[x] ****/
#elif defined ADC_START_SEC_CODE_ASIL_B_GLOBAL
    /*User pragmas here for PF[x]****/
#define ADC_START_SEC_CODE_ASIL_B_GLOBAL
#define MEMMAP_ERROR
#elif defined ADC_STOP_SEC_CODE_ASIL_B_GLOBAL
    /*User pragmas here for PF[x]****/
#define ADC_STOP_SEC_CODE_ASIL_B_GLOBAL
#define MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Adc_MemMap.h, wrong pragma command"
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The ADC driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the ADC driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is a part of the AUTOSAR stack that handles all the production errors reported by the BSW modules. The ADC driver reports all the production errors to the DEM modules through the

ADC driver

Dem_ReportErrorStatus() API. The user of the ADC driver must process all the production errors (fail / pass) reported to the DEM module through the Dem_ReportErrorStatus() API.

The Dem.h and Dem.c files are provided in the MCAL package as a stub code and needs to be replaced with a complete DEM module during the integration phase.

Note: Re-entrancy of the Adc_StopGroupConversion, Adc_DisableHardwareTrigger, Adc_RS0EventInterruptHandler, Adc_RS1EventInterruptHandler, Adc_RS2EventInterruptHandler and Adc_ChEventInterruptHandler APIs is dependent on the re-entrancy of the Dem_ReportErrorStatus() API. As per their design, the modules APIs are re-entrant for different hardware units. However, in case the Dem_ReportErrorStatus() API is implemented as non-reentrant, the APIs inherit the property of the same.

- **SchM**

The SchM module is a part of the RTE that manages the BSW Scheduler. The ADC driver uses the exclusive areas defined in the SchM_Adc.h file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the ADC driver are:

- KernelData
- SrcRegAccess

The SchM_Adc.h and SchM_Adc.c files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the ADC driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is as follows:

```
***** Sample implementation of SchM_Adc.c *****/
#include "Os.h"

void SchM_Enter_Adc_KernelData(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Adc_KernelData(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Adc_SrcRegAccess(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Adc_SrcRegAccess(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

ADC driver

The ADC driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API.

The driver performs only detection and reporting of the safety errors. The handling of the reported errors should be carried out by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The driver does not implement any notifications. However, it does report the completion of a group conversion through notification functions. These notification functions can be configured by the user in the EB tresos for each AdcChannelGroup separately.

- **Operating system (OS)**

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application.

The OS files provided by the MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

Note: The ADC driver updates the SRC registers of EVADC (`MODULE_SRC.VADC.G[] .SRx`) to clear the pending interrupt requests.

3.1.4.2 Multicore and Resource Manager

The ADC driver supports the execution of its APIs in parallel from all CPU cores. The user should allocate resources of the EVADC to the CPU cores at pre-compile time using the Resource Manager module. The following are the key points to be considered with respect to multicore in the driver:

- ADC hardware groups can be allocated to the CPU cores at pre-compile time. For example, EVADC_G0 and EVADC_G1 can be allocated to core-2, EVADC_G9 to core-3 and EVADC_G3 to core-0.
- An AdcChannel group belongs to a particular hardware group. Hence, all the AdcChannel groups belonging to a hardware group, get allocated to the same core as the hardware group.
- It must be ensured that AdcChannel group passed as a parameter while invoking an API belongs to the same core on which the API is invoked.
- DETs will be raised in case APIs are invoked with mismatch of core and AdcChannel group.
- Interrupts raised by a hardware group must be serviced by the CPU core to which the hardware group has been allocated to.
- Locating of constants, variables and configuration data to the correct memory space should be done by the user. Memory sections are marked GLOBAL(common to all cores) and CORE[x](specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

Code section

The executable code of the ADC driver is placed under single MemMap section. It can be relocated to any PFlash region.

Data section

The RAM variable memory sections marked as specific to a core should be re-located to the DPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region. In devices with no LMU, non-cached DPR can be used.

Configuration data and constants:

ADC driver

The configuration data sections marked as specific to a core should be re-located to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

Note: Relocating code, data or constants to a distant memory region would impact execution timings.

Note: If the driver operates from single (master) core, all the sections may be relocated to the PFlash/DSPR/DLMU of the same CPU core.

3.1.4.3 MCU support

The ADC driver is dependent on the MCU driver for the clock configuration and timer IP-related services. The initialization of the ADC driver must be started only after completion of the MCU initialization. The following must be considered while configuring the MCU driver in the EB tresos:

- CONVCTRL block must be used if more than one analog converters are used by the application. The configuration and programming of the CONVCTRL block is managed directly by the MCU driver.
- AdcChannel groups may require conversions triggered by a timer event generated by the GTM. The ADC driver uses the services of the MCU to configure the GTM-related trigger events during runtime. The GTM channels used by the ADC driver must be reserved in the MCU configuration for exclusive use by the ADC.
- AdcChannel groups may require conversions triggered by a hardware event generated for EICR-IGCR. The EICR-IGCR channels used by the ADC driver must be reserved in the MCU configuration for exclusive use by the ADC.

3.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the ADC driver through the PORT configuration and initialize the port pins prior to invoking the ADC initialization.

3.1.4.5 DMA support

The ADC driver may be configured such that the conversion results are directly transferred from the result register to the application buffers through the DMA move engines. The APIs and configuration parameters of the DMA driver may be used to achieve this. Enabling the DMA mode is a module-wide feature. When enabled, priority and queuing are not supported.

The result register event from EVADC triggers a service request, which is serviced by the DMA. The DMA move engine transfers the conversion results from result registers to the application buffers. Result event will be triggered on completion of conversion of the last channel of the AdcChannel group. A request source event is also triggered, which is serviced by the CPU and updates the status of the AdcChannel group.

The user must ensure the following points, while using this mode:

- Configuration to enable the DMA-based result transfer must be done through the EB tresos parameter: AdcResultHandlingImplementation.
- DMA channels intended to be used for the ADC driver must be reserved and configured through the DMA driver in the EB tresos.
- Consecutive result registers must be allocated to channels of an AdcChannel group in the EB tresos.
- ADC driver does not configure the DMA channels. The user of the ADC should invoke proper DMA APIs to start/stop the DMA channels before starting/stopping an AdcChannel group.
- Address space 0xD and 0xC should not be used for DMA-related usage. The MemMap sections allocating memory in the scratch pad RAM should always generate global addresses instead of local addresses.
- Since the Data CRC and Address CRC features of the DMA are not used for the ADC driver, the user should ensure that while using the DMA mode a plausibility check of the conversion result is performed either by redundancy or by other means.

ADC driver

3.1.4.6 Interrupt connections

The interrupt connections of the ADC driver are described in this section.

- **Result handling in interrupt mode**

Conversion result transfer in interrupt is selected when AdcResultHandlingImplementation is equal to ADC_INTERRUPT_MODE_RESULT_HANDLING. In this mode, the conversion results are transferred from result registers to the application buffers in the ISR of the Request Source Event. Each request source of an ADC hardware group is assigned a dedicated service request line. The channel event generated as a result of the limit check feature is also assigned a dedicated service request line. The following figure depicts the interrupt connections required by the ADC driver.

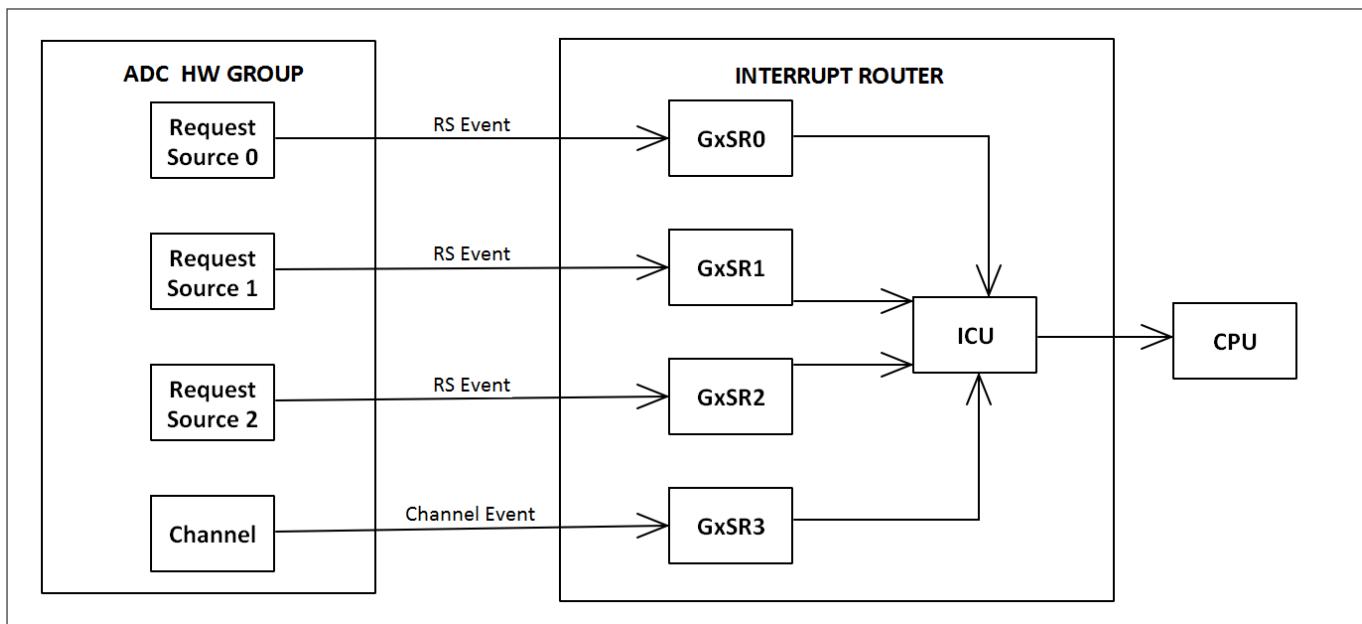


Figure 76 Result handling in the interrupt mode

ADC driver

Invoking the interrupt handlers provided by the driver must be done by the user. A sample invocation for EVADC_G0 and EVADC_G5 is shown as follows:

```
#include "Adc.h"
/**AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING**/

/******************EVADC_G0********************/
/******************SRC_VADC_G0_SR1*****/
ISR(ADC0SR0_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN0 of EVADC_G0*/
    Adc_RS0EventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}

/******************SRC_VADC_G0_SR1*****/
ISR(ADC0SR1_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN1 of EVADC_G0*/
    Adc_RS1EventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}

/******************SRC_VADC_G0_SR2*****/
ISR(ADC0SR2_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN2 of EVADC_G0*/
    Adc_RS2EventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}

/******************SRC_VADC_G0_SR3*****/
ISR(ADC0SR3_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN3 of EVADC_G0*/
    Adc_ChEventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}

/******************EVADC_G5*****/
/******************SRC_VADC_G5_SR0*****/
ISR(ADC5SR0_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN0 of EVADC_G5 */
    Adc_RS0EventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}

/******************SRC_VADC_G5_SR1*****/
ISR(ADC5SR1_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN1 of EVADC_G5 */
}
```

ADC driver

```

    Adc_RS1EventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}

/****************SRC_VADC_G5_SR2*******/
ISR(ADC5SR2_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN2 of EVADC_G5 */
    Adc_RS2EventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}

/****************SRC_VADC_G5_SR3*******/
ISR(ADC5SR3_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN3 of EVADC_G5 */
    Adc_ChEventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}

```

- **Result handling in DMA mode:**

Conversion result transfer through DMA is selected when AdcResultHandlingImplementation is equal to ADC_DMA_MODE_RESULT_HANDLING. In this mode, the conversion results are transferred from result registers to the application buffers by a DMA move engine. The result register event triggers a service request which is serviced by the DMA. The following figure represents the interrupt connectivity.

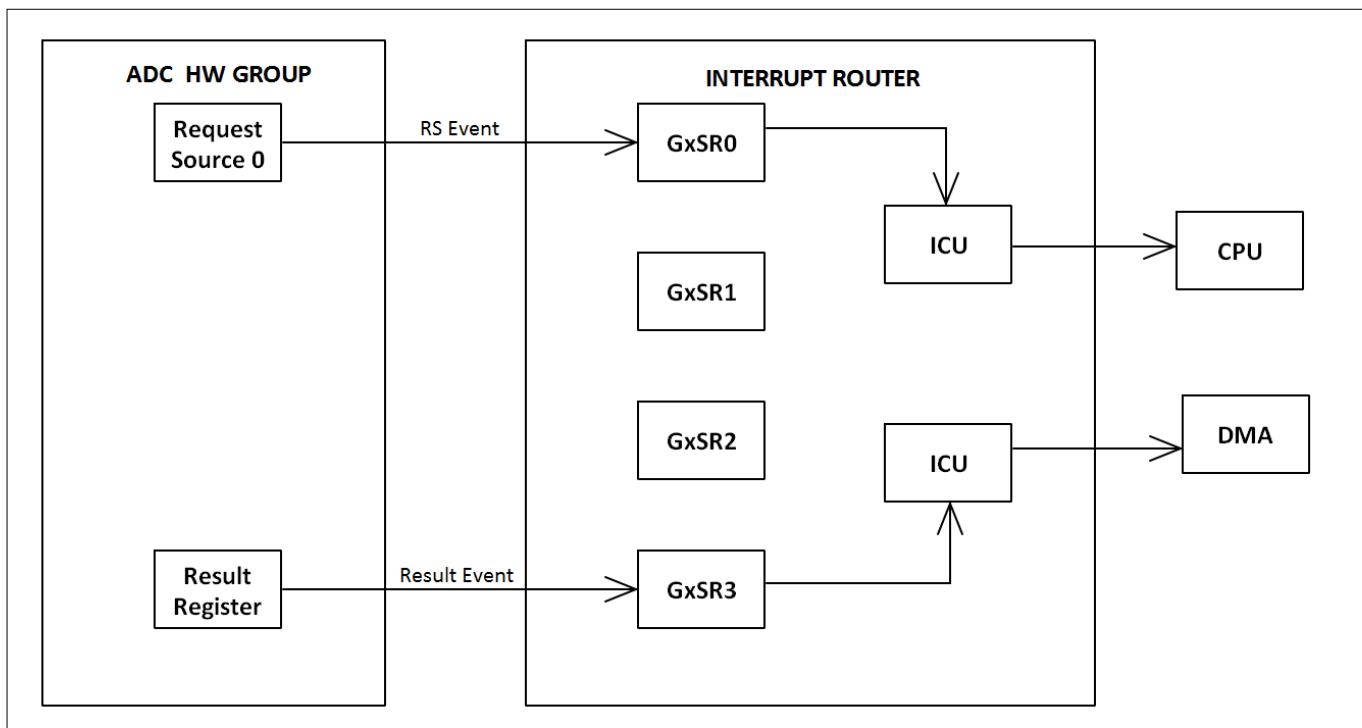


Figure 77

Result handling in the DMA mode

ADC driver

Invoking the interrupt handlers provided by the driver must be done by the user. A sample invocation for EVADC_G0 and EVADC_G5 is shown as follows:

```
#include "Adc.h"
/** AdcResultHandlingImplementation = ADC_MODE_RESULT_HANDLING ***/
/*****EVADC_G0*****
/*****SRC_VADC_G0_SR0*****
ISR(ADC0SR0_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN0 of EVADC_G0 */
    Adc_RS0EventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}
**** Service Request 3 is service by DMA ****/
**** Other Service Requests are unused ****/

/*****EVADC_G5*****
/*****SRC_VADC_G5_SR0*****
ISR(ADC5SR0_ISR)
{
    ENABLE(); /* Enable interrupts */
    /*The interrupt handler should be called from SRN0 of EVADC_G5*/
    Adc_RS0EventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}
**** Service Request 3 is service by DMA ****/
**** Other Service Requests are unused ****/
```

ADC driver**3.1.4.7 Example usage**

The following are some of the key use cases of the ADC driver.

Note: Refer to the comments in the code snippets for additional information.

Initialization of the driver

The code sequence for initializing the ADC driver is as follows:

ADC driver

```

/*
Configuration values mandatory for below code snippet:-
1. AdcResultHandlingImplementation = ADC_DMA_MODE_RESULT_HANDLING: Then
Dma_Init() is required prior to use of runtime ADC services.
2. AdcStartupCalibApi = TRUE: Then Adc_GetStartupCalStatus() and
Adc_TriggerStartupCal() can be invoked
*/

#include "Adc.h"
#include "Mcu.h"
#include "Port.h"
#include "Dma.h"
#include "Irq.h"

/* MCU and GTM Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Port Initialization */
Port_Init(&Port_Config);

#if(ADC_RESULT_HANDLING_IMPLEMENTATION == ADC_DMA_MODE_RESULT_HANDLING)
/* Dma Initialization, Only if DMA mode of Result
handling is used */
Dma_Init(&Dma_Config);
#endif

/* ADC Initialization */
Adc_Init(&Adc_Config);

/* ADC Startup Calibration */
Adc_TriggerStartupCal();
/* Wait till the Start Calibration is over*/
while(Adc_GetStartupCalStatus() != ADC_STARTUP_CALIB_OVER);

/* Initialize the SRPN and TOS for used interrupts */
IrqAdc_Init();

/* Enable Interrupts for used ADC HW units(x) */
SRC_VADC_Gx_SR0.B.SRE = 1U;
SRC_VADC_Gx_SR1.B.SRE = 1U;
SRC_VADC_Gx_SR2.B.SRE = 1U;

#if((ADC_RESULT_HANDLING_IMPLEMENTATION == ADC_DMA_MODE_RESULT_HANDLING) ||
(ADC_ENABLE_LIMIT_CHECK == STD_ON))
SRC_VADC_Gx_SR3.B.SRE = 1U;
#endif

```

ADC driver

```
/* Further APIs of ADC driver can be called now */
```

Start software-triggered conversion

The code sequence for starting a software triggered group conversion is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1. AdcEnableStartStopGroupApi = True
2. AdcGrpNotifCapability = True
3. AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING
*/

#include "Adc.h"
Adc_GroupType Group ;
Adc_ValueGroupType DataBufferPtr[10]; //Assuming buffer size of 10 is
sufficient for the AdcChannel group being depicted
Std_ReturnType lRetVal;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid SW group ID macro
generated in Adc_Cfg.h */
Group = AdcConf_AdcGroup_AdcXGroup_Y;

lRetVal = Adc_SetupResultBuffer(Group, &DataBufferPtr[0]);

if(lRetVal != E_NOT_OK)
{
    Adc_EnableGroupNotification(Group);
    Adc_StartGroupConversion(Group);
}
else
{
    /*Could not setup result buffer*/
}
```

Stop software-triggered conversion

ADC driver

The code sequence for stopping a software-triggered group conversion is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1. AdcEnableStartStopGroupApi = True
*/

#include "Adc.h"
Adc_GroupType Group ;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid SW group ID macro
generated in Adc_Cfg.h */
Group = AdcConf_AdcGroup_AdcXGroup_Y;

/*Make sure Group has already been started by calling
Adc_StopGroupConversion API*/

Adc_StopGroupConversion(Group);
```

Enable hardware trigger for conversion

ADC driver

The code sequence for enabling a hardware trigger for a group conversion is as follows:

```

/*
Configuration values mandatory for below code snippet:-
1.AdcHwTriggerApi = True
2.AdcEnableStartStopGroupApi = True
3.AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING
*/

#include "Adc.h"
Adc_GroupType Group ;
Adc_ValueGroupType DataBufferPtr[10]; //Assuming buffer size of 10 is
sufficient for the AdcChannel group being depicted
Std_ReturnType lRetVal;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid HW group ID macro generated in
Adc_Cfg.h */

Group = AdcConf_AdcGroup_AdcXGroup_Y;

lRetVal = Adc_SetupResultBuffer(Group, &DataBufferPtr[0]);

if(lRetVal != E_NOT_OK)
{
    Adc_EnableGroupNotification(Group);
    Adc_EnableHardwareTrigger(Group);
}
else
{
    /*Could not setup result buffer*/
}

```

Disable hardware trigger for conversion

The code sequence for disabling a hardware trigger for a group conversion is as follows:

```

/*
Configuration values mandatory for below code snippet:-
1. AdcHwTriggerApi = True
*/
#include "Adc.h"
Adc_GroupType Group ;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid HW group ID */
Group = AdcConf_AdcGroup_AdcXGroup_Y;

/*Make sure Group has already been started by calling Adc_
EnableHardwareTrigger API*/

/* Disable the HW trigger */
Adc_DisableHardwareTrigger(Group);

```

ADC driver

Enable hardware trigger for conversion with DMA result handling

The code sequence for enabling a hardware trigger for a group conversion in the DMA mode of result handling is as follows:

```

/*
Configuration values mandatory for below code snippet:-
1. AdcHwTriggerApi = True
2. AdcEnableStartStopGroupApi = True
3. AdcResultHandlingImplementation = ADC_DMA_MODE_RESULT_HANDLING
*/

#include "Adc.h"
#include "Dma.h"
Adc_GroupType Group ;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid HW group ID macro
generated in Adc_Cfg.h */

Group = AdcConf_AdcGroup_AdcXGroup_Y;
/* DMA channel initialization is already completed as part of Dma_Init*/
/* Update the source address to SFR address of result register and destination
address to RAM buffer */
Dma_ChUpdate(0U, PointerToChannelUpdateStruct, NULL_PTR)
/* Enable HW trigger for the DMA Channel used */
Dma_ChEnableHardwareTrigger(0U);

/* Enable notification and HW trigger for the Group*/
Adc_EnableGroupNotification(Group);
Adc_EnableHardwareTrigger(Group);

```

Read results

ADC driver

The code sequence for reading results of completed conversions is as follows:

```

/*
Configuration values mandatory for below code snippet:-
1. AdcReadGroupApi = True
2. AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING
*/

#include "Adc.h"
Adc_GroupType Group ;
Std_ReturnType Read_Err;
Adc_StatusType lRetVal;
Adc_ValueGroupType DataBufferPtr[10]; //Assuming buffer size of 10 is
sufficient for the AdcChannel group being depicted

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid group ID macro generated in
Adc_Cfg.h */

Group = AdcConf_AdcGroup_AdcXGroup_Y;

lRetVal = Adc_GetGroupStatus(Group);

if((lRetVal == ADC_STREAM_COMPLETED) || (lRetVal == ADC_COMPLETED))
{
    Read_Err = Adc_ReadGroup (Group, &DataBufferPtr[0]);
    if(Read_Err != E_NOT_OK)
    {
        /*Adc read group is successful*/
    }
}
else
{
    /*Results not ready*/
}

```

De-initialization of the driver

The code sequence for de-initialization of the driver is as follows:

```

/*
Configuration values mandatory for below code snippet:-
1. AdcDeInitApi = True
*/
#include "Adc.h"

/*Stop all the currently converting groups before calling Adc De-init so that
De-Initialization can go on successfully */

/* ADC De-Initialization */
Adc_DeInit();

```

Configuring an ADC channel group for GTM-based trigger and gate:

ADC driver

ADC channel groups conversions can be triggered on a timer event from the GTM Channel. The conversion for ADC channels in such a group occur every time a GTM timer event is detected. To address multiple application use cases the ADC driver provides multiple ways to configure the GTM channels as follows:

- **Time-based trigger using GTM**

In this mode, the ADC driver configures the GTM channel (through the APIs of MCU) every time the `Adc_EnableHardwareTrigger()` API is called for a group having GTM as a trigger source.

- The GTM channel intended to be used must be exclusively reserved in the MCU driver through the configuration parameters under the `McuGtmAllocationConf` container.
- The multiplexer for GTM to EVADC trigger connections must be configured in the MCU driver through the configuration parameters listed under the `GtmTriggerForAdc` container.
- The clock, related configuration of GTM must be done through the MCU driver configuration.
- The `AdcHwTriggerApi` configuration parameter must be set to TRUE.
- The correct GTM trigger line must be selected in the `AdcGroup/AdcHwExtTrigSelect` parameter.
- The edge of the external request signal must be selected in the `AdcGroup/AdcHwTrigSignal` parameter.
- The GTM timer and trigger interval must be configured in the container `AdcGroup/GtmTriggerTimerConfig`.

- **PWM signal-based trigger using GTM**

In this mode, the ADC driver relies upon the application or another driver to configure the GTM channel that generates the PWM signal.

- The GTM timer intended for use must be configured by the user outside the ADC driver.
- The multiplexer for GTM to EVADC trigger connections must be configured in the MCU driver through the configuration parameters listed under the `GtmTriggerForAdc` container.
- The clock, related configuration of GTM must be done through the MCU driver configuration.
- The `AdcHwTriggerApi` configuration parameter must be set to true.
- The correct GTM trigger line must be selected in the `AdcGroup/AdcHwExtTrigSelect` parameter.
- The edge of the external request signal must be selected in the `AdcGroup/AdcHwTrigSignal` parameter.

Similarly for gate signals, the GTM channel may be configured through the ADC driver, application or any other driver based on the use cases.

Configuring an ADC group for synchronous conversion

The ADC driver supports synchronous conversion of channels across the ADC hardware groups. This feature can be enabled through the `AdcSyncConvEnable`, `AdcSyncConvMode` and `AdcSyncConvChannelEnable` configuration parameters. The `AdcSyncConvMode` parameter determines the master and slave hardware groups. When an ADC channel group is triggered on the master hardware group then all the channels marked for synchronous conversion through `AdcSyncConvChannelEnable` will be triggered on the slave hardware groups also.

The driver configures the slave analog channels (CHCTR) with the same properties as that of the master analog channel. Hence, the input class used for the slave analog channels must have the same properties as the input class used for the master analog channel.

It is recommended to use the global input classes for the master channel, so that the slave is also configured with the same input class, eliminating different conversion properties. The conversion results for the channels of the slave hardware group will be stored in the application buffer after the conversion results for the channels of the master hardware group.

The buffer layout is explained in the diagram for the following scenario.

- Three channels are configured in an ADC channel group: CH9, CH1, and CH2
- Two streaming samples for each channel
- Kernel K0 synchronization master, kernel K1 and K2 synchronization slaves

ADC driver

- Channels CH9 and CH1 enabled for synchronous conversion

Note: If a channel on a slave kernel is converting or queued for conversion, the groups from the master kernel having the same slave channel configured as synchronous cannot be queued or started. Such a scenario triggers a DET.

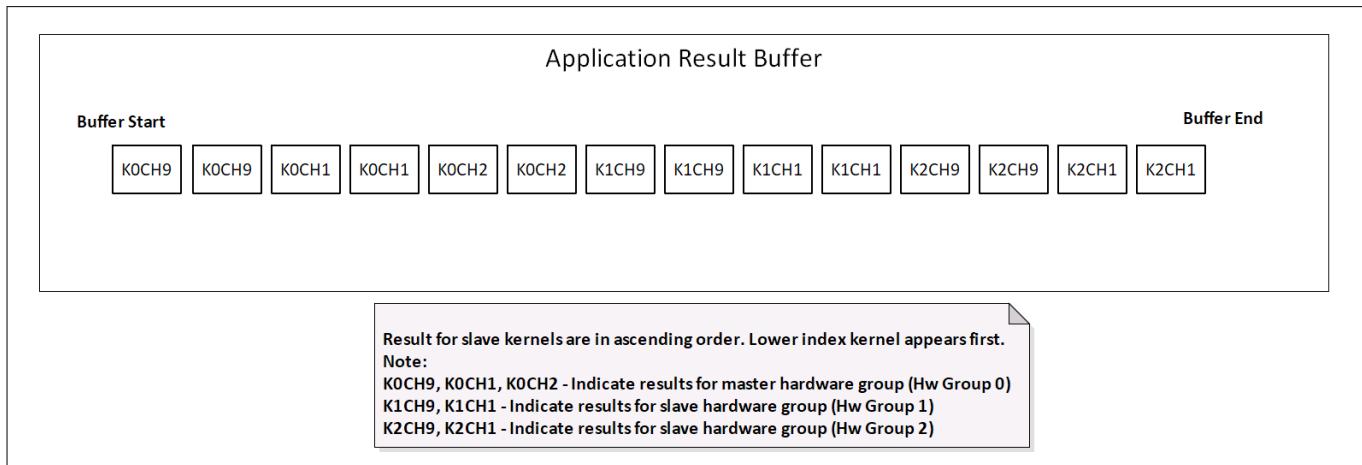


Figure 78 Synchronous conversion

3.1.5 Key architectural considerations

3.1.5.1 Modes of operation: priority and queuing

The ADC driver supports four modes of operation related to priority and queuing. The modes can be changed at pre-compile time by assigning appropriate value to AdcEnableQueueing and AdcPriorityImplementation. These modes are as follows:

- **Priority and queuing disabled**

In this mode of operation, only one AdcChannel group can execute on a particular ADC hardware group at a given time. Groups can be scheduled in parallel on different ADC hardware groups.

Configuration settings

- AdcEnableQueueing: False
- AdcPriorityImplementation: ADC_PRIORITY_NONE

- **Priority disabled and queuing enabled**

The ADC driver maintains one software queue per ADC hardware group in this mode of operation. The software queue operates in the FCFS mode. An AdcChannel group cannot be placed in the software queue, if it is already present in the queue or currently converting or waiting for hardware trigger.

Configuration settings

- AdcEnableQueueing: True
- AdcPriorityImplementation: ADC_PRIORITY_NONE

- **Hardware priority**

The prioritization mechanism used in this mode is completely based on the priority of the Request Sources (hardware arbiter). Request sources have fixed priority: RS0 has a priority level of 0 (lowest), RS1 has a priority level of 1, and RS2 has a priority level of 2 (highest). Since the ADC hardware supports only three priority levels, a mapping with AdcChannel group priority is established as follows:

- AdcChannel group priority number 0..253 maps to hardware priority 0 (lowest) and executes only on RS0

ADC driver

- AdcChannel group priority 254 maps to hardware priority 1 and executes only on RS1
- AdcChannel group priority 255 maps to hardware priority 2 (highest) and executes only on RS2

In this mode of operation, the AdcChannel group will be placed in the software queue if another AdcChannel group with the same priority is currently executing. The software queue operates in the FCFS mode. Hence, groups with the same priority in the queue will be executed in the order of the request. An AdcChannel group cannot be placed in the software queue, if it is already present in the queue or currently converting or triggered by hardware mechanism or sharing an analog channel / result register with another executing groups. Hardware triggered groups should never be placed in any software queue.

Configuration settings

- AdcEnableQueuing: False
- AdcPriorityImplementation: ADC_PRIORITY_HW

- **Hardware software priority**

This is the full-fledged prioritization mechanism using both hardware and software priority mechanisms. Request sources do not have a fixed priority in this mode. The AdcChannel groups can be allocated priority levels from 0 to 255 (highest). In this mode of operation, the ADC driver maintains a prioritized software queue per ADC hardware group. An AdcChannel group cannot be placed in the software queue, if it is already present in the queue or currently converting or triggered by hardware mechanism or sharing a analog channel/result register with another executing groups. Hardware-triggered groups should never be placed in any software queue.

Configuration settings

- AdcEnableQueuing: False
- AdcPriorityImplementation: ADC_PRIORITY_HW_SW

3.1.5.2 Modes of operation: result handling

The ADC driver supports two modes of operation related to result handling. The modes can be changed at pre-compile time by assigning appropriate value to AdcResultHandlingImplementation. The modes are listed as follows:

- **Interrupt-based result handling**

In this mode, the conversion results are transferred from the result registers to the application buffers in the ISR of the Request Source Event.

Each request source of an ADC hardware group is assigned a dedicated service request line.

The channel event generated as a result of the limit check feature is also assigned a dedicated service request line.

Configuration settings

- AdcResultHandlingImplementation: ADC_INTERRUPT_MODE_RESULT_HANDLING

- **DMA-based result handling**

In this mode, the conversion results are transferred from the result registers to the application buffers by a DMA move engine. When the DMA mode is enabled, priority and queuing are not supported (restricted through the tresos). Also, since the results are transferred to the application through the DMA, APIs related to the result buffer management and result reporting will not be available (restricted through the tresos). For more information, refer to the *DMA support* section.

Configuration settings

- AdcResultHandlingImplementation: ADC_DMA_MODE_RESULT_HANDLING

ADC driver

3.1.5.3 Re-entrancy of APIs

The ADC driver functions are re-entrant if the functions are called for different ADC channel groups. The re-entrancy of the API functions applies only if the caller takes care that there is no simultaneous invocation of different APIs for the same ADC Channel group. This is applicable for all API functions taking ADC Channel group as an input argument.

3.1.5.4 Accessing shared SFR

The ADC driver updates the SFR related to Service Request node and ERU. These SFR may be updated by the application software also. Hence, these updates must be done in a critical section or atomically.

- **Accessing ERU registers:** The ADC driver updates the MODULE_SCU.EICR and MODULE_SCU.IGCR registers to configure the trigger and gating signals. The update to these registers are done atomically by the driver. Any update to the MODULE_SCU.EICR and MODULE_SCU.IGCR by the application should be performed atomically, if the same register is used by the ADC driver also. This is required since two ERU channels share a common register. Therefore, update for one channel should not corrupt the ongoing write for another channel.
- **Accessing SRC registers:** The ADC driver updates the SRC registers of EVADC (MODULE_SRC.VADC.G[x].SRy) to clear the pending interrupt requests. The update to this register is done under a critical section using SchM locks. The application should update the MODULE_SRC.VADC.G[x].SRy under the SchM locks using the `SchM_Enter_Adc_SrcRegAccess()` and `SchM_Exit_Adc_SrcRegAccess()` functions. This will avoid corruption of the register, if updated by the driver and the application simultaneously.

3.1.5.5 Hardware trigger and gate mechanism

The ADC driver supports various hardware trigger and gate mechanisms to control the start or stop of the conversions.

- **GTM-based trigger and gate signal**

The ADC driver supports the trigger and gate through the GTM-TOM and GTM-ATOM channels. The driver provides the following flexibility to the user for configuring the GTM channel for trigger and gate signals:

- GTM channel can be configured through the tresos configuration parameters provided in the ADC driver. In such a case, the driver invokes APIs provided by the MCU driver to initialize the configured GTM channel at runtime.
- Configuration and initialization of the GTM channels for trigger and gate signal must be done by the user if the parameters related to the GTM trigger and gate signals are not configured inside the ADC driver.

- **ERU-based trigger and gate signal**

The ADC driver supports the trigger and gate through the ERU channels. The ERU channel must be configured through the tresos configuration parameters provided in the ADC driver. The initialization of the configured ERU channels is done at runtime by the ADC driver for the groups using ERU for trigger or gating.

ADC driver**3.2 Assumptions of Use (AoUs)**

The AoUs for the driver are as follows:

- **Correct group ID passed for runtime APIs**

The users of the ADC driver shall verify that the symbolic-name macros generated for ADC channel groups in the Adc_Cfg.h file holds the correct values, and use these macros when invoking the APIs of the driver.

[cover parentID ADC={97D3CDCB-F681-4523-B8FC-63A146B4D934}]

- **Correct power state passed for runtime APIs**

The users of the ADC driver shall verify that the symbolic-name macros generated for power states in the Adc_Cfg.h file holds the correct values, and use these macros when invoking APIs of the driver.

[cover parentID ADC={E345A665-AF41-4ad1-BF51-80987854F29D}]

- **Correctness of configuration pointer**

The users of the ADC driver shall ensure that correct pointer to the configuration structure is passed for initializing the driver.

[cover parentID ADC={1BAF8095-53CB-4ba6-A271-F4F686DE0A81}]

- **DMA channel initialization sequence**

The users of the ADC driver shall ensure that when the DMA mode of result handling is used, the DMA channel is setup-initialized before starting the ADC conversions. Also ensure that the DMA channel is stopped/deinitialized after stopping the ADC conversion.

[cover parentID ADC={F0B13815-84DD-4cc7-9A20-9542C8C85D2F}]

- **Priority of hardware trigger group**

The users of the ADC driver shall ensure that the priority of hardware-triggered group is higher compared to the software-triggered group in the continuous conversion mode. This ensures that hardware triggers are not missed when higher priority software-triggered groups are executing.

[cover parentID ADC={F5565B70-99D3-4fcf-80EA-B54CDFB19754}]

- **Groups using ERU as hardware trigger**

ADC channel groups having the same ERU (EICR/IGCR) channel as trigger/gating source shall not be triggered simultaneously. The driver does not perform any check if the ERU channel requested currently is already in use by another ADC channel group.

[cover parentID ADC={C8281063-A66F-4217-B299-21D0F17F33C9}]

- **Groups using GTM as hardware trigger**

ADC channel groups having the same GTM (TOM/ATOM) channel as trigger/gating source shall not be triggered simultaneously. The driver does not perform any check if the GTM channel requested currently is already in use by another ADC channel group.

[cover parentID ADC={1CB59FC5-2B01-4dbe-B658-454EE965A81F}]

- **Start-up calibration invocation-1**

The users of the ADC driver shall ensure that no conversion is ongoing before triggering the start-up calibration operation or started during the start-up calibration process. The status of the start-up calibration shall be checked through the Adc_GetStartupCalStatus API.

[cover parentID ADC={CE7E70BA-A387-4930-A6A3-0FF2BCF49550}]

- **Start-up calibration invocation-2**

The users of the ADC driver shall ensure that the Adc_TriggerStartupCal() API is invoked after completing the initialization and initialization check on all the cores.

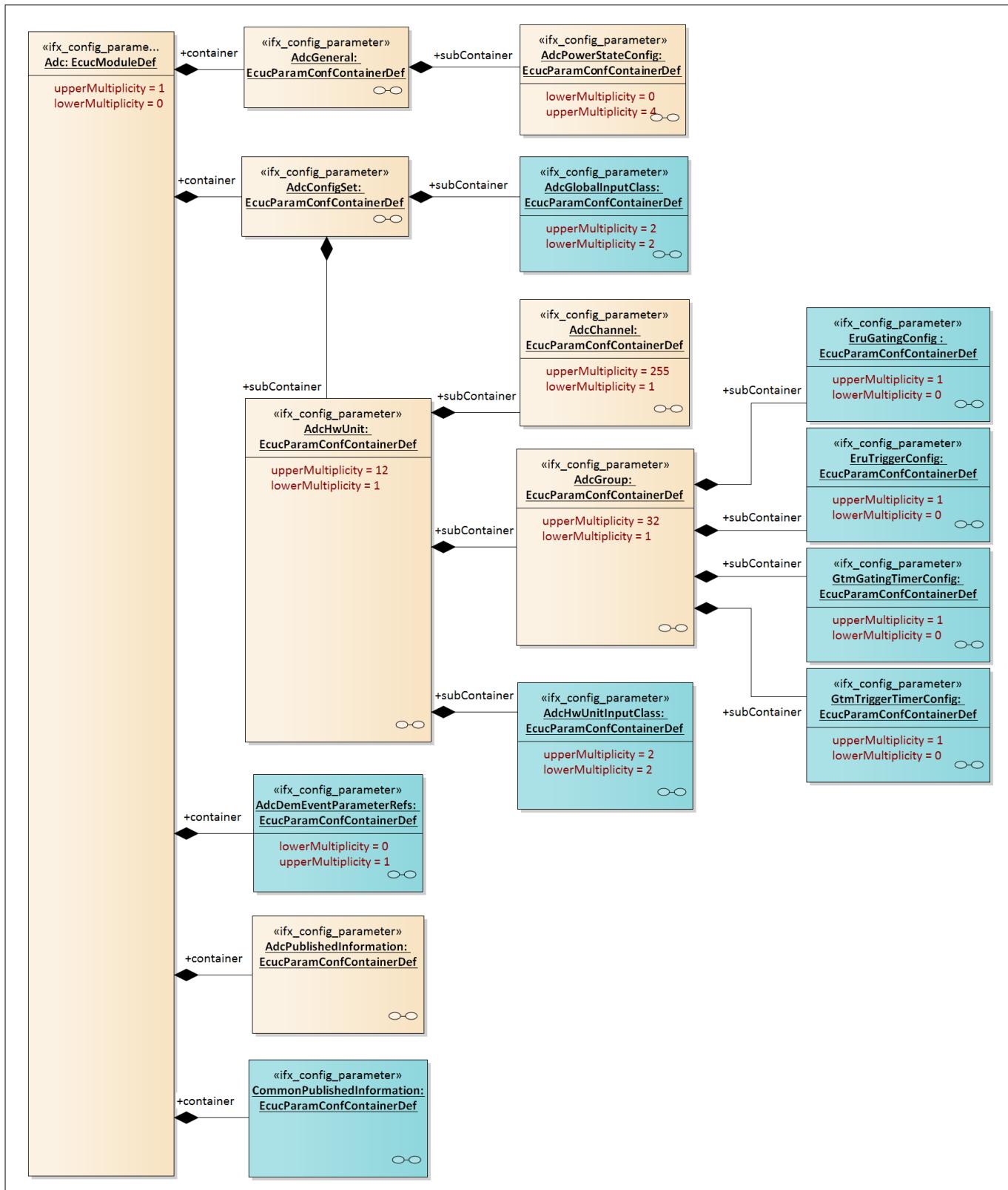
[cover parentID ADC={E27FCBC4-A8EC-4edf-859A-95654C24D655}]

- **Valid pointer and result buffer pointers passed**

ADC driver

The users of the ADC driver shall ensure a valid pointer and result buffer pointer with the required size is passed to achieve the expected behavior. The driver does not perform any check on the size of the buffers and pointers.

[cover parentID ADC={094F8CCC-0436-4bde-A6A0-7A93DD4A5055}]

ADC driver**3.3 Reference information****3.3.1 Configuration interfaces****Figure 79****Container hierarchy along with their configuration parameters**

ADC driver**3.3.1.1 Container: Adc**

Configuration of the ADC module

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.1.1 Config Variant**Table 12 Specification for Config Variant**

Name	Config Variant		
Description	Selects the config-variant for the ADC module. The default value of this parameter is set to VariantPostBuild as per AUTOSAR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VariantPostBuild: Post Build Support		
Default value	VariantPostBuild		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.2 Container: AdcChannel

This container contains the channel configuration (parameters). The short-name of all AdcChannels must be unique across the AdcHwUnits. Same AdcChannel can be assigned to multiple AdcGroups belonging to the same AdcHwUnit. The upper multiplicity of this container is restricted to a maximum of 255 logical channels per ADC hardware group. Note: Since AdcChannel can be a part of several AdcGroups, this container is not realized as a sub-container of AdcGroup but instead as a sub-container of AdcHwUnit.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.2.1 AdcAnChannelNum**Table 13 Specification for AdcAnChannelNum**

Name	AdcAnChannelNum		
Description	Parameter defines the analog channel number as per the hardware.. The possible range of values for analog channels is dependent on the value of the AdcHwUnitId. The default value of this parameter is the first physical channel for each hardware group.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

ADC driver**Table 13 Specification for AdcAnChannelNum (continued)**

Range	G[x]CH[y]: where x=Hardware Unit ID y=Channel Number		
Default value	G[x]CH[y]		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcHwUnitId		

3.3.1.2.2 AdcBWDEnable**Table 14 Specification for AdcBWDEnable**

Name	AdcBWDEnable		
Description	Parameter enables or disables the broken wire detection feature for the channel.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.2.3 AdcBWDPrechargeLevel**Table 15 Specification for AdcBWDPrechargeLevel**

Name	AdcBWDPrechargeLevel		
Description	Parameter determines the VAREF/VAGND pre-charging for the broken wire detection channel. The configuration of pre-charging for broken wire detection channel is editable only when AdcBWDEnable is enabled.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_BWD_PRECH_VAGND: VAGND is used for pre-charging.		

ADC driver**Table 15 Specification for AdcBWDPrechargeLevel (continued)**

	ADC_BWD_PRECH_VAREF: VAREF is used for pre-charging.		
Default value	ADC_BWD_PRECH_VAGND		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcBWDEnable		

3.3.1.2.4 AdcChannelConvTime**Table 16 Specification for AdcChannelConvTime**

Name	AdcChannelConvTime		
Description	<p>Parameter defines the conversion time during which the analog value is converted into its digital representation for each channel.</p> <p>AdcInputClassSelection controls the channel conversion time. Therefore, this parameter is disabled in tresos and cannot be edited.</p> <p>Note: The maximum value of this parameter is specified based on the maximum value generated from arxml.</p>		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	0 - 9223372036854775807		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.2.5 AdcChannelHighLimit**Table 17 Specification for AdcChannelHighLimit**

Name	AdcChannelHighLimit		
Description	<p>Parameter defines the upper limit used for limit checking.</p> <p>1. This parameter is configurable only if AdcChannelLimitCheck is set to TRUE, and AdcChannelRangeSelect is not equal to ADC_RANGE_UNDER_LOW and AdcChannelRangeSelect is not equal to ADC_RANGE_NOT_UNDER_LOW.</p>		

ADC driver**Table 17 Specification for AdcChannelHighLimit (continued)**

	2. AdcChannelHighLimit value should be greater than or equal to AdcChannelLowLimit. The default and the maximum value of this parameter is added based on the 12-bit ADC converter value supported by the hardware.		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	0 - 4095		
Default value	4095		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcChannelLowLimit, AdcChannelRangeSelect, AdcChannelLimitCheck		

3.3.1.2.6 AdcChannelId**Table 18 Specification for AdcChannelId**

Name	AdcChannelId		
Description	Parameter defines the numeric ID of the channel, which is the logical ID for the channel. It must be in consecutive sequence starting from zero for each ADC hardware unit. A symbolic name is generated for each channel. Note: The maximum number of logical channels per hardware group is 255 (ID 0 to ID 254)		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 254		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.2.7 AdcChannelLimitCheck**Table 19 Specification for AdcChannelLimitCheck**

Name	AdcChannelLimitCheck
Description	Parameter enables or disables the limit checking feature for the current channel.

ADC driver**Table 19 Specification for AdcChannelLimitCheck (continued)**

	<p>This parameter is set to FALSE by default and it can be TRUE only if AdcEnableLimitCheck is set to TRUE. ADC channels with limit checking feature enabled have to be assigned to AdcGroup which consists exactly of one AdcChannel (itself).</p> <p>Note: As per AUTOSAR, this is a pre-compile parameter. Since all channel related parameters are post-build, it is simpler software design to implement this parameter also as post-build instead of pre-compile.</p>		
Multiplicity	0..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcEnableLimitCheck		

3.3.1.2.8 AdcChannelLowLimit**Table 20 Specification for AdcChannelLowLimit**

Name	AdcChannelLowLimit		
Description	<p>Parameter defines the lower limit used for limit checking.</p> <ol style="list-style-type: none"> 1. This parameter is configurable only if AdcChannelLimitCheck is set to TRUE, and AdcChannelRangeSelect is not equal to ADC_RANGE_OVER_HIGH and AdcChannelRangeSelect is not equal to ADC_RANGE_NOT_OVER_HIGH. 2. AdcChannelLowLimit value should be less than or equal to AdcChannelHighLimit. The default and the maximum value of this parameter is added based on the 12-bit ADC converter value supported by the hardware. 		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	0 - 4095		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcChannelHighLimit, AdcChannelRangeSelect, AdcChannelLimitCheck		

ADC driver**3.3.1.2.9 AdcChannelRangeSelect****Table 21 Specification for AdcChannelRangeSelect**

Name	AdcChannelRangeSelect		
Description	<p>Parameter defines which conversion values are taken into account related to the limits defined with AdcChannelLowLimit and AdcChannelHighLimit. This parameter is configurable only if AdcChannelLimitCheck is set to TRUE.</p> <p>The default value of this parameter is set to ADC_RANGE_ALWAYS which effectively considers all the values as valid, as this is the behavior when limit check is disabled.</p>		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	<p>ADC_RANGE_ALWAYS: Complete range - independent from channel limit settings.</p> <p>ADC_RANGE_BETWEEN: Range between low limit and high limit - high limit value included.</p> <p>ADC_RANGE_NOT_BETWEEN: Range above high limit or below low limit - low limit value included.</p> <p>ADC_RANGE_NOT_OVER_HIGH: Range below high limit - high limit value included.</p> <p>ADC_RANGE_NOT_UNDER_LOW: Range above low limit.</p> <p>ADC_RANGE_OVER_HIGH: Range above high limit.</p> <p>ADC_RANGE_UNDER_LOW: Range below low limit - low limit value included.</p>		
Default value	ADC_RANGE_ALWAYS		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcChannelLimitCheck		

3.3.1.2.10 AdcChannelRefVoltsrcHigh**Table 22 Specification for AdcChannelRefVoltsrcHigh**

Name	AdcChannelRefVoltsrcHigh		
Description	<p>This parameter defines the upper reference voltage source for each channel. For Channel 0, value of this parameter should always be ADC_USE_VREF. AdcChannelRefVoltsrcHigh cannot have value as ADCUSES_CH0, if AdcAnChannelNum contains a channel with fixed reference.</p>		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	<p>ADCUSES_CH0: Use channel CH0 as a reference voltage for conversion.</p> <p>ADCUSES_VREF: Use Analog reference voltage as a reference voltage for conversion.</p>		
Default value	ADCUSES_VREF		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE

ADC driver**Table 22 Specification for AdcChannelRefVoltsrcHigh (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcAnChannelNum		

3.3.1.2.11 AdcChannelRefVoltsrcLow**Table 23 Specification for AdcChannelRefVoltsrcLow**

Name	AdcChannelRefVoltsrcLow		
Description	<p>Parameter defines the ground reference voltage source for each channel.</p> <p>The default value of this parameter is added to ADC_USES_VAGND as only EVADC analog ground pin is supported by the hardware.</p>		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	ADC_USES_VAGND: Use analog reference ground as a reference voltage for conversion.		
Default value	ADC_USES_VAGND		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.2.12 AdcChannelResolution**Table 24 Specification for AdcChannelResolution**

Name	AdcChannelResolution		
Description	<p>This parameter defines the channel resolution in bits. The converters operates only in 12-bit mode and hence the driver will update the result buffers with 12-bit conversion values only. Hence, this parameter will be un-editable is tresos and holds a default value of 12.</p>		
Multiplicity	0..1	Type	EcucIntegerParamDef
Range	1 - 63		
Default value	12		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE

ADC driver**Table 24 Specification for AdcChannelResolution (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.2.13 AdcChannelSampTime**Table 25 Specification for AdcChannelSampTime**

Name	AdcChannelSampTime		
Description	<p>Parameter defines configuration of sampling time during which the value is sampled for each channel. AdcInputClassSelection controls the channel sampling time. Therefore, the AdcChannelSampTime parameter is disabled in tresos and cannot be edited.</p> <p>Note: Maximum value of this parameter is specified based on the maximum value generated from arxml.</p>		
Multiplicity	0..1	Type	EcucIntegerParamDef
Range	0 - 9223372036854775807		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.2.14 AdcInputClassSelection**Table 26 Specification for AdcInputClassSelection**

Name	AdcInputClassSelection		
Description	<p>Parameter defines the input class for the analog channel. Input class determines the channel conversion properties such as sample time and conversion mode for each channel.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_GLOBAL_CLASS_0: Global Class 0 is used ADC_GLOBAL_CLASS_1: Global Class 1 is used ADC_HWUNIT_CLASS_0: Hardware Unit Class 0 is used ADC_HWUNIT_CLASS_1: Hardware Unit Class 1 is used		

ADC driver**Table 26 Specification for AdcInputClassSelection (continued)**

Default value	ADC_HWUNIT_CLASS_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.2.15 AdcSyncConvChannelEnable**Table 27 Specification for AdcSyncConvChannelEnable**

Name	AdcSyncConvChannelEnable		
Description	<p>Parameter enables the synchronous A to D conversion for the current channel. Conversion of this channel will trigger a simultaneous conversion for the analog channel with the same number on the slave hardware groups also.</p> <p>The default value of this parameter is set to FALSE. This parameter is editable only for channels of AdcHwUnit configured as ADC_SYNC_MASTER.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcSyncConvMode		

3.3.1.3 Container: AdcConfigSet

Container contains the configuration parameters and sub containers of the ADC module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

ADC driver**3.3.1.3.1 AdcSyncClockDisable****Table 28 Specification for AdcSyncClockDisable**

Name	AdcSyncClockDisable		
Description	<p>Parameter determines whether the analog clock is generated in synchronized mode or unsynchronized mode. The Converter Control (CONVCTRL) clock is configured by the MCU driver.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p> <p>Note: This parameter writes into the common SFRs of all the kernels and can be implemented as post-build. Hence it is represented inside the AdcConfigSet container.</p>		
Multiplicity	1..1		
Type	EcucBooleanParamDef		
Range	TRUE		
	FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.3.2 AdcSystemClock**Table 29 Specification for AdcSystemClock**

Name	AdcSystemClock		
Description	<p>Parameter refers to the system clock configured by the MCU driver. With this system clock, the reload value corresponding to the configured timer is calculated for the timer triggered AdcChannel groups.</p> <p>Note: This parameter writes into common SFRs of all the kernels and can be implemented as post-build. Hence it is represented inside the AdcConfigSet container.</p>		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

ADC driver**Table 29 Specification for AdcSystemClock (continued)**

Dependency	-
-------------------	---

3.3.1.4 Container: AdcDemEventParameterRefs

Container lists the production errors supported by the ADC driver. This container must be present when safety check is enabled.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.4.1 AdcClcFailureNotification**Table 30 Specification for AdcClcFailureNotification**

Name	AdcClcFailureNotification		
Description	Parameter defines whether CLC failure DEM notification is enabled or not. This parameter must be present when safety check is enabled.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	AdcSafetyEnable		

3.3.1.4.2 AdcConvStopTimeNotification**Table 31 Specification for AdcConvStopTimeNotification**

Name	AdcConvStopTimeNotification		
Description	Parameter determines whether conversion stop failure DEM notification is enabled or not. This parameter must be present when safety check is enabled.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE

ADC driver**Table 31 Specification for AdcConvStopTimeNotification (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	AdcSafetyEnable		

3.3.1.5 Container: AdcGeneral

Container contains all the general configuration parameters for the ADC driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.5.1 AdcDeInitApi**Table 32 Specification for AdcDeInitApi**

Name	AdcDeInitApi		
Description	Parameter adds or removes the Adc_DeInit() API from the code. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.5.2 AdcDevErrorDetect**Table 33 Specification for AdcDevErrorDetect**

Name	AdcDevErrorDetect		
Description	Parameter enables or disables the Default Error Tracer (DET) detection and reporting.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE		

ADC driver**Table 33 Specification for AdcDevErrorDetect (continued)**

	FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.5.3 AdcEnableLimitCheck**Table 34 Specification for AdcEnableLimitCheck**

Name	AdcEnableLimitCheck		
Description	Parameter enables or disables the limit checking feature in the ADC driver. It cannot be set to TRUE, when DMA mode of result handling is selected. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcResultHandlingImplementation		

3.3.1.5.4 AdcEnableQueueing**Table 35 Specification for AdcEnableQueueing**

Name	AdcEnableQueueing
Description	Parameter determines whether the queuing mechanism is active or not. It is editable and evaluated only when AdcPriorityImplementation is set as ADC_PRIORITY_NONE and AdcEnableStartStopGroupApi is set as TRUE and AdcResultHandlingImplementation is ADC_INTERRUPT_MODE_RESULT_HANDLING The default value of this parameter is set to FALSE to minimize the executable code size.

ADC driver**Table 35 Specification for AdcEnableQueueing (continued)**

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcResultHandlingImplementation, AdcEnableStartStopGroupApi, AdcPriorityImplementation		

3.3.1.5.5 AdcEnableStartStopGroupApi**Table 36 Specification for AdcEnableStartStopGroupApi**

Name	AdcEnableStartStopGroupApi		
Description	Parameter determines if the software trigger mechanisms (Adc_StartGroupConversion() and Adc_StopGroupConversion()) are available at runtime. When configured as TRUE, these APIs are available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.5.6 AdcGrpNotifCapability**Table 37 Specification for AdcGrpNotifCapability**

Name	AdcGrpNotifCapability
-------------	-----------------------

ADC driver**Table 37 Specification for AdcGrpNotifCapability (continued)**

Description	Parameter determines if the group notification mechanisms (Adc_EnableGroupNotification() and Adc_DisableGroupNotification()) are available at runtime. When configured as TRUE, these APIs are available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.5.7 AdcHwTriggerApi**Table 38 Specification for AdcHwTriggerApi**

Name	AdcHwTriggerApi		
Description	Parameter determines if the hardware trigger mechanisms (Adc_EnableHardwareTrigger() and Adc_DisableHardwareTrigger()) are available at runtime . When configured as TRUE, these APIs are available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.5.8 AdcInitCheckApi****Table 39 Specification for AdcInitCheckApi**

Name	AdcInitCheckApi		
Description	<p>Parameter adds or removes the Adc_InitCheck() API from the code. When configured as TRUE, the API is available at runtime.</p> <p>The default value of this parameter is set to FALSE to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.5.9 AdcInitDeInitApiMode**Table 40 Specification for AdcInitDeInitApiMode**

Name	AdcInitDeInitApiMode		
Description	<p>Configuration parameter defines the privilege mode in which the initialization and de-initialization APIs would operate.</p> <p>Since the ADC driver accesses the SFRs, it is efficient to operate the ADC driver in supervisory mode than the USER1 mode. Hence, the default mode of operation is the supervisory mode.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>ADC_MCAL_SUPERVISOR: Operating mode used is Supervisory.</p> <p>ADC_MCAL_USER1: Operating mode used is USER1.</p>		
Default value	ADC_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.5.10 AdcLowPowerStatesSupport****Table 41 Specification for AdcLowPowerStatesSupport**

Name	AdcLowPowerStatesSupport		
Description	<p>Parameter adds or removes all power state management related APIs (Adc_SetPowerState, Adc_GetCurrentPowerState, Adc_GetTargetPowerState, Adc_PreparePowerState) from the code. When configured as TRUE, these APIs are available at runtime.</p> <p>The default value of this parameter is set to FALSE to minimize the executable code size.</p>		
Multiplicity	0..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.5.11 AdcMaxChConvTimeCount**Table 42 Specification for AdcMaxChConvTimeCount**

Name	AdcMaxChConvTimeCount		
Description	<p>Parameter determines the maximum channel conversion time amongst all the configured channels. This parameter value is used by the ADC driver while trying to stop an ongoing conversion. This will make sure that the currently converting channel completes the conversion after a stop request and EVADC hardware unit goes to idle state before a new conversion is started by the driver.</p> <p>The default value of this parameter is set to be greater than the default time to stop the conversion. The default stop time is based on the input class configuration of the hardware.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16962		
Default value	6000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

ADC driver**Table 42 Specification for AdcMaxChConvTimeCount (continued)**

Dependency	AdcEnableQueuing, AdcPriorityImplementation		
-------------------	---	--	--

3.3.1.5.12 AdcMultiCoreErrorDetect**Table 43 Specification for AdcMultiCoreErrorDetect**

Name	AdcMultiCoreErrorDetect		
Description	<p>This parameter enables or disables the Multicore related default error tracer (DET) detection and reporting. It is applicable only when DETs are enabled.</p> <p>When set to TRUE, detection and reporting of multi-core related errors is enabled.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcDevErrorDetect		

3.3.1.5.13 AdcPowerStateAsynchTransitionMode**Table 44 Specification for AdcPowerStateAsynchTransitionMode**

Name	AdcPowerStateAsynchTransitionMode		
Description	<p>Parameter enables or disables the support of asynchronous power state transition.</p> <p>Note: Since power state transitions are always synchronous, this parameter is disabled in tresos.</p>		
Multiplicity	0..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile

ADC driver**Table 44 Specification for AdcPowerStateAsynchTransitionMode (continued)**

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.5.14 AdcPriorityImplementation**Table 45 Specification for AdcPriorityImplementation**

Name	AdcPriorityImplementation		
Description	<p>Parameter determines whether a priority mechanism for prioritization of the conversion requests is available. Two types of prioritization mechanisms can be selected.</p> <p>The hardware prioritization mechanism (ADC_PRIORITY_HW) uses the ADC hardware features for prioritization.</p> <p>The mixed hardware and software prioritization mechanism (ADC_PRIORITY_HW_SW) uses the ADC hardware features and software logic for prioritization of AdcChannel groups.</p> <p>The group priorities for software triggered groups are typically configured with lower priority levels than the group priorities for hardware triggered groups.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>ADC_PRIORITY_HW: Only hardware priority mechanism is available</p> <p>ADC_PRIORITY_HW_SW: Both hardware and software priority mechanisms are available</p> <p>ADC_PRIORITY_NONE: Priority mechanism is not available</p>		
Default value	ADC_PRIORITY_NONE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.5.15 AdcReadGroupApi**Table 46 Specification for AdcReadGroupApi**

Name	AdcReadGroupApi		
Description	<p>Parameter adds or removes the Adc_ReadGroup() API from the code. When set to TRUE, the API is available at runtime.</p> <p>AdcReadGroupApi should be FALSE, when DMA mode of result handling is selected.</p> <p>The default value of this parameter is set to FALSE to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef

ADC driver**Table 46 Specification for AdcReadGroupApi (continued)**

Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcResultHandlingImplementation		

3.3.1.5.16 AdcResultAlignment**Table 47 Specification for AdcResultAlignment**

Name	AdcResultAlignment		
Description	Parameter determines whether the raw ADC results in the ADC result buffer are left aligned or right aligned. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_ALIGN_LEFT: Raw results are left aligned ADC_ALIGN_RIGHT: Raw results are right aligned		
Default value	ADC_ALIGN_RIGHT		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.5.17 AdcResultHandlingImplementation**Table 48 Specification for AdcResultHandlingImplementation**

Name	AdcResultHandlingImplementation		
Description	Parameter determines the result handling mechanism for the ADC driver.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

ADC driver**Table 48 Specification for AdcResultHandlingImplementation (continued)**

Range	ADC_DMA_MODE_RESULT_HANDLING: Conversion results are transferred using a DMA channel. ADC_INTERRUPT_MODE_RESULT_HANDLING: Conversion results are transferred in the Request Source ISR.		
Default value	ADC_INTERRUPT_MODE_RESULT_HANDLING		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcPriorityImplementation		

3.3.1.5.18 AdcRuntimeApiMode**Table 49 Specification for AdcRuntimeApiMode**

Name	AdcRuntimeApiMode		
Description	Parameter defines the privilege mode, in which, the runtime APIs would operate. Since the ADC driver accesses the SFRs, it is efficient to operate the ADC driver in supervisory mode. Hence, the default mode of operation is supervisory mode. AdcRuntimeApiMode must be configured as User-1 mode if AdcInitDelnitApiMode is configured as User-1 mode.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_MCAL_SUPERVISOR: Operating mode used is Supervisory. ADC_MCAL_USER1: Operating mode used is USER1.		
Default value	ADC_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcInitDelnitApiMode		

3.3.1.5.19 AdcSafetyEnable**Table 50 Specification for AdcSafetyEnable**

Name	AdcSafetyEnable
Description	Parameter determines whether to enable or disable the safety check and reporting.

ADC driver**Table 50 Specification for AdcSafetyEnable (continued)**

	The detection of safety related errors is enabled, by default, to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.5.20 AdcSleepMode**Table 51 Specification for AdcSleepMode**

Name	AdcSleepMode		
Description	Parameter determines whether ADC driver accepts or rejects the sleep mode request from SCU. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_SLEEP_MODE_ACCEPT: Sleep Mode request from SCU is accepted ADC_SLEEP_MODE_REJECT: Sleep Mode request from SCU is rejected		
Default value	ADC_SLEEP_MODE_ACCEPT		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.5.21 AdcStartupCalibApi**Table 52 Specification for AdcStartupCalibApi**

Name	AdcStartupCalibApi
-------------	--------------------

ADC driver**Table 52 Specification for AdcStartupCalibApi (continued)**

Description	Parameter adds or removes the Adc_GetStartupCalStatus() and Adc_TriggerStartupCal() APIs from the code. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.5.22 AdcSupplyVoltage**Table 53 Specification for AdcSupplyVoltage**

Name	AdcSupplyVoltage		
Description	Parameter adjusts the analog circuitry to the supply voltage. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_VOLTAGE_3P3V: Fixed 3.3V power supply is connected ADC_VOLTAGE_5V: Fixed 5V power supply is connected ADC_VOLTAGE_CONTROLLED_BY_SUPPLY: Automatic control: Voltage range is controlled by power supply.		
Default value	ADC_VOLTAGE_CONTROLLED_BY_SUPPLY		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.5.23 AdcSyncConvEnable****Table 54 Specification for AdcSyncConvEnable**

Name	AdcSyncConvEnable		
Description	Parameter enables or disables the synchronous conversions across ADC hardware groups. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcResultHandlingImplementation		

3.3.1.5.24 AdcTriggerOneConversionEnable**Table 55 Specification for AdcTriggerOneConversionEnable**

Name	AdcTriggerOneConversionEnable		
Description	This parameter enables the execution of one dummy conversion for each configured hardware unit before start-up calibration is triggered in the API Adc_TriggerStartupCal(). (Refer Errata ADC_TC.083) The default value of this parameter is set to FALSE to minimize the executable code size. An error message will be raised if this parameter is set to TRUE while AdcStartupCalibApi is set to FALSE.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcStartupCalibApi		

ADC driver**3.3.1.5.25 AdcVersionInfoApi****Table 56 Specification for AdcVersionInfoApi**

Name	AdcVersionInfoApi		
Description	Parameter adds or removes the Adc_GetVersionInfo() API from the code. When set to TRUE, the API is available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.6 Container: AdcGlobalInputClass

Container defines the parameters for global input classes. The multiplicity of this parameter is added based on the number of global classes supported by the hardware. Note: This parameter writes into common SFRs of all the kernels and can be implemented as post-build. Hence, it is represented inside the AdcConfigSet container.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.6.1 AdcChConvMode**Table 57 Specification for AdcChConvMode**

Name	AdcChConvMode		
Description	Parameter defines the noise reduction level for standard conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_NOISE_REDUCTION_STEPS_0: Standard Conversion ADC_NOISE_REDUCTION_STEPS_1: 1 additional conversion step for Noise reduction ADC_NOISE_REDUCTION_STEPS_3: 3 additional conversion step for Noise reduction ADC_NOISE_REDUCTION_STEPS_7: 7 additional conversion step for Noise reduction		
Default value	ADC_NOISE_REDUCTION_STEPS_0		

ADC driver**Table 57 Specification for AdcChConvMode (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.6.2 AdcChPreChargeClkCycles**Table 58 Specification for AdcChPreChargeClkCycles**

Name	AdcChPreChargeClkCycles		
Description	Parameter defines the number of analog input precharge clock cycles for standard conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_INPUT_PRECHARGE_CYCLES_0: No precharge ADC_INPUT_PRECHARGE_CYCLES_16: Precharge for 16 clock cycles ADC_INPUT_PRECHARGE_CYCLES_32: Precharge for 32 clock cycles ADC_INPUT_PRECHARGE_CYCLES_8: Precharge for 8 clock cycles		
Default value	ADC_INPUT_PRECHARGE_CYCLES_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.6.3 AdcChSESPSEnable**Table 59 Specification for AdcChSESPSEnable**

Name	AdcChSESPSEnable		
Description	Parameter defines whether Spread Early Sample Point for standard conversions is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucBooleanParamDef

ADC driver**Table 59 Specification for AdcChSESPSEnable (continued)**

Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.6.4 AdcChSampleTime**Table 60 Specification for AdcChSampleTime**

Name	AdcChSampleTime		
Description	Parameter defines the number of clock cycles to be added to the minimum sample phase of two sample clock cycles. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.7 Container: AdcGroup

Container contains the group configuration (parameters). The short-name of all AdcGroups must be unique across the ADC hardware groups. That a minimum of one channel exists for a group is controlled through the lower multiplicity of AdcGroupDefinition parameter. The upper multiplicity of this container is added as 32 per ADC hardware unit, which is more than sufficient for application use case, given that the number of channels in a sequence is eight for primary hardware groups and sixteen for secondary hardware groups supported as per EVADC hardware unit.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

ADC driver**3.3.1.7.1 AdcChannel0Alias****Table 61 Specification for AdcChannel0Alias**

Name	AdcChannel0Alias		
Description	<p>Parameter configures the alias for Channel0. The hardware supports maximum of eight channels(0 to 7) for primary hardware groups and sixteen channels(0 to 15) for secondary hardware groups. The channel numbers upto 31 can be configured for diagnostics use cases.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p> <p>Note: A value of 0 here indicates that aliasing for Channel0 is disabled.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.7.2 AdcChannel1Alias**Table 62 Specification for AdcChannel1Alias**

Name	AdcChannel1Alias		
Description	<p>Parameter configures the alias for Channel1. The hardware supports maximum of eight channels(0 to 7) for primary hardware groups and sixteen channels(0 to 15) for secondary hardware groups. The channel numbers upto 31 can be configured for diagnostics use cases.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p> <p>Note: A value of 1 here indicates that aliasing for Channel1 is disabled.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 31		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.7.3 AdcGroupAccessMode****Table 63 Specification for AdcGroupAccessMode**

Name	AdcGroupAccessMode		
Description	<p>Parameter determines whether the group is a single access group or a streaming access group.</p> <p>1. ADC_ACCESS_MODE_STREAMING is not allowed for One-Shot software triggered groups (AdcGroupTriggSrc = ADC_TRIGG_SRC_SW and AdcGroupConvMode = ADC_CONV_MODE_ONESHOT).</p> <p>2. AdcGroupAccessMode is editable only when AdcResultHandlingImplementation is ADC_INTERRUPT_MODE_RESULT_HANDLING.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_ACCESS_MODE_SINGLE: Single value access mode ADC_ACCESS_MODE_STREAMING: Streaming access mode		
Default value	ADC_ACCESS_MODE_SINGLE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECU	Scope	LOCAL
Dependency	AdcGroupTriggSrc, AdcResultHandlingImplementation, AdcGroupConversionMode		

3.3.1.7.4 AdcGroupConversionMode**Table 64 Specification for AdcGroupConversionMode**

Name	AdcGroupConversionMode		
Description	Parameter determines whether a group is converting in single-shot mode or continuous conversion mode.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_CONV_MODE_CONTINUOUS: Conversions of an ADC channel group are performed continuously after a software API call (start). The conversions themselves are running automatically (no additional software or hardware trigger needed). ADC_CONV_MODE_ONESHOT: The conversion of an ADC channel group is performed once after a trigger.		
Default value	ADC_CONV_MODE_ONESHOT		
Post-build variant value	TRUE	Post-build variant multiplicity	-

ADC driver**Table 64 Specification for AdcGroupConversionMode (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.7.5 AdcGroupDefinition**Table 65 Specification for AdcGroupDefinition**

Name	AdcGroupDefinition		
Description	<p>Parameter defines the assignment of AdcChannel to an AdcGroup. Therefore, multiple nodes of AdcGroupDefinition builds a list of channels for the AdcGroup.</p> <p>The maximum number of channels in a sequence is eight for primary hardware groups and sixteen for secondary hardware groups. The upper multiplicity is added based on the queue depth as per EVADC hardware support.</p> <ol style="list-style-type: none"> 1.If the channels referred to do not belong to the same AdcHwUnitId an error message is raised. 2.If this container contains a channel with Limit Check enabled the number of channels is restricted to 1. 3.If the same analog channel is referred to more than once in the same group definition an error message is raised 		
Multiplicity	1..16	Type	EcucReferenceDef
Range	Reference to Node: AdcChannel		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcChannelLimitCheck, AdcHwUnitId		

3.3.1.7.6 AdcGroupId**Table 66 Specification for AdcGroupId**

Name	AdcGroupId
Description	Parameter defines the numeric ID of the ADC channel group. The maximum number of groups are limited to 32 in each ADC hardware unit. Numeric ID of the group is auto-generated by the configuration tool depending on the hardware unit.

ADC driver**Table 66 Specification for AdcGroupId (continued)**

	<p>The symbolic name of this parameter is to be used while calling the APIs. The symbolic name of each AdcChannel group is a macro defined with the numeric ID.</p> <p>Group ID per hardware unit:</p> <ul style="list-style-type: none"> 0 to 31 for HWUNIT_ADC0 32 to 63 for HWUNIT_ADC1 64 to 95 for HWUNIT_ADC2 96 to 127 for HWUNIT_ADC3 128 to 159 for HWUNIT_ADC4 160 to 191 for HWUNIT_ADC5 192 to 223 for HWUNIT_ADC6 224 to 255 for HWUNIT_ADC7 256 to 287 for HWUNIT_ADC8 288 to 319 for HWUNIT_ADC9 320 to 351 for HWUNIT_ADC10 352 to 383 for HWUNIT_ADC11 <p>Here, Bits 0 to 4 determine the Group ID and bits 5 to 8 determines the ADC hardware unit.</p> <p>The lowest group ID for a HW group is selected as the default value.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 383		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcHwUnitId		

3.3.1.7.7 AdcGroupPriority**Table 67 Specification for AdcGroupPriority**

Name	AdcGroupPriority		
Description	<p>Parameter configures the priority level of the AdcGroup. This node exists only when priority mechanism is enabled.</p> <p>The default value of this parameter is set to the lowest priority. Here, 0 is the lowest priority and 255 is the highest</p>		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		

ADC driver**Table 67 Specification for AdcGroupPriority (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcPriorityImplementation		

3.3.1.7.8 AdcGroupReplacement**Table 68 Specification for AdcGroupReplacement**

Name	AdcGroupReplacement		
Description	<p>Parameter provides the group replacement mechanism on AdcChannel group level if a group conversion is interrupted by a group which has a higher priority.</p> <p>This parameter is not editable since only the abort-restart mechanism is supported by the ADC driver.</p>		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	ADC_GROUP REPL_ABORT_RESTART: Abort or restart mechanism is used on the group level if a group is interrupted by a group with a higher priority. The complete conversion round of the interrupted group (all group channels) is restarted after the higher priority group conversion is finished. If the group is configured in streaming access mode only the results of the interrupted conversion round are discarded. Results of the previous conversion rounds that are already written to the result buffer are not affected.		
Default value	ADC_GROUP REPL_ABORT_RESTART		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.7.9 AdcGroupTriggSrc**Table 69 Specification for AdcGroupTriggSrc**

Name	AdcGroupTriggSrc
Description	Parameter defines the type of source event that starts a group conversion. Group conversion is started by a software API call for groups configured as ADC_TRIGG_SRC_SW and by a hardware event for groups configured as ADC_TRIGG_SRC_HW.

ADC driver**Table 69 Specification for AdcGroupTriggSrc (continued)**

	1.ADC_TRIGG_SRC_SW is allowed only when AdcEnableStartStopGroupApi is set to TRUE. 2.ADC_TRIGG_SRC_HW is allowed only when AdcHwTriggerApi is set to TRUE.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_TRIGG_SRC_HW: Hardware triggered (GTM/ERU/CCU6) group. ADC_TRIGG_SRC_SW: Software triggered group.		
Default value	ADC_TRIGG_SRC_SW		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcHwTriggerApi, AdcEnableStartStopGroupApi		

3.3.1.7.10 AdcHwExtGateSelect**Table 70 Specification for AdcHwExtGateSelect**

Name	AdcHwExtGateSelect		
Description	Parameter configures the hardware gating source for the group.The range of values is dependent on the device. 1. AdcHwExtGateSelect is editable for hardware triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_HW) 2. AdcHwExtTrigSelect should be ADC_GATE_NONE for software triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_SW) 3. AdcHwExtGateSelect should be CCU6 when trigger is through a GATE input (AdcHwExtTrigSelect = ADC_TRIG_15_GxREQTRP_GxREQGTySEL). The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_GATE_NONE: Hardware gate signal is not selected ADC_GATE_[x]_[y]: Hardware gate signal selected. [x] and [y] depends on the device		
Default value	ADC_GATE_NONE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

ADC driver**Table 70 Specification for AdcHwExtGateSelect (continued)**

Dependency	AdcGroupTriggSrc, AdcHwExtTrigSelect
-------------------	--------------------------------------

3.3.1.7.11 AdcHwExtTrigSelect**Table 71 Specification for AdcHwExtTrigSelect**

Name	AdcHwExtTrigSelect		
Description	<p>Parameter configures the hardware trigger source for the group. The range of values is dependent on the device.</p> <p>1. AdcHwExtTrigSelect is editable for hardware triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_HW).</p> <p>2. AdcHwExtTrigSelect should be ADC_TRIG_NONE for software triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_SW).</p> <p>3. AdcHwExtTrigSelect should not be ADC_TRIG_NONE for hardware triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_HW).</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>ADC_TRIG_NONE: Hardware trigger signal is not selected</p> <p>ADC_TRIG_[x]_[y]: Hardware trigger signal is selected.</p> <p>[x] and [y] depends on the device</p>		
Default value	ADC_TRIG_NONE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcGroupTriggSrc		

3.3.1.7.12 AdcHwGateSignal**Table 72 Specification for AdcHwGateSignal**

Name	AdcHwGateSignal		
Description	<p>Parameter configures the hardware gating signal level.</p> <p>1. AdcHwGateSignal is editable for hardware triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_HW)</p> <p>2. AdcHwGateSignal is editable when AdcHwExtGateSelect not equal to ADC_GATE_NONE and AdcHwExtTrigSelect is not equal to ADC_TRIG_15_GxREQTRP_GxREQGTySEL.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef

ADC driver**Table 72 Specification for AdcHwGateSignal (continued)**

Range	ADC_GATE_LVL_HIGH: Gate is enabled when a rising edge is detected and the signal remains high. ADC_GATE_LVL_LOW: Gate is enabled when a falling edge is detected and the signal remains low.		
Default value	ADC_GATE_LVL_HIGH		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcGroupTriggSrc, AdcHwExtTrigSelect, AdcHwExtGateSelect		

3.3.1.7.13 AdcHwTrigSignal**Table 73 Specification for AdcHwTrigSignal**

Name	AdcHwTrigSignal		
Description	<p>Parameter determines the edge of the external request signal on which the driver starts the conversion sequence.</p> <p>1.This parameter is editable and is applicable when AdcGroupTriggSrc is configured as ADC_TRIGG_SRC_HW.</p> <p>2.AdcHwTrigSignal should be ADC_HW_TRIG_RISING_EDGE,when AdcHwExtTrigSelect is GTM and GTM TOM or ATOM timer configuration for trigger signal is inside the ADC driver. The driver supports all the types of HW Trigger Signal(rising,falling and both) when the GTM TOM or ATOM configuration for trigger signal is outside the ADC driver.</p> <p>3.AdcHwTrigSignal should be ADC_HW_TRIG_RISING_EDGE, when AdcHwExtTrigSelect is through GATE_PIN and AdcHwExtGateSelect is GTM or CCU6 (Timer).</p>		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	ADC_HW_TRIG_BOTH_EDGES: Group is triggered on both the edges of the trigger signal. ADC_HW_TRIG_FALLING_EDGE: Group is triggered on the falling edge of the trigger signal ADC_HW_TRIG_RISING_EDGE: Group is triggered on rising edge of the trigger signal		
Default value	ADC_HW_TRIG_RISING_EDGE		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcHwExtGateSelect, AdcHwExtTrigSelect, AdcGroupTriggSrc		

ADC driver**3.3.1.7.14 AdcHwTrigTimer****Table 74 Specification for AdcHwTrigTimer**

Name	AdcHwTrigTimer		
Description	<p>Parameter specifies the reload value of the ADC module embedded timer (REQTM). For cyclic triggering of the ADC, GTM, or CCU6 trigger is used.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p> <p>Note: The maximum value of this parameter is specified based on the maximum value supported by the EVADC hardware trigger timer.</p>		
Multiplicity	0..1	Type	EcucIntegerParamDef
Range	0 - 1023		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.7.15 AdcNotification**Table 75 Specification for AdcNotification**

Name	AdcNotification		
Description	<p>Parameter is used by the ADC driver to invoke the user-defined function whenever the conversion of all the channels of a group is completed. The parameter can be configured as a name or an address(numeric value) of the notification function. If configured as NULL_PTR, notification is not issued by the ADC driver. The parameter is editable only when AdcGrpNotifCapability is set to TRUE.</p> <p>Note1: By default, the notification parameter will be NULL_PTR , to remove the dependency from the user defined functions.</p> <p>Note2: The ADC driver does not validate the configured function name or address for correctness and hence the responsibility falls on the user.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build

ADC driver**Table 75 Specification for AdcNotification (continued)**

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcGrpNotifCapability		

3.3.1.7.16 AdcResRegDefinition**Table 76 Specification for AdcResRegDefinition**

Name	AdcResRegDefinition		
Description	<p>Parameter configures the result register in which to store the ADC conversion results. Value of the parameter should be in ascending order. Therefore, the first configured channel should have the lowest result register index. So, a default value set to 0 . Possible set of result registers configured for a group having five channels may be as: 4, 5, 6, 7, 8 or 0, 1, 2, 3, 4 or 11, 12 ,13, 14, 15.</p> <ol style="list-style-type: none"> 1. If AdcResultRegister are not configured in contiguous-ascending order an error message is raised. 2. The number of nodes should be the same as the number of nodes configured for the corresponding AdcGroupDefinition container. 		
Multiplicity	1..16	Type	EcuIntegerParamDef
Range	0 - 15		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	IFX	Scope	LOCAL
Dependency	AdcGroupDefinition		

3.3.1.7.17 AdcStreamingBufferMode**Table 77 Specification for AdcStreamingBufferMode**

Name	AdcStreamingBufferMode
Description	<p>Parameter configures the streaming buffer as a "linear buffer" or as a "circular buffer".</p> <ol style="list-style-type: none"> 1. AdcStreamingBufferMode cannot be ADC_STREAM_BUFFER_CIRCULAR for software triggered One-Shot conversion (AdcGroupTriggSrc = 'ADC_TRIGG_SRC_SW' and AdcGroupConversionMode = 'ADC_CONV_MODE_ONESHOT' and AdcGroupAccessMode = 'ADC_ACCESS_MODE_SINGLE' and AdcResultHandlingImplementation = 'ADC_INTERRUPT_MODE_RESULT_HANDLING') 2. AdcStreamingBufferMode cannot be ADC_STREAM_BUFFER_LINEAR for software triggered continuous single access groups (AdcGroupTriggSrc = 'ADC_TRIGG_SRC_SW', AdcGroupConversionMode = 'ADC_CONV_MODE_CONTINUOUS' and AdcGroupAccessMode =

ADC driver**Table 77 Specification for AdcStreamingBufferMode (continued)**

	'ADC_ACCESS_MODE_SINGLE' and AdcResultHandlingImplementation = 'ADC_INTERRUPT_MODE_RESULT_HANDLING') 3. AdcStreamingBufferMode cannot be ADC_STREAM_BUFFER_LINEAR for hardware triggered single access groups (AdcGroupTriggSrc = 'ADC_TRIGG_SRC_HW' and AdcGroupAccessMode = 'ADC_ACCESS_MODE_SINGLE' and AdcResultHandlingImplementation = 'ADC_INTERRUPT_MODE_RESULT_HANDLING') 4. AdcStreamingBufferMode is editable only when AdcResultHandlingImplementation is ADC_INTERRUPT_MODE_RESULT_HANDLING.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_STREAM_BUFFER_CIRCULAR: The ADC driver continues the conversion even if the stream buffer is full (number of samples reached) by wrapping around the stream buffer itself. ADC_STREAM_BUFFER_LINEAR: The ADC driver stops the conversion as soon as the stream buffer is full (number of samples reached).		
Default value	ADC_STREAM_BUFFER_LINEAR		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcGroupAccessMode, AdcGroupTriggSrc, AdcResultHandlingImplementation, AdcGroupConversionMode		

3.3.1.7.18 AdcStreamingNumSamples**Table 78 Specification for AdcStreamingNumSamples**

Name	AdcStreamingNumSamples		
Description	Parameter configures the number of ADC values to be acquired per channel in streaming access mode. 1. AdcStreamingNumSamples should be greater than 1 for Streaming access groups (AdcGroupAccessMode = ADC_ACCESS_MODE_STREAMING and AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING) 2. AdcStreamingNumSamples should be equal to 1 for single access groups (AdcGroupAccessMode = ADC_ACCESS_MODE_SINGLE and AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING) 3. AdcStreamingNumSamples is editable only when AdcResultHandlingImplementation is ADC_INTERRUPT_MODE_RESULT_HANDLING.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 255		
Default value	1		

ADC driver**Table 78 Specification for AdcStreamingNumSamples (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	AdcGroupAccessMode, AdcResultHandlingImplementation		

3.3.1.8 Container: AdcHwUnit

Container consists of the configuration parameter and containers for ADC hardware group. The upper multiplicity of this parameter is added based on the total number of EVADC hardware units supported by the hardware.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.8.1 AdcAnalogClockSyncDelay**Table 79 Specification for AdcAnalogClockSyncDelay**

Name	AdcAnalogClockSyncDelay		
Description	Parameter configures analog clock synchronization delay. The delay should be less than or equal to AdcPrescale-1(DIVA). The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcPrescale		

3.3.1.8.2 AdcCalibrationSampleTime**Table 80 Specification for AdcCalibrationSampleTime**

Name	AdcCalibrationSampleTime		
Description	Parameter configures the calibration sample time. The default value of this parameter is set to the reset value of the corresponding SFR.		

ADC driver**Table 80 Specification for AdcCalibrationSampleTime (continued)**

Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_CAL_TIME_2_TIMES_TADCI: Calibration sample time = 2 x tADCI ADC_CAL_TIME_4_TIMES_TADCI: Calibration sample time = 4 x tADCI ADC_CAL_TIME_6_TIMES_TADCI: Calibration sample time = 6 x tADCI ADC_CAL_TIME_8_TIMES_TADCI: Calibration sample time = 8 x tADCI		
Default value	ADC_CAL_TIME_2_TIMES_TADCI		
Post-build variant value	TRUE		
Value configuration class	Post-Build		
Origin	IFX		
Dependency	-		

3.3.1.8.3 AdcClockSource**Table 81 Specification for AdcClockSource**

Name	AdcClockSource		
Description	Parameter defines the ADC module specific clock source. The clock to ADC module is controlled through AdcSystemClock, Therefore, this parameter is unused. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	ADC_SYSTEM_CLK: System clock for the EVADC		
Default value	ADC_SYSTEM_CLK		
Post-build variant value	TRUE		
Value configuration class	Post-Build		
Origin	AUTOSAR_ECUC		
Dependency	-		

3.3.1.8.4 AdcHwUnitId**Table 82 Specification for AdcHwUnitId**

Name	AdcHwUnitId
-------------	-------------

ADC driver**Table 82 Specification for AdcHwUnitId (continued)**

Description	Parameter is used to arrange the configuration parameters in the configuration structure according to ADC hardware unit ID. This parameter should have a unique value across AdcHwUnit container nodes for same config set, that is, a hardware unit can be configured only once in a config set. The lowest index ADC hardware group available is selected as the default value for this parameter.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_HWUNIT[x]: [x] = list of available Hardware units.		
Default value	ADC_HWUNIT[x]		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.8.5 AdcIdlePrechargeEnable**Table 83 Specification for AdcIdlePrechargeEnable**

Name	AdcIdlePrechargeEnable		
Description	Parameter determines whether pre-charging of the sampling capacitor is enabled or not. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.8.6 AdcInputBufferEnable****Table 84 Specification for AdcInputBufferEnable**

Name	AdcInputBufferEnable		
Description	<p>Parameter determines whether input buffering is enabled or not. If enabled buffering time is selected through AIPS(AdcChPreChargeClkCycles).</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.8.7 AdcMSBDoubleClkEnable**Table 85 Specification for AdcMSBDoubleClkEnable**

Name	AdcMSBDoubleClkEnable		
Description	<p>Parameter configures the number of clock cycles for MSB conversions.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.8.8 AdcPostCalibrationDisable****Table 86 Specification for AdcPostCalibrationDisable**

Name	AdcPostCalibrationDisable		
Description	Parameter configures whether post-calibration is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.8.9 AdcPrechargeReference**Table 87 Specification for AdcPrechargeReference**

Name	AdcPrechargeReference		
Description	Parameter determines whether VDDM/VSSM is used for pre-charging or not. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_VDD_VSM_NOT_USED: VAREF/VAGND for the conversion. ADC_VDD_VSM_USED: VDDM/VSSM for precharging and VAREF/VAGND for the final adjustment during a conversion.		
Default value	ADC_VDD_VSM_USED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.8.10 AdcPrescale****Table 88 Specification for AdcPrescale**

Name	AdcPrescale		
Description	Parameter configures the divider for generating the analog internal clock (fADCDI) from fADC. The lower multiplicity of this parameter is 1, because the divider for generating analog frequency is mandatorily for performing the conversion. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	2 - 32		
Default value	4		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.8.11 AdcReferencePrechargePhases**Table 89 Specification for AdcReferencePrechargePhases**

Name	AdcReferencePrechargePhases		
Description	Parameter configures the number of pre-charge clock phases for the reference input. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_PRECHARGE_PHASE_1: Precharge the reference input for 1 clock phase ADC_PRECHARGE_PHASE_2: Precharge the reference input for 2 clock phase		
Default value	ADC_PRECHARGE_PHASE_1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.8.12 AdcRequestSource0ConvMode****Table 90 Specification for AdcRequestSource0ConvMode**

Name	AdcRequestSource0ConvMode		
Description	<p>Parameter configures the conversion start mode for a request source. The configuration of the conversion start mode for request sources is editable and applicable only when priority mechanism is enabled.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>ADC_CANCEL_INJECT_REPEAT_MODE: Request source can cancel the ongoing conversion of other sources.</p> <p>ADC_WAIT_FOR_START_MODE: Request source waits for the completion of the current conversion from the other source.</p>		
Default value	ADC_WAIT_FOR_START_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcPriorityImplementation		

3.3.1.8.13 AdcRequestSource1ConvMode**Table 91 Specification for AdcRequestSource1ConvMode**

Name	AdcRequestSource1ConvMode		
Description	<p>Parameter configures the conversion start mode for a request source. The configuration of the conversion start mode for request sources is editable and applicable only when priority mechanism is enabled.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>ADC_CANCEL_INJECT_REPEAT_MODE: Request source can cancel the ongoing conversion of other sources.</p> <p>ADC_WAIT_FOR_START_MODE: Request source waits for the completion of current conversion from other source.</p>		
Default value	ADC_WAIT_FOR_START_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

ADC driver**Table 91 Specification for AdcRequestSource1ConvMode (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcPriorityImplementation		

3.3.1.8.14 AdcRequestSource2ConvMode**Table 92 Specification for AdcRequestSource2ConvMode**

Name	AdcRequestSource2ConvMode		
Description	Parameter configures the conversion start mode for a request source. The configuration of the conversion start mode for request sources is editable and applicable only when priority mechanism is enabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_CANCEL_INJECT_REPEAT_MODE: Request source can cancel the ongoing conversion of other sources. ADC_WAIT_FOR_START_MODE: Request source waits for the completion of current conversion from other source.		
Default value	ADC_WAIT_FOR_START_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcPriorityImplementation		

3.3.1.8.15 AdcSampleSyncEnable**Table 93 Specification for AdcSampleSyncEnable**

Name	AdcSampleSyncEnable		
Description	Parameter determines whether the ADC Sample Time Synchronization is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE		

ADC driver**Table 93 Specification for AdcSampleSyncEnable (continued)**

	FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcSyncClockDisable		

3.3.1.8.16 AdcSyncConvMode**Table 94 Specification for AdcSyncConvMode**

Name	AdcSyncConvMode		
Description	Parameter configures whether the hardware groups operate in standalone or synchronous mode. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_STAND_ALONE: Hardware group operates in stand-alone mode. ADC_SYNC_MASTER: Hardware group operates in master mode. ADC_SYNC_SLAVE: Hardware group operates in slave mode.		
Default value	ADC_STAND_ALONE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcSyncConvEnable		

3.3.1.9 Container: AdcHwUnitInputClass

Container defines the parameters for hardware unit input classes. The multiplicity of this parameter is added based on the number of hardware unit classes supported by the hardware.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

ADC driver**3.3.1.9.1 AdcChConvMode****Table 95 Specification for AdcChConvMode**

Name	AdcChConvMode		
Description	Parameter configures the noise reduction level for standard conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_NOISE_REDUCTION_STEPS_0: Standard conversion ADC_NOISE_REDUCTION_STEPS_1: One additional conversion step for noise reduction ADC_NOISE_REDUCTION_STEPS_3: Three additional conversion steps for noise reduction ADC_NOISE_REDUCTION_STEPS_7: Seven additional conversion step for noise reduction		
Default value	ADC_NOISE_REDUCTION_STEPS_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.9.2 AdcChPreChargeClkCycles**Table 96 Specification for AdcChPreChargeClkCycles**

Name	AdcChPreChargeClkCycles		
Description	Parameter configures the number of analog input precharge clock cycles for standard conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_INPUT_PRECHARGE_CYCLES_0: No pre-charge ADC_INPUT_PRECHARGE_CYCLES_16: Pre-charge for 16 clock cycles ADC_INPUT_PRECHARGE_CYCLES_32: Pre-charge for 32 clock cycles ADC_INPUT_PRECHARGE_CYCLES_8: Pre-charge for 8 clock cycles		
Default value	ADC_INPUT_PRECHARGE_CYCLES_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

ADC driver**Table 96 Specification for AdcChPreChargeClkCycles (continued)**

Dependency	-
-------------------	---

3.3.1.9.3 AdcChSESPSEnable**Table 97 Specification for AdcChSESPSEnable**

Name	AdcChSESPSEnable		
Description	Parameter configures whether Spread Early Sample Point for standard conversions is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.9.4 AdcChSampleTime**Table 98 Specification for AdcChSampleTime**

Name	AdcChSampleTime		
Description	Parameter configures the number of clock cycles to be added to the minimum sample phase of two sample clock cycles. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

ADC driver**Table 98 Specification for AdcChSampleTime (continued)**

Dependency	-
-------------------	---

3.3.1.10 Container: AdcPowerStateConfig

Container contains the configuration parameters related to the power states supported. The upper multiplicity of this container is added based on the total number of power modes supported by the EVADC hardware unit.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.10.1 AdcPowerState**Table 99 Specification for AdcPowerState**

Name	AdcPowerState		
Description	Parameter configures the availability of the different power state supported by the EVADC hardware unit. The available power modes are as follows: 0 - Normal operation (permanently on) 1 - Fast standby mode 2 - Slow standby mode 3 - Analog converter off Note: Each power state must be selected only once across all AdcPowerStateConfig instances. An error is generated if the same value is selected twice. The normal operation (full power) is selected as the default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 3		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.10.2 AdcPowerStateReadyCbkRef**Table 100 Specification for AdcPowerStateReadyCbkRef**

Name	AdcPowerStateReadyCbkRef
Description	Parameter determines the callback function for the power mode. Note: Because power state transitions are always synchronous this parameter is disabled in tresos.

ADC driver**Table 100 Specification for AdcPowerStateReadyCbkRef (continued)**

Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.11 Container: AdcPublishedInformation

Container contains the published information of the ADC driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.11.1 AdcChannelValueSigned**Table 101 Specification for AdcChannelValueSigned**

Name	AdcChannelValueSigned		
Description	Parameter provides the information whether the result value of the ADC driver has sign information (TRUE) or not (FALSE).		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.11.2 AdcGroupFirstChannelFixed****Table 102 Specification for AdcGroupFirstChannelFixed**

Name	AdcGroupFirstChannelFixed		
Description	Parameter provides the information whether the first channel of an ADC channel group can be configured (FALSE) or is fixed (TRUE) to a value determined by the ADC hardware unit.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.11.3 AdcMaxChannelResolution**Table 103 Specification for AdcMaxChannelResolution**

Name	AdcMaxChannelResolution		
Description	Parameter provides the information of the maximum channel resolution in bits.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 63		
Default value	12		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

3.3.1.12 Container: CommonPublishedInformation

Container contains the common published information of the ADC driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

ADC driver**3.3.1.12.1 ArMajorVersion****Table 104 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Parameter provides the major version of the AUTOSAR specification.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.12.2 ArMinorVersion**Table 105 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	Parameter provides the minor version of the AUTOSAR specification.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.12.3 ArPatchVersion**Table 106 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	Parameter provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		

ADC driver**Table 106 Specification for ArPatchVersion (continued)**

Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.12.4 ModuleId**Table 107 Specification for ModuleId**

Name	ModuleId		
Description	Parameter provides the Module Id of the ADC driver.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	123		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.12.5 Release**Table 108 Specification for Release**

Name	Release		
Description	Specifies the deriveate for which the configuration project is created.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per hardware derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

ADC driver**Table 108 Specification for Release (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.12.6 SwMajorVersion**Table 109 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Specifies the major version of the driver software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.12.7 SwMinorVersion**Table 110 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Specifies the minor version of the driver software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.12.8 SwPatchVersion****Table 111 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Specifies the patch version of the driver software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.12.9 VendorId**Table 112 Specification for VendorId**

Name	VendorId		
Description	Specifies the vendor ID for Infineon.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.13 Container: EruGatingConfig

Container contains the ERU configuration for the gating signal. The container EruGatingConfig is available only for hardware triggered groups and when AdcHwExtGateSelect is an ERU signal. If the group is configured as HW_ERU_GATE and the container EruGatingConfig is missing, an error message is raised.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

ADC driver**3.3.1.13.1 AdcEruErsInputPin****Table 113 Specification for AdcEruErsInputPin**

Name	AdcEruErsInputPin		
Description	Parameter determines the input pin for the selected ERS.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ERS_REQ_[x]: Range of literals varies based on target device.		
Default value	ERS_REQ_[x]		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.13.2 AdcEruErsRef**Table 114 Specification for AdcEruErsRef**

Name	AdcEruErsRef		
Description	Parameter is a reference to the ERU container in the MCU. It lists all the ERU-ERS channels available. If ERU-ERS channel referred to is not marked as to be used by ADC driver in the MCU driver an error message is raised.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuEruChannelInputLineConf		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcHwUnitId		

3.3.1.13.3 AdcEruOguRef**Table 115 Specification for AdcEruOguRef**

Name	AdcEruOguRef
-------------	--------------

ADC driver**Table 115 Specification for AdcEruOguRef (continued)**

Description	Parameter is a reference to the ERU container in the MCU. It lists all the ERU-ERS channels available. If ERU-OGU channel referred to is not marked as to be used by the ADC driver in MCU driver an error message is raised.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuEruChannelOutputUnitConf		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcHwUnitId		

3.3.1.14 Container: EruTriggerConfig

Container contains the ERU configuration for the trigger signal. The container EruTriggerConfig is available only for hardware triggered groups and when AdcHwExtTrigSelect is an ERU signal. If the group is configured as HW_ERU_TRIG and the container EruTriggerConfig is missing, an error message is raised.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

3.3.1.14.1 AdcEruErsInputPin

Table 116 Specification for AdcEruErsInputPin

Name	AdcEruErsInputPin		
Description	Parameter determines the input pin for the selected ERS.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ERS_REQ_[x]: Range of literals varies based on target device.		
Default value	ERS_REQ_[x]		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

ADC driver**3.3.1.14.2 AdcEruErsRef****Table 117 Specification for AdcEruErsRef**

Name	AdcEruErsRef		
Description	Parameter is a reference to the ERU container in the MCU. It lists all the ERU-ERS channels available. If ERU-ERS channel referred to is not marked as to be used by the ADC driver in the MCU driver an error message is raised.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuEruChannelInputLineConf		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcHwUnitId		

3.3.1.14.3 AdcEruOguRef**Table 118 Specification for AdcEruOguRef**

Name	AdcEruOguRef		
Description	Parameter is a reference to the ERU container in the MCU. It lists all the ERU-ERS channels available. If ERU-OGU channel referred to is not marked as to be used by the ADC driver in the MCU driver an error message is raised.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuEruChannelOutputUnitConf		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	AdcHwUnitId		

3.3.1.15 Container: GtmGatingTimerConfig

Container GtmGatingTimerConfig is available only for hardware triggered groups and when AdcHwExtgateSelect is a GTM signal. If the container is not configured for such a group, then the initialization of the GTM channel intended to be used for gate signal must be done outside of the ADC driver.

Post-Build Variant Multiplicity: FALSE

ADC driver

Multiplicity Configuration Class: Pre-Compile

3.3.1.15.1 GtmTimerCM0Ticks

Table 119 Specification for GtmTimerCM0Ticks

Name	GtmTimerCM0Ticks		
Description	Parameter specifies the period compare value (CM0) for the TOM/ATOM channel in ticks. For, TOM channel upper limit is restricted to 65535. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerUsed		

3.3.1.15.2 GtmTimerClockSelect

Table 120 Specification for GtmTimerClockSelect

Name	GtmTimerClockSelect		
Description	Parameter decides the clock source for the GTM timer. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_CONFIGURABLE_CLOCK_0: Configurable Clock 0 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_1: Configurable Clock 1 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_2: Configurable Clock 2 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_3: Configurable Clock 3 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_4: Configurable Clock 4 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_5: Configurable Clock 5 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_6: Configurable Clock 6 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_7: Configurable Clock 7 will be supplied to the Channel GTM_FIXED_CLOCK_0: Fixed Clock 0 will be supplied to the Channel GTM_FIXED_CLOCK_1: Fixed Clock 1 will be supplied to the Channel GTM_FIXED_CLOCK_2: Fixed Clock 2 will be supplied to the Channel GTM_FIXED_CLOCK_3: Fixed Clock 3 will be supplied to the Channel		

ADC driver**Table 120 Specification for GtmTimerClockSelect (continued)**

	GTM_FIXED_CLOCK_4: Fixed Clock 4 will be supplied to the Channel		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerUsed		

3.3.1.15.3 GtmTimerTimePeriod**Table 121 Specification for GtmTimerTimePeriod**

Name	GtmTimerTimePeriod		
Description	Parameter specifies timeout value for the GTM timer. If the calculated GTM ticks is greater than 0xFFFF for TOM channels or greater than 0xFFFFFFF for ATOM channels an error message is raised. This parameter is editable only when GtmTimerCM0Ticks is equal to 0 (user has not entered the period match ticks directly). A nominal time interval is configured as the default value for time-based trigger.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65536000		
Default value	20000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerCM0Ticks		

3.3.1.15.4 GtmTimerUsed**Table 122 Specification for GtmTimerUsed**

Name	GtmTimerUsed		
Description	Parameter defines the GTM timer channel used. If timer channel referred to, is not marked as to be used by the ADC driver in the MCU driver an error message is raised.		
Multiplicity	1..1	Type	EcuChoiceReferenceDef

ADC driver**Table 122 Specification for GtmTimerUsed (continued)**

Range	Reference to Node: McuGtmAtomChannelAllocationConf, McuGtmTomChannelAllocationConf		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.16 Container: GtmTriggerTimerConfig

Container GtmTriggerTimerConfig is available only for hardware triggered groups and when AdcHwExtTrigSelect is a GTM signal. If the container is not configured for such a group, then the initialization of the GTM channel intended to be used for trigger signal must be done outside of the ADC driver.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.16.1 GtmTimerCM0Ticks**Table 123 Specification for GtmTimerCM0Ticks**

Name	GtmTimerCM0Ticks		
Description	Parameter specifies the period compare value (CM0) for the TOM/ATOM channel in ticks. For, TOM channel upper limit is restricted to 65535. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerUsed		

ADC driver**3.3.1.16.2 GtmTimerClockSelect****Table 124 Specification for GtmTimerClockSelect**

Name	GtmTimerClockSelect		
Description	Parameter decides the clock source for the GTM timer. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_CONFIGURABLE_CLOCK_0: Configurable Clock 0 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_1: Configurable Clock 1 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_2: Configurable Clock 2 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_3: Configurable Clock 3 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_4: Configurable Clock 4 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_5: Configurable Clock 5 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_6: Configurable Clock 6 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_7: Configurable Clock 7 will be supplied to the Channel. GTM_FIXED_CLOCK_0: Fixed Clock 0 will be supplied to the Channel GTM_FIXED_CLOCK_1: Fixed Clock 1 will be supplied to the Channel GTM_FIXED_CLOCK_2: Fixed Clock 2 will be supplied to the Channel GTM_FIXED_CLOCK_3: Fixed Clock 3 will be supplied to the Channel GTM_FIXED_CLOCK_4: Fixed Clock 4 will be supplied to the Channel		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerUsed		

3.3.1.16.3 GtmTimerTimePeriod**Table 125 Specification for GtmTimerTimePeriod**

Name	GtmTimerTimePeriod		
Description	Parameter specifies timeout value for the GTM timer. If the calculated number of GTM ticks is greater than 0xFFFF for TOM channels or greater than 0xFFFFFFF for ATOM channels an error message is raised. This parameter is editable only when GtmTimerCM0Ticks is equal to 0 (user has not entered the period match ticks directly). A nominal time-interval is configured as the default value for time-based trigger.		
Multiplicity	1..1	Type	EcucIntegerParamDef

ADC driver**Table 125 Specification for GtmTimerTimePeriod (continued)**

Range	0 - 65536000		
Default value	20000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerCM0Ticks		

3.3.1.16.4 GtmTimerUsed**Table 126 Specification for GtmTimerUsed**

Name	GtmTimerUsed		
Description	Parameter defines the GTM timer channel used. If timer channel referred to, is not marked as to be used by the ADC driver in the MCU driver an error message is raised.		
Multiplicity	1..1	Type	EcucChoiceReference Def
Range	Reference to Node: McuGtmAtomChannelAllocationConf, McuGtmTomChannelAllocationConf		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.2 Functions - Type definitions**3.3.2.1 Adc_ChannelRangeSelectType****Table 127 Specification for Adc_ChannelRangeSelectType**

Syntax	Adc_ChannelRangeSelectType
Type	Enumeration
File	Adc.h

ADC driver**Table 127 Specification for Adc_ChannelRangeSelectType (continued)**

Range	0 - ADC_RANGE_UNDER_LOW	Range below low limit - low limit value included
	1 - ADC_RANGE_BETWEEN	Range between low limit and high limit - high limit value included
	2 - ADC_RANGE_OVER_HIGH	Range above high limit
	3 - ADC_RANGE_ALWAYS	Complete range - independent of channel limit settings
	4 - ADC_RANGE_NOT_UNDER_LOW	Range above low limit
	5 - ADC_RANGE_NOT_BETWEEN	Range above high limit or below low limit - low limit value included
	6 - ADC_RANGE_NOT_OVER_HIGH	Range below high limit - high limit value included
Description	Defines the type for which conversion values are taken into account related to the borders defined with AdcChannelLowLimit and AdcChannelHighLimit.	
Source	AUTOSAR	

3.3.2.2 Adc_ChannelType**Table 128 Specification for Adc_ChannelType**

Syntax	Adc_ChannelType	
Type	uint8	
File	Adc.h	
Range	0 - 255	
Description	Defines the type for numeric ID of an ADC channel. The Adc_ChannelType is used for both logical and physical channel ID. The maximum number of logical channels per hardware group is 255 (ID 0 to ID 254, here ID 255 is used as invalid channel ID) and the maximum number of physical channels per hardware group is 16 (ID 0 to ID 15). Hence, the data width is selected based on the maximum channel ID.	
Source	AUTOSAR	

3.3.2.3 Adc_ConfigType**Table 129 Specification for Adc_ConfigType**

Syntax	Adc_ConfigType	
Type	Structure	
File	Adc.h	
Range	--	The elements of the data structure are specific to the microcontroller.

ADC driver**Table 129 Specification for Adc_ConfigType (continued)**

Description	Defines the data structure containing the set of configuration parameters required for initializing the ADC driver and ADC hardware unit(s).
Source	AUTOSAR

3.3.2.4 Adc_ConversionTimeType**Table 130 Specification for Adc_ConversionTimeType**

Syntax	Adc_ConversionTimeType	
Type	uint8	
File	Adc.h	
Range	--	Unused
Description	Defines the type for conversion time during which the sampled analog value is converted into its digital representation. Since this type is not used by the driver, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.5 Adc_GroupAccessModeType**Table 131 Specification for Adc_GroupAccessModeType**

Syntax	Adc_GroupAccessModeType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_ACCESS_MODE_SINGLE	Single value access mode.
	1 - ADC_ACCESS_MODE_STREAMING	Streaming value access mode.
Description	Defines the type for configuring the access mode to group results. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.6 Adc_GroupConvModeType**Table 132 Specification for Adc_GroupConvModeType**

Syntax	Adc_GroupConvModeType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_CONV_MODE_ONESHOT	Exactly one conversion of each channel in an ADC channel group is performed after the configured trigger event. In case of group trigger source software, a

ADC driver**Table 132 Specification for Adc_GroupConvModeType (continued)**

		One Shot conversion, once started, can be stopped by a software API call. In case of group trigger source hardware, a One Shot conversion, once started, can be stopped by disabling the trigger event.
	1 - ADC_CONV_MODE_CONTINUOUS	Repeated conversions of each ADC channel in an ADC channel group are performed. Continuous conversion mode is only available for group trigger source software. A Continuous conversion, once started, can be stopped by a software API call.
Description	Defines the type for configuring the conversion mode for an ADC channel group. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.7 Adc_GroupDefType**Table 133 Specification for Adc_GroupDefType**

Syntax	Adc_GroupDefType	
Type	Structure	
File	Adc.h	
Range	--	The elements of the data structure are specific to the microcontroller.
Description	Defines the type for assignment of channels to a channel group (this is not an API type).	
Source	AUTOSAR	

3.3.2.8 Adc_GroupPriorityType**Table 134 Specification for Adc_GroupPriorityType**

Syntax	Adc_GroupPriorityType	
Type	uint8	
File	Adc.h	
Range	0..255	
Description	Defines the type for configuring the priority level of the group. The lowest priority is 0. Data width is specified based on the maximum priority level supported by AUTOSAR	
Source	AUTOSAR	

ADC driver**3.3.2.9 Adc_GroupReplacementType****Table 135 Specification for Adc_GroupReplacementType**

Syntax	Adc_GroupReplacementType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_GROUP_REPL_ABORT_RESTART	Abort/Restart mechanism is used on group level if a group is interrupted by a higher priority group. The complete conversion round of the interrupted group (all group channels) is restarted after the higher priority group conversion is finished. If the group is configured in streaming access mode only the results of the interrupted conversion round are discarded. Results of the previous conversion rounds which are already written to the result buffer are not affected.
	1 - ADC_GROUP_REPL_SUSPEND_RESUME	Suspend/Resume mechanism is used on group level if a group is interrupted by a higher priority group. The conversion round of the interrupted group is completed after the higher priority group conversion is finished. Results of the previous conversion rounds which are already written to the result buffer are not affected.
Description	Defines the type for replacement mechanism used on AdcChannel group level if a group conversion is interrupted by a group which has a higher priority. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.10 Adc_GroupType**Table 136 Specification for Adc_GroupType**

Syntax	Adc_GroupType	
Type	uint16	
File	Adc.h	
Range	0 - 383	The range of this type is microcontroller specific and has to be described by the supplier.
	Defines the type for numeric ID of an ADC channel group. The lower five bits represent the 'AdcChannel group ID', while the bits 5 to 8 represent the 'ADC hardware group' or	

ADC driver**Table 136 Specification for Adc_GroupType (continued)**

	'ADC hardware unit'. The ADC driver supports a maximum of 32 channel groups per EVADC hardware group. Hence, the data width is selected based on the maximum channel group ID possible across all hardware groups.
Source	AUTOSAR

3.3.2.11 Adc_HwTriggerSignalType**Table 137 Specification for Adc_HwTriggerSignalType**

Syntax	Adc_HwTriggerSignalType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_HW_TRIG_RISING_EDGE	React on the rising edge of the hardware trigger signal
	1 - ADC_HW_TRIG_FALLING_EDGE	React on the falling edge of the hardware trigger signal
	2 - ADC_HW_TRIG_BOTH_EDGES	React on both edges of the hardware trigger signal
Description	Defines the type for configuring on which edge of the hardware trigger signal the driver should react. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.12 Adc_HwTriggerTimerType**Table 138 Specification for Adc_HwTriggerTimerType**

Syntax	Adc_HwTriggerTimerType	
Type	uint16	
File	Adc.h	
Range	0 - 1023	
Description	Defines the type for reload value of the ADC module embedded timer. Data width is specified based on the range supported by the EVADC hardware trigger timer.	
Source	AUTOSAR	

3.3.2.13 Adc_NotifyFnPtrType**Table 139 Specification for Adc_NotifyFnPtrType**

Syntax	Adc_NotifyFnPtrType
Type	Pointer to a function of type void Function_Name (void)

ADC driver**Table 139 Specification for Adc_NotifyFnPtrType (continued)**

File	Adc.h
Description	Defines the function pointer type for call back functions (on group conversion completion).
Source	IFX

3.3.2.14 Adc_PowerStateRequestResultType**Table 140 Specification for Adc_PowerStateRequestResultType**

Syntax	Adc_PowerStateRequestResultType	
Type	Enumeration	
File	Adc.h	
Range	0 - ADC_SERVICE_ACCEPTED	Power state change executed.
	1 - ADC_NOT_INIT	ADC Module not initialized.
	2 - ADC_SEQUENCE_ERROR	Wrong API call sequence.
	3 - ADC_HW_FAILURE	The hardware module has a failure which prevents it to enter the required power state or the read power state from hardware SFR corresponds to invalid range.
	4 - ADC_POWER_STATE_NOT_SUPP	ADC Module does not support the requested power state.
	5 - ADC_TRANS_NOT_POSSIBLE	ADC Module cannot transition directly from the current power state to the requested power state or the hardware peripheral is still busy.
Description	Defines the type for result of the requests related to power state transitions.	
Source	AUTOSAR	

3.3.2.15 Adc_PowerStateType**Table 141 Specification for Adc_PowerStateType**

Syntax	Adc_PowerStateType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_FULL_POWER	Full Power
	1 - FAST_STANDBY_MODE	Fast standby mode
	2 - SLOW_STANDBY_MODE	Slow standby mode
	3 - ADC_OFF	Converter is off

ADC driver**Table 141 Specification for Adc_PowerStateType (continued)**

Description	Defines the type for power state currently active or set as target power state. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.
Source	AUTOSAR

3.3.2.16 Adc_PrescaleType**Table 142 Specification for Adc_PrescaleType**

Syntax	Adc_PrescaleType	
Type	uint8	
File	Adc.h	
Range	2 - 32	Range of values for prescaler
Description	Defines the type for clock pre-scaler used for analog clock divider (DIVA). Data width is specified based on range supported by the hardware.	
Source	AUTOSAR	

3.3.2.17 Adc_PriorityImplementationType**Table 143 Specification for Adc_PriorityImplementationType**

Syntax	Adc_PriorityImplementationType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_PRIORITY_NONE 1 - ADC_PRIORITY_HW 2 - ADC_PRIORITY_HW_SW	
Description	Priority mechanism is not available	Hardware priority mechanism is available only
Source	Hardware and software priority mechanism is available	
Description	Defines the type for configuring the prioritization mechanism. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.18 Adc_ResolutionType**Table 144 Specification for Adc_ResolutionType**

Syntax	Adc_ResolutionType	
Type	uint8	
File	Adc.h	
Range	--	Unused

ADC driver**Table 144 Specification for Adc_ResolutionType (continued)**

Description	Defines the type for channel resolution in number of bits. Since this type is not used by the driver, smallest data type supported by the TriCore is selected for type definition.
Source	

3.3.2.19 Adc_ResultAlignmentType**Table 145 Specification for Adc_ResultAlignmentType**

Syntax	Adc_ResultAlignmentType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_ALIGN_RIGHT	Results are right-aligned
	1 - ADC_ALIGN_LEFT	Results are left-aligned
Description	Defines the type for alignment of ADC raw results in ADC result buffer (left/right alignment). To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.20 Adc_SamplingTimeType**Table 146 Specification for Adc_SamplingTimeType**

Syntax	Adc_SamplingTimeType	
Type	uint8	
File	Adc.h	
Range	--	Unused
	Defines the type for sampling time during which the value is sampled, (in clock-cycles). Since this type is not used by the driver, smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.21 Adc_StartupCalibStatusType**Table 147 Specification for Adc_StartupCalibStatusType**

Syntax	Adc_StartupCalibStatusType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_STARTUP_CALIB_NOT_TRIGGERED	Startup Calibration not triggered
	1 - ADC_STARTUP_CALIB_ONGOING	Startup Calibration is ongoing

ADC driver**Table 147 Specification for Adc_StartupCalibStatusType (continued)**

	2 - ADC_STARTUP_CALIB_OVER	Startup calibration over
Description	Defines the type for startup calibration status of the ADC driver. Data width is specified based on range of calibration status supported by the hardware. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	IFX	

3.3.2.22 Adc_StatusType**Table 148 Specification for Adc_StatusType**

Syntax	Adc_StatusType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_IDLE 1 - ADC_BUSY 2 - ADC_COMPLETED 3 - ADC_STREAM_COMPLETED	The conversion of the specified group has not been started
		The conversion of the specified group has been started and is still going on and so far no result is available.
		A conversion round (which is not the final one) of the specified group has been finished. At least a result is available for all channels of the group.
		The result buffer is completely filled. For each channel of the selected group the number of samples to be acquired is available
Description	Defines the type for reporting the current status of the conversion of requested channel group. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.23 Adc_StreamBufferModeType**Table 149 Specification for Adc_StreamBufferModeType**

Syntax	Adc_StreamBufferModeType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_STREAM_BUFFER_LINEAR	The ADC Driver stops the conversion as soon as the stream buffer is full (number of samples reached).

ADC driver**Table 149 Specification for Adc_StreamBufferModeType (continued)**

	1 - ADC_STREAM_BUFFER_CIRCULAR	The ADC Driver continues the conversion even if the stream buffer is full (number of samples reached) by wrapping around the stream buffer itself.
Description	Defines the type for configuring the streaming access mode buffer type. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.24 Adc_StreamNumSampleType**Table 150 Specification for Adc_StreamNumSampleType**

Syntax	Adc_StreamNumSampleType	
Type	uint8	
File	Adc.h	
Range	0 - 255	
Description	Defines the type for configuring the number of group conversions in streaming access mode.	
Source	AUTOSAR	

3.3.2.25 Adc_SyncConvModeType**Table 151 Specification for Adc_SyncConvModeType**

Syntax	Adc_SyncConvModeType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_SYNC_CONV_MODE_NONE 1 - ADC_SYNC_CONV_MODE_MASTER 2 - ADC_SYNC_CONV_MODE_SLAVE	
Description	0 - ADC_SYNC_CONV_MODE_NONE	The Sync conversion mode of the specified group is none
	1 - ADC_SYNC_CONV_MODE_MASTER	The Sync conversion mode of the specified group is Master
	2 - ADC_SYNC_CONV_MODE_SLAVE	The Sync conversion mode of the specified group is Slave
Description	Defines the type for Sync conversion mode of the ADC driver. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	IFX	

ADC driver**3.3.2.26 Adc_TriggerSourceType****Table 152 Specification for Adc_TriggerSourceType**

Syntax	Adc_TriggerSourceType	
Type	uint8	
File	Adc.h	
Range	0 - ADC_TRIGG_SRC_SW	Group is triggered by a software API call.
	1 - ADC_TRIGG_SRC_HW	Group is triggered by a hardware event.
Description	Defines the type for configuring the trigger source for an ADC channel group. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	

3.3.2.27 Adc_ValueGroupType**Table 153 Specification for Adc_ValueGroupType**

Syntax	Adc_ValueGroupType	
Type	uint16	
File	Adc.h	
Range	0x0 - 0xFFFF	
Description	Defines the type for reading the converted value of channel group (raw, without further scaling, alignment according to precompile switch ADC_RESULT_ALIGNMENT). Data width is specified based on maximum resolution supported by the EVADC hardware.	
Source	AUTOSAR	

3.3.3 Functions - APIs

This section lists all the APIs of the ADC driver.

3.3.3.1 Adc_Init**Table 154 Specification for Adc_Init API**

Syntax	void Adc_Init (const Adc_ConfigType * const ConfigPtr)
Service ID	0
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for different CPU core

ADC driver**Table 154 Specification for Adc_Init API (continued)**

Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>API initializes the ADC hardware groups as per the configuration pointer passed and sets all the global variables to their initialized state.</p> <p>The SFRs of the configured ADC hardware unit are first reset to default values and then initialized as per the configuration. It also disables the notifications and hardware trigger capability. Groups are set to default ADC_IDLE state.</p> <p>This API must be invoked from all the cores using the ADC driver, as each call initializes only the SFRs and global variables of the hardware groups used by the invoking core. The SFRs and global variables common to all cores are initialized by the MCALs master core.</p> <p>The ADC initialization status is set at the end of Initialization function execution.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_ALREADY_INITIALIZED: Error code is reported if initialization is requested from a core, which is already in the initialized state.</p> <p>ADC_E_PARAM_CONFIG: Error code is reported if the API is invoked with invalid configuration pointer (null pointer in case the master core or the slave core invokes with configuration pointer other than the one passed by the master core).</p> <p>ADC_E_MASTER_CORE_UNINIT: Error code is reported if a slave core initialization is invoked prior to initialization of the master core.</p> <p>ADC_E_CORE_NOT_CONFIGURED: Error code is reported if the API is invoked from a core which has no ADC hardware group allocated.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>ADC_E_CLC_FAILURE: Error code is reported when enabling of CLC (module clock) fails.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	<ol style="list-style-type: none"> 1. ADC driver does not perform a NULL_PTR check on ConfigPtr, when DET is off. 2. Startup calibration must be triggered by the user once the initialization from all cores is completed. 3. Mcu_Init() and Port_Init() should be called before calling this API. 4. Interrupts should be in a disabled before calling this API. 	

ADC driver**3.3.3.2 Adc_DeInit****Table 155 Specification for Adc_DeInit API**

Syntax	<pre>void Adc_DeInit (void)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different CPU core	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>API resets all SFRs of the EVADC configured during initialization to their reset states including SFRs for enabling service requests. API must be invoked from all cores using the ADC driver, as each call resets only the SFRs and global variables of the hardware groups used by the calling core. The SFRs and global variables common to all cores are reset by the MCALs master core.</p> <p>This API is only available when AdcDeInitApi is configured as TRUE.</p> <p>Note: SFRs of hardware groups not configured during Adc_Init are not de-initialized by this API. The state of an ADC driver is set to UNINIT at the beginning of the de-Initialization function.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_BUSY: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group or hardware resources used by the AdcChannel group are currently busy.</p> <p>ADC_E_SLAVE_CORE_INIT: Error code is reported if de-initialization from master core is invoked while any of the slave cores is still in the initialized state.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	AdcDeInitApi	
User hints	1. Resetting of the SRC register for various SRNs, should be handled by the OS/Application.	

ADC driver**Table 155 Specification for Adc_DeInit API (continued)**

	2. The ADC module's environment must not call the function Adc_DeInit while any group is not in state ADC_IDLE.
--	---

3.3.3.3 Adc_SetupResultBuffer**Table 156 Specification for Adc_SetupResultBuffer API**

Syntax	<pre>Std_ReturnType Adc_SetupResultBuffer (const Adc_GroupType Group, const Adc_ValueGroupType * const DataBufferPtr)</pre>	
Service ID	0x0c	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Group DataBufferPtr	Numeric ID of requested ADC channel group. Pointer to the start of result data buffer
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Result buffer pointer initialized correctly E_NOT_OK: Operation failed or development error occurred
Description	<p>API sets up the start address of group specific result buffers, where the conversion results will be stored. The application has to ensure that the application buffer, where DataBufferPtr points to, can hold all the conversion results of the specified group.</p> <p>Note: This API is not available when AdcResultHandlingImplementation is set to DMA mode of result handling. In this scenario the start of the application result buffer is provided through DMA channels destination address.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_BUSY: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group or hardware resources used by the AdcChannel group are currently busy.</p> <p>ADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.</p> <p>ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.</p>	

ADC driver**Table 156 Specification for Adc_SetupResultBuffer API (continued)**

	<p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcResultHandlingImplementation
User hints	<ol style="list-style-type: none"> ADC module's environment must ensure that no group conversions are started without prior initialization of the according result buffer pointer to point to a valid result buffer. ADC module's environment must ensure that the application buffer, whose address is passed as parameter in Adc_SetupResultBuffer, has the according size to hold all group channel conversion results and if streaming access is selected, hold these results multiple times as specified with streaming sample parameter. ADC Driver does not work as expected if NULL_PTR is passed via DataBufferPtr when DET is off.

3.3.3.4 Adc_EnableGroupNotification**Table 157 Specification for Adc_EnableGroupNotification API**

Syntax	<pre>void Adc_EnableGroupNotification (const Adc_GroupType Group)</pre>	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	API enables the notification mechanism for the requested ADC channel group. <i>Note: This API is only available when AdcGrpNotifCapability is configured as TRUE.</i>	
Source	AUTOSAR	
Error handling	DET: ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API. ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.	

ADC driver**Table 157 Specification for Adc_EnableGroupNotification API (continued)**

	<p>ADC_E_NOTIF_CAPABILITY: Error code is reported if enable or disable notification functions is invoked with an AdcChannel group whose configuration set has no notification or NULL_PTR configured as notification.</p> <p>ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcGrpNotifCapability
User hints	None

3.3.3.5 Adc_DisableGroupNotification**Table 158 Specification for Adc_DisableGroupNotification API**

Syntax	<pre>void Adc_DisableGroupNotification (const Adc_GroupType Group)</pre>	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>API disables the notification mechanism for the requested ADC channel group.</p> <p><i>Note: This API is only available when AdcGrpNotifCapability is configured as TRUE.</i></p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.</p>	

ADC driver**Table 158 Specification for Adc_DisableGroupNotification API (continued)**

	<p>ADC_E_NOTIF_CAPABILITY: Error code is reported if enable or disable notification functions is invoked with an AdcChannel group whose configuration set has no notification or NULL_PTR configured as notification.</p> <p>ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcGrpNotifCapability
User hints	None

3.3.3.6 Adc_StartGroupConversion**Table 159 Specification for Adc_StartGroupConversion API**

Syntax	<pre>void Adc_StartGroupConversion (const Adc_GroupType Group)</pre>	
Service ID	0x02	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Reentrant for channel groups executing on different ADC hardware groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>API starts the conversion of all the channels of the requested AdcChannel group.</p> <p><i>Note: This API is available only when AdcEnableStartStopGroupApi is configured as TRUE.</i></p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_BUSY: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group or hardware resources used by the AdcChannel group are currently busy.</p>	

ADC driver**Table 159 Specification for Adc_StartGroupConversion API (continued)**

	<p>ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.</p> <p>ADC_E_WRONG_TRIGGER_SRC: Error code is reported if an AdcChannel group with trigger source as software is invoked through a hardware trigger API or an AdcChannel group with trigger source as hardware is invoked through a software trigger API.</p> <p>ADC_E_BUFFER_UNINIT: Error code is reported if a conversion start request is placed while the result buffer pointer is not initialized.</p> <p>ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.</p> <p>ADC_E_CONVERTOR_OFF: Error code is reported if start conversion or enable hardware trigger is invoked while the convertor is in OFF state.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ADC_SE_CALIB_ONGOING: Error code is reported if start conversion or enable hardware trigger is invoked while start-up calibration is ongoing.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcEnableStartStopGroupApi
User hints	<ol style="list-style-type: none"> 1. ADC Driver does not work as expected if wrong group is passed when DET is off 2. The ADC module's environment should call the function Adc_StartGroupConversion for groups configured with software trigger source only.

3.3.3.7 Adc_StopGroupConversion**Table 160 Specification for Adc_StopGroupConversion API**

Syntax	<pre>void Adc_StopGroupConversion (const Adc_GroupType Group)</pre>	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for channel groups executing on different ADC hardware groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

ADC driver**Table 160 Specification for Adc_StopGroupConversion API (continued)**

Description	API stops the conversion of the requested ADC channel group. It also disables the group notification and sets the group to IDLE state. Note: This API is available only when AdcEnableStartStopGroupApi is configured as TRUE.
Source	AUTOSAR
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_IDLE: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group is currently in the IDLE state.</p> <p>ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.</p> <p>ADC_E_WRONG_TRIGGER_SRC: Error code is reported if an AdcChannel group with trigger source as software is invoked through a hardware trigger API or an AdcChannel group with trigger source as hardware is invoked through a software trigger API.</p> <p>ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>ADC_E_CONV_STOP_TIME_FAILURE: Error code is reported when an ongoing conversion cannot be stopped due to unknown hardware failure.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcEnableStartStopGroupApi
User hints	ADC module's environment should call the function Adc_StopGroupConversion for groups configured with trigger source software only.

3.3.3.8 Adc_EnableHardwareTrigger**Table 161 Specification for Adc_EnableHardwareTrigger API**

Syntax	void Adc_EnableHardwareTrigger (const Adc_GroupType Group)	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for channel groups executing on different ADC hardware groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group.

ADC driver**Table 161 Specification for Adc_EnableHardwareTrigger API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	API enables the hardware trigger or gate for the requested ADC channel group. Note: This API is available only when AdcHwTriggerApi is configured as TRUE.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <ul style="list-style-type: none"> ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API. ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID. ADC_E_WRONG_CONV_MODE: Error code is reported if Adc_EnableHardwareTrigger or Adc_DisableHardwareTrigger is called for a group with the conversion mode configured as continuous. ADC_E_WRONG_TRIGG_SRC: Error code is reported if an AdcChannel group with trigger source as software is invoked through a hardware trigger API or an AdcChannel group with trigger source as hardware is invoked through a software trigger API. ADC_E_BUFFER_UNINIT: Error code is reported if a conversion start request is placed while the result buffer pointer is not initialized. ADC_E_BUSY: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group or hardware resources used by the AdcChannel group are currently busy. ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core. ADC_E_CONVERTOR_OFF: Error code is reported if start conversion or enable hardware trigger is invoked while the convertor is in OFF state. <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <ul style="list-style-type: none"> ADC_SE_CALIB_ONGOING: Error code is reported if start conversion or enable hardware trigger is invoked while start-up calibration is ongoing. <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	AdcHwTriggerApi	
User hints	<ol style="list-style-type: none"> 1. ADC module's environment should call the function Adc_EnableHardwareTrigger for groups configured in hardware trigger mode only. 2. ADC module's environment must guarantee that no concurrent conversions take place on the same hardware unit (happening of different hardware triggers at the same time). 	

ADC driver**3.3.3.9 Adc_DisableHardwareTrigger****Table 162 Specification for Adc_DisableHardwareTrigger API**

Syntax	<pre>void Adc_DisableHardwareTrigger (const Adc_GroupType Group)</pre>	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for channel groups executing on different ADC hardware groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>API stops the ongoing conversion and disables hardware trigger for the requested ADC channel group. It also disables the group notification and sets the group to IDLE state.</p> <p>Note: This API is available only when AdcHwTriggerApi is configured as TRUE.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.</p> <p>ADC_E_WRONG_CONV_MODE: Error code is reported if Adc_EnableHardwareTrigger or Adc_DisableHardwareTrigger is called for a group with the conversion mode configured as continuous.</p> <p>ADC_E_WRONG_TRIGG_SRC: Error code is reported if an AdcChannel group with trigger source as software is invoked through a hardware trigger API or an AdcChannel group with trigger source as hardware is invoked through a software trigger API.</p> <p>ADC_E_IDLE: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group is currently in the IDLE state.</p> <p>ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>ADC_E_CONV_STOP_TIME_FAILURE: Error code is reported when an ongoing conversion cannot be stopped due to unknown hardware failure.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

ADC driver**Table 162 Specification for Adc_DisableHardwareTrigger API (continued)**

Configuration dependencies	AdcHwTriggerApi
User hints	ADC module's environment should call the function Adc_DisableHardwareTrigger for groups configured in hardware trigger mode only.

3.3.3.10 Adc_GetGroupStatus**Table 163 Specification for Adc_GetGroupStatus API**

Syntax	Adc_StatusType Adc_GetGroupStatus (const Adc_GroupType Group)	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Adc_StatusType	Conversion status for the requested group.
Description	API returns the status of the requested ADC channel group.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.</p> <p>ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

ADC driver**3.3.3.11 Adc_ReadGroup****Table 164 Specification for Adc_ReadGroup API**

Syntax	<pre>Std_ReturnType Adc_ReadGroup (const Adc_GroupType Group, Adc_ValueGroupType * const DataBufferPtr)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different AdcChannel Groups	
Parameters (in)	Group	Numeric ID of requested ADC channel group.
Parameters (out)	-	-
Parameters (in - out)	DataBufferPtr	ADC results of all channels of the selected group are stored in the data buffer addressed with the pointer.
Return	Std_ReturnType	E_OK: Results are available and written to the data buffer. E_NOT_OK: No results are available or development error occurred
Description	<p>API reads the group conversion result of the last completed conversion round of the requested group and stores the channel conversion value starting from DataBufferPtr address. The channel conversion results are stored in ascending channel order.</p> <p>Note:</p> <ol style="list-style-type: none"> 1. This API is available only when AdcReadGroupApi is configured as TRUE. 2. This API is not available when AdcResultHandlingImplementation is set to DMA mode of result handling. 	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_IDLE: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group is currently in the IDLE state.</p> <p>ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.</p> <p>ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.</p> <p>ADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

ADC driver**Table 164 Specification for Adc_ReadGroup API (continued)**

Configuration dependencies	AdcReadGroupApi
User hints	ADC module's environment must ensure that a conversion has been completed for the requested group before requesting the conversion result.

3.3.3.12 Adc_GetStreamLastPointer**Table 165 Specification for Adc_GetStreamLastPointer API**

Syntax	Adc_StreamNumSampleType Adc_GetStreamLastPointer (const Adc_GroupType Group, Adc_ValueGroupType ** const PtrToSamplePtr)	
Service ID	0x0b	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different AdcChannel Groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group.
Parameters (out)	PtrToSamplePtr	Pointer to result buffer pointer.
Parameters (in - out)	-	-
Return	Adc_StreamNumSampleType	Number of valid samples per channel.
Description	<p>API returns the number of valid samples per channel, stored in the result buffer. It also updates the PtrToSamplePtr with the address (within the group result buffer) of the latest result sample of the 1st channel. With pointer and the return value, all valid group conversion results can be accessed.</p> <p>Note: This API is not available when AdcResultHandlingImplementation is set to DMA mode of result handling.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.</p> <p>ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.</p> <p>ADC_E_IDLE: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group is currently in the IDLE state.</p>	

ADC driver**Table 165 Specification for Adc_GetStreamLastPointer API (continued)**

	<p>ADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcResultHandlingImplementation
User hints	None

3.3.3.13 Adc_GetCurrentPowerState**Table 166 Specification for Adc_GetCurrentPowerState API**

Syntax	<pre>Std_ReturnType Adc_GetCurrentPowerState (Adc_PowerStateType * const CurrentPowerState, Adc_PowerStateRequestResultType * const Result)</pre>	
Service ID	0x11	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	CurrentPowerState Result	<p>The current power mode of the ADC HW Unit is returned in this parameter</p> <p>If the API returns E_OK:</p> <p>ADC_SERVICE_ACCEPTED: Current power mode was returned.</p> <p>If the API returns E_NOT_OK:</p> <p>ADC_NOT_INIT: ADC Module not initialized.</p> <p>ADC_HW_FAILURE: Current power state read from the SFR corresponds to invalid range.</p>
Parameters (in - out)	-	-
Return	Std_ReturnType	<p>E_OK: Power mode could be provided</p> <p>E_NOT_OK: Power mode could not be provided</p>
Description	<p>API returns the current power state of the ADC hardware groups assigned to the calling core.</p> <p>Note: This API is available only when AdcLowPowerStatesSupport is configured as TRUE.</p>	
Source	AUTOSAR	

ADC driver**Table 166 Specification for Adc_GetCurrentPowerState API (continued)**

Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ADC_SE_POWER_STATE_INVALID: Error code is reported if power state read from the hardware is invalid (not configured).</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcLowPowerStatesSupport
User hints	None

3.3.3.14 Adc_GetTargetPowerState**Table 167 Specification for Adc_GetTargetPowerState API**

Syntax	<pre>Std_ReturnType Adc_GetTargetPowerState (Adc_PowerStateType * const TargetPowerState, Adc_PowerStateRequestResultType * const Result)</pre>	
Service ID	0x12	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	TargetPowerState Result	The Target power mode of the ADC HW Unit is returned in this parameter If the API returns E_OK: ADC_SERVICE_ACCEPTED:Target power mode was returned. If the API returns E_NOT_OK: ADC_NOT_INIT: ADC Module not initialized. ADC_HW_FAILURE: Current power state read from the SFR corresponds to invalid range.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Power mode could be provided

ADC driver**Table 167 Specification for Adc_GetTargetPowerState API (continued)**

	E_NOT_OK: Power mode could not be provided
Description	API returns the power state successfully prepared by Adc_PrepPowerState API. If the power state was not prepared, then the current power state of hardware units assigned to the invoking core is returned. Note: This API is available only when AdcLowPowerStatesSupport is configured as TRUE.
Source	AUTOSAR
Error handling	DET: ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API. ADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter. Runtime Errors: None DEM: None Safety Errors: ADC_SE_POWER_STATE_INVALID: Error code is reported if power state read from the hardware is invalid (not configured). <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	AdcLowPowerStatesSupport
User hints	None

3.3.3.15 Adc_PrepPowerState**Table 168 Specification for Adc_PrepPowerState API**

Syntax	<pre>Std_ReturnType Adc_PrepPowerState (const Adc_PowerStateType PowerState, Adc_PowerStateRequestResultType * const Result)</pre>	
Service ID	0x13	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different CPU cores	
Parameters (in)	PowerState	The target power state intended to be attained
Parameters (out)	Result	If the API returns E_OK: ADC_SERVICE_ACCEPTED: ADC Module power state preparation was started. ADC_SEQUENCE_ERROR: Current power state of all hardware units is same as the target power state

ADC driver**Table 168 Specification for Adc_PrepPowerState API (continued)**

		If the API returns E_NOT_OK: ADC_NOT_INIT: ADC Module not initialized. ADC_POWER_STATE_NOT_SUPP: ADC Module does not support the requested power state. ADC_HW_FAILURE: Current power state read from the SFR corresponds to invalid range.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Preparation process started or Current power state of all the hardware units is same as the target power state E_NOT_OK: Service is rejected
Description	<p>API starts the needed process for the ADC hardware units assigned to the invoking core to enter the target power state. If current power state of the hardware units assigned to the invoking core is same as the target power state, then the API does not perform the intended action. The API must be invoked from all the cores using the ADC driver, as each call prepares the power state only for the ADC hardware groups used by the calling core.</p> <p>Note: This API is available only when AdcLowPowerStatesSupport is configured as TRUE.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_POWER_STATE_NOT_SUPPORTED: Error code is reported if the API is invoked with an unsupported power state.</p> <p>ADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ADC_SE_POWER_STATE_INVALID: Error code is reported if power state read from the hardware is invalid (not configured).</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	AdcLowPowerStatesSupport	
User hints	None	

3.3.3.16 Adc_SetPowerState**Table 169 Specification for Adc_SetPowerState API**

Syntax	Std_ReturnType Adc_SetPowerState (
---------------	---------------------------------------

ADC driver**Table 169 Specification for Adc_SetPowerState API (continued)**

	Adc_PowerStateRequestResultType * const Result)	
Service ID	0x10	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different CPU cores	
Parameters (in)	-	-
Parameters (out)	Result	If the API returns E_OK: ADC_SERVICE_ACCEPTED: Power state change executed. If the API returns E_NOT_OK: ADC_NOT_INIT: ADC Module not initialized. ADC_SEQUENCE_ERROR: wrong API call sequence. ADC_TRANS_NOT_POSSIBLE: ADC channel groups are not in state IDLE or notifications enabled.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Power Mode changed E_NOT_OK: Service is rejected
Description	<p>API sets the already prepared power states for all the ADC hardware units assigned to the invoking core. The API must be invoked from all the cores using the ADC driver, as each call sets the power state only for the ADC hardware groups used by the calling core.</p> <p>Note:</p> <ol style="list-style-type: none"> When the converter is set to 'full power mode', then the conversion is started immediately by the hardware after a request (no additional time delay). This is the default mode after modules initialization. When the converter is set to 'Fast standby mode' or 'Slow standby mode' and no conversion is requested, then it saves power. When a conversion is triggered then the converter wakes up automatically, but it needs certain wake-up time before conversions can be performed. There is no power saving while the conversions are being performed during this mode. During this phase the driver reports the current power state as 'Fast standby mode' or 'Slow standby mode'. In this mode, when the conversion is no longer active, then the converter saves power and a new conversion request will trigger the wake-up cycle again (time delay for wake-up). When the converter is set to 'off mode', it enters power saving mode and no further conversions are possible until a power mode transition is made explicitly to come out of 'off mode'. This API is available only when AdcLowPowerStatesSupport is configured as TRUE. 	
Source	AUTOSAR	
Error handling	DET:	

ADC driver**Table 169 Specification for Adc_SetPowerState API (continued)**

	<p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>ADC_E_NOT_DISENGAGED: Error code is reported if the API is invoked while one or more AdcChannel group is not in IDLE state or has notification enabled.</p> <p>ADC_E_PERIPHERAL_NOT_PREPARED: Error code is reported if the ADC is not prepared for the target power state.</p> <p>ADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcLowPowerStatesSupport
User hints	None

3.3.3.17 Adc_GetVersionInfo**Table 170 Specification for Adc_GetVersionInfo API**

Syntax	<pre>void Adc_GetVersionInfo (Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x0a	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where to store the version information of the ADC driver.
Parameters (in - out)	-	-
Return	void	-
Description	<p>API returns the version information of this driver.</p> <p><i>Note: This API is available only when AdcVersionInfoApi is configured as TRUE.</i></p>	
Source	AUTOSAR	
Error handling	DET:	

ADC driver**Table 170 Specification for Adc_GetVersionInfo API (continued)**

	<p>ADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcVersionInfoApi
User hints	-

3.3.3.18 Adc_TriggerStartupCal**Table 171 Specification for Adc_TriggerStartupCal API**

Syntax	Std_ReturnType Adc_TriggerStartupCal (void)	
Service ID	0x31	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Start-up calibration is triggered E_NOT_OK: Start-up calibration is not triggered
Description	<p>API triggers the start-up calibration. The API can be invoked from any core, it should be ensured that initialization sequence of all the cores is over before invoking the API.</p> <p><i>Note: This API is available only when AdcStartupCalibApi is configured as TRUE.</i></p>	
Source	IFX	
Error handling	<p>DET:</p> <p>ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p>	

ADC driver**Table 171 Specification for Adc_TriggerStartupCal API (continued)**

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	AdcStartupCalibApi
User hints	API should be triggered after the ADC initialization is completed in all the Cores.

3.3.3.19 Adc_GetStartupCalStatus**Table 172 Specification for Adc_GetStartupCalStatus API**

Syntax	Adc_StartupCalibStatusType Adc_GetStartupCalStatus (void)	
Service ID	0x30	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Adc_StartupCalibStatusType	ADC_STARTUP_CALIB_NOT_TRIGGERED: Startup Calibration not triggered or DET occurred ADC_STARTUP_CALIB_ONGOING: Startup Calibration is ongoing. ADC_STARTUP_CALIB_OVER: Startup calibration over
Description	API returns the status of the start-up calibration for all the ADC hardware groups assigned to invoking core. Note: This API is available only when AdcStartupCalibApi is configured as TRUE.	
Source	IFX	
Error handling	DET: ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	AdcStartupCalibApi	

ADC driver**Table 172 Specification for Adc_GetStartupCalStatus API (continued)**

User hints	None
-------------------	------

3.3.3.20 Adc_InitCheck**Table 173 Specification for Adc_InitCheck API**

Syntax	<pre>Std_ReturnType Adc_InitCheck (const Adc_ConfigType * const ConfigPtr)</pre>	
Service ID	0x32	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different CPU core	
Parameters (in)	ConfigPtr	Pointer to ADC configuration Set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Initialization check passed E_NOT_OK: Initialization check failed, passed ConfigPtr is NULL or initialization for current core is not completed
Description	<p>API checks whether the initialization was performed correctly or not. The check for correct initialization is only done in the context of the invoking core. The API is available when initialization check is explicitly enabled.</p> <p>Note: Init check should be performed in the following sequence:</p> <ol style="list-style-type: none"> 1. Call Adc_Init from a core. 2. Call Adc_InitCheck from the same core. 	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	AdcSafetyEnable,AdcInitCheckApi	
User hints	The ADC module's environment must ensure that Adc_InitCheck should be performed in the following sequence: <ol style="list-style-type: none"> 1. Call Adc_Init from a core. 2. Call Adc_InitCheck from the same core. 	

ADC driver**3.3.4 Notifications and Callbacks**

The driver does not support any notifications and callbacks.

3.3.5 Scheduled functions

The driver does not support any scheduled functions

3.3.6 Interrupt service routines

This section lists all the interrupt handlers of the driver.

3.3.6.1 Adc_ChEventInterruptHandler
Table 174 Specification for Adc_ChEventInterruptHandler API

Syntax	<pre>void Adc_ChEventInterruptHandler (const uint32 KernelId)</pre>	
Service ID	0x36	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different ADC hardware groups	
Parameters (in)	KernelId	Hardware group ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Handles the interrupts from a channel event for the passed ADC KernelId. Channel events are triggered when the conversion results are in range (as configured) for a limit checking group. Application will be notified if a notification is configured (in tresos) and enabled.</p> <p>Note: This API is available only when AdcLimitCheckApi is configured as TRUE.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>ADC_E_CONV_STOP_TIME_FAILURE: Error code is reported when an ongoing conversion cannot be stopped due to unknown hardware failure.</p> <p>Safety Errors:</p>	

ADC driver**Table 174 Specification for Adc_ChEventInterruptHandler API (continued)**

	<p>ADC_SE_INT_PLAUSIBILITY: Error code is reported if an unintended interrupt is triggering the ISR.</p> <p>ADC_SE_PARAM_KERNEL: Error code is reported if the kernel ID passed is not configured or it is beyond the available range in the hardware.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcEnableLimitCheck
User hints	User must call this interrupt handler from the ISR of GxSRN3 of each kernel and pass the kernel ID as the parameter.

3.3.6.2 Adc_RS0EventInterruptHandler**Table 175 Specification for Adc_RS0EventInterruptHandler API**

Syntax	<pre>void Adc_RS0EventInterruptHandler (const uint32 KernelId)</pre>	
Service ID	0x33	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different ADC hardware groups	
Parameters (in)	KernelId	Hardware group ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Handles the interrupts from Request Source 0 event for the passed ADC KernelId. Request Source 0 event is triggered when all channels of an AdcGroup installed on it, completes one round of conversion.</p> <p>Application will be notified if a notification is configured(in tresos) and enabled.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>ADC_E_CONV_STOP_TIME_FAILURE: Error code is reported when an ongoing conversion cannot be stopped due to unknown hardware failure.</p>	

ADC driver**Table 175 Specification for Adc_RS0EventInterruptHandler API (continued)**

	<p>Safety Errors:</p> <p>ADC_SE_INT_PLAUSIBILITY: Error code is reported if an unintended interrupt is triggering the ISR.</p> <p>ADC_SE_PARAM_KERNEL: Error code is reported if the kernel ID passed is not configured or it is beyond the available range in the hardware.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	User must call this interrupt handler from the ISR of GxSRN0 of each kernel and pass the kernel ID as the parameter.

3.3.6.3 Adc_RS1EventInterruptHandler**Table 176 Specification for Adc_RS1EventInterruptHandler API**

Syntax	<pre>void Adc_RS1EventInterruptHandler (const uint32 KernelId)</pre>	
Service ID	0x34	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different ADC hardware groups	
Parameters (in)	KernelId	Hardware group ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Handles the interrupts from Request Source 1 event for the passed ADC KernelId. Request Source 1 event is triggered when all channels of an AdcGroup installed on it, completes one round of conversion.</p> <p>Application will be notified if a notification is configured(in tresos) and enabled.</p> <p><i>Note: This function is available only if AdcPriorityImplementation is not equal to NONE.</i></p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM:</p>	

ADC driver**Table 176 Specification for Adc_RS1EventInterruptHandler API (continued)**

	<p>ADC_E_CONV_STOP_TIME_FAILURE: Error code is reported when an ongoing conversion cannot be stopped due to unknown hardware failure.</p> <p>Safety Errors:</p> <p>ADC_SE_INT_PLAUSIBILITY: Error code is reported if an unintended interrupt is triggering the ISR.</p> <p>ADC_SE_PARAM_KERNEL: Error code is reported if the kernel ID passed is not configured or it is beyond the available range in the hardware.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcPriorityImplementation
User hints	User must call this interrupt handler from the ISR of GxSRN1 of each kernel and pass the kernel ID as the parameter.

3.3.6.4 Adc_RS2EventInterruptHandler**Table 177 Specification for Adc_RS2EventInterruptHandler API**

Syntax	<pre>void Adc_RS2EventInterruptHandler (const uint32 KernelId)</pre>	
Service ID	0x35	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different ADC hardware groups	
Parameters (in)	KernelId	Hardware group ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Handles the interrupts from Request Source 2 event for the passed ADC KernelId. Request Source 2 event is triggered when all channels of an AdcGroup installed on it, completes one round of conversion.</p> <p>Application will be notified if a notification is configured(in tresos) and enabled.</p> <p><i>Note: This function is available only if AdcPriorityImplementation is not equal to NONE.</i></p>	
Source	IFX	
Error handling	DET: None	

ADC driver**Table 177 Specification for `Adc_RS2EventInterruptHandler` API (continued)**

	<p>Runtime Errors: None</p> <p>DEM:</p> <p>ADC_E_CONV_STOP_TIME_FAILURE: Error code is reported when an ongoing conversion cannot be stopped due to unknown hardware failure.</p> <p>Safety Errors:</p> <p>ADC_SE_INT_PLAUSIBILITY: Error code is reported if an unintended interrupt is triggering the ISR.</p> <p>ADC_SE_PARAM_KERNEL: Error code is reported if the kernel ID passed is not configured or it is beyond the available range in the hardware.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	AdcPriorityImplementation
User hints	User must call this interrupt handler from the ISR of GxSRN2 of each kernel and pass the kernel ID as the parameter.

3.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

3.3.7.1 Development errors

The following table lists all the development errors reported by the driver. Note: The following error IDs are also reported as safety errors.

Table 178 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.	AUTOSAR	ADC_E_UNINIT=0x0A	Adc_TriggerStartupCal, Adc_PrepPowerState, Adc_GetTargetPowerState , Adc_GetCurrentPowerStat e, Adc_SetPowerState, Adc_GetStreamLastPointe r, Adc_ReadGroup, Adc_GetGroupStatus, Adc_GetStartupCalStatus, Adc_EnableGroupNotifica tion, Adc_DisableGroupNotifica tion, Adc_SetupResultBuffer, Adc_EnableHardwareTrig ger, Adc_DisableHardwareTrig ger,

ADC driver**Table 178 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
			Adc_StopGroupConversion, Adc_StartGroupConversion, Adc_DeInit
Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group or hardware resources used by the AdcChannel group are currently busy.	AUTOSAR	ADC_E_BUSY=0x0B	Adc_SetupResultBuffer, Adc_EnableHardwareTrigger, Adc_StartGroupConversion, Adc_DeInit
Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group is currently in the IDLE state.	AUTOSAR	ADC_E_IDLE=0x0C	Adc_GetStreamLastPointer, Adc_ReadGroup, Adc_DisableHardwareTrigger, Adc_StopGroupConversion
Error code is reported if initialization is requested from a core, which is already in the initialized state.	AUTOSAR	ADC_E_ALREADY_INITIALIZED=0x0D	Adc_Init
Error code is reported if the API is invoked with invalid configuration pointer (null pointer in case the master core or the slave core invokes with configuration pointer other than the one passed by the master core).	AUTOSAR	ADC_E_PARAM_CONFIG=0x0E	Adc_Init
Error code is reported if the API is invoked with null-pointer as a parameter.	AUTOSAR	ADC_E_PARAM_POINTER=0x14	Adc_GetStreamLastPointer, Adc_GetCurrentPowerState, Adc_GetTargetPowerState, Adc_SetPowerState, Adc_PrepPowerState, Adc_ReadGroup, Adc_SetupResultBuffer, Adc_GetVersionInfo
Error code is reported if the API is invoked with an invalid group ID.	AUTOSAR	ADC_E_PARAM_GROUP=0x15	Adc_GetStreamLastPointer, Adc_ReadGroup, Adc_GetGroupStatus, Adc_EnableGroupNotifica

ADC driver**Table 178 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
			tion, Adc_DisableGroupNotification, Adc_SetupResultBuffer, Adc_EnableHardwareTrigger, Adc_DisableHardwareTrigger, Adc_StopGroupConversion, Adc_StartGroupConversion
Error code is reported if Adc_EnableHardwareTrigger or Adc_DisableHardwareTrigger is called for a group with the conversion mode configured as continuous.	AUTOSAR	ADC_E_WRONG_CONV_MODE=0x16	Adc_EnableHardwareTrigger, Adc_DisableHardwareTrigger
Error code is reported if an AdcChannel group with trigger source as software is invoked through a hardware trigger API or an AdcChannel group with trigger source as hardware is invoked through a software trigger API.	AUTOSAR	ADC_E_WRONG_TRIGG_SRC=0x17	Adc_EnableHardwareTrigger, Adc_DisableHardwareTrigger, Adc_StopGroupConversion, Adc_StartGroupConversion
Error code is reported if enable or disable notification functions is invoked with an AdcChannel group whose configuration set has no notification or NULL_PTR configured as notification.	AUTOSAR	ADC_E_NOTIF_CAPABILITY=0x18	Adc_EnableGroupNotification, Adc_DisableGroupNotification
Error code is reported if a conversion start request is placed while the result buffer pointer is not initialized.	AUTOSAR	ADC_E_BUFFER_UNINIT=0x19	Adc_EnableHardwareTrigger, Adc_StartGroupConversion
Error code is reported if the API is invoked while one or more AdcChannel group is not in IDLE state	AUTOSAR	ADC_E_NOT_DISENGAGED=0x1A	Adc_SetPowerState

ADC driver**Table 178 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
or has notification enabled.			
Error code is reported if the API is invoked with an unsupported power state.	AUTOSAR	ADC_E_POWER_STATE_NOT_SUPPORTED=0x1B	Adc_PrepPowerState
Error code is reported if the ADC is not prepared for the target power state.	AUTOSAR	ADC_E_PERIPHERAL_NOT_PREPARED=0x1D	Adc_SetPowerState
Error code is reported if a slave core initialization is invoked prior to initialization of the master core.	IFX	ADC_E_MASTER_CORE_UNINIT=0x66	Adc_Init
Error code is reported if de-initialization from master core is invoked while any of the slave cores is still in the initialized state.	IFX	ADC_E_SLAVE_CORE_INIT=0x67	Adc_DelInit
Error code is reported if the API is invoked from a core which has no ADC hardware group allocated.	IFX	ADC_E_CORE_NOT_CONFIGURED=0x64	Adc_Init
Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.	IFX	ADC_E_CORE_GROUP_MISMATCH=0x65	Adc_GetStreamLastPointer, Adc_ReadGroup, Adc_GetGroupStatus, Adc_EnableHardwareTrigger, Adc_DisableHardwareTrigger, Adc_StopGroupConversion, Adc_StartGroupConversion, Adc_EnableGroupNotification, Adc_DisableGroupNotification, Adc_SetupResultBuffer
Error code is reported if start conversion or enable hardware trigger is invoked while the convertor is in OFF state.	IFX	ADC_E_CONVERTOR_OFF=0x30	Adc_EnableHardwareTrigger, Adc_StartGroupConversion

ADC driver**3.3.7.2 Production errors**

The following table lists all the production errors reported by the driver.

Table 179 Description of production errors reported

Description	Source	Error code and value	Applicable APIs
Error code is reported when enabling of CLC (module clock) fails.	IFX	ADC_E_CLC_FAILURE=Assigned by DEM	Adc_Init
Error code is reported when an ongoing conversion cannot be stopped due to unknown hardware failure.	IFX	ADC_E_CONV_STOP_TIME_FAILURE=Assigned by DEM	Adc_ChEventInterruptHandler, Adc_RS2EventInterruptHandler, Adc_RS1EventInterruptHandler, Adc_RS0EventInterruptHandler, Adc_DisableHardwareTrigger, Adc_StopGroupConversion

3.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 180 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
Error code is reported if the kernel ID passed is not configured or it is beyond the available range in the hardware.	IFX	ADC_SE_PARAM_KERNEL=0xCB	Adc_ChEventInterruptHandler, Adc_RS2EventInterruptHandler, Adc_RS1EventInterruptHandler, Adc_RS0EventInterruptHandler
Error code is reported if start conversion or enable hardware trigger is invoked while start-up calibration is ongoing.	IFX	ADC_SE_CALIB_ONGOING=0xC8	Adc_EnableHardwareTrigger, Adc_StartGroupConversion
Error code is reported if an unintended interrupt is triggering the ISR.	IFX	ADC_SE_INT_PLAUSIBILITY=0xC9	Adc_ChEventInterruptHandler, Adc_RS2EventInterruptHandler, Adc_RS1EventInterruptHandler, Adc_RS0EventInterruptHandler

ADC driver

Table 180 Description of safety errors reported (continued)

Description	Source	Error code and value	Applicable APIs
Error code is reported if power state read from the hardware is invalid (not configured).	IFX	ADC_SE_POWER_STATE_INVALID=0x CA	Adc_PrepPowerState, Adc_GetTargetPowerState , Adc_GetCurrentPowerStat e

3.3.7.4 Runtime errors

The driver does not report any runtime errors.

3.3.8 Deviations and limitations

This section describes the deviations and limitations from software specification.

3.3.8.1 Deviations

This section describes the deviations from software specification.

Table 181 Known deviations

Reference	Deviation
ECUC_AdC_00389: AdcHwUnitId	The AdcHwUnitId parameter is implemented as pre-compile instead of post-build. The value assigned to AdcHwUnitId may change only at pre-compile time.
ECUC_AdC_00392: AdcChannelId	The AdcChannelId parameter is implemented as pre-compile instead of post-build. The value assigned to AdcChannelId may change only at pre-compile time.
Access of SRC registers	The ADC driver updates the SRC registers of EVADC (MODULE_SRC.VADC.G[x].SRy) to clear the pending interrupt requests. Refer to the Key architectural consideration section for information on how to update this shared resource simultaneously by the application and the MCAL driver.
Address and Data CRC in DMA mode	Since the Data CRC and Address CRC features of DMA are not used for ADC driver, the user shall ensure that, while using the DMA mode a plausibility check of the conversion result is performed either by redundancy or by other means.
Safety error for unintended service request	Refer to Reporting of unintended service requests .

3.3.8.2 Limitations

The following are the limitations for the ADC driver.

ADC driver**Table 182 Known limitations**

Reference	Limitation
Input Class for channels in synchronous conversion	The user must configure global input classes (for CHCTR.ICSEL through the AdcInputClassSelection configuration parameter) for the master and slave channels so that the conversion properties and timings for the master and slave channels are the same. If the global input class is not used, the properties of the kernel-specific input classes used for the master and slave channels must be identical.
Order of conversion for groups with same priority in HW-SW priority mode	The order of conversion for group with same priority in the hardware-software priority is not first come first serve always. The order is determined by hardware arbiter runtime. Among the groups with the same priority the one installed on RS0 is always executed first followed by the one on RS1 and then RS2.
Input parameter to the Adc_InitCheck() API is unused	Input parameter to the Adc_InitCheck() API is unused, instead the parameter passed during invocation of the Adc_Init() API is used for InitCheck evaluation.

3.3.9 **Unsupported hardware features**

The following hardware features of the EVADC are not supported:

- Wait-for-read mode
- Conversion result modification modes (Standard Data Reduction mode, Filtering mode and Difference mode)
- External multiplexer
- Fast compare channels
- FIFO mode of result registers
- Pull-down diagnostics
- Multiplexer diagnostics
- Converter diagnostics

4 BFX driver

4.1 User information

4.1.1 Description

The BFX library provides bit handling functionality for fixed-point data specified by AUTOSAR. The library provides support for 8-bit, 16-bit and 32-bit data. The library provides all its functionality, independent of any underlying hardware IP.

4.1.2 Hardware-software mapping

This section is not applicable for the BFX library.

4.1.3 File structure

4.1.3.1 C file structure

This section provides details of the C files of the BFX library and the interdependence between them.

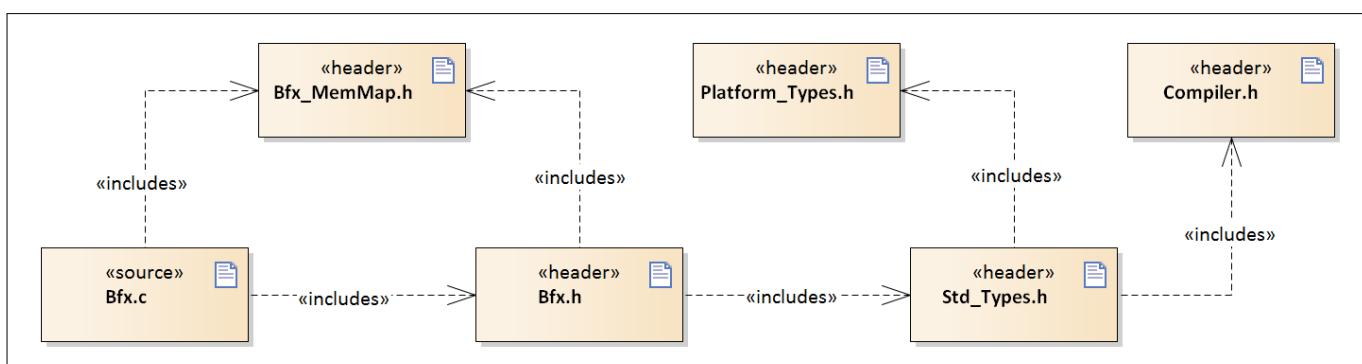


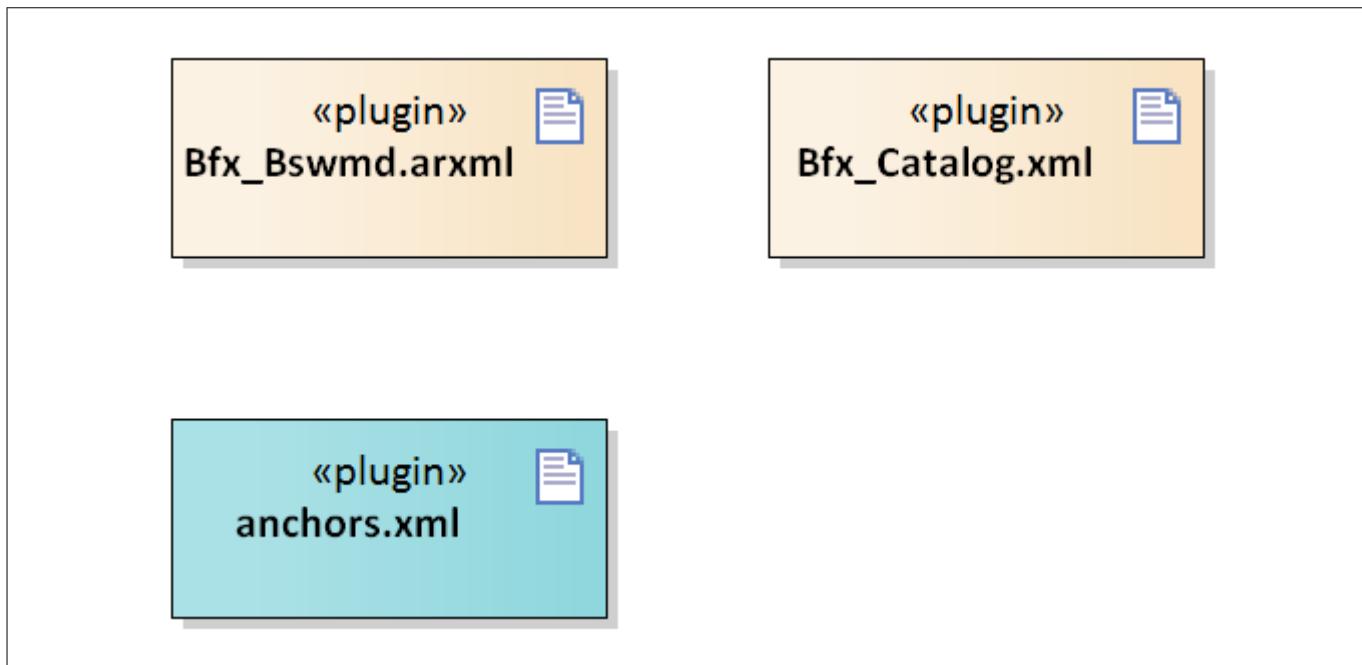
Figure 80 C file structure

Table 183 C file structure

File name	Description
Bfx.c	File (Static) containing implementation of all functions of the BFX library
Bfx.h	Header file (Static) defining prototypes of all APIs of the BFX library exposed to the upper layer
Bfx_MemMap.h	File (Static) containing the memory section definitions used by the BFX library
Compiler.h	Provides abstraction from compiler-specific keywords
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

4.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the BFX library.

BFX driver

Figure 81 **Code generator plugin files**
Table 184 **Code generator plugin files**

File name	Description
Bfx_Bswmd.arxml	AUTOSAR format module description file for the BFX library
Bfx_Catalog.xml	AUTOSAR format catalog file as per catalog_V3_0_0.ml.xsd for the BFX library
anchors.xml	Tresos anchors support file for the BFX library

4.1.4 **Integration hints**

This section describes the key points that an integrator or user of the BFX library must consider.

The BFX library does not require initialization phase to provide its intended functionality as the library does not have any associated hardware IP, SFRs or global variables, which need to be initialized. Shutdown phase is also not required for the BFX library as there is no initialization phase. The BFX library does not depend on any other module for its functionality.

The APIs of the BFX library may be invoked from several CPU cores simultaneously. However, the access to the shared resources, passed as API parameters, must be serialized.

4.1.4.1 **Integration with AUTOSAR stack**

This section lists the modules, which are not part of the MCAL, but are required to integrate the BFX library.

- **EcuM**

The EcuM module is not required for integrating the BFX library.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Bfx_MemMap.h` file.

BFX driver

The `Bfx_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are relocated to the correct memory region. A sample implementation listing the memory-section macros is shown as follows:

```

/* Code Section */
/*
To be used for mapping code to application block, boot block, external flash
etc.

{codePeriod} is the typical period time value and unit of the
ExecutableEntitys in this MemorySection.

The name part _{codePeriod} is optional.

Units are:
- US microsecond
- MS millisecond
- S second

For example 100US, 400US, 1MS, 5MS, 10MS, 20MS, 100MS, 1S
*/
#ifndef BFX_START_SEC_CODE_ASIL_B_GLOBAL
/* User Pragma for PF[x] */
#endif
#ifndef BFX_STOP_SEC_CODE_ASIL_B_GLOBAL
#ifndef MEMMAP_ERROR
#endif
#endif
#ifndef MEMMAP_ERROR
#error "BFX_MemMap.h, wrong pragma command"
#endif

```

- **DET**

The DET module is not required for integrating the BFX library.

- **DEM**

The DEM module is not required for integrating the BFX library.

- **SchM**

The SchM is not required for integrating the BFX library.

- **Safety error**

The BFX library does not report any safety errors.

- **Notification and callbacks**

The BFX library does not provide any notifications or callbacks.

- **OS**

The OS is not required for integrating the BFX library.

4.1.4.2 Multicore and Resource Manager

The BFX library supports execution of its APIs simultaneously from all CPU cores as long as the access to the shared resources, passed as parameters to the BFX APIs, is serialized. The following are the key points to be considered with respect to multicore in the BFX library:

BFX driver

- The BFX library does not access any SFRs or any shared resource, except in case where a shared resource is passed as parameter to a BFX API. AoU is provided to the user to serialize the access to such shared resources, which are passed as parameters to the BFX APIs.
- Locating text to correct memory space should be done by the user. All memory sections for BFX library are marked GLOBAL (common to all cores). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of the BFX library is placed under single MemMap section. This MemMap section can be relocated to any PFlash region. It is possible to access the code from all CPU cores.

Data section:

The BFX library does not define any RAM variables. Hence, data section is not required for the BFX library.

Configuration data and constants:

The BFX library does not define any configuration data or constants. Hence, configuration data section is not required for the BFX library.

Note: Relocating code to a distant memory space would impact execution timings

Note: If the driver operates from single core, the code section may be relocated to the PFlash/DSPR of the same CPU core.

4.1.4.3 MCU support

The BFX library does not use any services provided by the MCU driver.

4.1.4.4 Port support

The BFX library does not use any services provided by the PORT driver.

4.1.4.5 DMA support

The BFX library does not use any services provided by the DMA driver.

4.1.4.6 Interrupt connections

The BFX library does not use any interrupt source.

BFX driver**4.1.4.7 Example usage**

The BFX is a library module. All the BFX APIs can be called independently of each other; therefore, there is no example usage for the BFX library.

4.1.5 Key architectural considerations

There are no key architectural considerations for the BFX library.

BFX driver

4.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Valid pointer as parameter**

The user shall ensure that a valid pointer is passed to the BFX APIs wherever applicable because the driver does not have any mechanism to know if the passed address is valid or to report invalid address.

[cover parentID BFX={360E5AAD-9083-4e2d-BD5B-10DA92664475}]

- **Serialized access to shared resource**

The user shall implement an appropriate mechanism to serialize the access to the shared resources, which are passed as parameter to the BFX APIs by using SchM functions or spinlocks.

[cover parentID BFX={5A3CAADA-6377-43e1-8AE5-C5F043BC9CE6}]

- **Valid permission level**

The user shall not pass any SFR, for which the user application does not have appropriate access rights, as a parameter of any BFX API.

[cover parentID BFX={CBA42528-D0E6-400c-92F5-F1F8DE36D4A1}]

- **Parameter range check**

The user shall ensure all parameters are within the specified valid range as the input range checks are not performed by the BFX APIs.

[cover parentID BFX={FCE75C8A-8252-4996-87B2-E66CE25EEEC7}]

- **Status assumption**

The user shall note that if the value of the Status parameter, for any SetBits API, is not 0, the Status parameter value is treated as 1.

[cover parentID BFX={3BD8FAE0-AB19-414d-B079-8918BD47E8F6}]

BFX driver

4.3 Reference information

4.3.1 Configuration interfaces

The BFX library does not support any configuration options that may affect the functional behavior of the routines.

4.3.2 Functions - Type definitions

The BFX library does not define any data types.

4.3.3 Functions - APIs

This section provides details of all the APIs of the BFX library.

4.3.3.1 Bfx_SetBit_u8u8

Table 185 Specification for Bfx_SetBit_u8u8 API

Syntax	<pre>void Bfx_SetBit_u8u8 (uint8 * const Data, const uint8 BitPn)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitPn	Bit position (Valid range: 0 to 7)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_SetBit_u8u8 function sets the logical status of the bit at BitPn bit position of the Data parameter to 1.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	

BFX driver**Table 185 Specification for `Bfx_SetBit_u8u8` API (continued)**

User hints	The API is used for modifying a bit of the 8-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be modified, is 0 to 7.
-------------------	---

4.3.3.2 `Bfx_SetBit_u16u8`**Table 186 Specification for `Bfx_SetBit_u16u8` API**

Syntax	<pre>void Bfx_SetBit_u16u8 (uint16 * const Data, const uint8 BitPn)</pre>	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitPn	Bit position (Valid range: 0 to 15)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_SetBit_u16u8 function sets the logical status of the bit at BitPn bit position of the Data parameter to 1.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying a bit of the 16-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be modified, is 0 to 15.	

4.3.3.3 `Bfx_SetBit_u32u8`**Table 187 Specification for `Bfx_SetBit_u32u8` API**

Syntax	<pre>void Bfx_SetBit_u32u8 (</pre>
---------------	------------------------------------

BFX driver**Table 187 Specification for `Bfx_SetBit_u32u8` API (continued)**

	<pre> uint32 * const Data, const uint8 BitPn) </pre>	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitPn	Bit position (Valid range: 0 to 31)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_SetBit_u32u8 function sets the logical status of the bit at BitPn bit position of the Data parameter to 1.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying a bit of the 32-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be modified, is 0 to 31.	

4.3.3.4 `Bfx_ClrBit_u8u8`**Table 188 Specification for `Bfx_ClrBit_u8u8` API**

Syntax	<pre> void Bfx_ClrBit_u8u8 (uint8 * const Data, const uint8 BitPn) </pre>
Service ID	0x06
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API

BFX driver**Table 188 Specification for `Bfx_ClrBit_u8u8` API (continued)**

Parameters (in)	BitPn	Bit position (Valid range: 0 to 7)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ClrBit_u8u8 function clears the logical status of the bit at BitPn bit position of the Data parameter to 0.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying a bit of the 8-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be modified, is 0 to 7.	

4.3.3.5 Bfx_ClrBit_u16u8**Table 189 Specification for `Bfx_ClrBit_u16u8` API**

Syntax	<pre>void Bfx_ClrBit_u16u8 (uint16 * const Data, const uint8 BitPn)</pre>	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitPn	Bit position (Valid range: 0 to 15)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-

BFX driver**Table 189 Specification for `Bfx_ClrBit_u16u8` API (continued)**

Description	The Bfx_ClrBit_u16u8 function clears the logical status of the bit at BitPn bit position of the Data parameter to 0.
Source	AUTOSAR
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	The API is used for modifying a bit of the 16-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be modified, is 0 to 15.

4.3.3.6 `Bfx_ClrBit_u32u8`**Table 190 Specification for `Bfx_ClrBit_u32u8` API**

Syntax	<pre>void Bfx_ClrBit_u32u8 (uint32 * const Data, const uint8 BitPn)</pre>	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitPn	Bit position (Valid range: 0 to 31)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ClrBit_u32u8 function clears the logical status of the bit at BitPn bit position of the Data parameter to 0.	
Source	AUTOSAR	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

BFX driver**Table 190 Specification for `Bfx_ClrBit_u32u8` API (continued)**

Configuration dependencies	-
User hints	The API is used for modifying a bit of the 32-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be modified, is 0 to 31.

4.3.3.7 `Bfx_GetBit_u8u8_u8`**Table 191 Specification for `Bfx_GetBit_u8u8_u8` API**

Syntax	<pre>boolean Bfx_GetBit_u8u8_u8 (const uint8 Data, const uint8 BitPn)</pre>	
Service ID	0x0a	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data BitPn	Input data Bit position (Valid range: 0 to 7)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Status as per the extracted bit TRUE : Extracted bit is 1 FALSE : Extracted bit is 0
Description	The Bfx_GetBit_u8u8_u8 function returns TRUE when the logical status of the bit at BitPn bit position of the Data input parameter is 1, otherwise the function returns FALSE.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for extracting a bit from the 8-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be extracted, is 0 to 7.	

BFX driver**4.3.3.8 Bfx_GetBit_u16u8_u8****Table 192 Specification for Bfx_GetBit_u16u8_u8 API**

Syntax	<pre>boolean Bfx_GetBit_u16u8_u8 (const uint16 Data, const uint8 BitPn)</pre>	
Service ID	0x0b	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data BitPn	Input data Bit position (Valid range: 0 to 15)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Status as per the extracted bit TRUE : Extracted bit is 1 FALSE : Extracted bit is 0
Description	The Bfx_GetBit_u16u8_u8 function returns TRUE when the logical status of the bit at BitPn bit position of the Data input parameter is 1, otherwise the function returns FALSE.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for extracting a bit from the 16-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be extracted, is 0 to 15.	

4.3.3.9 Bfx_GetBit_u32u8_u8**Table 193 Specification for Bfx_GetBit_u32u8_u8 API**

Syntax	<pre>boolean Bfx_GetBit_u32u8_u8 (const uint32 Data, const uint8 BitPn)</pre>	
---------------	---	--

BFX driver**Table 193 Specification for `Bfx_GetBit_u32u8_u8` API (continued)**

Service ID	0x0c	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data BitPn	Input data Bit position (Valid range: 0 to 31)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Status as per the extracted bit TRUE : Extracted bit is 1 FALSE : Extracted bit is 0
Description	The Bfx_GetBit_u32u8_u8 function returns TRUE when the logical status of the bit at BitPn bit position of the Data input parameter is 1, otherwise the function returns FALSE.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for extracting a bit of the 32-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be extracted, is 0 to 31.	

4.3.3.10 `Bfx_SetBits_u8u8u8u8`**Table 194 Specification for `Bfx_SetBits_u8u8u8u8` API**

Syntax	<pre>void Bfx_SetBits_u8u8u8u8 (uint8 * const Data, const uint8 BitStartPn, const uint8 BitLn, const uint8 Status)</pre>
Service ID	0x20
Sync/Async	Synchronous
ASIL Level	B

BFX driver**Table 194 Specification for `Bfx_SetBits_u8u8u8u8` API (continued)**

Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitStartPn	Start bit position (Valid range: 0 to 7)
	BitLn	Bit field length (Valid range: 1 to (8 - BitStartPn))
	Status	Status value to be set
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_SetBits_u8u8u8u8</code> function clears the logical status of the bits of the Data parameter starting from BitStartPn bit position for BitLn number of bits to 0 when the value of Status parameter is zero. Otherwise, for non-zero value of Status parameter, the function sets the logical status of the bits of the Data parameter starting from BitStartPn bit position for BitLn number of bits to 1.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying some bits of the 8-bit Data parameter. Hence, the valid ranges for input parameters are as follows: 1. The valid range for the BitStartPn parameter, which indicates the start position of the bits to be modified, is 0 to 7. 2. The valid range for the BitLn parameter, which indicates the number of bits to be modified, is 1 to (8 - BitStartPn).	

4.3.3.11 `Bfx_SetBits_u16u8u8u8`**Table 195 Specification for `Bfx_SetBits_u16u8u8u8` API**

Syntax	<pre>void Bfx_SetBits_u16u8u8u8 (uint16 * const Data, const uint8 BitStartPn, const uint8 BitLn, const uint8 Status)</pre>
Service ID	0x21
Sync/Async	Synchronous

BFX driver**Table 195 Specification for `Bfx_SetBits_u16u8u8u8` API (continued)**

ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitStartPn BitLn Status	Start bit position (Valid range: 0 to 15) Bit field length (Valid range: 1 to (16 - BitStartPn)) Status value to be set
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_SetBits_u16u8u8u8</code> function clears the logical status of the bits of the Data parameter starting from BitStartPn bit position for BitLn number of bits to 0 when the value of Status parameter is zero. Otherwise, for non-zero value of Status parameter, the function sets the logical status of the bits of the Data parameter starting from BitStartPn bit position for BitLn number of bits to 1.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying some bits of the 16-bit Data parameter. Hence, the valid ranges for input parameters are as follows: 1. The valid range for the BitStartPn parameter, which indicates the start position of the bits to be modified, is 0 to 15. 2. The valid range for the BitLn parameter, which indicates the number of bits to be modified, is 1 to (16 - BitStartPn).	

4.3.3.12 `Bfx_SetBits_u32u8u8u8`**Table 196 Specification for `Bfx_SetBits_u32u8u8u8` API**

Syntax	void <code>Bfx_SetBits_u32u8u8u8</code> (<code>uint32 * const Data,</code> <code>const uint8 BitStartPn,</code> <code>const uint8 BitLn,</code> <code>const uint8 Status</code>)
Service ID	0x22

BFX driver**Table 196 Specification for `Bfx_SetBits_u32u8u8u8` API (continued)**

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitStartPn BitLn Status	Start bit position (Valid range: 0 to 31) Bit field length (Valid range: 1 to (32 - BitStartPn)) Status value to be set
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_SetBits_u32u8u8u8 function clears the logical status of the bits of the Data parameter starting from BitStartPn bit position for BitLn number of bits to 0 when the value of Status parameter is zero. Otherwise, for non-zero value of Status parameter, the function sets the logical status of the bits of the Data parameter starting from BitStartPn bit position for BitLn number of bits to 1.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying some bits of the 32-bit Data parameter. Hence, the valid ranges for input parameters are as follows: 1. The valid range for the BitStartPn parameter, which indicates the start position of the bits to be modified, is 0 to 31. 2. The valid range for the BitLn parameter, which indicates the number of bits to be modified, is 1 to (32 - BitStartPn).	

4.3.3.13 `Bfx_GetBits_u8u8u8_u8`**Table 197 Specification for `Bfx_GetBits_u8u8u8_u8` API**

Syntax	<pre>uint8 Bfx_GetBits_u8u8u8_u8 (const uint8 Data, const uint8 BitStartPn, const uint8 BitLn)</pre>
Service ID	0x26

BFX driver**Table 197 Specification for `Bfx_GetBits_u8u8u8_u8` API (continued)**

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data BitStartPn BitLn	Input data Start bit position (Valid range: 0 to 7) Bit field length (Valid range: 1 to (8 - BitStartPn))
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint8	Bits extracted from the input parameter
Description	The Bfx_GetBits_u8u8u8_u8 function returns the bits of the Data input parameter starting from BitStartPn bit position for BitLn number of bits.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for extracting some bits of the 8-bit Data parameter. Hence, the valid ranges for input parameters are as follows: 1. The valid range for the BitStartPn parameter, which indicates the start position of the bits to be extracted, is 0 to 7. 2. The valid range for the BitLn parameter, which indicates the number of bits to be extracted, is 1 to (8 - BitStartPn).	

4.3.3.14 `Bfx_GetBits_u16u8u8_u16`**Table 198 Specification for `Bfx_GetBits_u16u8u8_u16` API**

Syntax	<pre>uint16 Bfx_GetBits_u16u8u8_u16 (const uint16 Data, const uint8 BitStartPn, const uint8 BitLn)</pre>
Service ID	0x27
Sync/Async	Synchronous
ASIL Level	B

BFX driver**Table 198 Specification for `Bfx_GetBits_u16u8u8_u16` API (continued)**

Re-entrancy	Reentrant	
Parameters (in)	Data BitStartPn BitLn	Input data Start bit position (Valid range: 0 to 15) Bit field length (Valid range: 1 to (16 - BitStartPn))
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint16	Bits extracted from the input parameter
Description	The Bfx_GetBits_u16u8u8_u16 function returns the bits of the Data input parameter starting from BitStartPn bit position for BitLn number of bits.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for extracting some bits of the 16-bit Data parameter. Hence, the valid ranges for input parameters are as follows: 1. The valid range for the BitStartPn parameter, which indicates the start position of the bits to be extracted, is 0 to 15. 2. The valid range for the BitLn parameter, which indicates the number of bits to be extracted, is 1 to (16 - BitStartPn).	

4.3.3.15 `Bfx_GetBits_u32u8u8_u32`**Table 199 Specification for `Bfx_GetBits_u32u8u8_u32` API**

Syntax	uint32 Bfx_GetBits_u32u8u8_u32 (const uint32 Data, const uint8 BitStartPn, const uint8 BitLn)
Service ID	0x28
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant

BFX driver**Table 199 Specification for `Bfx_GetBits_u32u8u8_u32` API (continued)**

Parameters (in)	Data BitStartPn BitLn	Input data Start bit position (Valid range: 0 to 31) Bit field length (Valid range: 1 to (32 - BitStartPn))
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Bits extracted from the input parameter
Description	The Bfx_GetBits_u32u8u8_u32 function returns the bits of the Data input parameter starting from BitStartPn bit position for BitLn number of bits.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying some bits of the 32-bit Data parameter. Hence, the valid ranges for input parameters are as follows: 1. The valid range for the BitStartPn parameter, which indicates the start position of the bits to be extracted, is 0 to 31. 2. The valid range for the BitLn parameter, which indicates the number of bits to be extracted, is 1 to (32 - BitStartPn).	

4.3.3.16 `Bfx_SetBitMask_u8u8`**Table 200 Specification for `Bfx_SetBitMask_u8u8` API**

Syntax	void Bfx_SetBitMask_u8u8 (uint8 * const Data, const uint8 Mask)
Service ID	0x2a
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API
Parameters (in)	Mask
	Mask value

BFX driver**Table 200 Specification for `Bfx_SetBitMask_u8u8` API (continued)**

Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_SetBitMask_u8u8</code> function sets the logical status of the bits of the Data parameter to 1, for all the bit positions for which the logical status of bit in the Mask parameter is set to 1. The remaining bits of the Data parameter will retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

4.3.3.17 `Bfx_SetBitMask_u16u16`**Table 201 Specification for `Bfx_SetBitMask_u16u16` API**

Syntax	<pre>void Bfx_SetBitMask_u16u16 (uint16 * const Data, const uint16 Mask)</pre>	
Service ID	0x2b	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Mask	Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_SetBitMask_u16u16</code> function sets the logical status of the bits of the Data parameter to 1, for all the bit positions for which the logical status of bit in the Mask parameter is set to 1. The remaining bits of the Data parameter will retain their original values.	

BFX driver**Table 201 Specification for `Bfx_SetBitMask_u16u16` API (continued)**

Source	AUTOSAR
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	The API is implemented as inline function to achieve performance optimization.

4.3.3.18 `Bfx_SetBitMask_u32u32`**Table 202 Specification for `Bfx_SetBitMask_u32u32` API**

Syntax	<pre>void Bfx_SetBitMask_u32u32 (uint32 * const Data, const uint32 Mask)</pre>	
Service ID	0x2c	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Mask	Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_SetBitMask_u32u32</code> function sets the logical status of the bits of the <code>Data</code> parameter to 1, for all the bit positions for which the logical status of bit in the <code>Mask</code> parameter is set to 1. The remaining bits of the <code>Data</code> parameter will retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	

BFX driver**Table 202 Specification for `Bfx_SetBitMask_u32u32` API (continued)**

User hints	The API is implemented as inline function to achieve performance optimization.
-------------------	--

4.3.3.19 Bfx_ClrBitMask_u8u8**Table 203 Specification for `Bfx_ClrBitMask_u8u8` API**

Syntax	<pre>void Bfx_ClrBitMask_u8u8 (uint8 * const Data, const uint8 Mask)</pre>	
Service ID	0x30	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Mask	Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ClrBitMask_u8u8 function clears the logical status of the bits of the Data parameter to 0, for all the bit positions for which the logical status of bit in the Mask parameter is set to 1. The remaining bits of Data parameter will retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

4.3.3.20 Bfx_ClrBitMask_u16u16**Table 204 Specification for `Bfx_ClrBitMask_u16u16` API**

Syntax	<pre>void Bfx_ClrBitMask_u16u16 (uint16 * const Data,</pre>
---------------	--

BFX driver**Table 204 Specification for `Bfx_ClrBitMask_u16u16` API (continued)**

	const uint16 Mask)	
Service ID	0x31	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Mask	Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_ClrBitMask_u16u16</code> function clears the logical status of the bits of the Data parameter to 0, for all the bit positions for which the logical status of bit in the Mask parameter is set to 1. The remaining bits of Data parameter will retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

4.3.3.21 `Bfx_ClrBitMask_u32u32`**Table 205 Specification for `Bfx_ClrBitMask_u32u32` API**

Syntax	void <code>Bfx_ClrBitMask_u32u32</code> (<code>uint32 * const Data,</code> <code>const uint32 Mask</code>)
Service ID	0x32
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API

BFX driver**Table 205 Specification for `Bfx_ClrBitMask_u32u32` API (continued)**

Parameters (in)	Mask	Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_ClrBitMask_u32u32</code> function clears the logical status of the bits of the Data parameter to 0, for all the bit positions for which the logical status of bit in the Mask parameter is set to 1. The remaining bits of Data parameter will retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

4.3.3.22 `Bfx_TstBitMask_u8u8_u8`**Table 206 Specification for `Bfx_TstBitMask_u8u8_u8` API**

Syntax	<pre>boolean Bfx_TstBitMask_u8u8_u8 (const uint8 Data, const uint8 Mask)</pre>	
Service ID	0x36	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data	Input data
	Mask	Mask value
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Test result TRUE : All bits defined in mask are set in input parameter

BFX driver**Table 206 Specification for `Bfx_TstBitMask_u8u8_u8` API (continued)**

	FALSE: At least one bit defined in mask is not set in input parameter
Description	The Bfx_TstBitMask_u8u8_u8 function returns TRUE when the logical status of all the bits defined in the Mask parameter are also set at the same bit position in the Data input parameter, otherwise the function returns FALSE.
Source	AUTOSAR
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

4.3.3.23 `Bfx_TstBitMask_u16u16_u8`**Table 207 Specification for `Bfx_TstBitMask_u16u16_u8` API**

Syntax	boolean Bfx_TstBitMask_u16u16_u8 (const uint16 Data, const uint16 Mask)	
Service ID	0x37	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data Mask	Input data Mask value
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Test result TRUE : All bits defined in mask are set in input parameter FALSE: At least one bit defined in mask is not set in input parameter
Description	The Bfx_TstBitMask_u16u16_u8 function returns TRUE when the logical status of all the bits defined in the Mask parameter are also set at the same bit position in the Data input parameter, otherwise the function returns FALSE.	

BFX driver**Table 207 Specification for `Bfx_TstBitMask_u16u16_u8` API (continued)**

Source	AUTOSAR
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

4.3.3.24 `Bfx_TstBitMask_u32u32_u8`**Table 208 Specification for `Bfx_TstBitMask_u32u32_u8` API**

Syntax	<pre>boolean Bfx_TstBitMask_u32u32_u8 (const uint32 Data, const uint32 Mask)</pre>	
Service ID	0x38	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data Mask	Input data Mask value
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Test result TRUE : All bits defined in mask are set in input parameter FALSE: At least one bit defined in mask is not set in input parameter
Description	The <code>Bfx_TstBitMask_u32u32_u8</code> function returns TRUE when the logical status of all the bits defined in the Mask parameter are also set at the same bit position in the Data input parameter, otherwise the function returns FALSE.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None	

BFX driver**Table 208 Specification for `Bfx_TstBitMask_u32u32_u8` API (continued)**

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

4.3.3.25 `Bfx_TstBitLnMask_u8u8_u8`**Table 209 Specification for `Bfx_TstBitLnMask_u8u8_u8` API**

Syntax	boolean Bfx_TstBitLnMask_u8u8_u8 (const uint8 Data, const uint8 Mask)	
Service ID	0x3a	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data Mask	Input data Mask value
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Test result TRUE : At least one bit defined in mask is set in input parameter FALSE: No bit defined in mask is set in input parameter
Description	The Bfx_TstBitLnMask_u8u8_u8 function returns TRUE when the logical status of at least one bit defined in the Mask parameter is also set at the same bit position in the Data input parameter, otherwise the function returns FALSE.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

BFX driver**4.3.3.26 Bfx_TstBitLnMask_u16u16_u8****Table 210 Specification for Bfx_TstBitLnMask_u16u16_u8 API**

Syntax	<pre>boolean Bfx_TstBitLnMask_u16u16_u8 (const uint16 Data, const uint16 Mask)</pre>	
Service ID	0x3b	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data Mask	Input data Mask value
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Test result TRUE : At least one bit defined in mask is set in input parameter FALSE: No bit defined in mask is set in input parameter
Description	The Bfx_TstBitLnMask_u16u16_u8 function returns TRUE when the logical status of at least one bit defined in the Mask parameter is also set at the same bit position in the Data input parameter, otherwise the function returns FALSE.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

4.3.3.27 Bfx_TstBitLnMask_u32u32_u8**Table 211 Specification for Bfx_TstBitLnMask_u32u32_u8 API**

Syntax	<pre>boolean Bfx_TstBitLnMask_u32u32_u8 (const uint32 Data, const uint32 Mask)</pre>	
---------------	--	--

BFX driver**Table 211 Specification for Bfx_TstBitLnMask_u32u32_u8 API (continued)**

Service ID	0x3c	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data Mask	Input data Mask value
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Test result TRUE : At least one bit defined in mask is set in input parameter FALSE: No bit defined in mask is set in input parameter
Description	The Bfx_TstBitLnMask_u32u32_u8 function returns TRUE when the logical status of at least one bit defined in the Mask parameter is also set at the same bit position in the Data input parameter, otherwise the function returns FALSE.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

4.3.3.28 Bfx_TstParityEven_u8_u8**Table 212 Specification for Bfx_TstParityEven_u8_u8 API**

Syntax	boolean Bfx_TstParityEven_u8_u8 (const uint8 Data)	
Service ID	0x40	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data	Input Data

BFX driver**Table 212 Specification for `Bfx_TstParityEven_u8_u8` API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Test result TRUE : Parity of input parameter is even FALSE: Parity of input parameter is odd
Description	The <code>Bfx_TstParityEven_u8_u8</code> function returns TRUE when the number of bits whose logical status is set to 1 in the Data input parameter is even, otherwise the function returns FALSE.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

4.3.3.29 `Bfx_TstParityEven_u16_u8`**Table 213 Specification for `Bfx_TstParityEven_u16_u8` API**

Syntax	<pre>boolean Bfx_TstParityEven_u16_u8 (const uint16 Data)</pre>	
Service ID	0x41	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data	Input Data
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Test result TRUE : Parity of input parameter is even FALSE: Parity of input parameter is odd

BFX driver**Table 213 Specification for `Bfx_TstParityEven_u16_u8` API (continued)**

Description	The <code>Bfx_TstParityEven_u16_u8</code> function returns TRUE when the number of bits whose logical status is set to 1 in the Data input parameter is even, otherwise the function returns FALSE.
Source	AUTOSAR
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

4.3.3.30 `Bfx_TstParityEven_u32_u8`**Table 214 Specification for `Bfx_TstParityEven_u32_u8` API**

Syntax	boolean <code>Bfx_TstParityEven_u32_u8</code> (const uint32 Data)	
Service ID	0x42	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Data	Input Data
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	boolean	Test result TRUE : Parity of input parameter is even FALSE: Parity of input parameter is odd
Description	The <code>Bfx_TstParityEven_u32_u8</code> function returns TRUE when the number of bits whose logical status is set to 1 in the Data input parameter is even, otherwise the function returns FALSE.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None	

BFX driver**Table 214 Specification for `Bfx_TstParityEven_u32_u8` API (continued)**

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

4.3.3.31 Bfx_ToggleBits_u8**Table 215 Specification for `Bfx_ToggleBits_u8` API**

Syntax	void Bfx_ToggleBits_u8 (uint8 * const Data)	
Service ID	0x46	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ToggleBits_u8 function toggles all the bits of the Data parameter (1's complement of the Data parameter).	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

BFX driver**4.3.3.32 Bfx_ToggleBits_u16****Table 216 Specification for Bfx_ToggleBits_u16 API**

Syntax	<pre>void Bfx_ToggleBits_u16 (uint16 * const Data)</pre>	
Service ID	0x47	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ToggleBits_u16 function toggles all the bits of the Data parameter (1's complement of the Data parameter).	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

4.3.3.33 Bfx_ToggleBits_u32**Table 217 Specification for Bfx_ToggleBits_u32 API**

Syntax	<pre>void Bfx_ToggleBits_u32 (uint32 * const Data)</pre>	
Service ID	0x48	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	

BFX driver**Table 217 Specification for `Bfx_ToggleBits_u32` API (continued)**

Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_ToggleBits_u32</code> function toggles all the bits of the Data parameter (1's complement of the Data parameter).	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

4.3.3.34 `Bfx_ToggleBitMask_u8u8`**Table 218 Specification for `Bfx_ToggleBitMask_u8u8` API**

Syntax	<pre>void Bfx_ToggleBitMask_u8u8 (uint8 * const Data, const uint8 Mask)</pre>	
Service ID	0x4a	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Mask	Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-

BFX driver**Table 218 Specification for `Bfx_ToggleBitMask_u8u8` API (continued)**

Description	The <code>Bfx_ToggleBitMask_u8u8</code> function toggles the logical status of the bits of the Data parameter, for all the bit positions for which the logical status of bit in the Mask parameter is set to 1. The remaining bits of the Data parameter will retain their original values.
Source	AUTOSAR
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	The API is implemented as inline function to achieve performance optimization.

4.3.3.35 `Bfx_ToggleBitMask_u16u16`**Table 219 Specification for `Bfx_ToggleBitMask_u16u16` API**

Syntax	<pre>void Bfx_ToggleBitMask_u16u16 (uint16 * const Data, const uint16 Mask)</pre>	
Service ID	0x4b	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Mask	Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_ToggleBitMask_u16u16</code> function toggles the logical status of the bits of the Data parameter, for all the bit positions for which the logical status of bit in the Mask parameter is set to 1. The remaining bits of the Data parameter will retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None	

BFX driver**Table 219 Specification for `Bfx_ToggleBitMask_u16u16` API (continued)**

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	The API is implemented as inline function to achieve performance optimization.

4.3.3.36 `Bfx_ToggleBitMask_u32u32`**Table 220 Specification for `Bfx_ToggleBitMask_u32u32` API**

Syntax	<pre>void Bfx_ToggleBitMask_u32u32 (uint32 * const Data, const uint32 Mask)</pre>	
Service ID	0x4c	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Mask	Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ToggleBitMask_u32u32 function toggles the logical status of the bits of the Data parameter, for all the bit positions for which the logical status of bit in the Mask parameter is set to 1. The remaining bits of the Data parameter will retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

BFX driver**4.3.3.37 Bfx_ShiftBitRt_u8u8****Table 221 Specification for Bfx_ShiftBitRt_u8u8 API**

Syntax	<pre>void Bfx_ShiftBitRt_u8u8 (uint8 * const Data, const uint8 ShiftCnt)</pre>	
Service ID	0x50	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Shift right count (Valid range: 0 to 7)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ShiftBitRt_u8u8 function shifts the bits of the Data parameter to the right by ShiftCnt count. The most significant bit (left-most bit) is replaced by a 0 bit and the least significant bit (right-most bit) is discarded for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for shifting bits of the 8-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be shifted, is 0 to 7.	

4.3.3.38 Bfx_ShiftBitRt_u16u8**Table 222 Specification for Bfx_ShiftBitRt_u16u8 API**

Syntax	<pre>void Bfx_ShiftBitRt_u16u8 (uint16 * const Data, const uint8 ShiftCnt)</pre>
---------------	--

BFX driver**Table 222 Specification for `Bfx_ShiftBitRt_u16u8` API (continued)**

Service ID	0x51	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Shift right count (Valid range: 0 to 15)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ShiftBitRt_u16u8 function shifts the bits of the Data parameter to the right by ShiftCnt count. The most significant bit (left-most bit) is replaced by a 0 bit and the least significant bit (right-most bit) is discarded for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for shifting bits of the 16-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be shifted, is 0 to 15.	

4.3.3.39 Bfx_ShiftBitRt_u32u8**Table 223 Specification for `Bfx_ShiftBitRt_u32u8` API**

Syntax	<pre>void Bfx_ShiftBitRt_u32u8 (uint32 * const Data, const uint8 ShiftCnt)</pre>
Service ID	0x52
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API

BFX driver**Table 223 Specification for `Bfx_ShiftBitRt_u32u8` API (continued)**

Parameters (in)	ShiftCnt	Shift right count (Valid range: 0 to 31)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ShiftBitRt_u32u8 function shifts the bits of the Data parameter to the right by ShiftCnt count. The most significant bit (left-most bit) is replaced by a 0 bit and the least significant bit (right-most bit) is discarded for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for shifting bits of the 32-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be shifted, is 0 to 31.	

4.3.3.40 `Bfx_ShiftBitLt_u8u8`**Table 224 Specification for `Bfx_ShiftBitLt_u8u8` API**

Syntax	<pre>void Bfx_ShiftBitLt_u8u8 (uint8 * const Data, const uint8 ShiftCnt)</pre>	
Service ID	0x56	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Shift left count (Valid range: 0 to 7)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified

BFX driver**Table 224 Specification for `Bfx_ShiftBitLt_u8u8` API (continued)**

Return	void	-
Description	The Bfx_ShiftBitLt_u8u8 function shifts the bits of the Data parameter to the left by ShiftCnt count. The least significant bit (right-most bit) is replaced by a 0 bit and the most significant bit (left-most bit) is discarded for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for shifting bits of the 8-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be shifted, is 0 to 7.	

4.3.3.41 `Bfx_ShiftBitLt_u16u8`**Table 225 Specification for `Bfx_ShiftBitLt_u16u8` API**

Syntax	<pre>void Bfx_ShiftBitLt_u16u8 (uint16 * const Data, const uint8 ShiftCnt)</pre>	
Service ID	0x57	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Shift left count (Valid range: 0 to 15)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ShiftBitLt_u16u8 function shifts the bits of the Data parameter to the left by ShiftCnt count. The least significant bit (right-most bit) is replaced by a 0 bit and the most significant bit (left-most bit) is discarded for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None	

BFX driver**Table 225 Specification for `Bfx_ShiftBitLt_u16u8` API (continued)**

	<p>Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	<ol style="list-style-type: none"> 1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for shifting bits of the 16-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be shifted, is 0 to 15.

4.3.3.42 `Bfx_ShiftBitLt_u32u8`**Table 226 Specification for `Bfx_ShiftBitLt_u32u8` API**

Syntax	<pre>void Bfx_ShiftBitLt_u32u8 (uint32 * const Data, const uint8 ShiftCnt)</pre>	
Service ID	0x58	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Shift left count (Valid range: 0 to 31)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_ShiftBitLt_u32u8 function shifts the bits of the Data parameter to the left by ShiftCnt count. The least significant bit (right-most bit) is replaced by a 0 bit and the most significant bit (left-most bit) is discarded for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	<p>DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	

BFX driver**Table 226 Specification for `Bfx_ShiftBitLt_u32u8` API (continued)**

User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for shifting bits of the 32-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be shifted, is 0 to 31.
-------------------	---

4.3.3.43 Bfx_RotBitRt_u8u8**Table 227 Specification for `Bfx_RotBitRt_u8u8` API**

Syntax	<pre>void Bfx_RotBitRt_u8u8 (uint8 * const Data, const uint8 ShiftCnt)</pre>	
Service ID	0x5a	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Rotate right count (Valid range: 0 to 7)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_RotBitRt_u8u8 function rotates the bits of the Data parameter to the right by ShiftCnt count. The least significant bit (right-most bit) is rotated to the most significant bit (left-most bit) location for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for rotating bits of the 8-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be rotated, is 0 to 7.	

BFX driver**4.3.3.44 Bfx_RotBitRt_u16u8****Table 228 Specification for Bfx_RotBitRt_u16u8 API**

Syntax	<pre>void Bfx_RotBitRt_u16u8 (uint16 * const Data, const uint8 ShiftCnt)</pre>	
Service ID	0x5b	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Rotate right count (Valid range: 0 to 15)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_RotBitRt_u16u8 function rotates the bits of the Data parameter to the right by ShiftCnt count. The least significant bit (right-most bit) is rotated to the most significant bit (left-most bit) location for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for rotating bits of the 16-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be rotated, is 0 to 15.	

4.3.3.45 Bfx_RotBitRt_u32u8**Table 229 Specification for Bfx_RotBitRt_u32u8 API**

Syntax	<pre>void Bfx_RotBitRt_u32u8 (uint32 * const Data, const uint8 ShiftCnt)</pre>
---------------	--

BFX driver**Table 229 Specification for Bfx_RotBitRt_u32u8 API (continued)**

Service ID	0x5c	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Rotate right count (Valid range: 0 to 31)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_RotBitRt_u32u8 function rotates the bits of the Data parameter to the right by ShiftCnt count. The least significant bit (right-most bit) is rotated to the most significant bit (left-most bit) location for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for rotating bits of the 32-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be rotated, is 0 to 31.	

4.3.3.46 Bfx_RotBitLt_u8u8**Table 230 Specification for Bfx_RotBitLt_u8u8 API**

Syntax	<pre>void Bfx_RotBitLt_u8u8 (uint8 * const Data, const uint8 ShiftCnt)</pre>
Service ID	0x60
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API

BFX driver**Table 230 Specification for Bfx_RotBitLt_u8u8 API (continued)**

Parameters (in)	ShiftCnt	Rotate left count (Valid range: 0 to 7)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_RotBitLt_u8u8 function rotates the bits of the Data parameter to the left by ShiftCnt count. The most significant bit (left-most bit) is rotated to the least significant bit (right-most bit) location for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for rotating bits of the 8-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be rotated, is 0 to 7.	

4.3.3.47 Bfx_RotBitLt_u16u8**Table 231 Specification for Bfx_RotBitLt_u16u8 API**

Syntax	<pre>void Bfx_RotBitLt_u16u8 (uint16 * const Data, const uint8 ShiftCnt)</pre>	
Service ID	0x61	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Rotate left count (Valid range: 0 to 15)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified

BFX driver**Table 231 Specification for Bfx_RotBitLt_u16u8 API (continued)**

Return	void	-
Description	The Bfx_RotBitLt_u16u8 function rotates the bits of the Data parameter to the left by ShiftCnt count. The most significant bit (left-most bit) is rotated to the least significant bit (right-most bit) location for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for rotating bits of the 16-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be rotated, is 0 to 15.	

4.3.3.48 Bfx_RotBitLt_u32u8**Table 232 Specification for Bfx_RotBitLt_u32u8 API**

Syntax	void Bfx_RotBitLt_u32u8 (uint32 * const Data, const uint8 ShiftCnt)	
Service ID	0x62	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	ShiftCnt	Rotate left count (Valid range: 0 to 31)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_RotBitLt_u32u8 function rotates the bits of the Data parameter to the left by ShiftCnt count. The most significant bit (left-most bit) is rotated to the least significant bit (right-most bit) location for every single bit shift cycle.	
Source	AUTOSAR	
Error handling	DET: None	

BFX driver**Table 232 Specification for `Bfx_RotBitLt_u32u8` API (continued)**

	<p>Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	<ol style="list-style-type: none"> 1. The API is implemented as inline function to achieve performance optimization. 2. The API is used for rotating bits of the 32-bit Data parameter. Hence, the valid range for the ShiftCnt parameter, which indicates the count by which the bits are to be rotated, is 0 to 31.

4.3.3.49 `Bfx_CopyBit_u8u8u8u8`**Table 233 Specification for `Bfx_CopyBit_u8u8u8u8` API**

Syntax	<pre>void Bfx_CopyBit_u8u8u8u8 (uint8 * const DestinationData, const uint8 DestinationPosition, const uint8 SourceData, const uint8 SourcePosition)</pre>	
Service ID	0x66	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	DestinationPosition SourceData SourcePosition	Destination bit position (Valid range: 0 to 7) Source data Source bit position (Valid range: 0 to 7)
Parameters (out)	-	-
Parameters (in - out)	DestinationData	Pointer to destination data which is to be modified
Return	void	-
Description	The <code>Bfx_CopyBit_u8u8u8u8</code> function copies a bit at <code>SourcePosition</code> bit position of the <code>SourceData</code> parameter to <code>DestinationPosition</code> bit position of the <code>DestinationData</code> parameter.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

BFX driver**Table 233 Specification for `Bfx_CopyBit_u8u8u8u8` API (continued)**

Configuration dependencies	-
User hints	<p>The API is used for modifying a bit of the 8-bit SourceData parameter. Hence, the valid range for input parameters are as follows:</p> <ol style="list-style-type: none"> 1. The valid range for the SourcePosition parameter, which indicates the position of the bit to be copied, is 0 to 7. 2. The valid range for the DestinationPosition parameter, which indicates the position of the bit to be modified, is 0 to 7.

4.3.3.50 `Bfx_CopyBit_u16u8u16u8`**Table 234 Specification for `Bfx_CopyBit_u16u8u16u8` API**

Syntax	<pre>void Bfx_CopyBit_u16u8u16u8 (uint16 * const DestinationData, const uint8 DestinationPosition, const uint16 SourceData, const uint8 SourcePosition)</pre>	
Service ID	0x67	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	DestinationPosition SourceData SourcePosition	Destination bit position (Valid range: 0 to 15) Source data Source bit position (Valid range: 0 to 15)
Parameters (out)	-	-
Parameters (in - out)	DestinationData	Pointer to destination data which is to be modified
Return	void	-
Description	The Bfx_CopyBit_u16u8u16u8 function copies a bit at SourcePosition bit position of the SourceData parameter to DestinationPosition bit position of the DestinationData parameter.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

BFX driver**Table 234 Specification for `Bfx_CopyBit_u16u8u16u8` API (continued)**

Configuration dependencies	-
User hints	<p>The API is used for modifying a bit of the 16-bit SourceData parameter. Hence, the valid range for input parameters are as follows:</p> <ol style="list-style-type: none"> 1. The valid range for the SourcePosition parameter, which indicates the position of the bit to be copied, is 0 to 15. 2. The valid range for the DestinationPosition parameter, which indicates the position of the bit to be modified, is 0 to 15.

4.3.3.51 `Bfx_CopyBit_u32u8u32u8`**Table 235 Specification for `Bfx_CopyBit_u32u8u32u8` API**

Syntax	<pre>void Bfx_CopyBit_u32u8u32u8 (uint32 * const DestinationData, const uint8 DestinationPosition, const uint32 SourceData, const uint8 SourcePosition)</pre>	
Service ID	0x68	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	DestinationPosition SourceData SourcePosition	Destination bit position (Valid range: 0 to 31) Source data Source bit position (Valid range: 0 to 31)
Parameters (out)	-	-
Parameters (in - out)	DestinationData	Pointer to destination data which is to be modified
Return	void	-
Description	The Bfx_CopyBit_u32u8u32u8 function copies a bit at SourcePosition bit position of the SourceData parameter to DestinationPosition bit position of the DestinationData parameter.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

BFX driver**Table 235 Specification for `Bfx_CopyBit_u32u8u32u8` API (continued)**

Configuration dependencies	-
User hints	<p>The API is used for modifying a bit of the 32-bit SourceData parameter. Hence, the valid range for input parameters are as follows:</p> <ol style="list-style-type: none"> 1. The valid range for the SourcePosition parameter, which indicates the position of the bit to be copied, is 0 to 31. 2. The valid range for the DestinationPosition parameter, which indicates the position of the bit to be modified, is 0 to 31.

4.3.3.52 `Bfx_PutBits_u8u8u8u8`**Table 236 Specification for `Bfx_PutBits_u8u8u8u8` API**

Syntax	<pre>void Bfx_PutBits_u8u8u8u8 (uint8 * const Data, const uint8 BitStartPn, const uint8 BitLn, const uint8 Pattern)</pre>	
Service ID	0x70	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitStartPn BitLn Pattern	Start bit position (Valid range: 0 to 7) Bit field length (Valid range: 1 to (8 - BitStartPn)) Bit pattern to be set
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to destination data which is to be modified
Return	void	-
Description	The Bfx_PutBits_u8u8u8u8 function copies the bit pattern from the Pattern parameter starting from 0 bit position for BitLn number of bits into the Data parameter at the bit positions starting from BitStartPn bit position for BitLn number of bits.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

BFX driver**Table 236 Specification for `Bfx_PutBits_u8u8u8u8` API (continued)**

Configuration dependencies	-
User hints	<p>The API is used for modifying some bits of the 8-bit Data parameter. Hence, the valid ranges for input parameters are as follows:</p> <ol style="list-style-type: none"> 1. The valid range for the BitStartPn parameter, which indicates the start position of the bits to be modified, is 0 to 7. 2. The valid range for the BitLn parameter, which indicates the number of bits to be modified, is 1 to (8 - BitStartPn).

4.3.3.53 `Bfx_PutBits_u16u8u8u16`**Table 237 Specification for `Bfx_PutBits_u16u8u8u16` API**

Syntax	<pre>void Bfx_PutBits_u16u8u8u16 (uint16 * const Data, const uint8 BitStartPn, const uint8 BitLn, const uint16 Pattern)</pre>	
Service ID	0x71	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitStartPn BitLn Pattern	Start bit position (Valid range: 0 to 15) Bit field length (Valid range: 1 to (16 - BitStartPn)) Bit pattern to be set
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to destination data which is to be modified
Return	void	-
Description	The Bfx_PutBits_u16u8u8u16 function copies the bit pattern from the Pattern parameter starting from 0 bit position for BitLn number of bits into the Data parameter at the bit positions starting from BitStartPn bit position for BitLn number of bits.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

BFX driver**Table 237 Specification for Bfx_PutBits_u16u8u8u16 API (continued)**

Configuration dependencies	-
User hints	<p>The API is used for modifying some bits of the 16-bit Data parameter. Hence, the valid ranges for input parameters are as follows:</p> <ol style="list-style-type: none"> 1. The valid range for the BitStartPn parameter, which indicates the start position of the bits to be modified, is 0 to 15. 2. The valid range for the BitLn parameter, which indicates the number of bits to be modified, is 1 to (16 - BitStartPn).

4.3.3.54 Bfx_PutBits_u32u8u8u32**Table 238 Specification for Bfx_PutBits_u32u8u8u32 API**

Syntax	<pre>void Bfx_PutBits_u32u8u8u32 (uint32 * const Data, const uint8 BitStartPn, const uint8 BitLn, const uint32 Pattern)</pre>	
Service ID	0x72	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitStartPn BitLn Pattern	Start bit position (Valid range: 0 to 31) Bit field length (Valid range: 1 to (32 - BitStartPn)) Bit pattern to be set
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to destination data which is to be modified
Return	void	-
Description	The Bfx_PutBits_u32u8u8u32 function copies the bit pattern from the Pattern parameter starting from 0 bit position for BitLn number of bits into the Data parameter at the bit positions starting from BitStartPn bit position for BitLn number of bits.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

BFX driver**Table 238 Specification for `Bfx_PutBits_u32u8u8u32` API (continued)**

Configuration dependencies	-
User hints	<p>The API is used for modifying some bits of the 32-bit Data parameter. Hence, the valid ranges for input parameters are as follows:</p> <ol style="list-style-type: none"> 1. The valid range for the BitStartPn parameter, which indicates the start position of the bits to be modified, is 0 to 31. 2. The valid range for the BitLn parameter, which indicates the number of bits to be modified, is 1 to (32 - BitStartPn).

4.3.3.55 `Bfx_PutBitsMask_u8u8u8`**Table 239 Specification for `Bfx_PutBitsMask_u8u8u8` API**

Syntax	<pre>void Bfx_PutBitsMask_u8u8u8 (uint8 * const Data, const uint8 Pattern, const uint8 Mask)</pre>	
Service ID	0x80	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Pattern Mask	Bit pattern to be set Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to destination data which is to be modified
Return	void	-
Description	The <code>Bfx_PutBitsMask_u8u8u8</code> function copies the bit pattern from the <code>Pattern</code> parameter into the <code>Data</code> parameter, for all the bit positions for which the logical status of bit in the <code>Mask</code> parameter is set to 1. The remaining bits of the <code>Data</code> parameter retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	

BFX driver**Table 239 Specification for `Bfx_PutBitsMask_u8u8u8` API (continued)**

User hints	The API is implemented as inline function to achieve performance optimization.
-------------------	--

4.3.3.56 `Bfx_PutBitsMask_u16u16u16`**Table 240 Specification for `Bfx_PutBitsMask_u16u16u16` API**

Syntax	<pre>void Bfx_PutBitsMask_u16u16u16 (uint16 * const Data, const uint16 Pattern, const uint16 Mask)</pre>	
Service ID	0x81	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Pattern Mask	Bit pattern to be set Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to destination data which is to be modified
Return	void	-
Description	The <code>Bfx_PutBitsMask_u16u16u16</code> function copies the bit pattern from the <code>Pattern</code> parameter into the <code>Data</code> parameter, for all the bit positions for which the logical status of bit in the <code>Mask</code> parameter is set to 1. The remaining bits of the <code>Data</code> parameter retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

BFX driver**4.3.3.57 Bfx_PutBitsMask_u32u32u32****Table 241 Specification for Bfx_PutBitsMask_u32u32u32 API**

Syntax	<pre>void Bfx_PutBitsMask_u32u32u32 (uint32 * const Data, const uint32 Pattern, const uint32 Mask)</pre>	
Service ID	0x82	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	Pattern Mask	Bit pattern to be set Mask value
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to destination data which is to be modified
Return	void	-
Description	The Bfx_PutBitsMask_u32u32u32 function copies the bit pattern from the Pattern parameter into the Data parameter, for all the bit positions for which the logical status of bit in the Mask parameter is set to 1. The remaining bits of the Data parameter retain their original values.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is implemented as inline function to achieve performance optimization.	

4.3.3.58 Bfx_PutBit_u8u8u8**Table 242 Specification for Bfx_PutBit_u8u8u8 API**

Syntax	<pre>void Bfx_PutBit_u8u8u8 (uint8 * const Data, const uint8 BitPn, const boolean Status)</pre>
---------------	---

BFX driver**Table 242 Specification for `Bfx_PutBit_u8u8u8` API (continued)**

Service ID	0x85	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitPn Status	Bit position (Valid range: 0 to 7) Status value (Valid values: TRUE or FALSE)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The <code>Bfx_PutBit_u8u8u8</code> function updates the logical status of the bit at BitPn bit position of the Data parameter to 1 when the value of Status parameter is TRUE; otherwise, the function updates the logical status of the bit at BitPn bit position of the Data parameter to 0.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying a bit of the 8-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be modified, is 0 to 7.	

4.3.3.59 `Bfx_PutBit_u16u8u8`**Table 243 Specification for `Bfx_PutBit_u16u8u8` API**

Syntax	<pre>void Bfx_PutBit_u16u8u8 (uint16 * const Data, const uint8 BitPn, const boolean Status)</pre>
Service ID	0x86
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API

BFX driver**Table 243 Specification for `Bfx_PutBit_u16u8u8` API (continued)**

Parameters (in)	BitPn Status	Bit position (Valid range: 0 to 15) Status value (Valid values: TRUE or FALSE)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified
Return	void	-
Description	The Bfx_PutBit_u16u8u8 function updates the logical status of the bit at BitPn bit position of the Data parameter to 1 when the value of Status parameter is TRUE; otherwise, the function updates the logical status of the bit at BitPn bit position of the Data parameter to 0.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying a bit of the 16-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be modified, is 0 to 15.	

4.3.3.60 `Bfx_PutBit_u32u8u8`**Table 244 Specification for `Bfx_PutBit_u32u8u8` API**

Syntax	<pre>void Bfx_PutBit_u32u8u8 (uint32 * const Data, const uint8 BitPn, const boolean Status)</pre>	
Service ID	0x87	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for pointer to distinct memory location passed as parameter to the API	
Parameters (in)	BitPn Status	Bit position (Valid range: 0 to 31) Status value (Valid values: TRUE or FALSE)
Parameters (out)	-	-
Parameters (in - out)	Data	Pointer to data which is to be modified

BFX driver**Table 244 Specification for `Bfx_PutBit_u32u8u8` API (continued)**

Return	void	-
Description	The Bfx_PutBit_u32u8u8 function updates the logical status of the bit at BitPn bit position of the Data parameter to 1 when the value of Status parameter is TRUE; otherwise, the function updates the logical status of the bit at BitPn bit position of the Data parameter to 0.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The API is used for modifying a bit of the 32-bit Data parameter. Hence, the valid range for the BitPn parameter, which indicates the position of the bit to be modified, is 0 to 31.	

4.3.3.61 Bfx_GetVersionInfo**Table 245 Specification for `Bfx_GetVersionInfo` API**

Syntax	<pre>void Bfx_GetVersionInfo (Std_VersionInfoType * const Versioninfo)</pre>	
Service ID	0xff	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	Versioninfo	Pointer to memory location where the version information of this module is to be stored
Parameters (in - out)	-	-
Return	void	-
Description	The Bfx_GetVersionInfo function returns the version information of BFX library.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None	

BFX driver
Table 245 Specification for `Bfx_GetVersionInfo` API (continued)

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

4.3.4 Notifications and Callbacks

The BFX library does not support any notifications or callbacks.

4.3.5 Scheduled functions

The BFX library does not support any scheduled functions.

4.3.6 Interrupt service routines

The BFX library does not support any interrupts.

4.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

4.3.7.1 Development errors

The driver does not report any development errors.

4.3.7.2 Production errors

The driver does not report any production errors.

4.3.7.3 Safety errors

The driver does not report any safety errors.

4.3.7.4 Runtime errors

The driver does not report any runtime errors.

4.3.8 Deviations and limitations

4.3.8.1 Deviations

The BFX library does not report any deviation with respect to the requirements.

4.3.8.2 Limitations

The section describes the limitation from software specification.

BFX driver**Table 246 Known limitations**

Reference	Limitation
Multicore capability and reentrancy of the BFX APIs with pointer parameters	The BFX library does not have any mechanism to serialize the access to a shared resource, which is passed as parameter to a BFX API. Therefore, the BFX APIs are multicore capable and reentrant only for distinct pointer instances as parameters. The onus is on the user to implement an appropriate mechanism to serialize the access to such shared resources, which are passed as parameters to BFX APIs.

4.3.9 Unsupported hardware features

The BFX library does not have any associated hardware.

Can_17_McmCan driver

5 **Can_17_McmCan driver**

5.1 **User information**

5.1.1 **Description**

The CAN driver is responsible for providing standard CAN communication services specified by AUTOSAR. The M_CAN unit is the underlying CAN hardware unit, which consists of nodes (called as controllers in AUTOSAR) sharing the message RAM (called as hardware objects in AUTOSAR). The CAN driver provides services for:

- Initialization of CAN controllers to control the behavior and state of the CAN controllers
- Setting and modifying the baud-rate configuration of the CAN controller
- CAN and CAN FD frame transmission and reception is supported
- Successful frame transmission notification, reception of dedicated and FIFO messages and bus-off event notification in the polling and interrupt modes
- Data reception using the receive FIFO functionality
- Pretended networking mode handling
- Multiple read/write period functionality support
- Multiplexed transmission using Tx queue

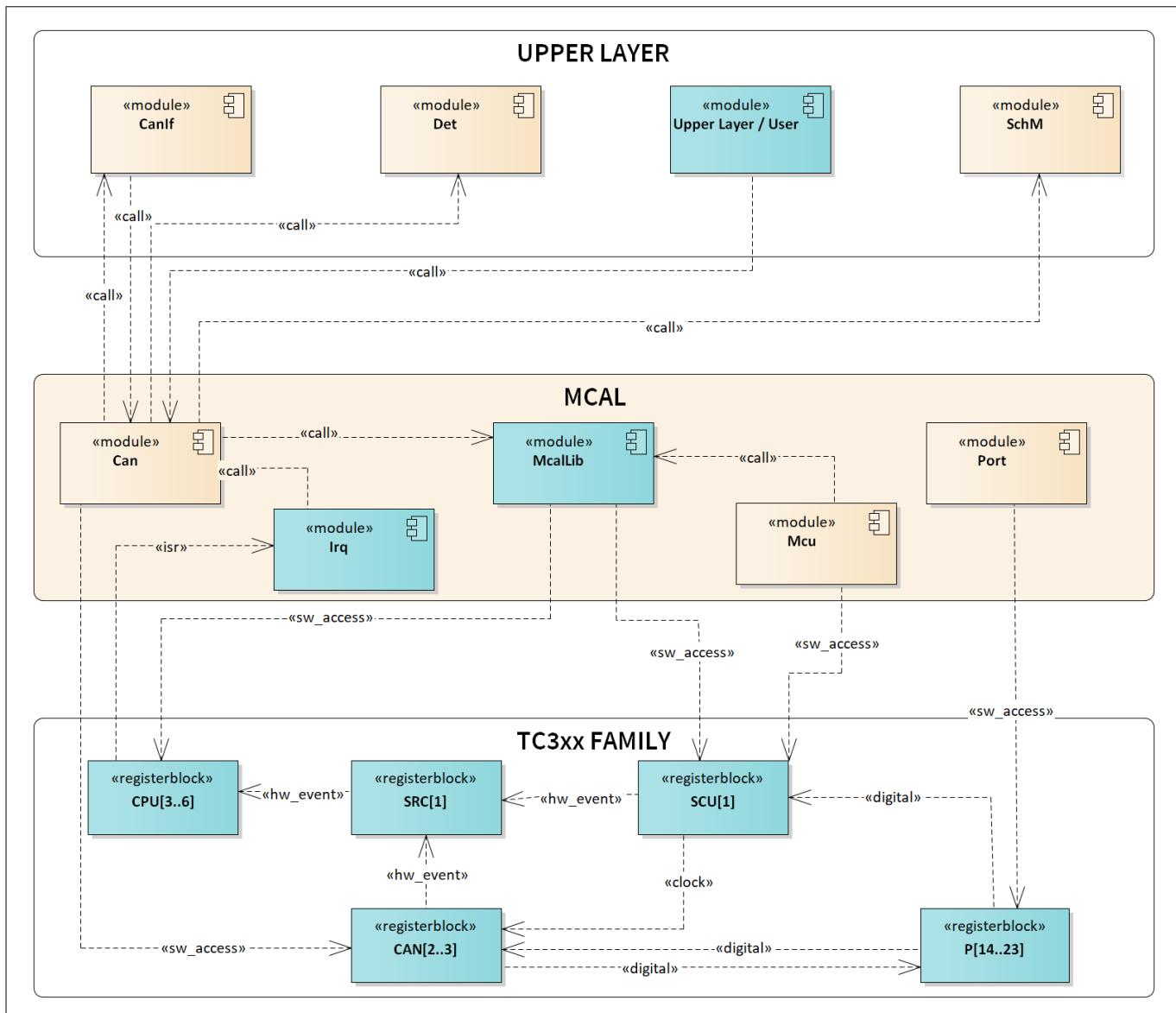
Individual interrupt lines are routed for the handling of the following events of each CAN node:

- Bus-off event handling
- Transmit event handling
- Dedicated message receive event handling
- Receive FIFO 0 and FIFO 1 watermark and FIFO full event handling

The CAN driver is delivered as a Post-Build variant. Therefore, the driver supports configuration parameters with pre-compile and post-build configuration classes. The APIs provided by the CAN driver are multi-core capable, which means that they may be invoked from several cores simultaneously.

5.1.2 **Hardware-software mapping**

This section describes the system view of the driver and peripherals administered by it.

Can_17_McmCan driver

Figure 82 **Mapping of hardware-software interfaces**

5.1.2.1 SRC: dependent hardware peripheral

Hardware functional features

The CAN driver depends on the interrupt router for raising an interrupt to the CPU based on the CAN Tx-complete/ CAN Rx-complete/ CAN Rx-FIFO complete/ CAN bus-off errors.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software. The CAN driver uses the interrupt router module to route the different CAN Tx, Rx, Rx-FIFO and Bus-Off activities for every CAN controller configured.

Hardware diagnostic features

The SMU alarms configured for interrupt router are not monitored by the CAN driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU. The CAN driver provides interrupt handlers as software interfaces, which must be invoked from the ISR.

Can_17_McmCan driver

5.1.2.2 M_CAN: primary hardware peripheral

Hardware functional features

The CAN driver uses the M_CAN to communicate according to the ISO 11898-1. In addition, the M_CAN supports communication according to the CAN FD protocol specification 1.0. The key hardware functional features used by the driver are:

- All the CAN controllers and message RAM available in the M_CAN module are used to implement the CAN driver
- CAN FD with up to 64 data bytes supported
- Up to 64 dedicated receive buffers
- Up to 32 dedicated transmit buffers
- Two configurable receive FIFOs
- Configurable transmit queue
- Four individual interrupts are configured per controller they are dedicated Rx, Rx FIFOs, Tx and bus-off events.

The unsupported features of the M_CAN are:

- Event-synchronized time-triggered communication
- CAN error logging
- High-priority messages

Users of the hardware

The CAN driver exclusively utilizes the M_CAN module.

Hardware diagnostic features

The SMU alarms configured for the M_CAN are not monitored by the CAN driver.

Hardware events

The CAN driver uses the following hardware events from the M_CAN IP:

- Successful transmission of a CAN / CAN FD frame is notified by flag (relevant bit in the IR register) as well as interrupt. The CAN driver uses the TxEvent FIFO new entry to handle notifications to upper layer
- Successful reception of a CAN / CAN FD frame is notified by flag (relevant bit in the IR register) as well as interrupt. The CAN driver uses the receive interrupt raised
- Bus-Off event is notified by flag (relevant bit in the IR register) as well as interrupt. The CAN driver uses the bus-off interrupt which is raised
- Both Rx FIFO0 watermark reached, RxFIFO0 Full, RxFIFO 1 watermark reached event, RxFIFO1 Full events are routed to same ISR. All the listed flags are handled in the CAN driver to process the received data through FIFO

5.1.2.3 SCU: dependent hardware peripheral

Hardware functional features

The CAN driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB clock signal for functioning. The clock frequency for M_CAN depends on the clock configuration in the SCU.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

Can_17_McmCan driver

The SMU alarms configured for the SCU IP are not monitored by the CAN driver.

Hardware events

Hardware events from the SCU are not used by the CAN driver.

5.1.2.4 Port: dependent hardware peripheral**Hardware functional features**

The receive and transmit signals are routed to the MCMCAM through the port pads. The signals as mentioned are configured and enabled through the PORT driver.

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from port pads are not used by the CAN driver.

5.1.3 File structure**5.1.3.1 C File Structure**

This section provides details of the C files of the CAN driver.

Can_17_McmCan driver

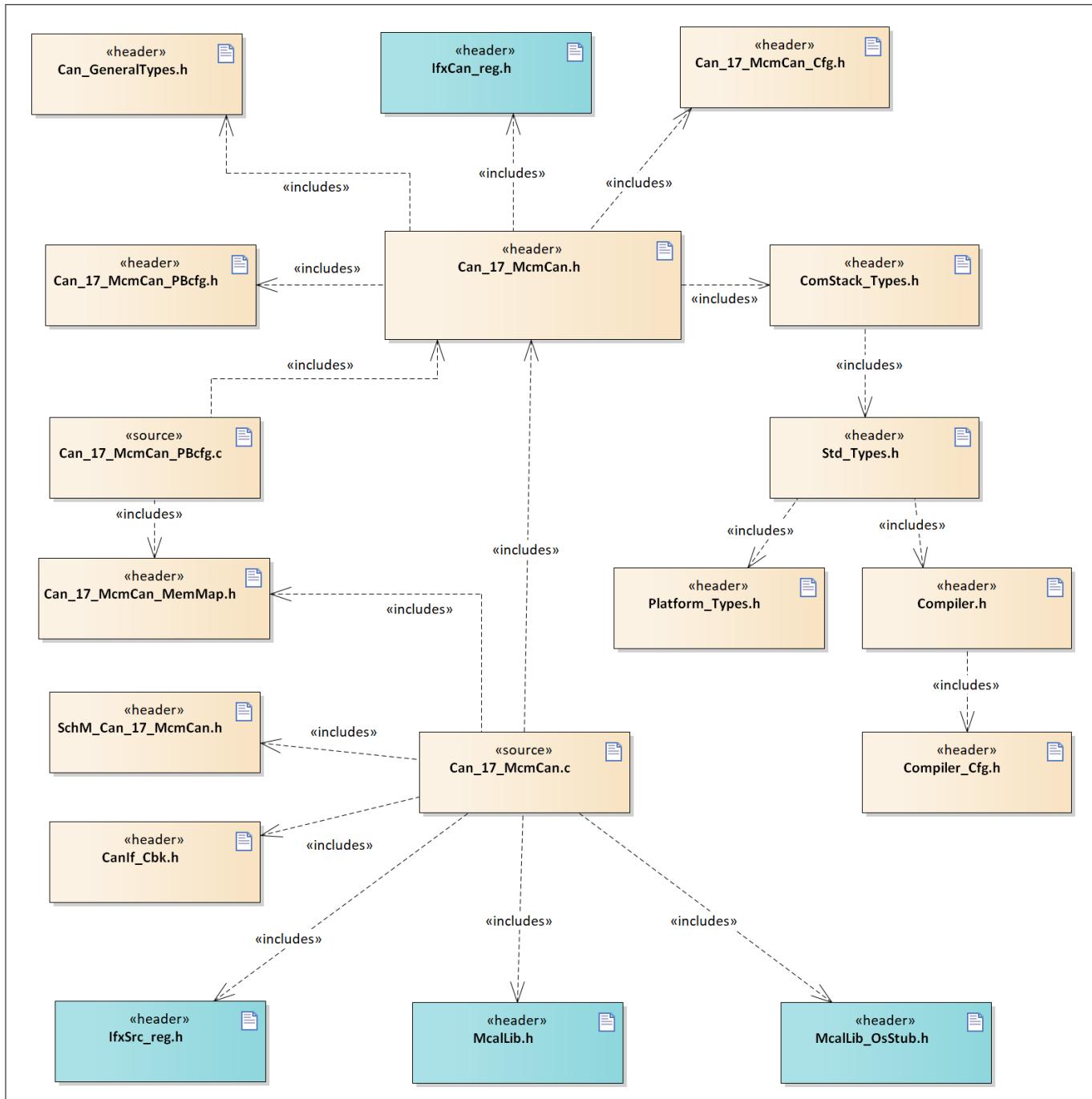


Figure 83 C File Structure

Table 247 C File Structure

File name	Description
CanIf_Cbk.h	Header file containing declarations of the CanIf callbacks
Can_17_McmCan.c	Implementation of the CAN driver functionality
Can_17_McmCan.h	Export of the CAN driver functionality
Can_17_McmCan_Cfg.h	The configuration data of the CAN driver is declared here Note: All pre-compile time configuration parameters shall be defined as pre-processor directives (#define)

Can_17_McmCan driver
Table 247 C File Structure (continued)

File name	Description
Can_17_McmCan_MemMap.h	Mapping of code and data (variables, constant variables) to specific memory sections
Can_17_McmCan_PBcfg.c	Post Build configuration data of the CAN driver is defined here
Can_17_McmCan_PBcfg.h	Header file (generated) containing declaration of the post-build configuration data structures
Can_GeneralTypes.h	Contains all types and constants that are shared among the AUTOSAR CAN modules Can, CanIf and CanTrcv
ComStack_Types.h	Type Definition for Com stack
Compiler.h	Provides abstraction from compiler-specific keywords
Compiler_Cfg.h	Configuration header file for compiler abstraction
IfxCAN_Reg.h	SFR header file for CAN
IfxSrc_Reg.h	SFR header file for Interrupt Controller
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore™. This shall be included by other drivers to call OS APIs.
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Can_17_McmCan.h	Functions to enable/disable interrupts are declared here.
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

5.1.3.2 Code Generator Plugin Files

This section provides details of the code generator plugin files of the CAN driver.

Can_17_McmCan driver

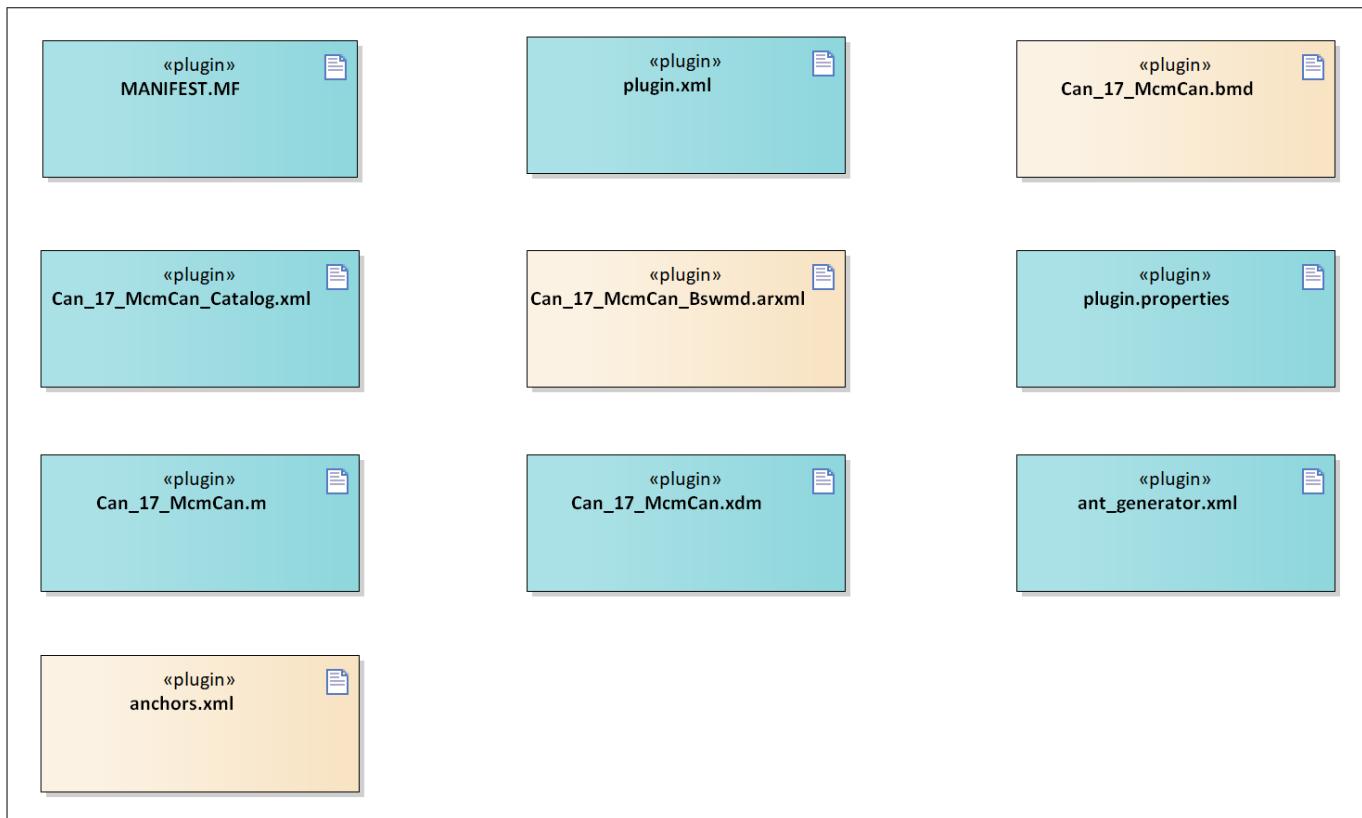


Figure 84 **Code Generator Plugin Files**

Table 248 **Code Generator Plugin Files**

File name	Description
Can_17_McmCan.bmd	AUTOSAR format XML data model schema file for the CAN driver
Can_17_McmCan.m	Code template macro file for the CAN driver
Can_17_McmCan.xdm	Tresos format XML data model schema file
Can_17_McmCan_Bswmd.arxml	AUTOSAR format module description file
Can_17_McmCan_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the metadata for the CAN driver
anchors.xml	AUTOSAR format module description file
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point
plugin.properties	Tresos plugin support file for the CAN driver
plugin.xml	Tresos plugin support file for the CAN driver

5.1.4 Integration hints

This section lists the key points that an integrator or user of the CAN driver must consider.

Can_17_McmCan driver

5.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the CAN driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **CAN Interface (CanIf)**

The CanIf module is a part of the AUTOSAR stack that provides upper layers a hardware independent interface to the CAN communication system comprising multiple CAN controllers.

The `CanIf_Cbk.c` and `CanIf_Cbk.h` files are provided as stub code and needs to be replaced with complete CanIf module during integration phase. The CAN driver uses the APIs of CanIf to provide notifications as listed.

`CanIf_ControllerModeIndication()`: Notification for a successful state transition that was triggered for a controller

`CanIf_TxConfirmation()`: Notification for a successfully processed transmission of a CAN Tx pdu.

`CanIf_RxIndication()`: Notification for a successful reception of a received CAN Rx l-pdu to the CanIf after passing all filters and validation checks.

`CanIf_ControllerBusOff()`: Notification for a Controller BusOff event referring to the corresponding CAN Controller.

`CanIf_CurrentIcomConfiguration()`: Notification to inform about the change of the Icom configuration of a CAN controller.

`CanIf_TriggerTransmit()`: Within this API, the CanIf shall check whether the available data fits into the buffer size reported by `PduInfoPtr->SduLength`. If it fits, it shall copy its data into the buffer provided by `PduInfoPtr`.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user specific memory regions. To achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Can_17_McmCan_MemMap.h`.

The file `Can_17_McmCan_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

Can_17_McmCan driver

```

***** GLOBAL RAM DATA -- NON CLEARED LMU *****/
#if defined CAN_17_MCMCAN_START_SEC_VAR_CLEARED_QM_GLOBAL_32
/*User pragmas here for non- cached LMU*******/
#define CAN_17_MCMCAN_START_SEC_VAR_CLEARED_QM_GLOBAL_32
#define MEMMAP_ERROR
#elif defined CAN_17_MCMCAN_STOP_SEC_VAR_CLEARED_QM_GLOBAL_32
/*User pragmas here for non- cached LMU*******/
#define CAN_17_MCMCAN_STOP_SEC_VAR_CLEARED_QM_GLOBAL_32
#define MEMMAP_ERROR

***** CORE[x] CONFIG DATA --PF[x] *****/
#elif defined CAN_17_MCMCAN_START_SEC_CONFIG_DATA_QM_CORE0_UNSPECIFIED
/*User pragmas here for PF[x]*******/
#define CAN_17_MCMCAN_START_SEC_CONFIG_DATA_QM_CORE0_UNSPECIFIED
#define MEMMAP_ERROR

#elif defined CAN_17_MCMCAN_STOP_SEC_CONFIG_DATA_QM_CORE0_UNSPECIFIED
/*User pragmas here for PF[x]*******/
#define CAN_17_MCMCAN_STOP_SEC_CONFIG_DATA_QM_CORE0_UNSPECIFIED
#define MEMMAP_ERROR

***** CODE -- PF[x] *****/
#elif defined CAN_17_MCMCAN_START_SEC_CODE_QM_GLOBAL
/*User pragmas here for PF[x]*******/
#define CAN_17_MCMCAN_START_SEC_CODE_QM_GLOBAL
#define MEMMAP_ERROR
#endif

#if defined MEMMAP_ERROR
#error "Can_17_McmCan_MemMap.h, wrong pragma command"
#endif

```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The CAN driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the CAN driver must process all the errors reported to the DET module through the API `Det_ReportError()`.

The files `Det.h` and `Det.c` are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is not required for integrating the CAN driver.

- **SchM**

Can_17_McmCan driver

The SchM module is a part of the RTE that manages the Basic Software Scheduler. The CAN driver uses the exclusive areas defined in `SchM_Can_17_McmCan.h` to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the CAN driver are:

CanIntCtrl

CanWrMO

IcomMsgCntrVal

The `SchM_Can_17_McmCan.h` and `SchM_Can_17_McmCan.c` files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the CAN driver as suspend / resume of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is depicted below:

```
***** Sample implementation of SchM_Can_17_McmCan.c ****/
void SchM_Enter_Can_17_McmCan_CanIntCtrl()
{
    /* Start critical section */
    SuspendAllInterrupts(); /* Suspend CPU core interrupts */
}

void SchM_Exit_Can_17_McmCan_CanIntCtrl()
{
    /* End of critical section */
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Can_17_McmCan_CanWrMO()
{
    /* Start critical section */
    SuspendAllInterrupts(); /* Suspend CPU core interrupts */
}

void SchM_Exit_Can_17_McmCan_CanWrMO()
{
    /* End of critical section */
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Can_17_McmCan_IcomMsgCntrVal()
{
    /* Start critical section */
    SuspendAllInterrupts(); /* Suspend CPU core interrupts */
}

void SchM_Exit_Can_17_McmCan_IcomMsgCntrVal()
{
    /* End of critical section */
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

The CAN driver does not report any safety errors.

Can_17_McmCan driver

- **Notifications and callbacks**

The CAN driver does not implement any notifications. However, it does report transmit confirmation, mode change indication, bus-off and wake up identification, pretended network activation or de-activation completion and successful reception through the CanIf module call backs.

- **Operating system (OS)**

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of the interrupts must also be managed by the OS or application. The operating system files provided by MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

5.1.4.2 Multicore and Resource Manager

The CAN driver supports execution of its APIs simultaneously from all CPU cores. The user should allocate resources of CAN to CPU cores at pre-compile time using the Resource Manager module. The following are the key points to be considered with respect to multicore in the driver:

- CAN controllers of the CAN driver can be allocated to CPU cores at pre-compile time.
- CAN controllers that are not allocated to a CPU core shall be by default allocated to the master core.
- Initialization of the CAN controller must start with the master core initialization only after the successful initialization of the master core should there be a trigger for a slave core initialization. CAN driver of the slave cores can be initialized simultaneously.
- De-initialization of the CAN driver for different slave cores can be initiated simultaneously. The master core de-initialization of the CAN driver should be carried out only after the de-initialization of the CAN driver in all the slave cores.
- DETs will be raised in case APIs are invoked with mismatch of CPU core and controller IDs or hardware object IDs.
- Interrupts raised by a hardware group must be serviced by the CPU core to which the hardware group has been allocated to.
- Locating constants, variables and configuration data to correct memory space should be done by the user. Memory sections are marked GLOBAL(common to all cores) and CORE[x] (specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of CAN driver is placed under single MemMap section. It can be relocated to any PFlash region.

Data section:

The RAM variable memory sections marked as specific to a core should be relocated to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region. In devices with no LMU, non-cached DSPR can be used.

Configuration data and constants:

The configuration data sections marked as specific to a core should be re-located to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

Note: Relocating code, data or constants to a distant memory region would impact execution timings.

Note: If the driver operates from single (master) core, all the sections may be relocated to the PFlash/ DSPR/DLMU of the same CPU core.

Can_17_McmCan driver

5.1.4.3 MCU support

The CAN driver is dependent on the MCU driver for clock configuration. The initialization of CAN driver must be started only after the completion of MCU initialization. The following must be considered while configuring the MCU driver in EB Tresos:

- McuMCanClockSourceSelection - Used to select the different clock source.
- McuMCanFrequency - To be set if the McuMCanClockSourceSelection is MCAN_CLOCK_SOURCE_MCANI_SEL1.
- McuMainOscillatorFrequency - To be set if the McuMCanClockSourceSelection is MCAN_CLOCK_SOURCE_OSC_SEL2.

5.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the CAN driver through the PORT configuration and initialize the port pins prior to invoking of CAN initialization.

The TxD and RxD pins (corresponding to the Rx Pin selection made in CAN driver) of the different CAN controllers must be configured with respective direction and configuration in the PORT driver.

5.1.4.5 DMA support

The CAN driver does not use any services provided by the DMA driver.

5.1.4.6 Interrupt connections

The interrupt connections of the CAN driver are described in this section.

Table 249 Handling CAN interrupt lines

Controller	Signal	Service type	Function to be called
Controller 0	CAN0SR0_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER00_ID);
Controller 0	CAN0SR1_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER00_ID);
Controller 0	CAN0SR2_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER00_ID);
Controller 0	CAN0SR3_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER00_ID);

Can_17_McmCan driver**Table 249 Handling CAN interrupt lines (continued)**

Controller 1	CAN0SR4_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER01_ID);
Controller 1	CAN0SR5_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER01_ID);
Controller 1	CAN0SR6_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER01_ID);
Controller 1	CAN0SR7_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER01_ID);
Controller 2	CAN0SR8_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 2	CAN0SR9_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 2	CAN0SR10_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 2	CAN0SR11_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 3	CAN0SR12_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);

Can_17_McmCan driver**Table 249 Handling CAN interrupt lines (continued)**

Controller 3	CAN0SR13_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);
Controller 3	CAN0SR14_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);
Controller 3	CAN0SR15_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);
Controller 4	CAN1SR0_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID, CAN_17_MCMCAN_HWMCMCONTROLLER00_ID);
Controller 4	CAN1SR1_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID, CAN_17_MCMCAN_HWMCMCONTROLLER00_ID);
Controller 4	CAN1SR2_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID, CAN_17_MCMCAN_HWMCMCONTROLLER00_ID);
Controller 4	CAN1SR3_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID, CAN_17_MCMCAN_HWMCMCONTROLLER00_ID);
Controller 5	CAN1SR4_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID, CAN_17_MCMCAN_HWMCMCONTROLLER01_ID);
Controller 5	CAN1SR5_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID, CAN_17_MCMCAN_HWMCMCONTROLLER01_ID);

Can_17_McmCan driver**Table 249 Handling CAN interrupt lines (continued)**

Controller 5	CAN1SR6_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MC MCAN_HWMCMKERNEL1_ID ,CAN_17_MCMCAN_HWMCMCONTROLLER01_ID);
Controller 5	CAN1SR7_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MC MCAN_HWMCMKERNEL1_ID ,CAN_17_MCMCAN_HWMCMCONTROLLER01_ID);
Controller 6	CAN1SR8_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID,CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 6	CAN1SR9_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID,CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 6	CAN1SR10_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MC MCAN_HWMCMKERNEL1_ID ,CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 6	CAN1SR11_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MC MCAN_HWMCMKERNEL1_ID ,CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 7	CAN1SR12_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID,CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);
Controller 7	CAN1SR13_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL1_ID,CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);
Controller 7	CAN1SR14_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MC MCAN_HWMCMKERNEL1_ID ,CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);

Can_17_McmCan driver**Table 249 Handling CAN interrupt lines (continued)**

Controller 7	CAN1SR15_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MC_MCAN_HWMCMKERNEL1_ID,CAN_17_MCMCAN_HWMCM_CONTROLLER03_ID);
Controller 8	CAN2SR0_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL2_ID,CAN_17_MCMCAN_HWMCM_CONTROLLER00_ID);
Controller 8	CAN2SR1_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL2_ID,CAN_17_MCMCAN_HWMCM_CONTROLLER00_ID);
Controller 8	CAN2SR2_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MC_MCAN_HWMCMKERNEL2_ID,CAN_17_MCMCAN_HWMCM_CONTROLLER00_ID);
Controller 8	CAN2SR3_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MC_MCAN_HWMCMKERNEL2_ID,CAN_17_MCMCAN_HWMCM_CONTROLLER00_ID);
Controller 9	CAN2SR4_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL2_ID,CAN_17_MCMCAN_HWMCM_CONTROLLER01_ID);
Controller 9	CAN2SR5_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCMKERNEL2_ID,CAN_17_MCMCAN_HWMCM_CONTROLLER01_ID);
Controller 9	CAN2SR6_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MC_MCAN_HWMCMKERNEL2_ID,CAN_17_MCMCAN_HWMCM_CONTROLLER01_ID);
Controller 9	CAN2SR7_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MC_MCAN_HWMCMKERNEL2_ID,CAN_17_MCMCAN_HWMCM_CONTROLLER01_ID);

Can_17_McmCan driver**Table 249 Handling CAN interrupt lines (continued)**

Controller 10	CAN2SR8_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCKERNEL2_ID,CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 10	CAN2SR9_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCKERNEL2_ID,CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 10	CAN2SR10_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MCMCAN_HWMCKERNEL2_ID,CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 10	CAN2SR11_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MCMCAN_HWMCKERNEL2_ID,CAN_17_MCMCAN_HWMCMCONTROLLER02_ID);
Controller 11	CAN2SR12_ISR	Service on CAN data Reception through dedicated buffer.	Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCKERNEL2_ID,CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);
Controller 11	CAN2SR13_ISR	Service for the transmission completion new entry event on TX FIFO Event.	Can_17_McmCan_IsrTransmitHandler(CAN_17_MCMCAN_HWMCKERNEL2_ID,CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);
Controller 11	CAN2SR14_ISR	Service on CAN controller within Bus Off mode.	Can_17_McmCan_IsrBusOffHandler(CAN_17_MCMCAN_HWMCKERNEL2_ID,CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);
Controller 11	CAN2SR15_ISR	Service on CAN receive FIFO water mark level or FIFO full level reached on Rx FIFO0 or Rx FIFO1.	Can_17_McmCan_IsrRxFIFOHandler(CAN_17_MCMCAN_HWMCKERNEL2_ID,CAN_17_MCMCAN_HWMCMCONTROLLER03_ID);

Can_17_McmCan driver

Invoking of interrupt handlers provided by the driver must be done by the user. A sample invocation for controller 0, dedicated Rx interrupt is shown as follows:

```
#include "Can_17_McmCan.h"
ISR(CAN0SR0_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();

    /* Call CAN Rx Interrupt function for dedicated buffer */

    Can_17_McmCan_IsrReceiveHandler(CAN_17_MCMCAN_HWMCMKERNEL0_ID, CAN_17_MCMCAN_HWM
    CMCONTROLLER00_ID);
}
```

Can_17_McmCan driver

5.1.4.7 Example usage

Configuring the driver and related modules

The AUTOSAR configuration parameter CanControllerBaseAddress is used with a selection of address for the mapping of CAN controller in the hardware to the configured CAN controller Id.

Note: Kernel specific RAM allocation per controller is based on the CAN hardware objects allocated to that controller. In the case of a controller with FD baudrate configured the RAM allocated per hardware object is 4 times higher. Hence, in order to have an optimised RAM memory utilisation it is recommended that a controller with FD baudrate configured have only hardware objects that use CAN FD communication.

Configuring of CAN hardware object

The MCMCAN hardware is supported with Rx FIFO and dedicated Rx buffer for reception and Tx Queue and Tx dedicated buffer for the transmission operation. The user can select the hardware object buffer type while configuring hardware object handler (HOH).

The following rules are considered while configuring the buffer type selection for HRH and HTH:

- If the CanObjectType value is configured with RECEIVE and CanHwObjectCount value is equal to 1 then the buffer type of HRH is assigned as Rx dedicated buffer.
- If the CanObjectType value is configured with RECEIVE and CanHwObjectCount value is greater than 1 then the buffer type of HRH is assigned as Rx FIFO. First instance shall be considered with buffer type as Rx FIFO0 and if second one is available for the same controller, it is considered with buffer type as Rx FIFO1.
- The user can configure Rx dedicated buffer or Rx FIFO (0 and 1) or the combination of the two types of the receive operations.
- If the CanObjectType value is configured with TRANSMIT and CanHwObjectCount value is greater than 1 then the buffer type of HTH is assigned as Tx Queue. The CanMultiplexedTransmission check needs to be done to make CanHwObjectCount value greater than 1 for a TRANSMIT message.
- If the CanObjectType value is configured with TRANSMIT and CanHwObjectCount value is equal to 1 then the buffer type of HTH is assigned as Tx dedicated buffer.

CanObjectId configuration rules

- CanObjectId shall be unique and shall start with 0 and continue without any gap
- HRHs (CanObjectId) belonging to a controller shall be grouped together.
- HTHs (CanObjectId) belonging to a controller shall be grouped together.

Ensure HRHs of all controllers are grouped before the HTHs of all controllers, then the entire HRH id shall have lower CanObjectId than all HTH.

Initializing the CAN driver

```

/* Mcu Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock ();
/* Port Initialization */
Port_Init(&Port_Config);
/* CAN Initialization */
Can_17_McmCan_Init(&Can_17_McmCan_Config);
/* Further APIs of CAN driver can be called now */

```

CAN controller mode change

Can_17_McmCan driver

After CAN initialization the following sequence may be followed.

```
/* Set the controller with state as START */
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_0,CAN_T_START);
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_1,CAN_T_START);
```

Disabling and enabling CAN controller interrupts

```
/* Disable the interruption by CAN event */
Can_17_McmCan_DisableControllerInterrupts(Can_17_McmCanConf_CanController_CanController_0);
/* Request Write operation */
Can_17_McmCan_Write(10, &PduInfo_ExtId[0]) ;
/* Enable the interruption by CAN event */
Can_17_McmCan_EnableControllerInterrupts(Can_17_McmCanConf_CanController_CanController_0);
/* Notification can be expected now */
```

Re-initializing CAN controller baudrate

```
/* Set the controller with state as STOP */
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_0,CAN_T_STOP);
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_1,CAN_T_STOP);
/* Set the baudrate */
#if (CAN_17_MCMCAN_SET_BAUDRATE_API == STD_ON)
Can_17_McmCan_SetBaudrate(Can_17_McmCanConf_CanController_CanController_0, 0);
#endif

/* Set the baudrate */
#if (CAN_17_MCMCAN_SET_BAUDRATE_API == STD_ON)
Can_17_McmCan_SetBaudrate(Can_17_McmCanConf_CanController_CanController_1, 0);
#endif
/* Set the controller with state as START */
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_0,CAN_T_START);
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_1,CAN_T_START);
```

Can_17_McmCan driver**Activating and de-activating the pretended networking**

```
/* Set the controller with state as START */
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_0,CAN_T_START);
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_1,CAN_T_START);
/* Activate Pretended networking */
#if (CAN_17_MCMCAN_PUBLIC_ICOM_SUPPORT == STD_ON)
Can_17_McmCan_SetIcomConfiguration(Can_17_McmCanConf_CanController_CanController_0, 1);
#endif
/* Activate Pretended networking */
#if (CAN_17_MCMCAN_PUBLIC_ICOM_SUPPORT == STD_ON)
Can_17_McmCan_SetIcomConfiguration(Can_17_McmCanConf_CanController_CanController_1, 2);
#endif

/* Deactivate Pretended networking */
#if (CAN_17_MCMCAN_PUBLIC_ICOM_SUPPORT == STD_ON)
Can_17_McmCan_SetIcomConfiguration(Can_17_McmCanConf_CanController_CanController_0, 0);
#endif
/* Deactivate Pretended networking */
#if (CAN_17_MCMCAN_PUBLIC_ICOM_SUPPORT == STD_ON)
Can_17_McmCan_SetIcomConfiguration(Can_17_McmCanConf_CanController_CanController_1, 0);
#endif
```

Can_17_McmCan driver**De-initializing the CAN driver**

```
/* Mcu Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock ();
/* Port Initialization */
Port_Init(&Port_Config);
/* CAN Initialization */
Can_17_McmCan_Init(&Can_17_McmCan_Config);

/* Set the controller with state as START */
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_0,CAN_T_START);
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_1,CAN_T_START);

/* Data transmission by Controller 0 to 1 */
Can_17_McmCan_Write(8, &PduInfo_1[0]) ;

Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_0,CAN_T_STOP);
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_1,CAN_T_STOP);

/* Call CAN de-Initialization function */
Can_17_McmCan_DeInit();
```

Can_17_McmCan driver

Transmission and reception in polling Mode

```

/* Mcu Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock ();
/* Port Initialization */
Port_Init(&Port_Config);
/* CAN Initialization */
Can_17_McmCan_Init(&Can_17_McmCan_Config);

/* Set the controller with state as START */
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_0,CAN_T_START);
Can_17_McmCan_SetControllerMode
(Can_17_McmCanConf_CanController_CanController_1,CAN_T_START);

/* Data transmission by Controller 0 to 1 */
Can_17_McmCan_Write(8, &PduInfo_1[0]);

/* In Scheduled function call poll for the reception of the message */
/* Reception is polled for and shall raise Can If notification in controller 1 */
*/
Can_17_McmCan_MainFunction_Read_x();

/* Transmission is polled for and shall raise a Can If notification in
controller 0 */
Can_17_McmCan_MainFunction_Write_x();

```

Possible values of CanControllerBaseAddress container

Table 250 Controller base address List

Controller	Base address
Controller 0 (Node 00)	0xF0208100
Controller 1 (Node 01)	0xF0208500
Controller 2 (Node02)	0xF0208900
Controller 3 (Node03)	0xF0208D00
Controller 4 (Node10)	0xF0218100
Controller 5 (Node11)	0xF0218500
Controller 6 (Node12)	0xF0218900
Controller 7 (Node 13)	0xF0218D00
Controller 8 (Node20)	0xF0228100
Controller 9 (Node 21)	0xF0228500
Controller 10 (Node 22)	0xF0228900

Can_17_McmCan driver

Table 250 Controller base address List (continued)

Controller	Base address
Controller 11 (Node 23)	0xF0228D00

Above list contains the base address of the controller nodes supported, these address are to be updated by the integrator in the respective CanController configuration for mapping the controller to respective CAN hardware object (refer to CanControllerBaseAddress in the CanController container).

5.1.5 Key architectural considerations

5.1.5.1 CAN interrupt handling

Tx event handling

TEFN (Tx Event FIFO New Entry) is the only event enabled for handling the Tx event FIFO.

Note: *TEFN will be triggered for every new status event added in the event FIFO. TEFL(Tx Event FIFO Lost) and TEFF (Tx Event FIFO Full) events are cleared in the same handler. If TEFL is SET, this indicates that the Tx event is lost for which the CAN driver will raise a DET as CAN_17_MCMCAN_E_DATALOST. The same DET is raised during multi-period transmit as well Can_17_McmCan_MainFunction_Write_x. This indicates that the bus is loaded and no sufficient time is provided to process the Tx notifications to upper layer.*

Rx dedicated handling

DRX (Message stored to Dedicated Rx Buffer) event is raised when one of the dedicated buffer is updated with the message from CAN bus. Independent bits are present in NDAT1 and NDAT2, which will indicate that the message is copied to which of the dedicated buffer.

Rx FIFO handling

RFxW (Rx FIFO x Watermark reached) and RFxF (Rx FIFO x Full) are the two events that are enabled for handling the Rx FIFO (x represents FIFO 0 or FIFO 1).

Note: *Watermark is used for CAN HW to trigger the interrupt when certain number of messages are received in FIFO. If RFxL is SET, this indicates that the FIFO message is lost for which CAN driver will raise the DET as CAN_17_MCMCAN_E_DATALOST and RFxL is cleared by CAN driver. This indicates that the bus is loaded and no sufficient time is provided to process the received frames.*

Bus Off handling

Bus off interrupt is enabled in order to indicate that the bus is faulty to notify upper layer for handling the erroneous bus.

Note: *Due to possibility of hardware updating the TEFN bit in background in same cycle when software is trying to clear this bit, IR remains updated due to hardware write. Due to this, software has to clear the flag repeatedly and make sure that the intended flag is cleared before processing the interrupt. Same behavior can be observed for other flags like RF0W, RF1W and DRX. So all the above listed flags are cleared in an loop with a exit condition for maximum of three retry. The retry mechanism is not implemented for Bus off interrupt since the bit setting and clearing from software cannot occur in same clock cycle.*

Can_17_McmCan driver

5.1.5.2 Multi-period Tx and Rx

- The multi-period Tx and Rx main function calls Can_17_McmCan_MainFunction_Read_(x) and Can_17_McmCan_MainFunction_Write_(x), where number of main function calls generated is based on the number of main function period entries configured in Can/CanGeneral/CanMainFunctionRWPeriods configured are generated with the suffix 'x' changing based on the index of Can/CanGeneral/CanMainFunctionRWPeriods/*[] configured.
- The multi-period Tx and Rx will be available only when the number of Can/CanGeneral/CanMainFunctionRWPeriods configured is greater than one and at least one controller is configured with Rx (for Can_17_McmCan_MainFunction_Read_(x)) or Tx (for Can_17_McmCan_MainFunction_Write_(x)) in polling mode.
- The default function call Can_17_McmCan_MainFunction_Read will not be available when Can_17_McmCan_MainFunction_Read_(x) is generated.
- The default function call Can_17_McmCan_MainFunction_Write will not be available when Can_17_McmCan_MainFunction_Write_(x) is generated.
- If at least one controller is configured to have Tx as polling mode and CanMainFunctionPeriod is greater than one, then the function Can_17_McmCan_MainFunction_Write_(x) is generated for all values of 'x' available in the index of CanMainFunctionPeriod, even if no controller Tx operating is polling mode is referring to that CanMainFunctionPeriod index, this function generated for such CanMainFunctionPeriod is effectively an empty function with no actions performed.
- If at least one controller is configured to have Rx as polling mode and CanMainFunctionPeriod is greater than one, then the function Can_17_McmCan_MainFunction_Read_(x) is generated for all values of 'x' available in the index of CanMainFunctionPeriod, even if no controller Rx operating is polling mode is referring to that CanMainFunctionPeriod index, this function generated for such CanMainFunctionPeriod is effectively an empty function with no actions performed.
- The multi-period Tx and Rx functions generated with a controller polling associated to its CanMainFunctionPeriod will give out notifications only for hardware object handle events (Tx / Rx) associated to that CanMainFunctionPeriod.

5.1.5.3 Mixed mode Rx processing

Few applications have requirements to handle the Receive FIFO data through polling since these messages are not time critical. However, the dedicated buffers are to be managed through interrupts to manage the tight timing constraints.

MIXED mode Rx processing allows dedicated Rx processing in INTERRUPT mode and RX FIFO in POLLING mode. This can be achieved through the following steps:

1. CanRxProcessing is set to INTERRUPT and the Can_17_McmCan_MainFunction_Read API is called in periodic raster.
2. In mixed mode, the behavior will be unpredictable in case the SRE of the RX FIFO is enabled by application during initialization and the Can_17_McmCan_MainFunction_Read API is also called in periodic raster.
3. In case the user decides to use POLLING for FIFO's, all controller FIFO's will be processed in POLLING mode only.
4. CanRxProcessing should be chosen as either INTERRUPT or POLLING uniformly for all controllers. This has to be done to ensure the desired functionality of mixed mode.

Note that multiperiod (Can_17_McmCan_Main_function_Read_x) function for this feature will not be supported.

Can_17_McmCan driver

5.2 Assumptions of Use (AoUs)

There are no AoUs for the driver.

Can_17_McmCan driver

5.3 Reference information

5.3.1 Configuration interfaces

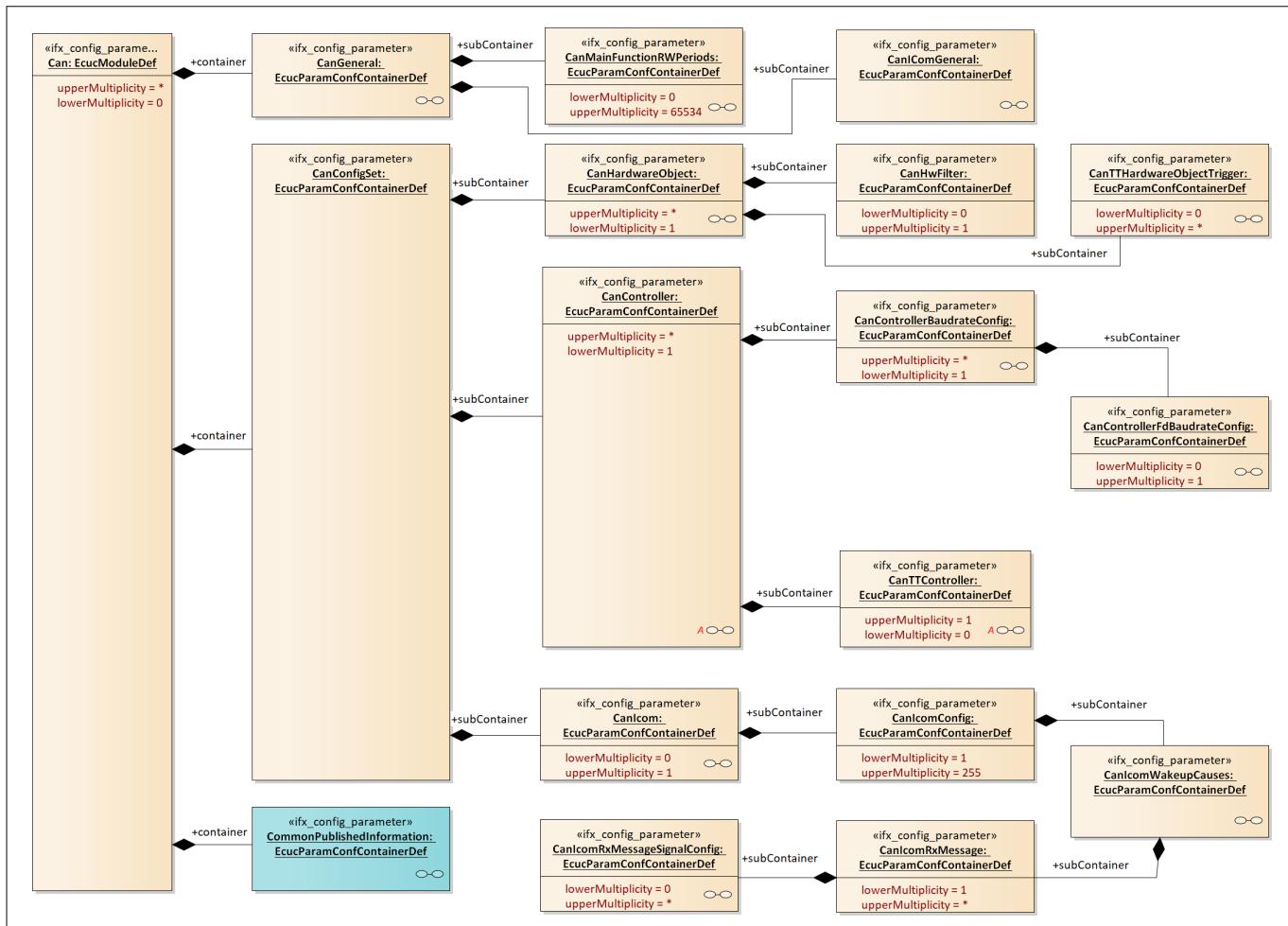


Figure 85 Container hierarchy along with their configuration parameters

5.3.1.1 Container: CanConfigSet

This container contains the configuration parameters and sub containers of the AUTOSAR CAN driver.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.2 Container: CanController

This container contains the configuration parameters of the CAN controller(s).

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

Can_17_McmCan driver

5.3.1.2.1 CanBusoffProcessing

Table 251 Specification for CanBusoffProcessing

Name	CanBusoffProcessing		
Description	<p>Specifies the way bus off event on the controller is notified.</p> <p>Enables/disables the Can_17_McmCan_MainFunction_BusOff() API for handling bus-off events in the polling mode.</p> <p>It is applicable only when CanControllerActivation is set to TRUE.</p> <p>The default value is set to INTERRUPT to set all the CAN driver configuration parameter default values to be interrupt compatible.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>INTERRUPT: event is notified by the interrupt mechanism</p> <p>POLLING: event is notified when polled</p>		
Default value	INTERRUPT		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation		

5.3.1.2.2 CanControllerActivation

Table 252 Specification for CanControllerActivation

Name	CanControllerActivation		
Description	<p>Defines if a CAN controller is used in the configuration</p> <p>The default value is set to TRUE as a new controller added is automatically taken as activated. De-activated CAN controllers are a special case.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL

Can_17_McmCan driver
Table 252 Specification for CanControllerActivation (continued)

Dependency	-
-------------------	---

5.3.1.2.3 CanControllerBaseAddress
Table 253 Specification for CanControllerBaseAddress

Name	CanControllerBaseAddress		
Description	<p>Specifies the CAN controller base address.</p> <p>It is applicable only when CanControllerActivation is set to TRUE.</p> <p>The default value is set to the base address of CAN controller 0.</p> <p>The controller base address values for each controller is mentioned in the UM. In case the controller for the particular device is not present the configuration of the base address wrt the particular controller will give a configuration error.</p> <p>The selection of address (not editable by the user) for the mapping of CAN controller in the hardware to the configured CAN controller Id. The Base address of CAN controller is map with the CAN Node Registers ACCENNODExY address (Node y Access Enable Register 0) of the same controller</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	4028662016		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation		

5.3.1.2.4 CanControllerDefaultBaudrate
Table 254 Specification for CanControllerDefaultBaudrate

Name	CanControllerDefaultBaudrate		
Description	<p>Reference to baudrate configuration container configured for the CAN controller.</p> <p>It is applicable only when CanControllerActivation is set to TRUE</p>		
Multiplicity	1..1	Type	EcuReferenceDef
Range	Reference to Node: CanControllerBaudrateConfig		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 254 Specification for CanControllerDefaultBaudrate (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation		

5.3.1.2.5 CanControllerId
Table 255 Specification for CanControllerId

Name	CanControllerId		
Description	<p>Provides the controller ID, which is unique in a given CAN driver. The value for this parameter starts with 0 and continues without any gaps.</p> <p>The value 'n' depends on the number of CAN controllers supported by the hardware and is dependent on the device being used.</p> <p>It is applicable only when CanControllerActivation is set to TRUE</p> <p>The default value of CanControllerId is set to 0 representing the first index.</p> <p>Note: TC356_ADAS has 7 controllers. The user will be able to add 8 controllers in the configuration, however, the controller with the unavailable base address will trigger an error during EB tresos configuration.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - n-1		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanControllerActivation		

5.3.1.2.6 CanControllerLoopbackEnable
Table 256 Specification for CanControllerLoopbackEnable

Name	CanControllerLoopbackEnable		
Description	<p>Specifies whether the internal loop back mode is enabled or not for the controller</p> <p>This setting is applicable only when CanControllerActivation is set to TRUE</p> <p>By default, the optional interface APIs are disabled to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef

Can_17_McmCan driver
Table 256 Specification for CanControllerLoopbackEnable (continued)

Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CanControllerActivation		

5.3.1.2.7 CanCpuClockRef
Table 257 Specification for CanCpuClockRef

Name	CanCpuClockRef		
Description	Reference to the CPU clock configuration, which is set in the MCU driver configuration It is applicable only when CanControllerActivation is set to TRUE. It also depends on the McuClockReferencePoint. CanCpuClockRef configuration parameter is made as non-editable as MCMCAN driver makes use of CPU peripheral bus clock for its clock, the CPU peripheral bus clock is referenced by the container CanPeripheralBusClockRef. The configuration parameter, even though not used, shall be present in the schema to maintain the AUTOSAR schema.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuClockReferencePoint		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	McuClockReferencePoint		

5.3.1.2.8 CanPeripheralBusClockRef
Table 258 Specification for CanPeripheralBusClockRef

Name	CanPeripheralBusClockRef
-------------	--------------------------

Can_17_McmCan driver
Table 258 Specification for CanPeripheralBusClockRef (continued)

Description	Reference to the CPU peripheral bus clock configuration, which is set in the MCU driver configuration It is applicable only when CanControllerActivation is set to TRUE. It also depends on the McuClockReferencePointConfig.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockReferencePointConfig, CanControllerActivation		

5.3.1.2.9 CanRxInputSelection

Table 259 Specification for CanRxInputSelection

Name	CanRxInputSelection		
Description	Provides alternative port pin selection for receive input line It is applicable only when CanControllerActivation is set to TRUE and CanControllerLoopbackEnable is FALSE. Default value: CANxx_RXDz: Receive input line CANxx_RXDz. Where 'z' will vary depending on device variant. The default value shall be set to CANxx_RXDA as it is the first Rx input selection available for all CAN controllers.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CANxx_RXDz: Receive input line CANxx_RXDz. Where, 'z' will vary depending on the device variant The default value is set to CANxx_RXDA		
Default value	CANxx_RXDz		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CanControllerLoopbackEnable, CanControllerActivation		

Can_17_McmCan driver**5.3.1.2.10 CanRxProcessing****Table 260 Specification for CanRxProcessing**

Name	CanRxProcessing		
Description	<p>Specifies the way reception event on the controller is notified. It is applicable only when CanControllerActivation is set to TRUE.</p> <p>The default value is set to INTERRUPT to set all the CAN driver configuration parameter default values to be interrupt compatible. CanRxProcessing field to be set to INTERRUPT to handle the MIXED mode processing.</p> <p>CanRxProcessing field to be set to INTERRUPT to handle the MIXED mode processing by scheduling Can_17_McmCan_MainFunction_Read()</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>INTERRUPT: event is notified by the interrupt mechanism</p> <p>POLLING: event is notified when polled</p>		
Default value	INTERRUPT		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation		

5.3.1.2.11 CanTxProcessing**Table 261 Specification for CanTxProcessing**

Name	CanTxProcessing		
Description	<p>Specifies the way transmission event on the controller is notified.</p> <p>Enables/disables Can_17_McmCan_MainFunction_Write() API for handling the PDU reception events in the polling mode.</p> <p>It is applicable only if CanControllerActivation is set to TRUE</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>INTERRUPT: event is notified by the interrupt mechanism</p> <p>POLLING: event is notified when polled</p>		
Default value	INTERRUPT		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 261 Specification for CanTxProcessing (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation		

5.3.1.2.12 CanWakeupFunctionalityAPI
Table 262 Specification for CanWakeupFunctionalityAPI

Name	CanWakeupFunctionalityAPI		
Description	Adds/removes the Can_17_McmCan_CheckWakeup() service from the code True: Can_17_McmCan_CheckWakeup can be used False: Can_17_McmCan_CheckWakeup cannot be used It is applicable only when both CanControllerActivation and CanWakeupSupport are set to TRUE. The CanWakeupFunctionalityAPI configuration parameter is made non-editable as the CAN driver does not support wakeup over CAN bus. The configuration parameter even though not used shall be present in the schema to maintain the AUTOSAR schema.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanWakeupSupport, CanControllerActivation		

5.3.1.2.13 CanWakeupProcessing
Table 263 Specification for CanWakeupProcessing

Name	CanWakeupProcessing
Description	Specifies the way wake up event on the controller is notified. Enables/disables Can_17_McmCan_MainFunction_Wakeup() API for handling the wakeup events in the polling mode.

Can_17_McmCan driver
Table 263 Specification for CanWakeUpProcessing (continued)

	<p>It is applicable only when CanControllerActivation is set to TRUE.</p> <p>Wake up processing follows the Rx processing parameter configuration.</p> <p>The default value is set to INTERRUPT to set all the CAN driver configuration parameter default values to be interrupt compatible.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>INTERRUPT: event is notified by the interrupt mechanism</p> <p>POLLING: event is notified when polled</p>		
Default value	INTERRUPT		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanRxProcessing, CanControllerActivation		

5.3.1.2.14 CanWakeUpSourceRef
Table 264 Specification for CanWakeUpSourceRef

Name	CanWakeUpSourceRef		
Description	<p>Contains a reference to the wakeup source for this controller as defined in the ECU State Manager.</p> <p>Implementation type: reference to Ecum_WakeupSourceType</p> <p>It is applicable only when both CanControllerActivation and CanWakeUpSupport are set to TRUE.</p> <p>CanWakeUpSourceRef configuration parameter is made as non-editable as MCMCAN driver does not support wakeup over CAN bus.</p> <p>The configuration parameter, even though not used, shall be present in the schema to maintain the AUTOSAR schema.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: EcumWakeupSource		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile

Can_17_McmCan driver
Table 264 Specification for CanWakeupsourceref (continued)

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	EcumWakeupSource, CanWakeupsupport, CanControlleractivation		

5.3.1.2.15 CanWakeupsupport
Table 265 Specification for CanWakeupsupport

Name	CanWakeupsupport		
Description	Enable/disable the CAN driver support for wakeup over CAN bus. It is applicable only when CanControlleractivation is set to TRUE. By default, the optional interface APIs are disabled to minimize the executable code size. CanWakeupsupport configuration parameter is made non-editable as the MCMCAN driver does not support wakeup over CAN bus. The configuration parameter, even though not used, shall be present in the schema to maintain the AUTOSAR schema.		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.3 Container: CanControllerBaudrateConfig

This container contains bit timing related configuration parameters of the CAN controller(s).

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.3.1 CanControllerBaudRate
Table 266 Specification for CanControllerBaudRate

Name	CanControllerBaudRate
Description	Specifies the baudrate of the controller (in Kbps).

Can_17_McmCan driver
Table 266 Specification for CanControllerBaudRate (continued)

	<p>It is dependent on values of CanControllerActivation, CanPeripheralBusClockRef, CanControllerPropSeg, CanControllerSeg1, CanControllerSeg2 and CanControllerSyncJumpWidth.</p> <p>The range is limited from 40 to 1000 kbps as the MCMCAN driver hardware supports only this range of baud rate accurately.</p> <p>The default value is set to 500 kbps, as it is the most commonly used baud rate.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	40 - 1000		
Default value	500		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerSeg2, CanControllerSeg1, CanControllerPropSeg, CanControllerSyncJumpWidth, CanPeripheralBusClockRef, CanControllerActivation		

5.3.1.3.2 CanControllerBaudRateConfigID
Table 267 Specification for CanControllerBaudRateConfigID

Name	CanControllerBaudRateConfigID		
Description	<p>Uniquely identifies a specific baud rate configuration. This ID is used by the SetBaudrate API.</p> <p>It is applicable only when both CanControllerActivation and CanSetBaudrateApi are set to TRUE.</p> <p>The default value is set to 0, as it is the start ID for the first configuration.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanSetBaudrateApi, CanControllerActivation		

Can_17_McmCan driver**5.3.1.3.3 CanControllerPropSeg****Table 268 Specification for CanControllerPropSeg**

Name	CanControllerPropSeg		
Description	<p>Specifies the propagation delay in time quanta.</p> <p>Configuration rule:</p> <ul style="list-style-type: none"> - The sum of CanControllerPropSeg and CanControllerSeg1 should be within 2 (included) and 256 (included). - The sum of 1, CanControllerPropSeg, CanControllerSeg1 and CanControllerSeg2 should be within 4 (included) and 385 (included). <p>The range is limited from 1 to 255 as the MCMCAN driver hardware supports this range of propagation segment value.</p> <p>The default value is set to 47 as the value is set to obtain the most commonly used baud rate of 500 kbps.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 255		
Default value	47		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation, CanControllerSyncJumpWidth, CanControllerSeg2, CanControllerSeg1, CanControllerBaudRate, CanPeripheralBusClockRef		

5.3.1.3.4 CanControllerSeg1**Table 269 Specification for CanControllerSeg1**

Name	CanControllerSeg1		
Description	<p>Specifies phase segment 1 in time quanta.</p> <p>Configuration rule:</p> <ul style="list-style-type: none"> - The sum of CanControllerPropSeg and CanControllerSeg1 should be within 2 (included) and 256 (included). - The sum of 1, CanControllerPropSeg, CanControllerSeg1 and CanControllerSeg2 should be within 4 (included) and 385 (included). <p>The default value is set to 16 as the value is set to obtain the most commonly used baud rate of 500 kbps.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 255		
Default value	16		

Can_17_McmCan driver
Table 269 Specification for CanControllerSeg1 (continued)

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation, CanControllerSyncJumpWidth, CanControllerSeg2, CanControllerPropSeg, CanControllerBaudRate, CanPeripheralBusClockRef		

5.3.1.3.5 CanControllerSeg2

Table 270 Specification for CanControllerSeg2

Name	CanControllerSeg2		
Description	Specifies phase segment 2 in time quanta. Configuration rule: The sum of 1, CanControllerPropSeg, CanControllerSeg1 and CanControllerSeg2 should be within 4 (included) and 385 (included). The default value is set to 16 as the value is set to obtain the most commonly used baud rate of 500 kbps.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	2 - 128		
Default value	16		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation, CanControllerSyncJumpWidth, CanControllerSeg1, CanControllerBaudRate, CanControllerPropSeg, CanPeripheralBusClockRef		

5.3.1.3.6 CanControllerSyncJumpWidth

Table 271 Specification for CanControllerSyncJumpWidth

Name	CanControllerSyncJumpWidth		
Description	Specifies the synchronization jump width for the controller in time quanta. The default value is set to 4 as the value is set to obtain the most commonly used baud rate of 500 kbps.		
Multiplicity	1..1	Type	EcuIntegerParamDef

Can_17_McmCan driver
Table 271 Specification for CanControllerSyncJumpWidth (continued)

Range	1 - 128		
Default value	4		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation, CanPeripheralBusClockRef, CanControllerBaudRate, CanControllerPropSeg, CanControllerSeg1, CanControllerSeg2		

5.3.1.4 Container: CanControllerFdBaudrateConfig

This optional container contains bit timing related configuration parameters of the CAN controller(s) for payload and CRC of a CAN FD frame. If this container exists the controller supports CAN FD frames.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.4.1 CanControllerFdBaudRate

Table 272 Specification for CanControllerFdBaudRate

Name	CanControllerFdBaudRate		
Description	<p>Specifies the data segment baud rate of the controller (in kbps).</p> <p>It is dependent on the values of CanControllerActivation, CanPeripheralBusClockRef, CanControllerPropSeg, CanControllerSeg1, CanControllerSeg2 and CanControllerSyncJumpWidth.</p> <p>The range is limited from 40 to 5000 kbps as the MCMCAN driver hardware supports only this range of baud rate accurately.</p> <p>The default value is set to 2500 kbps, as it is the most commonly used baud rate.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	40 - 5000		
Default value	2500		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerSyncJumpWidth, CanControllerSeg2, CanControllerSeg1, CanControllerPropSeg, CanPeripheralBusClockRef, CanControllerActivation		

Can_17_McmCan driver**5.3.1.4.2 CanControllerPropSeg****Table 273 Specification for CanControllerPropSeg**

Name	CanControllerPropSeg		
Description	<p>Specifies the propagation delay in time quanta.</p> <p>Configuration rule:</p> <ul style="list-style-type: none"> - The sum of CanControllerPropSeg and CanControllerSeg1 should be within 1 (included) and 32 (included). - The sum of 1, CanControllerPropSeg, CanControllerSeg1 and CanControllerSeg2 should be within 4 (included) and 49 (included). <p>The default value is set to 1 as the value is set to obtain the most commonly used baud rate of 2500 kbps.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 31		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation, CanControllerSyncJumpWidth, CanControllerSeg2, CanControllerSeg1, CanControllerFdBaudRate, CanPeripheralBusClockRef		

5.3.1.4.3 CanControllerSeg1**Table 274 Specification for CanControllerSeg1**

Name	CanControllerSeg1		
Description	<p>Specifies phase segment 1 in time quanta.</p> <p>Configuration rule:</p> <ul style="list-style-type: none"> - The sum of CanControllerPropSeg and CanControllerSeg1 should be within 1 (included) and 32 (included). - The sum of 1, CanControllerPropSeg, CanControllerSeg1 and CanControllerSeg2 should be within 4 (included) and 49 (included). <p>The default value is set to 2 as the value is set to obtain the most commonly used baud rate of 2500 kbps.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 32		
Default value	2		

Can_17_McmCan driver
Table 274 Specification for CanControllerSeg1 (continued)

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation, CanControllerSyncJumpWidth, CanControllerSeg2, CanControllerPropSeg, CanControllerFdBaudRate, CanPeripheralBusClockRef		

5.3.1.4.4 CanControllerSeg2

Table 275 Specification for CanControllerSeg2

Name	CanControllerSeg2		
Description	Specifies phase segment 2 in time quanta. Configuration rule: - The sum of 1, CanControllerPropSeg, CanControllerSeg1 and CanControllerSeg2 should be within 4 (included) and 49 (included). The default value is set to 1 as the value is set to obtain the most commonly used baud rate of 2500 kbps.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 16		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation, CanControllerSyncJumpWidth, CanControllerSeg1, CanControllerPropSeg, CanControllerFdBaudRate, CanPeripheralBusClockRef		

5.3.1.4.5 CanControllerSyncJumpWidth

Table 276 Specification for CanControllerSyncJumpWidth

Name	CanControllerSyncJumpWidth		
Description	Specifies the synchronization jump width for the controller in time quanta. The default value is set to 1 as the value is set to obtain the most commonly used baud rate of 2500 kbps.		
Multiplicity	1..1	Type	EcuIntegerParamDef

Can_17_McmCan driver
Table 276 Specification for CanControllerSyncJumpWidth (continued)

Range	1 - 16		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation, CanControllerSeg1, CanControllerSeg2, CanControllerPropSeg, CanControllerFdBaudRate, CanPeripheralBusClockRef		

5.3.1.4.6 CanControllerTrcvDelayCompensationOffset
Table 277 Specification for CanControllerTrcvDelayCompensationOffset

Name	CanControllerTrcvDelayCompensationOffset		
Description	<p>Specifies the transceiver delay compensation offset (in ns). This value needs to be provided in nano seconds and not in MTQ(Minimum Time Quantas).</p> <p>By default the optional interface APIs are disabled to minimize the executable code size.</p> <p>Example:</p> <p>CAN Module clock frequency(McuMCanFrequency) = 40MHz</p> <p>- MTQ(Minimum Time Quanta) = $1/40 * 10^{-6}$ s = 0.025 us = 25ns</p> <p>CAN FD Baud Rate(CanControllerFdBaudRate) = 2MBit/s</p> <p>- FD BitTime = $1/(2 * 10^6)$ s/Bit = $0.5 * 10^{-6}$ = 500ns/Bit</p> <p>CanControllerTrcvDelayCompensationOffset = (SSP %) * FD BitTime = $0.80 * 500\text{ns} = 400\text{ ns}$</p> <p>The range of this parameter is deviated from the AUTOSAR value of 0-400 to 0-65535 (in ns) to accommodate larger values transceivers delay compensations required by different CAN FD baud rates.</p>		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation, CanPeripheralBusClockRef		

Can_17_McmCan driver

5.3.1.4.7 CanControllerTxBitRateSwitch

Table 278 Specification for CanControllerTxBitRateSwitch

Name	CanControllerTxBitRateSwitch		
Description	<p>Specifies if the bit rate switching shall be used for transmissions</p> <p>If FALSE, the CAN FD frames shall be sent without bit rate switching.</p> <p>The default value of the CanControllerTxBitRateSwitch configuration parameter is set to TRUE. CAN FD being used without bitrate switch enabled is a special case.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanControllerActivation		

5.3.1.5 Container: CanHwFilter

This container is only valid for HRHs and contains the configuration (parameters) of one hardware filter.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.5.1 CanHwFilterCode

Table 279 Specification for CanHwFilterCode

Name	CanHwFilterCode		
Description	<p>Specifies (together with the filter mask) the identifiers range that passes the hardware filter.</p> <p>The referenced hardware object with CanObjectType as RECEIVE type.</p> <p>The default value is set to 2047 as this will match all the standard ID type 11-bit identifiers.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	2047		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 279 Specification for CanHwFilterCode (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanIdType, CanObjectType		

5.3.1.5.2 CanHwFilterMask
Table 280 Specification for CanHwFilterMask

Name	CanHwFilterMask		
Description	<p>Describes a mask for the hardware-based filtering of CAN identifiers. The CAN identifiers of the incoming messages are masked with the appropriate CanFilterMask bits holding a 0, which means do not care, that is, do not compare the message identifier in the respective bit position.</p> <p>The mask should be built by filling with leading 0. In case of CanIdType EXTENDED or MIXED, a 29-bit mask should be built. In case of CanIdType STANDARD, an 11-bit mask should be built</p> <p>The default value is set to 2047 as this will mask all the standard ID type 11-bit identifiers.</p> <p>Note: The CanHwFilterMask value shall be applicable only in the case of an Rx Fifo being used (i.e. CanHwObjectCount value greater than 1), in the case of dedicated Rx filter (i.e. CanHwObjectCount value equal to 1) range filtering is not applicable and will be set to non editable.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	2047		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanHwObjectCount, CanObjectType, CanIdType		

5.3.1.6 Container: CanIcom

This container contains the parameters for configuring pretended networking

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

Can_17_McmCan driver

5.3.1.7 Container: CanIcomConfig

This container contains the configuration parameters of the ICOM configuration. It is enabled only when CanPublicIcomSupport is enabled. The upper multiplicity of the CanIcomConfig configuration parameter is limited to 255 as this is the maximum ICOM configurations supported by the CAN driver.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.7.1 CanIcomConfigId

Table 281 Specification for CanIcomConfigId

Name	CanIcomConfigId		
Description	Identifies the ID of the ICOM configuration. The default value is set to 1 as it is the start value of the config ID value.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 255		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanPublicIcomSupport		

5.3.1.7.2 CanIcomWakeOnBusOff

Table 282 Specification for CanIcomWakeOnBusOff

Name	CanIcomWakeOnBusOff		
Description	Defines that the MCU should wake if the bus-off is detected or not. The default value is set to TRUE as bus-off error detection is commonly enabled in the communication systems.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

Can_17_McmCan driver
Table 282 Specification for CanIcomWakeOnBusOff (continued)

Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanPublicIcomSupport		

5.3.1.8 Container: CanIComGeneral

This container contains the general configuration parameters of the ICOM configuration.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

5.3.1.8.1 CanIcomLevel

Table 283 Specification for CanIcomLevel

Name	CanIcomLevel		
Description	<p>Defines the level of the pretended networking.</p> <p>The default value is set to CAN_ICOM_LEVEL_ONE as the CAN driver supports only this level of pretended networking.</p> <p>The CanIcomLevel configuration parameter is made non-editable as the CAN driver only supports one ICOM-level type.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>CAN_ICOM_LEVEL_ONE: first level of pretended networking is supported</p> <p>CAN_ICOM_LEVEL_TWO: second level of pretended networking is supported</p>		
Default value	CAN_ICOM_LEVEL_ONE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanPublicIcomSupport		

5.3.1.8.2 CanIcomVariant

Table 284 Specification for CanIcomVariant

Name	CanIcomVariant		
Description	<p>Defines the variant, which is supported by this CanController.</p> <p>The default value is set to CAN_ICOM_VARIANT_SW as the CAN driver supports only software variant of ICOM.</p>		

Can_17_McmCan driver
Table 284 Specification for CanIcomVariant (continued)

	The CanIcomVariant configuration parameter is made non-editable as the CAN driver does not support variants of ICOM other than the default type mentioned.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CAN_ICOM_VARIANT_HW: pretended networking is supported only by hardware CAN_ICOM_VARIANT_NONE: pretended networking is not supported CAN_ICOM_VARIANT_SW: pretended networking is supported only by software		
Default value	CAN_ICOM_VARIANT_SW		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanPublicIcomSupport		

5.3.1.9 Container: CanIcomRxMessage

This container contains the configuration parameters for the wakeup causes for matching the received messages. It has to be configured as often as received messages are defined as wakeup cause. Constraint: For all CanIcomRxMessage instances, the message IDs which are defined in CanIcomMessageld and in CanIcomMessageldMask should not overlap.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.9.1 CanIcomCounterValue

Table 285 Specification for CanIcomCounterValue

Name	CanIcomCounterValue		
Description	Defines that the MCU should wake when the message with the ID is received 'n' times on the communication channel. The default value is set to 1 as this is the minimum value for this parameter.		
Multiplicity	0..1	Type	EcucIntegerParamDef
Range	1 - 65536		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	ECU

Can_17_McmCan driver
Table 285 Specification for CanIcomCounterValue (continued)

Dependency	CanPublicIcomSupport		
-------------------	----------------------	--	--

5.3.1.9.2 CanIcomMessageId
Table 286 Specification for CanIcomMessageId

Name	CanIcomMessageId		
Description	<p>Defines the message ID for which the wakeup causes of this CanIcomRxMessage are configured for. In addition a mask (CanIcomMessageIdMask) can be defined, in that case it is possible to define a range of Rx messages, which can create a wakeup condition.</p> <p>The default value is set to 1 as this is the minimum value for the configuration parameter.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 536870912		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanPublicIcomSupport		

5.3.1.9.3 CanIcomMessageIdMask
Table 287 Specification for CanIcomMessageIdMask

Name	CanIcomMessageIdMask		
Description	<p>Describes a mask for filtering the CAN identifiers. The CAN identifiers of incoming messages are masked with CanIcomMessageIdMask. If the masked identifier matches the masked value of CanIcomMessageId, it can create a wakeup condition for CanIcomRxMessage. Bits holding a 0 signifies do not care, that is, do not compare the message identifier in the respective bit position. The mask should be built by filling with leading 0.</p> <p>The default value is set to 0 as this is the minimum value for the configuration parameter.</p>		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	0 - 536870912		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile

Can_17_McmCan driver
Table 287 Specification for CanIcomMessageIdMask (continued)

Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanPublicIcomSupport		

5.3.1.9.4 CanIcomMissingMessageTimerValue
Table 288 Specification for CanIcomMissingMessageTimerValue

Name	CanIcomMissingMessageTimerValue		
Description	<p>Defines that the MCU should wake when the message with the configured ICOM Message ID is not received for a specific time in seconds on the communication channel.</p> <p>This parameter would be disabled for editing as MCMCAN does not support wakeup over CAN bus.</p> <p>The configuration parameter value range is limited from 1 ms to 65.535 s as per the timer ability of the micro-controller.</p> <p>The default value is set to 1 s to comply with common timer settings.</p>		
Multiplicity	0..1	Type	EcucFloatParamDef
Range	0.000001 - 65.535		
Default value	1.0		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanPublicIcomSupport		

5.3.1.9.5 CanIcomPayloadLength
Table 289 Specification for CanIcomPayloadLength

Name	CanIcomPayloadLength		
Description	Defines the payload length that should be compared with the payload length of the received message. The MCU shall wake when the message with the selected ID is having a payload length mismatch.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 8		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 289 Specification for CanIcomPayloadLength (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CanPublicIcomSupport		

5.3.1.9.6 CanIcomPayloadLengthError

Table 290 Specification for CanIcomPayloadLengthError

Name	CanIcomPayloadLengthError		
Description	Defines that the MCU should wake when a payload error occurs. If the received payload length does not match the configured payload length, this would act as a wake up condition. The default value is set to FALSE as the ICOM payload length error is a special feature of ICOM.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanPublicIcomSupport		

5.3.1.10 Container: CanIcomRxMessageSignalConfig

This container contains the configuration parameters for the wakeup causes for the matching signals. It has to be configured as often as a signal is defined as wakeup cause. When at least one Signal conditions defined in CanIcomRxMessageSignalConfig evaluates to TRUE or when no CanIcomRxMessageSignalConfig is defined, the whole wakeup condition is considered to be TRUE. All instances of this container refer to the same frame/pdu (see CanIcomMessageId).

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.10.1 CanIcomSignalMask

Table 291 Specification for CanIcomSignalMask

Name	CanIcomSignalMask
-------------	-------------------

Can_17_McmCan driver
Table 291 Specification for CanIcomSignalMask (continued)

Description	This parameter should be used to mask a signal in the payload of a CAN message. The mask is binary AND with the signal payload. The result will be used in combination of the operations defined in CanIcomSignalOperation with the CanIcomSignalValue. The configuration parameter is non-editable as the mask value is taken from the CanIcomSignalMaskUpper32bits and CanIcomSignalMaskLower32bits container, the following split of the CanIcomSignalMask configuration parameter is due to the limitation of configuration tool to support the full range of this parameter.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 9223372036854775807		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanIcomMessageldMask, CanPubliclcomSupport		

5.3.1.10.2 CanIcomSignalMaskLower32bits

Table 292 Specification for CanIcomSignalMaskLower32bits

Name	CanIcomSignalMaskLower32bits		
Description	Defines the lower 32 bit value of the parameter, which is used to mask a signal in the payload of a CAN message. The default value is set to the maximum range to accept all the messages.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 4294967295		
Default value	4294967295		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CanIcomMessageldMask, CanPubliclcomSupport		

Can_17_McmCan driver

5.3.1.10.3 CanIcomSignalMaskUpper32bits

Table 293 Specification for CanIcomSignalMaskUpper32bits

Name	CanIcomSignalMaskUpper32bits		
Description	<p>Defines the upper32 bit value of parameter, which is used to mask a signal in the payload of a CAN message.</p> <p>The default value is set to the maximum range to accept all the messages.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	4294967295		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CanIcomMessageIdMask, CanPublicIcomSupport		

5.3.1.10.4 CanIcomSignalOperation

Table 294 Specification for CanIcomSignalOperation

Name	CanIcomSignalOperation		
Description	<p>Defines the operation, which should be used to verify whether the signal value creates a wakeup condition or not.</p> <p>The default value is set to the most commonly used one-on-one message mapping of the EQUAL operation.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>AND: the received signal value masked by CanIcomSignalMask has at least one bit set in common with CanIcomSignalValue (binary AND).</p> <p>EQUAL: the received signal value masked by CanIcomSignalMask is equal to CanIcomSignalValue.</p> <p>GREATER: the received signal value masked by CanIcomSignalMask is strictly greater than CanIcomSignalValue.</p> <p>Values are interpreted as unsigned integers.</p> <p>SMALLER: the received signal value masked by CanIcomSignalMask is strictly smaller than CanIcomSignalValue.</p> <p>Values are interpreted as unsigned integers.</p> <p>XOR: the received signal value masked by CanIcomSignalMask then XORed to CanIcomSignalValue is not null.</p>		
Default value	EQUAL		

Can_17_McmCan driver
Table 294 Specification for CanIcomSignalOperation (continued)

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanIcomMessageIdMask, CanPublicIcomSupport		

5.3.1.10.5 CanIcomSignalRef

Table 295 Specification for CanIcomSignalRef

Name	CanIcomSignalRef		
Description	References to the COM layer signal that ICOM should use as a reference parameter. The CanIcomSignalRef configuration parameter is made non-editable as the McmCan driver does not support matching of the ICOM message with the messages in the upper layer. To comply with the AUTOSAR schema this configuration parameter is added but not used in the generator files.		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node:		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanIcomMessageIdMask		

5.3.1.10.6 CanIcomSignalValue

Table 296 Specification for CanIcomSignalValue

Name	CanIcomSignalValue
Description	This parameter should be used to define a signal value, which shall be compared (CanIcomSignalOperation) with the masked CanIcomSignalMask value of the received signal (CanIcomSignalRef). The configuration parameter is non-editable as the value is taken from CanIcomSignalValueUpper32bits and CanIcomSignalValueLower32bits container, the following split of the CanIcomSignalValue configuration parameter is due to the limitation of configuration tool to support the full range of this parameter.

Can_17_McmCan driver
Table 296 Specification for CanIcomSignalValue (continued)

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 9223372036854775807		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanIcomMessageIdMask, CanPublicIcomSupport		

5.3.1.10.7 CanIcomSignalValueLower32bits
Table 297 Specification for CanIcomSignalValueLower32bits

Name	CanIcomSignalValueLower32bits		
Description	Defines the lower 32 bit value of the parameter which is used to compare (CanIcomSignalOperation) with the masked CanIcomSignalMask value of the received signal (CanIcomSignalRef). The default value is set to the maximum range to accept all the messages.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	4294967295		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CanIcomMessageIdMask, CanPublicIcomSupport		

5.3.1.10.8 CanIcomSignalValueUpper32bits
Table 298 Specification for CanIcomSignalValueUpper32bits

Name	CanIcomSignalValueUpper32bits		
Description	Defines the upper32 bit value of the parameter which is used to compare (CanIcomSignalOperation) with the masked CanIcomSignalMask value of the received signal (CanIcomSignalRef). The default value is set to the maximum range to accept all the messages.		

Can_17_McmCan driver
Table 298 Specification for CanIcomSignalValueUpper32bits (continued)

Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 4294967295		
Default value	4294967295		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CanIcomMessageIdMask, CanPublicIcomSupport		

5.3.1.11 Container: CanIcomWakeupCauses

This container contains the configuration parameters of the wakeup causes to leave the power saving mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

5.3.1.12 Container: CanTTController

This container contains the configuration parameters of the TTCAN controller(s) (which are needed in addition to the configuration parameters of the CAN controller(s)) to support the TTCAN feature. The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN-related configurations are kept for following the AUTOSAR schema.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.12.1 CanTTControllerApplWatchdogLimit

Table 299 Specification for CanTTControllerApplWatchdogLimit

Name	CanTTControllerApplWatchdogLimit		
Description	<p>Defines the maximum time period (unit is 256 times NTU) after which the application has to serve the watchdog.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to the TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 299 Specification for CanTTControllerApplWatchdogLimit (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.2 CanTTControllerCycleCountMax
Table 300 Specification for CanTTControllerCycleCountMax

Name	CanTTControllerCycleCountMax		
Description	<p>Defines the value for cycle_count_max.</p> <p>Allowed values:</p> <ul style="list-style-type: none"> 0x00: 1 basic cycle 0x01: 2 basic cycles 0x03: 4 basic cycles 0x07: 8 basic cycles 0x0F: 16 basic cycles 0x1F: 32 basic cycles 0x3F: 64 basic cycles <p>The TTCAN is not supported by the CAN driver module and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 63		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.3 CanTTControllerExpectedTxTrigger
Table 301 Specification for CanTTControllerExpectedTxTrigger

Name	CanTTControllerExpectedTxTrigger
Description	Defines the number of expected_tx_trigger.

Can_17_McmCan driver
Table 301 Specification for CanTTControllerExpectedTxTrigger (continued)

	The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.4 CanTTControllerExternalClockSynchronisation
Table 302 Specification for CanTTControllerExternalClockSynchronisation

Name	CanTTControllerExternalClockSynchronisation		
Description	<p>Enables/disables the external clock synchronization.</p> <p>TRUE: external clock synchronization enabled</p> <p>FALSE: external clock synchronization disabled</p> <p>This parameter should only be configurable when the CanTTControllerLevel2 parameter is set to TRUE.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

Can_17_McmCan driver**5.3.1.12.5 CanTTControllerGlobalTimeFiltering****Table 303 Specification for CanTTControllerGlobalTimeFiltering**

Name	CanTTControllerGlobalTimeFiltering		
Description	<p>Enables/disables the global time filtering.</p> <p>TRUE: global time filtering enabled.</p> <p>FALSE: global time filtering disabled.</p> <p>This parameter should only be configurable when the CanTTControllerLevel2 parameter is set to TRUE.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.6 CanTTControllerInitialRefOffset**Table 304 Specification for CanTTControllerInitialRefOffset**

Name	CanTTControllerInitialRefOffset		
Description	<p>Defines the initial value for ref trigger offset.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 127		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

Can_17_McmCan driver
Table 304 Specification for CanTTControllerInitialRefOffset (continued)

Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.7 CanTTControllerInterruptEnable
Table 305 Specification for CanTTControllerInterruptEnable

Name	CanTTControllerInterruptEnable		
Description	<p>Enables/disables the respective interrupts.</p> <p>Bit position set to 1: enable respective interrupt.</p> <p>Bit position set to 0: disable respective interrupt.</p> <p>Bit position / interrupt source:</p> <ul style="list-style-type: none"> 10: application watchdog. 9: watch trigger reached. 8: initialization watch trigger reached. 7: change of error level. 6: Tx overflow. 5: Tx underflow. 4: global time error. 3: gap. 2: start of cycle. 1: time discontinuity. 0: master state change. <p>Bit position - 1: Time Discontinuity and - 4: Global Time Error should only be configurable when the CanTTControllerLevel2 parameter is set to TRUE.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 1023		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Can_17_McmCan driver**5.3.1.12.8 CanTTControllerLevel2****Table 306 Specification for CanTTControllerLevel2**

Name	CanTTControllerLevel2		
Description	<p>Defines whether Level 2 or Level 1 is used.</p> <p>TRUE: Level 2 FALSE: Level 1</p> <p>If the CanTTControllerLevel2 parameter is set to FALSE, all parameters with dependency to the CanTTControllerLevel2 parameter need not be configured.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity			
Range	1..1	Type	EcucBooleanParamDef
Default value		FALSE	
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.12.9 CanTTControllerNTUConfig**Table 307 Specification for CanTTControllerNTUConfig**

Name	CanTTControllerNTUConfig		
Description	<p>Defines the config value for the NTU (network time unit).</p> <p>The value is expressed in microseconds. The value configured should be greater than 0.</p> <p>Together with the local oscillator period, the TUR (time unit ratio) can be derived from the NTU. This parameter should only be configurable when the CanTTControllerLevel2 parameter is set to TRUE.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity			
Range	1..1	Type	EcucFloatParamDef
Default value		0	

Can_17_McmCan driver
Table 307 Specification for CanTTControllerNTUConfig (continued)

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.10 CanTTControllerOperationMode
Table 308 Specification for CanTTControllerOperationMode

Name	CanTTControllerOperationMode		
Description	<p>Defines the operation mode.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	CAN_TT_EVENT_SYNC_TIME_TRIGGERED: synchronous time-triggered event mode CAN_TT_EVENT_TRIGGERED: event triggered mode CAN_TT_TIME_TRIGGERED: time triggered mode		
Default value	CAN_TT_TIME_TRIGGERED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.11 CanTTControllerSyncDeviation
Table 309 Specification for CanTTControllerSyncDeviation

Name	CanTTControllerSyncDeviation		
Description	<p>Defines the maximum synchronization deviation.</p> <p>Given as a percentage value of the NTU (network time unit). The value configured should be greater than 0.</p> <p>This parameter should only be configurable when the CanTTControllerLevel2 parameter is set to TRUE.</p>		

Can_17_McmCan driver
Table 309 Specification for CanTTControllerSyncDeviation (continued)

	The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0 - 100		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.12.12 CanTTControllerTURRestore
Table 310 Specification for CanTTControllerTURRestore

Name	CanTTControllerTURRestore		
Description	<p>Enables/disables the TUR restore.</p> <p>Note that the value configured for the TUR can be derived from the value configured for the NTU and the local oscillator period.</p> <p>TRUE: TUR restore enabled FALSE: TUR restore disabled</p> <p>This parameter should only be configurable when the CanTTControllerLevel2 parameter is set to TRUE.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL

Can_17_McmCan driver**Table 310 Specification for CanTTControllerTURRestore (continued)**

Dependency	-
-------------------	---

5.3.1.12.13 CanTTControllerTimeMaster**Table 311 Specification for CanTTControllerTimeMaster**

Name	CanTTControllerTimeMaster		
Description	<p>Defines whether the controller acts as a potential time master.</p> <p>TRUE: potential time master.</p> <p>FALSE: time slave.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.14 CanTTControllerTimeMasterPriority**Table 312 Specification for CanTTControllerTimeMasterPriority**

Name	CanTTControllerTimeMasterPriority		
Description	<p>Defines the time master priority.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 312 Specification for CanTTControllerTimeMasterPriority (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.15 CanTTControllerTxEnableWindowLength
Table 313 Specification for CanTTControllerTxEnableWindowLength

Name	CanTTControllerTxEnableWindowLength		
Description	<p>Length of the Tx enable window is expressed in CAN bit times.</p> <p>The CanTTControllerTxEnableWindowlength definition parameter is used such that:</p> $\text{Length of enable window} = \text{CanTTControllerTxEnableWindowLength} + 1$ <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 16		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.12.16 CanTTControllerWatchTriggerGapTimeMark
Table 314 Specification for CanTTControllerWatchTriggerGapTimeMark

Name	CanTTControllerWatchTriggerGapTimeMark		
Description	<p>Defines the watch trigger time mark after a gap.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	0		

Can_17_McmCan driver
Table 314 Specification for CanTTControllerWatchTriggerGapTimeMark (continued)

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.12.17 CanTTControllerWatchTriggerTimeMark
Table 315 Specification for CanTTControllerWatchTriggerTimeMark

Name	CanTTControllerWatchTriggerTimeMark		
Description	<p>Defines the watch trigger time mark.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.12.18 CanTTIRQProcessing
Table 316 Specification for CanTTIRQProcessing

Name	CanTTIRQProcessing		
Description	<p>Enables/disables Can_MainFunction_BusOff() API for handling the bus-off events in the polling mode.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	INTERRUPT: event is notified by the interrupt mechanism		

Can_17_McmCan driver
Table 316 Specification for CanTTIRQProcessing (continued)

	POLLING: event is notified when polled		
Default value	INTERRUPT		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.13 Container: CanTTHardwareObjectTrigger

This container contains the configuration (parameters) of TTCAN triggers for hardware objects, which are additional to the configuration (parameters) of the CAN hardware objects. The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema..

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.13.1 CanTTHardwareObjectBaseCycle

Table 317 Specification for CanTTHardwareObjectBaseCycle

Name	CanTTHardwareObjectBaseCycle		
Description	<p>Defines the cycle_offset.</p> <p>CanTTHardwareObjectBaseCycle must be not greater than cycle_count_max.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 63		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

Can_17_McmCan driver

5.3.1.13.2 CanTTHardwareObjectCycleRepetition

Table 318 Specification for CanTTHardwareObjectCycleRepetition

Name	CanTTHardwareObjectCycleRepetition		
Description	<p>Defines the repeat_factor.</p> <p>CanTTHardwareObjectCycleRepetition should be a power of two (2), greater than cycle_offset but not greater than cycle_count_max + 1.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 64		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.13.3 CanTTHardwareObjectTimeMark

Table 319 Specification for CanTTHardwareObjectTimeMark

Name	CanTTHardwareObjectTimeMark		
Description	<p>Defines the point in time, when the trigger will be activated.</p> <p>Value is expressed in cycle time.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Can_17_McmCan driver

5.3.1.13.4 CanTTHardwareObjectTriggerId

Table 320 Specification for CanTTHardwareObjectTriggerId

Name	CanTTHardwareObjectTriggerId		
Description	<p>Sequential number which allows separation of different TTCAN triggers configured for one and the same hardware object.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 63		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.13.5 CanTTHardwareObjectTriggerType

Table 321 Specification for CanTTHardwareObjectTriggerType

Name	CanTTHardwareObjectTriggerType		
Description	<p>Defines the type of the trigger associated with the hardware object. This parameter depends on plain CAN parameter CAN_OBJECT_TYPE.</p> <p>The TTCAN is not supported by the CAN driver and, therefore, the set of configurations related to TTCAN are made non-editable and not used in the generator files. The TTCAN related configurations are kept for following the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CAN_TT_RX_TRIGGER: TT CAN with receive triggering CAN_TT_TX_REF_TRIGGER: TTCAN with reference triggered transmission CAN_TT_TX_REF_TRIGGER_GAP: TTCAN with reference triggered gap in transmission CAN_TT_TX_TRIGGER_EXCLUSIVE: TTCAN with exclusive trigger transmission CAN_TT_TX_TRIGGER_MERGED: TTCAN with merged triggered transmission CAN_TT_TX_TRIGGER_SINGLE: TTCAN with single trigger transmission		
Default value	CAN_TT_RX_TRIGGER		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 321 Specification for CanTTHardwareObjectTriggerType (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanObjectType		

5.3.1.14 Container: CommonPublishedInformation

General configuration of CAN driver common container, aggregated by all modules. It contains published information about vendor and versions.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

5.3.1.14.1 ArMajorVersion

Table 322 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	This parameter provides the major version of the AUTOSAR specification. The default value is set to 4 as the CAN driver is following the AUTOSAR version 4.2.2.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.14.2 ArMinorVersion

Table 323 Specification for ArMinorVersion

Name	ArMinorVersion		
Description	This parameter provides the minor version of the AUTOSAR specification. The default value is set to 2 as the CAN driver is following the AUTOSAR version 4.2.2.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	2		

Can_17_McmCan driver
Table 323 Specification for ArMinorVersion (continued)

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.14.3 ArPatchVersion
Table 324 Specification for ArPatchVersion

Name	ArPatchVersion		
Description	This parameter provides the patch version of the AUTOSAR specification. The default value is set to 2 as the CAN driver is following the AUTOSAR version 4.2.2.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.14.4 ModuleId
Table 325 Specification for ModuleId

Name	ModuleId		
Description	This parameter provides the module Id. The default value is set to 80 as this is the module ID of the CAN driver.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 65535		
Default value	80		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 325 Specification for ModuleId (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.14.5 Release
Table 326 Specification for Release

Name	Release		
Description	<p>This parameter indicates the TC3xx device derivative used for the implementation.</p> <p>The default value is derived from the property file and represents the hardware derivative of the micro controller for which the CAN driver is being configured.</p>		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per the hardware derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.14.6 SwMajorVersion
Table 327 Specification for SwMajorVersion

Name	SwMajorVersion		
Description	<p>This parameter provides the major version of the software.</p> <p>The default value is set to the software version that will be incremented per release of the code.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the software version		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 327 Specification for SwMajorVersion (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.14.7 SwMinorVersion
Table 328 Specification for SwMinorVersion

Name	SwMinorVersion		
Description	<p>This parameter provides the minor version of the software.</p> <p>The default value is set to the software version that will be incremented per update of the code</p>		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	As per the software version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.14.8 SwPatchVersion
Table 329 Specification for SwPatchVersion

Name	SwPatchVersion		
Description	<p>This parameter provides the patch version of the software.</p> <p>The default value is set to the software version that will be incremented per patch set of the code after release.</p>		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	As per the software version		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 329 Specification for SwPatchVersion (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.14.9 VendorApiInfix
Table 330 Specification for VendorApiInfix

Name	VendorApiInfix		
Description	<p>This parameter is used to specify the vendor specific name.</p> <p>The default value is set to McmCan as this is the unique name of the CAN driver provided by Infineon.</p>		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	McmCan		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.14.10 VendorId
Table 331 Specification for VendorId

Name	VendorId		
Description	<p>This parameter provides the vendor Id</p> <p>The default value is set to 17 as this is the Infineon vendor ID.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

Can_17_McmCan driver
Table 331 Specification for VendorId (continued)

Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.15 Container: Can

This container holds the configuration of a single CAN driver.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

5.3.1.16 Container: CanGeneral

This container contains the parameters related each CAN driver unit.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

5.3.1.16.1 CanDeInitApi
Table 332 Specification for CanDeInitApi

Name	CanDeInitApi		
Description	Switches the Can_17_McmCan_DeInit () API to ON or OFF. By default, the optional interface APIs are disabled to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

5.3.1.16.2 CanDevErrorDetection
Table 333 Specification for CanDevErrorDetection

Name	CanDevErrorDetection
Description	Switches the DET detection and notification to ON or OFF - TRUE: enabled (ON)

Can_17_McmCan driver
Table 333 Specification for CanDevErrorDetection (continued)

	- FALSE: disabled (OFF)		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.16.3 CanIndex
Table 334 Specification for CanIndex

Name	CanIndex		
Description	Specifies the InstanceId of the module instance. If only one instance is present it shall have the Id 0. The default value is set as 0 assuming there is only one instance of the CAN driver.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.16.4 CanInitDeInitApiMode
Table 335 Specification for CanInitDeInitApiMode

Name	CanInitDeInitApiMode
Description	Defines the mode in which the Init and Deinit APIs will be used.

Can_17_McmCan driver
Table 335 Specification for CanInitDeInitApiMode (continued)

	The default value of this parameter is set to Supervisor to enable maximum access rights to the registers used by the CAN driver.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CAN_17_MCMCAN_MCAL_SUPERVISOR: operating mode used is Supervisory CAN_17_MCMCAN_MCAL_USER1: operating mode used is USER-1		
Default value	CAN_17_MCMCAN_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.16.5 CanLPduReceiveCalloutFunction
Table 336 Specification for CanLPduReceiveCalloutFunction

Name	CanLPduReceiveCalloutFunction		
Description	<p>Specifies the name of a callout function that is called after a successful reception of a received CAN Rx L-PDU. If this parameter is configured with NULL_PTR, no callout will take place.</p> <p>The L-PDU callout function is mapped in a separate memory section.</p> <p>The L-PDU call out configuration parameter is set to non-editable as the CAN driver implemented is not an external CAN controller using any form of communication for interaction with the hardware.</p> <p>The default value is set to NULL_PTR as this configuration parameter is not being used.</p> <p>The configuration parameter even though not used, should be present in the schema to maintain the AUTOSAR schema.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL

Can_17_McmCan driver
Table 336 Specification for CanLPduReceiveCalloutFunction (continued)

Dependency	-
-------------------	---

5.3.1.16.6 CanMainFunctionBusoffPeriod

Table 337 Specification for CanMainFunctionBusoffPeriod

Name	CanMainFunctionBusoffPeriod		
Description	<p>Describes the period for cyclic call to Can_17_McmCan_MainFunction_Busoff. The unit is expressed in seconds.</p> <p>The default value is set to 5 ms. This is done to keep all the communication module main function periodicity to a common value.</p>		
Multiplicity	0..1	Type	EcucFloatParamDef
Range	0.001 - 65.535		
Default value	0.005		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.16.7 CanMainFunctionModePeriod

Table 338 Specification for CanMainFunctionModePeriod

Name	CanMainFunctionModePeriod		
Description	<p>Describes the period for the cyclic call to Can_17_McmCan_MainFunction_Mode. The unit is expressed in seconds.</p> <p>The default value is set to 5 ms. This is done to keep all the communication module main function periodicity to a common value.</p> <p>The Can_17_McmCan_MainFunction_Mode configuration parameter is made non-editable as the CAN driver has a synchronous mode setting mechanism and does not support the Can_17_McmCan_MainFunction_Mode() function.</p> <p>The configuration parameter, even though not used, shall be present in the schema to maintain the AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.001 - 65.535		
Default value	0.005		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 338 Specification for CanMainFunctionModePeriod (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.16.8 CanMainFunctionWakeupPeriod
Table 339 Specification for CanMainFunctionWakeupPeriod

Name	CanMainFunctionWakeupPeriod		
Description	<p>Describes the period for the cyclic call to Can_17_McmCan_MainFunction_Wakeup. Unit is expressed in seconds.</p> <p>The default value is set to 5 ms. This is done to keep all the communication module main function periodicity to a common value.</p>		
Multiplicity	0..1	Type	EcucFloatParamDef
Range	0.001 - 65.535		
Default value	0.005		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.16.9 CanMultiCoreErrorDetect
Table 340 Specification for CanMultiCoreErrorDetect

Name	CanMultiCoreErrorDetect		
Description	<p>Switches the multi-core error detection and notification to ON or OFF.</p> <ul style="list-style-type: none"> - TRUE: enabled (ON) - FALSE: disabled (OFF) <p>Note: If the CanMultiCoreErrorDetect parameter is set to TRUE with the CanDevErrorDetection parameter set to FALSE, an error is generated.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		

Can_17_McmCan driver
Table 340 Specification for CanMultiCoreErrorDetect (continued)

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CanDevErrorDetection		

5.3.1.16.10 CanMultiplexedTransmission
Table 341 Specification for CanMultiplexedTransmission

Name	CanMultiplexedTransmission		
Description	Enables/disables multiplexed transmission feature support. By default, the optional interface APIs are disabled to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECU	Scope	ECU
Dependency	-		

5.3.1.16.11 CanOsCounterRef
Table 342 Specification for CanOsCounterRef

Name	CanOsCounterRef		
Description	Contains a reference to the OsCounter, which can be used by the CAN driver. The CanOsCounterRef configuration parameter is made non-editable as the CAN driver should make use of the internal counter values. The configuration parameter, even though not used, should be present in the schema to maintain the AUTOSAR schema.		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: OsCounter		

Can_17_McmCan driver
Table 342 Specification for CanOsCounterRef (continued)

Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.16.12 CanPublicIcomSupport
Table 343 Specification for CanPublicIcomSupport

Name	CanPublicIcomSupport		
Description	Selects the support of pretended network features in the CAN driver. TRUE: enabled FALSE: disabled The CAN driver uses this parameter for enabling/disabling the pretended network feature support API Can_17_McmCan_SetIcomConfiguration (). By default, the optional interface APIs are disabled to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.16.13 CanSetBaudrateApi
Table 344 Specification for CanSetBaudrateApi

Name	CanSetBaudrateApi
Description	Used for enabling/disabling the support of Can_17_McmCan_SetBaudrate () and Can_17_McmCan_CheckBaudrate () APIs.

Can_17_McmCan driver
Table 344 Specification for CanSetBaudrateApi (continued)

	It is applicable only when both CanControllerActivation and CanSetBaudrateApi are set to TRUE. By default, the optional interface APIs are disabled to minimize the executable code size.		
Multiplicity	0..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.16.14 CanSupportTTCANRef
Table 345 Specification for CanSupportTTCANRef

Name	CanSupportTTCANRef		
Description	Refers to the CanIfSupportTTCAN parameter in the CAN interface module configuration. The CanIfSupportTTCAN parameter defines whether TTCAN is supported. The CanSupportTTCANRef configuration parameter is made non-editable as the CAN driver should not support TTCAN. The configuration parameter, even though not used, should be present in the schema to maintain the AUTOSAR schema.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: CanIfPrivateCfg		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

Can_17_McmCan driver

5.3.1.16.15 CanTimeoutDuration

Table 346 Specification for CanTimeoutDuration

Name	CanTimeoutDuration		
Description	<p>Specifies the maximum time for the blocking function until a timeout is detected. The unit is expressed in seconds.</p> <p>The default value is set to 1ms for the CanTimeoutDuration configuration parameter considering that no hardware action should take more than 1ms to execute.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.000001 - 65.535		
Default value	0.001		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.16.16 CanVersionInfoApi

Table 347 Specification for CanVersionInfoApi

Name	CanVersionInfoApi		
Description	<p>Switches the Can_17_McmCan_GetVersionInfo() API to ON or OFF.</p> <p>The default value is set as FALSE to reduce the code foot print as version information is seldom used in the development phase.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Can_17_McmCan driver

5.3.1.17 Container: CanMainFunctionRWPeriods

This container contains the parameter for configuring the period for the cyclic call to Can_17_McmCan_MainFunction_Read or Can_17_McmCan_MainFunction_Write depending on the referring item. The multiplicity range of the CanMainFunctionRWPeriods configuration parameter has been altered to 254 to keep a controllable upper limit to the number of instances.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.17.1 CanMainFunctionPeriod

Table 348 Specification for CanMainFunctionPeriod

Name	CanMainFunctionPeriod		
Description	Describes the period for the cyclic call to Can_17_McmCan_MainFunction_Read or Can_17_McmCan_MainFunction_Write depending on the referring item. The unit is expressed in seconds. The different poll-cycles will be configurable when more than one CanMainFunctionPeriod is configured. In this case, multiple Can_17_McmCan_MainFunction_Read() or Can_17_McmCan_MainFunction_Write() will be provided by the CAN driver. The default value is set to 5 ms. This is done to keep all the communication module main function periodicity to a common value.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.001 - 65.535		
Default value	0.005		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.18 Container: CanHardwareObject

This container contains the configuration (parameters) of the CAN hardware objects.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

5.3.1.18.1 CanControllerRef

Table 349 Specification for CanControllerRef

Name	CanControllerRef		
Description	Reference to the CAN controller to which the HOH (hardware object handle) is associated to		
Multiplicity	1..1	Type	EcucReferenceDef

Can_17_McmCan driver
Table 349 Specification for CanControllerRef (continued)

Range	Reference to Node: CanController		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

5.3.1.18.2 CanFdPaddingValue
Table 350 Specification for CanFdPaddingValue

Name	CanFdPaddingValue		
Description	<p>Specifies the value which is used to pad unspecified data in the CAN FD frames greater than 8 bytes for transmission. This is necessary due to the discrete possible values of the DLC (data length count) if greater than 8 bytes.</p> <p>If the length of a PDU which was requested to be sent does not match the allowed DLC values, the remaining bytes up to the next possible value should be padded with this value.</p> <p>It is applicable only when CanObjectType is of transmit type and CAN FD is enabled.</p>		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanControllerRef, CanControllerFdBaudrateConfig, CanObjectType		

5.3.1.18.3 CanHandleType
Table 351 Specification for CanHandleType

Name	CanHandleType
Description	<p>Specifies the type (FULL-CAN or BASIC-CAN) of a hardware object.</p> <p>As FULL CAN feature is most commonly used, the default value of the CanHandleType configuration parameter is set to FULL.</p>

Can_17_McmCan driver
Table 351 Specification for CanHandleType (continued)

Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	BASIC: for several L-PDUs handled by the hardware object FULL: for only one L-PDU (identifier) handled by the hardware object		
Default value	FULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.18.4 CanHwFIFOThreshold
Table 352 Specification for CanHwFIFOThreshold

Name	CanHwFIFOThreshold		
Description	Specifies the threshold size (less than or equal to the FIFO size) at which interrupt is triggered to copy the data(that is, the threshold size of Rx FIFO buffer at which the interrupt is triggered to copy the data). CanHwFIFOThreshold should be less than or equal to CanFifoSize CanObjectType should be RECEIVE type and CanHwObjectCount should be greater than 1.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 64		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CanObjectType, CanHwObjectCount		

5.3.1.18.5 CanHwObjectCount
Table 353 Specification for CanHwObjectCount

Name	CanHwObjectCount
Description	Number of the hardware objects used to implement one HOH.

Can_17_McmCan driver
Table 353 Specification for CanHwObjectCount (continued)

	In case of an HRH this parameter defines the number of elements in the hardware FIFO (for HRH objects the range is from 1 to 64). In case of a HTH it defines the number of elements in the Tx queue used for multiplexed transmission (for HTH objects the range is from 1 to 32). The maximum hardware object count is limited to 64 per controller. The limitation comes from the memory assigned per controller.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 64		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanMultiplexedTransmission, CanObjectType		

5.3.1.18.6 CanIdType
Table 354 Specification for CanIdType

Name	CanIdType		
Description	Specifies whether the CanHwFilterCode value is of following type: - standard identifier - extended identifier - mixed mode ImplementationType: Can_IdType The two most significant bits specify the frame type 00 CAN message with Standard CAN ID 01 CAN FD frame with Standard CAN ID 10 CAN message with Extended CAN ID 11 CAN FD frame with Extended CAN ID The default value of the CanIdType configuration parameter is set to STANDARD as it is the commonly used CanId.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	EXTENDED: all the CANIDs are of extended type only (29 bit). MIXED: The type of CANIDs can be both standard and extended type. STANDARD: all the CANIDs are of standard type only (11bit).		
Default value	STANDARD		

Can_17_McmCan driver
Table 354 Specification for CanIdType (continued)

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.18.7 CanMainFunctionRWPeriodRef
Table 355 Specification for CanMainFunctionRWPeriodRef

Name	CanMainFunctionRWPeriodRef		
Description	Reference to CanMainFunctionPeriod It is dependent on CanMainFunctionRWPeriods. It is applicable only when the referenced CAN controllers CanRxProcessing or CanTxProcessing or both are POLLING.		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: CanMainFunctionRWPeriods		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanObjectType, CanTxProcessing, CanRxProcessing		

5.3.1.18.8 CanObjectId
Table 356 Specification for CanObjectId

Name	CanObjectId
Description	Holds the handle ID of HRH or HTH. The value of this parameter is unique in a given CAN driver, and it should start with 0 and continue without any gaps. The HRH and HTH IDs share a common ID range. Example: HRH0-0, HRH1-1, HTH0-2, HTH1-3 Configuration rules to be followed: HRHs belonging to a controller should be grouped together HTHs belonging to a controller should be grouped together

Can_17_McmCan driver
Table 356 Specification for CanObjectId (continued)

	All HRHs should have lower CanObjectId than all HTFs Configuration example: HRHs of Controller0 is from 0 to 4 HRHs of Controller1 is from 5 to 9 HRHs of Controller2 is from 10 to 14 HRHs of Controller3 is from 15 to 19 HTFs of Controller0 is from 20 to 24 HTFs of Controller1 is from 25 to 29 HTFs of Controller2 is from 30 to 34 HTFs of Controller3 is from 35 to 39 Note: 'N' is the maximum number of hardware objects that can be configured and depends on the hardware device being used.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - N-1		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.18.9 CanObjectType
Table 357 Specification for CanObjectType

Name	CanObjectType		
Description	Specifies if the HardwareObject is used as a transmit or receive object The default value is set to RECEIVE because when configuring hardware objects, first the RECEIVE objects should be configured followed by the TRANSMIT objects.		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	RECEIVE: Receive HOH TRANSMIT: Transmit HOH		
Default value	RECEIVE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Can_17_McmCan driver
Table 357 Specification for CanObjectType (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

5.3.1.18.10 CanTriggerTransmitEnable
Table 358 Specification for CanTriggerTransmitEnable

Name	CanTriggerTransmitEnable		
Description	Defines whether or not the CAN supports the trigger-transmit API for this handle. By default, the optional interface APIs are disabled to minimize the executable code size.		
Multiplicity	0..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanObjectType		

5.3.2 Functions - Type definitions
5.3.2.1 CanTrcv_TrcvModeType
Table 359 Specification for CanTrcv_TrcvModeType

Syntax	CanTrcv_TrcvModeType	
Type	Enumeration	
File	Can_GeneralTypes.h	
Range	0 - CANTRCV_TRCVMODE_NORMAL	Transceiver mode Normal
	1 - CANTRCV_TRCVMODE_SLEEP	Transceiver mode Sleep
	2 - CANTRCV_TRCVMODE_STANDBY	Transceiver mode StandBy
Description	Operating modes of the CAN transceiver driver.	

Can_17_McmCan driver
Table 359 Specification for CanTrcv_TrcvModeType (continued)

Source	AUTOSAR
---------------	---------

5.3.2.2 **CanTrcv_TrcvWakeupModeType**

Table 360 Specification for CanTrcv_TrcvWakeupModeType

Syntax	CanTrcv_TrcvWakeupModeType	
Type	Enumeration	
File	Can_GeneralTypes.h	
Range	0 - CANTRCV_WUMODE_ENABLE	The notification for wakeup events are enabled on the addressed transceiver.
	1 - CANTRCV_WUMODE_DISABLE	The notification for wakeup events are disabled on the addressed transceiver.
	2 - CANTRCV_WUMODE_CLEAR	The stored notification events are cleared on the addressed transceiver.
Description	This type should be used to control the CanTrcv concerning the wakeup events and wakeup notifications.	
Source	AUTOSAR	

5.3.2.3 **CanTrcv_TrcvWakeupReasonType**

Table 361 Specification for CanTrcv_TrcvWakeupReasonType

Syntax	CanTrcv_TrcvWakeupReasonType	
Type	Enumeration	
File	Can_GeneralTypes.h	
Range	0 - CANTRCV_WU_ERROR	Due to an error wake up reason is not detected. This value may only be reported when the error is reported to the DEM before.
	1 - CANTRCV_WU_NOT_SUPPORTED	The transceiver does not support any information for the wake up reason.
	2 - CANTRCV_WU_BY_BUS	The transceiver has detected that the network has caused the wake up of the ECU.
	3 - CANTRCV_WU INTERNALLY	The transceiver has detected that the network has been woken up by the ECU through a request to the NORMAL mode.
	4 - CANTRCV_WU_RESET	The transceiver has detected, that the wakeup is due to an ECU reset.

Can_17_McmCan driver
Table 361 Specification for CanTrcv_TrsvWakeupReasonType (continued)

	5 - CANTRCV_WU_POWER_ON	The transceiver has detected, that the wakeup is due to an ECU reset after power on.
	6 - CANTRCV_WU_BY_PIN	The transceiver has detected, that the wakeup is due to a state held at the pin.
	7 - CANTRCV_WU_BY_SYSERR	The transceiver has detected, that the wake up of the ECU was caused by a hardware related device failure.
Description	This type denotes the wake up reason detected by the CanTrcv in detail.	
Source	AUTOSAR	

5.3.2.4 Can_HwHandleType
Table 362 Specification for Can_HwHandleType

Syntax	Can_HwHandleType	
Type	uint16	
File	Can_GeneralTypes.h	
Range	0x00 - 0xFFFF	By default, extended type is defined
Description	Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 hardware objects, uses the extended range. The CAN driver uses uint16 as the module hardware supports more 255 hardware objects.	
Source	AUTOSAR	

5.3.2.5 Can_HwType
Table 363 Specification for Can_HwType

Syntax	Can_HwType	
Type	Structure	
File	Can_GeneralTypes.h	
Range	Can_IdType CanId	Standard/Extended CAN ID of CAN L-PDU
	Can_HwHandleType Hoh	ID of the corresponding Hardware Object Range
	uint8 ControllerId	ControllerId provided by CanIf clearly identify the corresponding controller
Description	This type defines a data structure which clearly provides a hardware object handle including its corresponding CAN controller and therefore CanDrv as well as the specific CanId.	
Source	AUTOSAR	

Can_17_McmCan driver**5.3.2.6 Can_17_McmCan_ConfigType****Table 364 Specification for Can_17_McmCan_ConfigType**

Syntax	Can_17_McmCan_ConfigType	
Type	Structure	
File	Can_17_McmCan.h	
Range	--	The elements of the data structure are specific to the micro-controller
Description	This is the type of the external data structure containing the overall initialization data for the CAN driver and SFR settings affecting all controllers. Furthermore it contains pointers to controller configuration structures. That is it contains the definition of the implementation-specific post build configuration structure of the CAN driver.	
Source	AUTOSAR	

5.3.2.7 Can_PduType**Table 365 Specification for Can_PduType**

Syntax	Can_PduType	
Type	Structure	
File	Can_GeneralTypes.h	
Range	PduldType swPduHandle	Software PDU handle
	uint8 length	Number of SDU data bytes
	Can_IdType id	Formatted CAN message identifier
	uint8 * sdu	Pointer to data bytes
Description	This type unites Pduld (swPduHandle), SduLength (length), SduData (sdu), and CanId (id) for any CAN L-SDU.	
Source	AUTOSAR	

5.3.2.8 Can_IdType**Table 366 Specification for Can_IdType**

Syntax	Can_IdType	
Type	uint32	
File	Can_GeneralTypes.h	
Range	0x00- 0xFFFFFFFF	By default, extended 32-bit is defined
Description	Represents the identifier of an L-PDU. The two most significant bits specify the frame type: 00 CAN message with Standard CAN ID 01 CAN FD frame with Standard CAN ID 10 CAN message with Extended CAN ID 11 CAN FD frame with Extended CAN ID The type can be either uint16 or uint32 (type can be uint16 when all HOH's are of STANDARD type otherwise the type should be uint32). The CAN driver should support both uint16	

Can_17_McmCan driver
Table 366 Specification for Can_IdType (continued)

	and uint32. Standard32Bit - 0 to 0x400007FF Standard16Bit - 0 to 0x47FF Extended32Bit - 0 to 0xFFFFFFFF
Source	AUTOSAR

5.3.2.9 Can_StateTransitionType
Table 367 Specification for Can_StateTransitionType

Syntax	Can_StateTransitionType	
Type	Enumeration	
File	Can_GeneralTypes.h	
Range	0 - CAN_T_START	CAN controller transition value to request state STARTED.
	1 - CAN_T_STOP	CAN controller transition value to request state STOPPED.
	2 - CAN_T_SLEEP	CAN controller transition value to request state SLEEP.
	3 - CAN_T_WAKEUP	CAN controller transition value to request state STOPPED from state SLEEP.
Description	State transitions that are used by the CAN_SetControllerMode function.	
Source	AUTOSAR	

5.3.2.10 Can_ReturnType
Table 368 Specification for Can_ReturnType

Syntax	Can_ReturnType	
Type	Enumeration	
File	Can_GeneralTypes.h	
Range	0 - CAN_OK	Success
	1 - CAN_NOT_OK	Error or wakeup event occurred during sleep transition
	2 - CAN_BUSY	Transmit request could not be processed because no transmit object was available
Description	Represents the return values of the CAN driver APIs.	
Source	AUTOSAR	

Can_17_McmCan driver**5.3.3 Functions - APIs****5.3.3.1 Can_17_McmCan_CheckBaudrate****Table 369 Specification for Can_17_McmCan_CheckBaudrate API**

Syntax	<pre>Std_ReturnType Can_17_McmCan_CheckBaudrate (const uint8 Controller, const uint16 Baudrate)</pre>	
Service ID	0	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	Controller Baudrate	Associated CAN controller Baudrate to be checked
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: service request accepted, checking of baud rate started. E_NOT_OK: service request not accepted
Description	This function checks the baud rate of the CAN controller.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_PARAM_CONTROLLER: API service is called with the parameter controller which is out of the range</p> <p>CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>CAN_17_MCMCAN_E_PARAM_BAUDRATE: API service is called with the parameter</p> <p>CAN_17_MCMCAN_E_NOT_CONFIGURED: resource is not configured to the calling core</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	CanSetBaudrateApi	
User hints	None	

Can_17_McmCan driver**5.3.3.2 Can_17_McmCan_Init****Table 370 Specification for Can_17_McmCan_Init API**

Syntax	<pre>void Can_17_McmCan_Init (const Can_17_McmCan_ConfigType * const Config)</pre>	
Service ID	0	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Config	Pointer to the CAN driver root configuration
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>CAN driver module initialization function.</p> <p>The controllers initialized shall be configured to reject reception of CAN frames with remote transmission requests (i.e. Frames with RTR bit set)</p> <p>Can_17_McmCan_Init() should be called from the slave core only after the successful initialization of the master core by calling function Can_17_McmCan_Init().</p> <p>The clock and kernel specific common resource initialization of the CAN driver module shall take place during Master core initialization. During the slave core initialization the core specific variable and register initialization takes place.</p> <p><i>Note:</i> If Can_17_McmCan_Init is invoked after the CAN driver has already been initialized and the DETs are enabled then the driver will report CAN_17_MCMCAN_E_TRANSITION DET. Hence the user shall invoke the Can_17_McmCan_DelInit API before calling again Can_17_McmCan_Init API.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_MASTER_CORE_UNINIT: Master core is in uninitialized state when the Can_17_McmCan_Init() API is called in the slave core</p> <p>CAN_17_MCMCAN_E_NOT_CONFIGURED: resource is not configured to the calling core</p> <p>CAN_17_MCMCAN_E_TRANSITION: API service is called with invalid transition or in an unintended state.</p> <p>CAN_17_MCMCAN_E_INIT_FAILED: Can_17_McmCan_Init() API service is called with an invalid configuration set selection.</p> <p>Runtime Errors: None</p> <p>DEM: None</p>	

Can_17_McmCan driver
Table 370 Specification for Can_17_McmCan_Init API (continued)

	Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

5.3.3.3 Can_17_McmCan_GetVersionInfo

Table 371 Specification for Can_17_McmCan_GetVersionInfo API

Syntax	void Can_17_McmCan_GetVersionInfo (Std_VersionInfoType * const versioninfo)	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to the location to store the version information of this module.
Parameters (in - out)	-	-
Return	void	-
Description	This function provides the version information of the CAN driver The Can_17_McmCan_GetVersionInfo() function is available only when, CanVersionInfoApi is enabled.	
Source	AUTOSAR	
Error handling	DET: CAN_17_MCMCAN_E_PARAM_POINTER: API service is called with a NULL pointer as its parameter. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	CanVersionInfoApi	
User hints	None	

Can_17_McmCan driver**5.3.3.4 Can_17_McmCan_SetBaudrate****Table 372 Specification for Can_17_McmCan_SetBaudrate API**

Syntax	<pre>Std_ReturnType Can_17_McmCan_SetBaudrate (const uint8 Controller, const uint16 BaudRateConfigID)</pre>	
Service ID	0x0f	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different controllers. Non-reentrant for the same controller.	
Parameters (in)	Controller BaudRateConfigID	CAN controller for which the, baud rate needs to be set Unique Id with a specific baud rate configuration
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: service request accepted, setting of (new) baud rate started E_NOT_OK: service request not accepted
Description	<p>This service sets the baud rate configuration of the CAN controller.</p> <p>The Can_17_McmCan_SetBaudrate() function is available only when, CanSetBaudrateApi is enabled.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_PARAM_CONTROLLER: API service is called with the parameter controller which is out of the range</p> <p>CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>CAN_17_MCMCAN_E_PARAM_BAUDRATE: API service is called with the parameter</p> <p>CAN_17_MCMCAN_E_NOT_CONFIGURED: resource is not configured to the calling core</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	CanControllerActivation,CanSetBaudrateApi	
User hints	None	

Can_17_McmCan driver**5.3.3.5 Can_17_McmCan_SetControllerMode****Table 373 Specification for Can_17_McmCan_SetControllerMode API**

Syntax	<pre>Can_ReturnType Can_17_McmCan_SetControllerMode (const uint8 Controller, const Can_StateTransitionType Transition)</pre>	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Controller Transition	CAN controller for which the mode has to be changed Transition value to request new CAN controller state
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Can_ReturnType	CAN_OK: request accepted CAN_NOT_OK: request not accepted, a development error occurred
Description	<p>This function performs software triggered state transitions of the CAN controller state machine.</p> <p>The function is implemented synchronous as the change in the mode is done synchronously by the hardware.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_TRANSITION: API service is called with invalid transition or in an unintended state.</p> <p>CAN_17_MCMCAN_E_PARAM_CONTROLLER: API service is called with the parameter controller which is out of the range</p> <p>CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>CAN_17_MCMCAN_E_NOT_CONFIGURED: resource is not configured to the calling core</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

Can_17_McmCan driver**5.3.3.6 Can_17_McmCan_DisableControllerInterrupts****Table 374 Specification for Can_17_McmCan_DisableControllerInterrupts API**

Syntax	<pre>void Can_17_McmCan_DisableControllerInterrupts (const uint8 Controller)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different controllers.	
Parameters (in)	Controller	CAN controller for which interrupts need to be disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function disables all interrupts for the given CAN controller	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>CAN_17_MCMCAN_E_NOT_CONFIGURED: resource is not configured to the calling core</p> <p>CAN_17_MCMCAN_E_PARAM_CONTROLLER: API service is called with the parameter controller which is out of the range</p> <p>CAN_17_MCMCAN_E_PARAM_POINTER: API service is called with a NULL pointer as its parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

5.3.3.7 Can_17_McmCan_EnableControllerInterrupts**Table 375 Specification for Can_17_McmCan_EnableControllerInterrupts API**

Syntax	<pre>void Can_17_McmCan_EnableControllerInterrupts (</pre>
---------------	--

Can_17_McmCan driver**Table 375 Specification for Can_17_McmCan_EnableControllerInterrupts API (continued)**

	const uint8 Controller)	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different controllers.	
Parameters (in)	Controller	CAN controller for which interrupts shall be re-enabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This functions re-enables the allowed interrupts of the given CAN controller	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_PARAM_CONTROLLER: API service is called with the parameter controller which is out of the range</p> <p>CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>CAN_17_MCMCAN_E_NOT_CONFIGURED: resource is not configured to the calling core</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

5.3.3.8 Can_17_McmCan_Write**Table 376 Specification for Can_17_McmCan_Write API**

Syntax	Can_ReturnType Can_17_McmCan_Write (const Can_HwHandleType Hth, const Can_PduType * const PduInfo)
Service ID	0x06
Sync/Async	Synchronous
ASIL Level	QM

Can_17_McmCan driver
Table 376 Specification for Can_17_McmCan_Write API (continued)

Re-entrancy	Reentrant and thread safe.	
Parameters (in)	Hth PduInfo	Information which hardware transmit handle should be used for transmit. Implicitly this is also the information about the controller to use because the Hth numbers are unique inside a hardware unit. Pointer to the SDU user memory, DLC and Identifier
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Can_ReturnType	CAN_OK: write command has been accepted CAN_NOT_OK: development error occurred CAN_BUSY: no Tx hardware buffer available
Description	This function is used to transmit CAN/CAN FD frame based on the information passed to it. The CAN driver will only transmit messages with remote transmission request (RTR) bit at reset state (that is, no remote transmission request will be accepted by the CAN driver).	
Source	AUTOSAR	
Error handling	DET: CAN_17_MCMCAN_E_PARAM_POINTER: API service is called with a NULL pointer as its parameter. CAN_17_MCMCAN_E_PARAM_DLC: Can_17_McmCan_Write () API service is called with an inappropriate length as its parameter. CAN_17_MCMCAN_E_PARAM_HANDLE: Can_17_McmCan_Write() API service is called with the Hth parameter, which is not a configured hardware transmit handle. CAN_17_MCMCAN_E_PARAM_MSGID: API service is called with an inappropriate CAN message identifier as its parameter. CAN_17_MCMCAN_E_UNINIT: API service is used without initialization CAN_17_MCMCAN_E_NOT_CONFIGURED: resource is not configured to the calling core Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

Can_17_McmCan driver**5.3.3.9 Can_17_McmCan_SetIcomConfiguration****Table 377 Specification for Can_17_McmCan_SetIcomConfiguration API**

Syntax	<pre>Std_ReturnType Can_17_McmCan_SetIcomConfiguration (const uint8 Controller, const IcomConfigIdType ConfigurationId)</pre>	
Service ID	0xf	
Sync/Async	Asynchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different Controllers. Non reentrant for the same Controller.	
Parameters (in)	Controller ConfigurationId	CAN controller for which the status shall be changed. Requested configuration. An ID greater than 0 identifies a configuration with which pretended networking is activated for the Controller. An ID value of 0 deactivates the Pretended Networking identifier that is activated for the Controller.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: CAN driver succeeded in setting a configuration with a valid Configuration id. E_NOT_OK: CAN driver failed to set a configuration with a valid Configuration id.
Description	This service should change the Icom configuration of a CAN controller to the requested one. The Can_17_McmCan_SetIcomConfiguration() function is available only when, CanPublicIcomSupport is enabled.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_ICOM_CONFIG_INVALID: Can_17_McmCan_SetIcomConfiguration () API service is called with an invalid ICOM configuration ID</p> <p>CAN_17_MCMCAN_E_PARAM_CONTROLLER: API service is called with the parameter controller which is out of the range</p> <p>CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>CAN_17_MCMCAN_E_NOT_CONFIGURED: resource is not configured to the calling core</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

Can_17_McmCan driver
Table 377 Specification for Can_17_McmCan_SetIcomConfiguration API (continued)

Configuration dependencies	CanPublicIcomSupport
User hints	None

5.3.3.10 Can_17_McmCan_DeInit
Table 378 Specification for Can_17_McmCan_DeInit API

Syntax	void Can_17_McmCan_DeInit (void)	
Service ID	0x65	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This function de-initializes the CAN driver.</p> <p>If the function is called without the initialization of CAN driver while the DET is ON, development error CAN_E_UNINIT is reported.</p> <p>The Can_17_McmCan_DeInit() function is available only when, CanDeInitApi is enabled.</p> <p>The Can_17_McmCan_DeInit() API should be called from the master core only after the Can_17_McmCan_Deinit() API is successfully called for all the CAN-initialized slave cores.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>CAN_17_MCMCAN_E_SLAVE_CORE_INIT: The Can_17_McmCan_Deinit() API is called by the master core before de-initialization of the slave cores is not yet completed</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	CanDeInitApi	

Can_17_McmCan driver
Table 378 Specification for Can_17_McmCan_DeInit API (continued)

User hints	None
------------	------

5.3.4 Notifications and callbacks

The CAN driver does not implement any notifications or callbacks.

5.3.5 Scheduled functions

The CAN driver supports the following scheduled functions

5.3.5.1 Can_17_McmCan_MainFunction_Write

Table 379 Specification for Can_17_McmCan_MainFunction_Write API

Syntax	void Can_17_McmCan_MainFunction_Write (void)	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function performs the polling of transmit confirmation when CanTxProcessing is set to POLLING. The function will exist when elements in the parameter list CanMainFunctionRWPeriods is 0 or 1.	
Source	AUTOSAR	
Error handling	DET: CAN_17_MCMCAN_E_UNINIT: API service is used without initialization CAN_17_MCMCAN_E_DATALOST: When received CAN message in FIFO or tx event is lost Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

Can_17_McmCan driver
Table 379 Specification for Can_17_McmCan_MainFunction_Write API (continued)

Configuration dependencies	CanMainFunctionRWPeriods
User hints	None

5.3.5.2 Can_17_McmCan_MainFunction_Write_(x)

Table 380 Specification for Can_17_McmCan_MainFunction_Write_(x) API

Syntax	void Can_17_McmCan_MainFunction_Write_(x) (void)	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This function performs the polling of transmit confirmation when CanTxProcessing is set to POLLING.</p> <p>The function name shall be appended with _x, when the number of elements in the parameter list CanMainFunctionRWPeriods is greater than 1.</p> <p>For example: Elements in the CanMainFunctionRWPeriods parameter list are two (that is, greater than 1), then the following two functions are generated: Can_17_McmCan_MainFunction_Write_0 and Can_17_McmCan_MainFunction_Write_1. These functions poll for the hth configured for their respective periods.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>CAN_17_MCMCAN_E_DATALOST: When received CAN message in FIFO or tx event is lost</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

Can_17_McmCan driver
Table 380 Specification for Can_17_McmCan_MainFunction_Write_(x) API (continued)

Configuration dependencies	CanMainFunctionRWPeriods
User hints	-

5.3.5.3 Can_17_McmCan_MainFunction_Read

Table 381 Specification for Can_17_McmCan_MainFunction_Read API

Syntax	void Can_17_McmCan_MainFunction_Read (void)	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function is called in periodic raster to read the received messages in dedicated and FIFO. The RX processing starts when the threshold value of FIFO reaches watermark.	
Source	AUTOSAR	
Error handling	DET: CAN_17_MCMCAN_E_DATALOST: When received CAN message in FIFO or tx event is lost CAN_17_MCMCAN_E_UNINIT: API service is used without initialization Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	CanMainFunctionRWPeriods	
User hints	None	

Can_17_McmCan driver**5.3.5.4 Can_17_McmCan_MainFunction_Read_(x)****Table 382 Specification for Can_17_McmCan_MainFunction_Read_(x) API**

Syntax	void Can_17_McmCan_MainFunction_Read_(x) (void)	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This function performs the polling of receive indication when CanRxProcessing is set to POLLING.</p> <p>The function name shall be appended with _x, when the number of elements in the parameter list CanMainFunctionRWPeriods is greater than 1.</p> <p>e.g.: Elements in the parameter list CanMainFunctionRWPeriods is 2 (i.e. greater than 1), then two functions will be generated namely: Can_17_McmCan_MainFunction_Read_0 and Can_17_McmCan_MainFunction_Read_1 these functions will poll for the hrh configured for their respective periods.</p> <p>Multi-period main function does not support MIXED mode.</p> <p>The RX processing starts when the threshold value of FIFO reaches watermark.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_DATALOST: When received CAN message in FIFO or tx event is lost</p> <p>CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	CanMainFunctionRWPeriods	
User hints	None	

Can_17_McmCan driver**5.3.5.5 Can_17_McmCan_MainFunction_BusOff****Table 383 Specification for Can_17_McmCan_MainFunction_BusOff API**

Syntax	void Can_17_McmCan_MainFunction_BusOff (void)	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This function performs the polling of bus-off events that are configured statically as 'to be polled'.</p> <p>Bus-off notification will be provided to the upper layer only once when the hardware detects a bus-off. If the bus-off is prevalent even after the first notification, no further notifications will be provided to the upper layer.</p>	
Source	AUTOSAR	
Error handling	<p>DET: CAN_17_MCMCAN_E_UNINIT: API service is used without initialization</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

5.3.5.6 Can_17_McmCan_MainFunction_Wakeup**Table 384 Specification for Can_17_McmCan_MainFunction_Wakeup API**

Syntax	void Can_17_McmCan_MainFunction_Wakeup (void)
---------------	--

Can_17_McmCan driver
Table 384 Specification for Can_17_McmCan_MainFunction_Wakeup API (continued)

Service ID	0x0a	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function performs the polling of wake-up events that are configured statically as 'to be polled'.	
Source	AUTOSAR	
Error handling	DET: CAN_17_MCMCAN_E_UNINIT: API service is used without initialization Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

5.3.5.7 Can_17_McmCan_MainFunction_Mode

Table 385 Specification for Can_17_McmCan_MainFunction_Mode API

Syntax	void Can_17_McmCan_MainFunction_Mode (void)	
Service ID	0x0c	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-

Can_17_McmCan driver
Table 385 Specification for Can_17_McmCan_MainFunction_Mode API (continued)

Parameters (in - out)	-	-
Return	void	-
Description	This function is supposed to poll for the CAN controller mode transitions. Since in MCMCAN all the state transitions happen in a synchronous way, this function is implemented as an empty function.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

5.3.6 Interrupt service routines

5.3.6.1 Can_17_McmCan_IsrBusOffHandler

Table 386 Specification for Can_17_McmCan_IsrBusOffHandler API

Syntax	<pre>void Can_17_McmCan_IsrBusOffHandler (const uint8 HwKernelId, const uint8 NodeIdIndex)</pre>	
Service ID	-	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	HwKernelId NodeIdIndex	The CAN controller which is to be processed, is associated with the passed Kernel The CAN node which is to be processed
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

Can_17_McmCan driver
Table 386 Specification for Can_17_McmCan_IsrBusOffHandler API (continued)

Description	This function checks the occurrence of bus-off events on the given CAN controller and gives corresponding notification to the upper layer. This function resets the controller state to the stop state The Can_17_McmCan_IsrBusOffHandler() handler is available only when, CanBusoffProcessing is enabled
Source	IFX
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	CanBusoffProcessing
User hints	None

5.3.6.2 Can_17_McmCan_IsrReceiveHandler

Table 387 Specification for Can_17_McmCan_IsrReceiveHandler API

Syntax	<pre>void Can_17_McmCan_IsrReceiveHandler (const uint8 HwKernelId, const uint8 NodeIdIndex)</pre>	
Service ID	-	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	HwKernelId NodeIdIndex	The CAN controller which is to be processed, is associated with the passed Kernel The CAN node which is to be processed
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function should handle receive interrupts from dedicated receive buffers during CAN controller STARTED state.	

Can_17_McmCan driver
Table 387 Specification for Can_17_McmCan_IsrReceiveHandler API (continued)

	For dedicated reception the hardware filter code alone is considered, the receive mask available shall not be used during the filtering or processing of the message.
Source	IFX
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	CanRxProcessing
User hints	None

5.3.6.3 Can_17_McmCan_IsrRxFIFOHandler
Table 388 Specification for Can_17_McmCan_IsrRxFIFOHandler API

Syntax	<pre>void Can_17_McmCan_IsrRxFIFOHandler (const uint8 HwKernelId, const uint8 NodeIdIndex)</pre>	
Service ID	-	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	HwKernelId NodeIdIndex	The CAN controller which is to be processed, is associated with the passed Kernel The CAN node which is to be processed
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function shall handle receive interrupts from FIFO 0 and FIFO 1 during CAN controller STARTED state. The ISR is triggered for FIFO0 or for FIFO 1 on Watermark or on FIFO full event. Messages are read through FIFO and freed by acknowledging the slot to receive successive packet. Rx FIFO interrupt processes maximum of configured FIFO elements and if messages are received while the Rx FIFO messages are in progress and if number of messages received is greater	

Can_17_McmCan driver
Table 388 Specification for Can_17_McmCan_IsrRxFIFOHandler API (continued)

	<p>than the configured threshold level on exit of interrupt handler watermark interrupt will not be triggered therefore all messages will be processed only on FULL interrupt.</p> <p>If FIFO overflow is set, a DET with CAN_17_MCMCAN_E_DATALOST is raised to indicate that few messages may be lost.</p>
Source	IFX
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_DATALOST: When received CAN message in FIFO or tx event is lost</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	CanRxProcessing
User hints	None

5.3.6.4 Can_17_McmCan_IsrTransmitHandler
Table 389 Specification for Can_17_McmCan_IsrTransmitHandler API

Syntax	<pre>void Can_17_McmCan_IsrTransmitHandler (const uint8 HwKernelId, const uint8 NodeIdIndex)</pre>	
Service ID	-	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	HwKernelId NodeIdIndex	The CAN controller which is to be processed, is associated with the passed Kernel The CAN node which is to be processed
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Identifies the message object belonging to the given CAN controller for which the transmission request was successful, extracts its corresponding software PDU handle, and gives notification to upper layer.	

Can_17_McmCan driver
Table 389 Specification for Can_17_McmCan_IsrTransmitHandler API (continued)

	The Can_17_McmCan_IsrTransmitHandler() handler is available only when, CanTxProcessing is enabled
Source	IFX
Error handling	<p>DET:</p> <p>CAN_17_MCMCAN_E_DATALOST: When received CAN message in FIFO or tx event is lost</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	CanTxProcessing
User hints	None

5.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

5.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Table 390 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
API service is called with a NULL pointer as its parameter.	AUTOSAR	CAN_17_MCMCAN_E_PARAM_POINT ER=0x01	Can_17_McmCan_GetVersionInfo, Can_17_McmCan_DisableControllerInterrupts, Can_17_McmCan_Write
Can_17_McmCan_Write() API service is called with the Hth parameter, which is not a configured hardware transmit handle.	AUTOSAR	CAN_17_MCMCAN_E_PARAM_HANDLE=0x02	Can_17_McmCan_Write
Can_17_McmCan_Write() API service is called with an inappropriate length as its parameter.	AUTOSAR	CAN_17_MCMCAN_E_PARAM_DLC=0x03	Can_17_McmCan_Write
API service is called with the parameter controller which is out of the range	AUTOSAR	CAN_17_MCMCAN_E_PARAM_CONTROLLER=0x04	Can_17_McmCan_DisableControllerInterrupts, Can_17_McmCan_SetComConfiguration, Can_17_McmCan_SetControllerMode,

Can_17_McmCan driver
Table 390 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
			Can_17_McmCan_SetBaudrate, Can_17_McmCan_CheckBaudrate, Can_17_McmCan_EnableControllerInterrupts
API service is used without initialization	AUTOSAR	CAN_17_MCMCAN_E_UNINIT=0x05	Can_17_McmCan_MainFunction_Read, Can_17_McmCan_MainFunction_Write_(x), Can_17_McmCan_MainFunction_BusOff, Can_17_McmCan_MainFunction_Write, Can_17_McmCan_SetIcomConfiguration, Can_17_McmCan_SetControllerMode, Can_17_McmCan_SetBaudrate, Can_17_McmCan_CheckBaudrate, Can_17_McmCan_MainFunction_Wakeup, Can_17_McmCan_MainFunction_Read_(x), Can_17_McmCan_Delnit, Can_17_McmCan_Write, Can_17_McmCan_EnableControllerInterrupts, Can_17_McmCan_DisableControllerInterrupts
API service is called with invalid transition or in an unintended state.	AUTOSAR	CAN_17_MCMCAN_E_TRANSITION=0x06	Can_17_McmCan_Init, Can_17_McmCan_SetControllerMode
When received CAN message in FIFO or tx event is lost.	AUTOSAR	CAN_17_MCMCAN_E_DATALOST=0x07	Can_17_McmCan_MainFunction_Write_(x), Can_17_McmCan_MainFunction_Write, Can_17_McmCan_IsrTransmitHandler, Can_17_McmCan_MainFunction_Read, Can_17_McmCan_IsrRxFIFOHandler, Can_17_McmCan_MainFunction_Read_(x)

Can_17_McmCan driver
Table 390 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
API service is called with the parameter.	AUTOSAR	CAN_17_MCMCAN_E_PARAM_BAUDRATE=0x08	Can_17_McmCan_SetBaudrate, Can_17_McmCan_CheckBaudrate
Can_17_McmCan_SetIcom Configuration () API service is called with an invalid ICOM configuration ID.	AUTOSAR	CAN_17_MCMCAN_E_ICOM_CONFIG_INVALID=0x09	Can_17_McmCan_SetIcomConfiguration
Can_17_McmCan_Init() API service is called with an invalid configuration set selection.	AUTOSAR	CAN_17_MCMCAN_E_INIT_FAILED=0x0A	Can_17_McmCan_Init
API service is called with an inappropriate CAN message identifier as its parameter.	IFX	CAN_17_MCMCAN_E_PARAM_MSGID =0x0D	Can_17_McmCan_Write
resource is not configured to the calling core.	IFX	CAN_17_MCMCAN_E_NOT_CONFIGURED=0x64	Can_17_McmCan_Init, Can_17_McmCan_SetBaudrate, Can_17_McmCan_SetIcomConfiguration, Can_17_McmCan_Write, Can_17_McmCan_EnableControllerInterrupts, Can_17_McmCan_DisableControllerInterrupts, Can_17_McmCan_SetControllerMode, Can_17_McmCan_CheckBaudrate
Master core is in uninitialized state when the Can_17_McmCan_Init() API is called in the slave core.	IFX	CAN_17_MCMCAN_E_MASTER_CORE_UNINIT=0x66	Can_17_McmCan_Init
The Can_17_McmCan_Deinit() API is called by the master core before de-initialization of the slave cores is not yet completed.	IFX	CAN_17_MCMCAN_E_SLAVE_CORE_INIT=0x67	Can_17_McmCan_Deinit

5.3.7.2 Production errors

The driver does not report any production errors.

Can_17_McmCan driver

5.3.7.3 Safety errors

The driver does not report any safety errors.

5.3.7.4 Runtime errors

The driver does not report any runtime errors.

5.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

5.3.8.1 Deviations

The section describes the deviations from software specification.

Table 391 known deviations

Reference	Deviation
Specification of CAN Driver AUTOSAR Release 4.2.2 - [SWS_Can_00360] : Can_CheckWakeup	The CAN driver does not support the Can_17_McmCan_CheckWakeup() API listed in AUTOSAR. The CAN driver does not support wake up from sleep over the CAN bus. Hence, the configuration parameter CanWakeUpFunctionalityAPI that is associated with the enabling of the Can_17_McmCan_CheckWakeup() API is made non-editable.
Specification of CAN Driver AUTOSAR Release 4.2.2 - [ECUC_Can_00480]: CanControllerTrcvDelayCompensationOffset	For CanControllerTrcvDelayCompensationOffset parameter, the range has been extended to 65535 in order to accommodate CAN FD with the higher baudrates of > 2Mbps.
Specification of CAN Driver AUTOSAR Release 4.2.2 - [SWS_Can_00420] : The Can module shall reset the interrupt flag at the end of the ISR (if not done automatically by hardware).	RF0WE, RF1WE, RF0FE, RF1FE, TEFN, BO and DRX are the interrupts configured for the CAN module, in-order to ensure that the successive events are registered in IR during the processing of interrupts of previous received messages, interrupt flags are cleared at the start of the ISR. Note that the BO is the only exception and clears the interrupt at end of the ISR.
Specification of CAN Driver AUTOSAR Release 4.2.2 - [SWS_Can_00180]: The Can module may implement the function Can_MainFunction_Read as empty define in case no polling at all is used	Due to mixed mode implementation, application may configure the CanRxProcessing to Interrupt mode and would still call the Can_17_McmCan_MainFunction_Read to handle the Rx FIFO in polling mode, hence this requirement is deviated. Application must avoid calling this function when none of the receives are to be processed through polling mechanism.

5.3.8.2 Limitations

The section describes the limitations from software specification.

Can_17_McmCan driver
Table 392 Known limitations

Reference	Limitation
CanIf_RxIndication	CanIf_RxIndication contains the pdu information stored in the CAN driver internal memory so the upper layer must make a copy of pdu information once the CanIf_RxIndication is called by the CAN driver and should not be reusing the pointer passed by CAN driver.
RX FIFO handling in Interrupt mode	In the CAN driver for handling Rx FIFO through interrupt mode, watermark and FIFO full conditions are enabled. Rx FIFO interrupt processes maximum of configured FIFO elements and if messages are received while the Rx FIFO messages are in progress and if number of messages received is greater than the configured threshold level on exit of interrupt handler watermark interrupt will not be triggered therefore all messages will be processed only on FULL interrupt.
Mixed mode handling	Multiperiod read (Main_function_Read_x) for this feature will not be supported
Rx processing configuration of controllers	Rx processing of all configured controllers shall be either in polling or interrupt. Few controllers in CanRxProcessing in polling and few controllers in CanRxProcessing interrupt is not supported.
Rx processing (Dedicated and FIFO) only through interrupts	If application is handling the data receive through interrupts only without the need for processing FIFO Rx in polling mode, application must not call Can_17_McmCan_MainFunction_Read API from the scheduled function.

5.3.9 Unsupported hardware features

The following hardware features of McmCan are not supported:

- CAN error handling apart from bus-off notification
- CAN analysis: analyze mode, bit timing analysis
- Gateway mode
- Foreign remote requests
- Remote frames
- TTCAN
- Transmit FIFO
- Non-ISO frame transmission
- Protocol exception handling feature

CanTrcv_17_V9251 driver

6 CanTrcv_17_V9251 driver

6.1 User information

6.1.1 Description

The CAN transceiver is a hardware device, which adapts the signal levels that are used on the CAN bus to the logical (digital) signal levels recognized by a microcontroller. CAN Transceiver is part of the ECU Abstraction layer and works as an interface between the CAN protocol controller and the physical differential bus. CAN Transceiver driver is implemented to support the Infineon TLE9251V hardware. It supports the wake-up functionality through the bus, which wakes up only for valid wake-up pattern (WUP). The DIO interface is used to control the modes of the CAN Transceiver. The CAN transceiver, TLE9251V, supports the NORMAL and STANDBY modes. The CanTrcv_17_V9251 driver provides the services for:

- Initialization of the CAN Transceivers
- Controlling the operation mode of CAN Transceivers through the DIO
- Enabling/disabling the wake-up of the CAN Transceivers

6.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

CanTrcv_17_V9251 driver

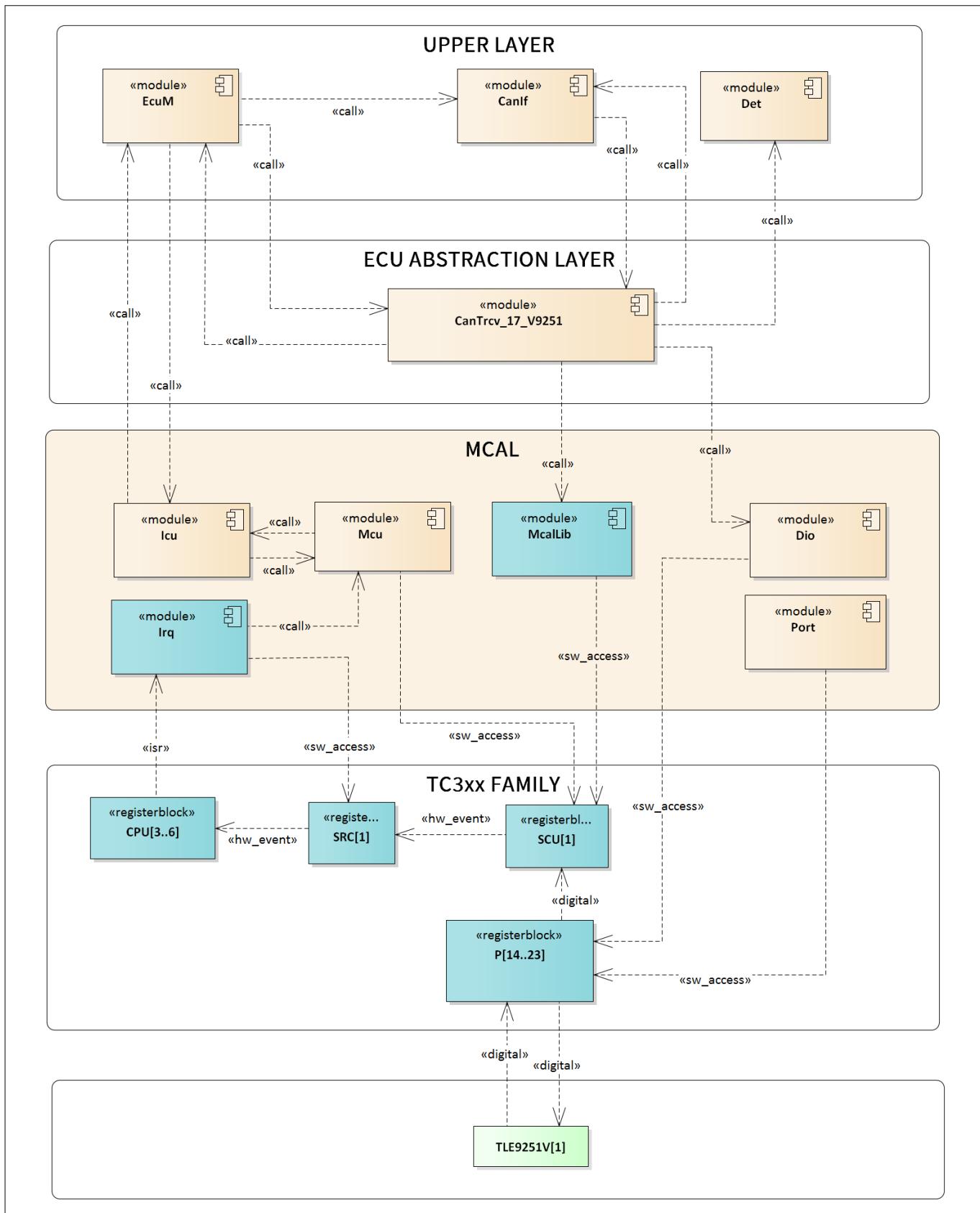


Figure 86

Mapping of hardware-software interfaces

CanTrcv_17_V9251 driver

6.1.2.1 TLE9251V: primary hardware peripheral

Hardware functional features

The CAN Transceiver driver uses the TLE9251V to provide an interface between the physical CAN bus layer and the CAN protocol controller. The key hardware functional features used by the driver are:

- Interface between CAN controller and CAN physical bus
- Supports Normal and Standby operation modes
- Supports BUS wake up, i.e. wake up by valid Wake-up Pattern only

The unsupported feature of the TLE9251V is:

- Forced-receive-only mode

Users of the hardware

The CAN Transceiver driver exclusively utilizes the TLE9251V module.

Hardware diagnostic features

Not applicable.

Hardware events

Not applicable.

6.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The CAN Transceiver driver depends on the SCU IP for the clock and reset functionalities.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the CAN transceiver driver.

Hardware events

Hardware events from the SCU are not used by the CAN transceiver driver.

6.1.2.3 Port: dependent hardware peripheral

Hardware functional features

The digital signals are routed to the CAN transceiver hardware through the digital port pads. The port pads are configured and enabled through the PORT driver. The CanTrcv_17_V9251 driver depends on the PORT driver for configuring the RxD, TxD, STB pins of the CAN transceiver hardware.

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Not applicable.

6.1.2.4 SRC: dependent hardware peripheral

Hardware functional features

CanTrcv_17_V9251 driver

The CAN transceiver driver depends on the ICU for interrupt handling. The ICU depends on the interrupt router for raising an interrupt to the CPU based on the wake-up events, which indicates wake-up activity on the RxD pin of the transceiver. The RxD pin is connected to the edge detection channel of the ICU.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software.

Hardware diagnostic features

The SMU alarms configured for interrupt router are not monitored by the CAN transceiver driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU. The CAN transceiver driver depends on the ICU driver, which provides interrupt handlers as software interfaces that must be invoked from the ISR.

6.1.3 File structure

6.1.3.1 C file structure

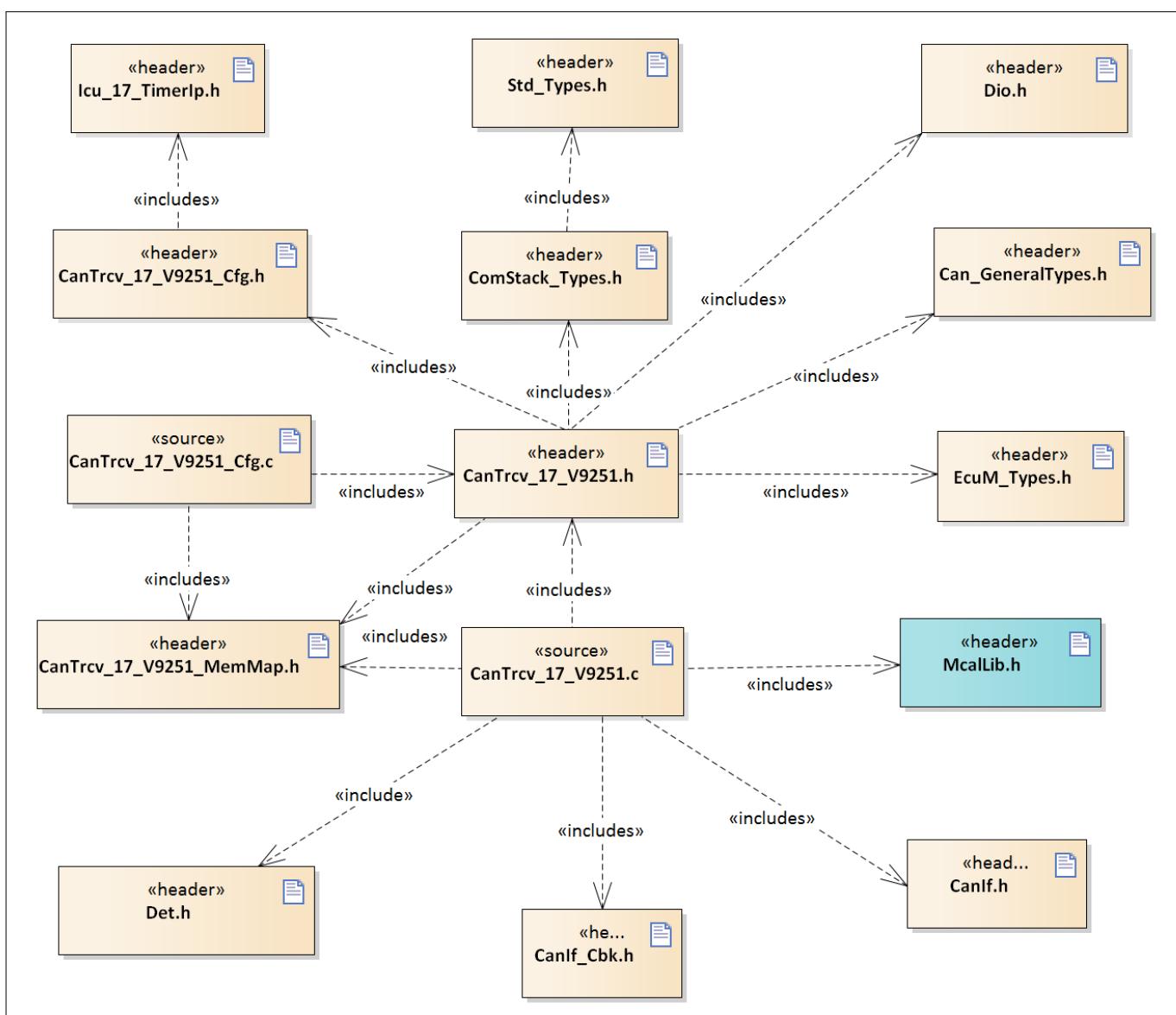


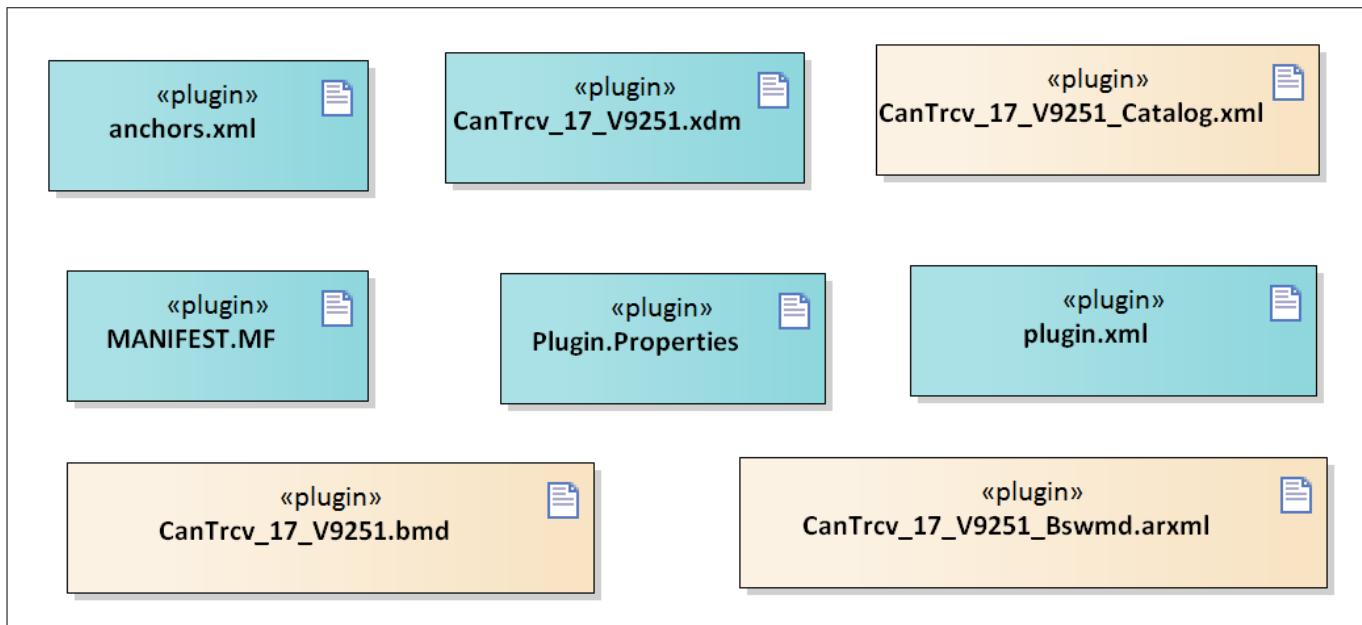
Figure 87

C file structure

CanTrcv_17_V9251 driver
Table 393 C file structure

File name	Description
CanIf.h	Header file containing the exported interfaces of CanIf
CanIf_Cbk.h	Header file containing declarations of the CanIf callbacks
CanTrcv_17_V9251.c	File (Static) containing implementation of CanTrcv_17_V9251 APIs
CanTrcv_17_V9251.h	Header file (Static) defining prototypes of data structures and APIs of CanTrcv_17_V9251 driver
CanTrcv_17_V9251_Cfg.c	File (Generated) containing definition of the configuration data structures for the CanTrcv_17_V9251 driver
CanTrcv_17_V9251_Cfg.h	Header file (Generated) containing CanTrcv_17_V9251 module constants and pre-processor macros as #defines
CanTrcv_17_V9251_MemMap.h	File (Static) containing the memory section definitions used by the CanTrcv_17_V9251 driver
Can_GeneralTypes.h	Contains all types and constants that are shared among the AUTOSAR CAN modules Can, CanIf and CanTrcv
ComStack_Types.h	Type Definition for Com stack
Det.h	Provides the exported interfaces of Development Error Tracer
Dio.h	Header file (Static) defining prototypes of data structures and APIs
EcuM_Types.h	Header file containing type declarations of the EcuM
Icu_17_TimerIp.h	Header file (static) defining prototypes of configuration data structures and APIs
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

6.1.3.2 Code generator plugin files

CanTrcv_17_V9251 driver

Figure 88 **Code generator plugin files**
Table 394 **Code generator plugin files**

File name	Description
CanTrcv_17_V9251.bmd	AUTOSAR format XML data model schema file
CanTrcv_17_V9251.xdm	Tresos format XML data model schema file
CanTrcv_17_V9251_Bswmd.arxml	AUTOSAR format module description file
CanTrcv_17_V9251_Catalog.xml	AUTOSAR format catalog file as per catalog_V3_0_0.ml.xsd
MANIFEST.MF	Tresos plugin support file containing the metadata for the CanTrcv_17_V9251 driver
Plugin.Properties	Tresos plugin support file for the CanTrcv_17_V9251 driver
anchors.xml	Tresos anchors support file for the CanTrcv_17_V9251 driver
plugin.xml	Tresos plugin support file for the CanTrcv_17_V9251 driver

6.1.4 Integration hints

This section lists the key points that an integrator or user of the CanTrcv_17_V9251 driver must consider.

6.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the CanTrcv_17_V9251 driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of the MCAL, the EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced

CanTrcv_17_V9251 driver

with a complete EcuM module during the integration phase. The CAN transceiver driver uses the API of EcuM to provide notifications as listed.

`EcuM_SetWakeupEvent()`: indication to EcuM for a valid wake-up from a transceiver channel.

- **CAN Interface (CanIf)**

The CanIf module is a part of the AUTOSAR stack that provides upper layers a hardware independent interface to the CAN communication system comprising multiple CAN controllers and CAN transceivers .The `CanIf_Cbk.c` and `CanIf_Cbk.h` files are provided as stub code and needs to be replaced with complete CanIf module during integration phase. The CanTrcv driver uses the API of CanIf to provide notifications as listed.

`CanIf_TrcvModeIndication()`: notification for a successful mode transition that was triggered for a transceiver channel.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `CanTrcv_17_V9251_MemMap.h` file.

The `CanTrcv_17_V9251_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are relocated to the correct memory region. A sample implementation listing the memory section macros is shown as follows.

CanTrcv_17_V9251 driver

```

*****GLOBAL DATA SECTION *****
#if defined CANTRCV_17_V9251_START_SEC_VAR_CLEARED_QM_LOCAL_UNSPECIFIED
/* User Pragma here */
#define CANTRCV_17_V9251_START_SEC_VAR_CLEARED_QM_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined CANTRCV_17_V9251_STOP_SEC_VAR_CLEARED_QM_LOCAL_UNSPECIFIED
/* User Pragma here */
#define CANTRCV_17_V9251_STOP_SEC_VAR_CLEARED_QM_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined CANTRCV_17_V9251_START_SEC_VAR_CLEARED_QM_LOCAL_8
/* User Pragma here */
#define CANTRCV_17_V9251_START_SEC_VAR_CLEARED_QM_LOCAL_8
#define MEMMAP_ERROR
#elif defined CANTRCV_17_V9251_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
/* User Pragma here */
#define CANTRCV_17_V9251_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
#define MEMMAP_ERROR
#elif defined CANTRCV_17_V9251_START_SEC_VAR_CLEARED_QM_LOCAL_16
/* User Pragma here */
#define CANTRCV_17_V9251_START_SEC_VAR_CLEARED_QM_LOCAL_16
#define MEMMAP_ERROR
#elif defined CANTRCV_17_V9251_STOP_SEC_VAR_CLEARED_QM_LOCAL_16
/* User Pragma here */
#define CANTRCV_17_V9251_STOP_SEC_VAR_CLEARED_QM_LOCAL_16
#define MEMMAP_ERROR

***** CANTRCV_17_V9251 MODULE CONFIG DATA *****
#elif defined CANTRCV_17_V9251_START_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
/* User Pragma here */
#define CANTRCV_17_V9251_START_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined CANTRCV_17_V9251_STOP_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
/* User Pragma here */
#define CANTRCV_17_V9251_STOP_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined CANTRCV_17_V9251_START_SEC_CONFIG_DATA_QM_LOCAL_8
/* User Pragma here */
#define CANTRCV_17_V9251_START_SEC_CONFIG_DATA_QM_LOCAL_8
#define MEMMAP_ERROR
#elif defined CANTRCV_17_V9251_STOP_SEC_CONFIG_DATA_QM_LOCAL_8
/* User Pragma here */
#define CANTRCV_17_V9251_STOP_SEC_CONFIG_DATA_QM_LOCAL_8
#define MEMMAP_ERROR

***** CANTRCV_17_V9251 MODULE CODE SECTION *****
#elif defined CANTRCV_17_V9251_START_SEC_CODE_QM_LOCAL
/* User Pragma here */
#define CANTRCV_17_V9251_START_SEC_CODE_QM_LOCAL
#define MEMMAP_ERROR
#elif defined CANTRCV_17_V9251_STOP_SEC_CODE_QM_LOCAL
/* User Pragma here */
#define CANTRCV_17_V9251_STOP_SEC_CODE_QM_LOCAL

```

CanTrcv_17_V9251 driver

```
#undef MEMMAP_ERROR
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The CanTrcv_17_V9251 driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the CanTrcv_17_V9251 driver must process all the errors reported to the DET module through the `Det_ReportError()` API. The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is not required for integrating the CanTrcv_17_V9251 driver.

- **SchM**

The CanTrcv_17_V9251 driver does not use any SchM services.

- **Safety error**

The CanTrcv_17_V9251 driver does not report any safety errors.

- **Notifications and callbacks**

The CanTrcv_17_V9251 driver does not implement any notifications. However, the CanTrcv_17_V9251 driver notifies the upper layer with the help of the following functions:

`CanIf_TrsvModeIndication()`: mode change indication to the CanIf layer after successful mode change of the CAN transceiver.

`EcuM_SetWakeupEvent()`: indication to the EcuM for a valid wake-up from the CAN transceiver

- **Operating system (OS)**

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application. The OS files provided by the MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

6.1.4.2 Multicore and Resource Manager

The CanTrcv_17_V9251 driver does not support execution on multiple cores simultaneously.

6.1.4.3 MCU support

The CanTrcv_17_V9251 driver is dependent on the MCU driver for the ERU channel allocation and system clock configuration. The initialization of the CanTrcv_17_V9251 driver must be started only after completion of the MCU initialization. The following must be considered while configuring the MCU driver in the EB tresos:

Select the `McuHardwareResourceAllocationConf` container and allocate the ERU input and output channels to the ICU driver from the `McuEruAllocationConf` subcontainer.

The corresponding ERU input and output channels have to be referred in `ERUInputConfiguration` container in the ICU channel, which is configured for wake-up and edge detection.

6.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the CanTrcv_17_V9251 driver through the PORT configuration and initialize the port pins prior to invoking of the CanTrcv_17_V9251 driver initialization. The TxD and RxD pins (corresponding to the Rx pin selection made in the CAN driver) of the different CAN controllers must be configured with respective direction and

CanTrcv_17_V9251 driver

configuration in the PORT driver. The STB pin of the CanTrcv TLE9251V must be configured as INPUT pin in the PORT driver configuration.

6.1.4.5 DMA support

The CanTrcv_17_V9251 driver does not use any service provided by the DMA driver.

6.1.4.6 Interrupt connections

The CanTrcv_17_V9251 driver does not use any interrupt source.

CanTrcv_17_V9251 driver

6.1.4.7 Example usage

This section describes how the CanTrcv_17_V9251 driver can be configured and how to use different APIs provided by the driver. All the APIs should be provided with valid input parameters. To detect the invalid function parameters, the DET should be enabled. The behavior of the APIs is undefined if DET is disabled and wrong parameters are passed.

Configuration of the driver

The CanTrcv_17_V9251 driver configuration involves the following steps.

1. In the MCU driver, configure the system clock and allocate ERU channels for the ICU driver.
2. In the Port driver, configure the port pin referred by the CAN transceiver TLE9251V STB as input pin.
3. In the DIO driver, configure the referred port pin to control the CAN transceiver TLE9251V hardware as an individual channel.
4. In the ICU driver, configure the ICU wake-up capable channel to detect the FALLING EDGE of the CAN transceiver TLE9251V RxD pin. This needs ERU channel configuration.
5. The IRQ driver configuration is required to configure the interrupt priorities for interrupts used by the ICU.
6. The MCALLIB driver configuration is required for the timing services used by the CanTrcv_17_V9251 driver.
7. In the EcuM, configure the wake-up source, and the same wake-up source must be configured in the CanTrcv_17_V9251 and the ICU configuration.
8. In the CanTrcv_17_V9251 driver, configure for required channels with Normal or Standby mode, the CanTrcvWakeUpByBusUsed parameter must be enabled for wake-up support for the corresponding channel.
9. In the CanTrcv_17_V9251 channel configuration, for CanTrcvIcuChannelRef parameter, refer to the ICU channel configured for wake up.
10. In the CanTrcv_17_V9251 channel configuration, CanTrcvDioChannel1Access parameter must refer to the DIO channel configured for controlling the CAN transceiver TLE9251 RxD pin.

When the CAN transceiver is in the standby mode, if it receives a valid wake-up pattern, the RxD pin of CAN transceiver will change its state from high to low. This falling edge is detected using the ICU module with the help of the ERU, therefore, pin connection should be ensured from the CAN transceiver RxD pin to the ERU input pin configured for the ERU in the ICU channel.

Initialization of CanTrcv_17_V9251 driver

The CanTrcv_17_V9251 driver is dependent on the ICU driver for edge detection. The initialization of the CanTrcv_17_V9251 driver must be started only after completion of the ICU initialization. Since the CAN transceiver TLE9251 supports wake-up only with interrupt mode, the ICU must be put to the sleep mode, and wake up for the corresponding channel should be enabled to support the wake-up functionality.

```

/*MCU Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock ();
/* Port Initialization */
Port_Init(&Port_Config);
/*ICU Initialization */
Icu_17_TimerIp_Init(&Icu_17_TimerIp_Config);
/* CanTrcv_17_V9251 Initialization */
CanTrcv_17_V9251_Init(NULL_PTR);
/* Further APIs of CanTrcv driver can be called now */

```

CAN Transceiver operation mode change:

CanTrcv_17_V9251 driver

After the CAN transceiver initialization, the following sequence can be followed for changing the operation mode.

```
/* CanTrcv_17_V9251 operation mode change */
CanTrcv_17_V9251_SetOpMode(0, CANTRCV_TRCVMODE_NORMAL);
```

CAN Transceiver wake-up mode change:

After the CAN transceiver initialization, the following sequence can be followed for changing the wake-up mode.

```
/* CanTrcv_17_V9251 wake-up mode change */
CanTrcv_17_V9251_SetWakeUpMode(0, CANTRCV_WUMODE_ENABLE);
```

6.1.5 Key architectural considerations

6.1.5.1 CAN transceiver wake up: only Interrupt mode is supported

The CAN transceiver driver supports the wake up functionality with the help of interrupts generated by ICU driver. Wake up by polling is not supported due to hardware limitations. In the CAN transceiver TLE9251V hardware, the wake-up activity is indicated by the RxD pin. In the standby mode, if the transceiver receives a valid WUP, the RxD pin of the transceiver changes its state from high to low and follows the CAN bus after a delay (less than 5us). The RxD pin is connected to the ERU, once the ICU driver gets the wake-up interrupt from the RxD transition from the ERU, the ICU driver informs the wake-up event to the EcuM.

6.1.5.2 User mode is not supported

The CanTrcv_17_V9251 driver does not support the User mode configuration for any of its APIs. Therefore, all the APIs of the driver shall be executed in the Supervisor mode.

[cover parentID CANTRCVW9251={D9AE5D75-3561-47e5-B0E0-49B9D0C1092A}]

CanTrcv_17_V9251 driver

6.2 Assumptions of Use (AoUs)

There are no AoUs for the driver.

CanTrcv_17_V9251 driver

6.3 Reference information

6.3.1 Configuration interfaces

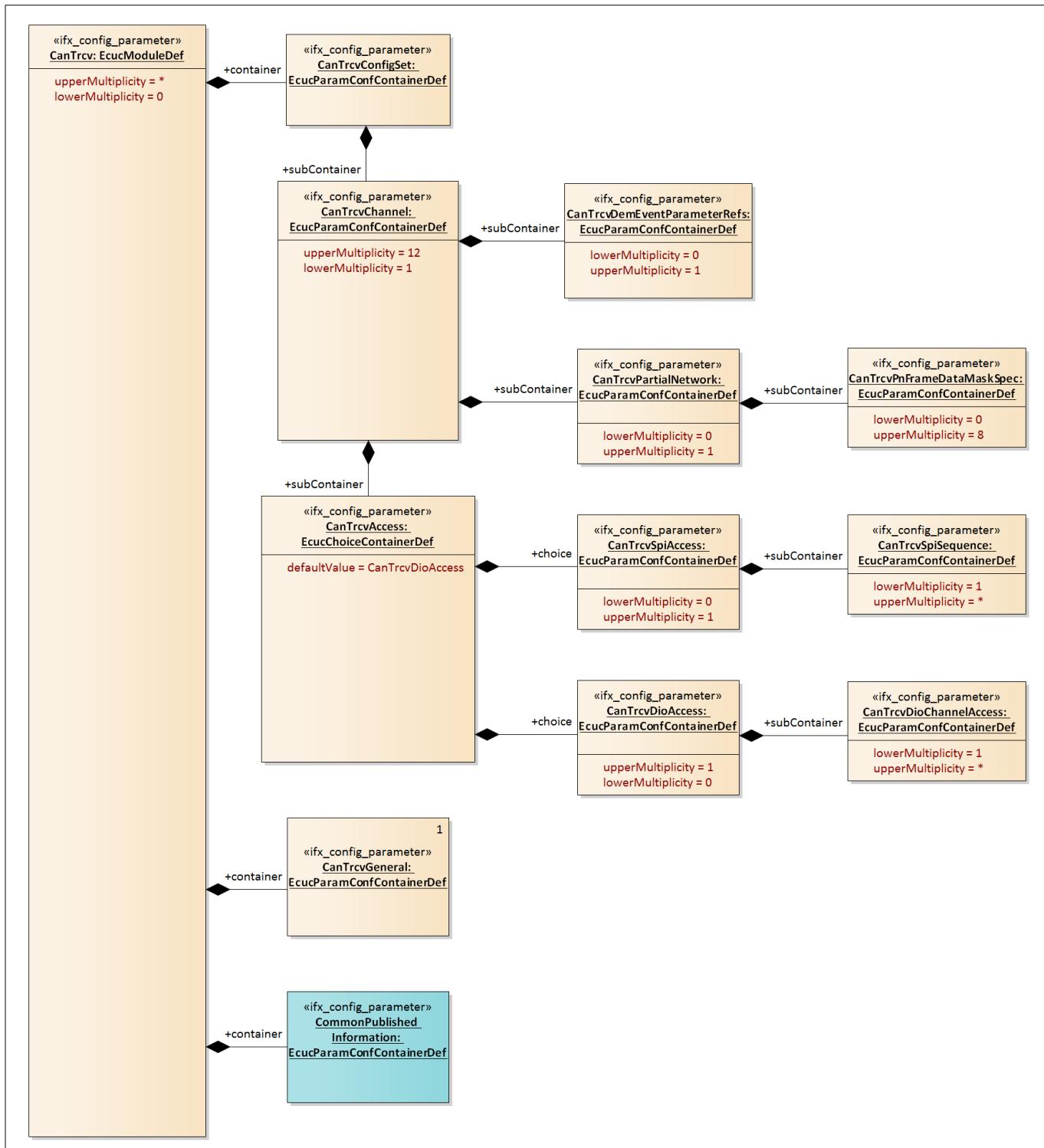


Figure 89 Container hierarchy along with their configuration parameters

CanTrcv_17_V9251 driver

6.3.1.1 Container: CommonPublished Information

This container contains the common published information of the CanTrcv_17_V9251 driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

6.3.1.1.1 ArMajorVersion

Table 395 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	This parameter provides the major version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

6.3.1.1.2 ArMinorVersion

Table 396 Specification for ArMinorVersion

Name	ArMinorVersion		
Description	This parameter provides the minor version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

CanTrcv_17_V9251 driver**6.3.1.1.3 ArPatchVersion****Table 397 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	This parameter provides the patch version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

6.3.1.1.4 ModuleId**Table 398 Specification for ModuleId**

Name	ModuleId		
Description	This parameter provides the module Id for the CanTrcv_17_V9251 driver.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	70		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

6.3.1.1.5 Release**Table 399 Specification for Release**

Name	Release		
Description	This parameter specifies the derivative for which configuration project is created.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		

CanTrcv_17_V9251 driver
Table 399 Specification for Release (continued)

Default value	As per hardware derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

6.3.1.1.6 SwMajorVersion
Table 400 Specification for SwMajorVersion

Name	SwMajorVersion		
Description	This parameter provides the software major version of the CanTrcv_17_V9251 driver.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

6.3.1.1.7 SwMinorVersion
Table 401 Specification for SwMinorVersion

Name	SwMinorVersion		
Description	This parameter provides the Software minor version of the CanTrcv_17_V9251 driver.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

CanTrcv_17_V9251 driver
Table 401 Specification for SwMinorVersion (continued)

Origin	IFX	Scope	LOCAL
Dependency	-		

6.3.1.1.8 SwPatchVersion
Table 402 Specification for SwPatchVersion

Name	SwPatchVersion		
Description	This parameter provides the software patch version of the CanTrcv_17_V9251 driver.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

6.3.1.1.9 VendorApiInfix
Table 403 Specification for VendorApiInfix

Name	VendorApiInfix		
Description	This parameter is used to specify the vendor specific name of the CanTrcv_17_V9251 driver.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	V9251		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

CanTrcv_17_V9251 driver**6.3.1.1.10 VendorId****Table 404 Specification for VendorId**

Name	VendorId		
Description	This parameter provides the Vendor Id for CanTrcv_17_V9251 driver.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

6.3.1.2 Container: CanTrcv

Specifies the configuration of the CAN Transceiver driver module.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

6.3.1.2.1 Config Variant**Table 405 Specification for Config Variant**

Name	Config Variant		
Description	This parameter indicates the selected configuration variant for CanTrcv_17_V9251 driver.		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	VariantPreCompile: The CanTrcv_17_V9251 driver supports only Pre- Compile variant.		
Default value	VariantPreCompile		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

CanTrcv_17_V9251 driver

6.3.1.3 Container: CanTrcvConfigSet

This container contains the configuration parameters and sub containers of the AUTOSAR CAN transceiver module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

6.3.1.3.1 CanTrcvSPICommRetries

Table 406 Specification for CanTrcvSPICommRetries

Name	CanTrcvSPICommRetries		
Description	<p>This parameter indicates the maximum number of communication retries in case of a failed SPI communication (applies both to timed out communication and to errors/NACK in the response data).</p> <p>If configured value is '0', no retry is allowed (communication is expected to succeed at first try).</p> <p>Note: Since CAN transceiver TLE9251V is not supporting SPI interface this parameter is not supported and made non-editable.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.3.2 CanTrcvSPICommTimeout

Table 407 Specification for CanTrcvSPICommTimeout

Name	CanTrcvSPICommTimeout		
Description	<p>This parameter indicates the maximum time allowed to the CAN transceiver for replying (either positively or negatively) to a SPI command.</p> <p>Timeout is configured in milliseconds. Timeout value of '0' means that no specific timeout is to be used by CAN transceiver and the communication is executed at the best of the SPI HW capacity.</p> <p>Note: Since CAN transceiver TLE9251V is not supporting SPI interface this parameter is not supported and made non-editable.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 100		

CanTrcv_17_V9251 driver
Table 407 Specification for CanTrcvSPICommTimeout (continued)

Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.4 Container: CanTrcvChannel

This container gives CAN transceiver driver information for a single CAN transceiver channel.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

6.3.1.4.1 CanTrcvAccess

Table 408 Specification for CanTrcvAccess

Name	CanTrcvAccess		
Description	This container gives CAN transceiver driver information about access to a single CAN transceiver. Note: CanTrcv_17_V9251 supports only DIO Interface.		
Multiplicity	1..1	Type	EcucChoiceContainerDef
Range			
Default value	CanTrcvDioAccess		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.4.2 CanTrcvChannelId

Table 409 Specification for CanTrcvChannelId

Name	CanTrcvChannelId
Description	This parameter specifies unique identifier of the CAN transceiver channel.

CanTrcv_17_V9251 driver
Table 409 Specification for CanTrcvChannelId (continued)

	<p>Note: The channel id should be less than number of channels configured. Zero is selected as the default value.</p> <p>Note: As per AUTOSAR the range of this parameter is 0 to 255, since AURIX TC3xx CAN controller has 12 nodes, CanTrcv_17_V9251 driver supports only 12 channels and range is from 0 to 11.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 11		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

6.3.1.4.3 CanTrcvChannelUsed
Table 410 Specification for CanTrcvChannelUsed

Name	CanTrcvChannelUsed		
Description	<p>This parameter Specifies if the configured channel is used or not.</p> <p>Note: This parameter is used to enable/disable the configured channel.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.4.4 CanTrcvControlsPowerSupply
Table 411 Specification for CanTrcvControlsPowerSupply

Name	CanTrcvControlsPowerSupply
-------------	----------------------------

CanTrcv_17_V9251 driver**Table 411 Specification for CanTrcvControlsPowerSupply (continued)**

Description	This parameter indicates the ECU power supply controlling method. TRUE = Controlled by transceiver. FALSE = Not controlled by transceiver. Note: Since CAN transceiver TLE9251V does not control the ECU power supply, this parameter is set FALSE and made non-editable.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.4.5 CanTrcvHwPnSupport**Table 412 Specification for CanTrcvHwPnSupport**

Name	CanTrcvHwPnSupport		
Description	This parameter indicates whether the CAN transceiver supports the selective wake-up function TRUE = Selective wake up feature is supported by the transceiver FALSE = Selective wake up functionality is not available in transceiver. Note: Since CAN transceiver TLE9251V does not support selective wake up functionality the default value of this parameter is made false and made non-editable.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL

CanTrcv_17_V9251 driver
Table 412 Specification for CanTrcvHwPnSupport (continued)

Dependency	-
-------------------	---

6.3.1.4.6 **CanTrcvIcuChannelRef**

Table 413 Specification for CanTrcvIcuChannelRef

Name	CanTrcvIcuChannelRef		
Description	<p>Reference to the ICU channel for detecting the wakeups.</p> <p>Note: Since the name of the dependent parameter is user configurable, the default value is set to NULL.</p> <p>This parameter is made non editable when CanTrcvWakeupByBusUsed is configured as FALSE.</p>		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: IcuChannel		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.4.7 **CanTrcvInitState**

Table 414 Specification for CanTrcvInitState

Name	CanTrcvInitState		
Description	<p>This parameter specifies the state of CAN transceiver after call to CanTrcv_17_V9251_Init.</p> <p>Note: CAN transceiver TLE9251V will support only normal mode and standby mode. Normal mode is set as default value, by assuming user expects CAN transceiver to work in normal mode. User is allowed to change the mode after initialization.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>CANTRCV_17_V9251_OP_MODE_NORMAL: Normal operation mode.</p> <p>CANTRCV_17_V9251_OP_MODE_STANDBY: Standby operation mode.</p>		
Default value	CANTRCV_17_V9251_OP_MODE_NORMAL		
Post-build variant value	FALSE	Post-build variant multiplicity	-

CanTrcv_17_V9251 driver
Table 414 Specification for CanTrcvInitState (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.4.8 CanTrcvMaxBaudrate
Table 415 Specification for CanTrcvMaxBaudrate

Name	CanTrcvMaxBaudrate		
Description	<p>This parameter specifies the max baud rate supported by CAN transceiver. Value shall be configured by configuration tool based on transceiver hardware type.</p> <p>Note: Default value is maximum baud rate supported by CAN transceiver. The baud rate will be in kbps. The baud rate range exceeds the AUTOSAR specified range. This parameter does not have any significance and it gives the information on maximum baud rate supported, so this parameter is not used anywhere in the implemented design.</p> <p>Note: Range and default vale of the baud rate is modified. Range is extended to 5kbps since the hardware supports CAN FD.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 5000		
Default value	5000		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.4.9 CanTrcvPorWakeupSourceRef
Table 416 Specification for CanTrcvPorWakeupSourceRef

Name	CanTrcvPorWakeupSourceRef
Description	<p>This parameter specifies the symbolic name reference to indicate the wake up sources configured to report the wake up source events.</p> <p>This reference is mandatory if the CAN transceiver supports POR flag.</p> <p>Since the name of the dependent parameter is user configurable, the default value is set to NULL.</p>

CanTrcv_17_V9251 driver**Table 416 Specification for CanTrcvPorWakeupSourceRef (continued)**

	Note: Since CAN transceiver TLE9251V is not supporting POR detection, this parameter is not supported and made non-editable. This configuration parameter is not used in the code but it is listed for AUTOSAR compatibility.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: EcuMWakeupSource		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

6.3.1.4.10 CanTrcvSyserrWakeupSourceRef**Table 417 Specification for CanTrcvSyserrWakeupSourceRef**

Name	CanTrcvSyserrWakeupSourceRef		
Description	<p>This parameter specifies the symbolic name reference to indicate the wake up sources configured to report the wake up source events.</p> <p>This reference is mandatory if the CAN transceiver supports SYSERR flag. Since the name of the dependent parameter is user configurable, the default value is set to NULL.</p> <p>Note: Since CAN transceiver TLE9251V is not supporting SYSERR detection, this parameter is not supported and made non-editable. This configuration parameter is not used in the code but it is listed for AUTOSAR compatibility.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: EcuMWakeupSource		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

CanTrcv_17_V9251 driver

6.3.1.4.11 **CanTrcvWakeupByBusUsed**

Table 418 Specification for CanTrcvWakeupByBusUsed

Name	CanTrcvWakeupByBusUsed		
Description	<p>This parameter Indicates whether wake up by bus is supported or not. If CAN transceiver hardware does not support wake up by bus value is always FALSE. If CAN transceiver hardware supports wake up by bus value is TRUE or FALSE depending whether it is used or not.</p> <p>Note: Since CAN transceiver TLE9251V supports wake up only through bus, user can use this parameter to switch the wake up functionality of CAN transceiver.</p> <p>TRUE = Is used and wake up functionality is supported for respective channel.</p> <p>FALSE = Is not used and wake up functionality is not supported for respective channel.</p> <p>If CanTrcvWakeupByBusUsed is FALSE then user is not allowed to configure wake up related configuration parameters like CanTrcvWakeupSourceRef and CanTrcvIcuChannelRef.</p> <p>Since Cantcv_17_V9251 driver depends on this parameter, so this parameter is always needed, hence the lower multiplicity of this parameter is set to 1.</p> <p>Since CAN transceiver TLE9251V supports wake up functionality all the time, this parameter does not depends on CanTrcvWakeUpSupport parameter.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanTrcvWakeUpSupport		

6.3.1.4.12 **CanTrcvWakeupSourceRef**

Table 419 Specification for CanTrcvWakeupSourceRef

Name	CanTrcvWakeupSourceRef
Description	<p>This parameter is a reference to a wake up source configured in the EcuM configuration.</p> <p>This reference is only needed when CanTrcvWakeupByBusUsed is true.</p> <p>Note: Since the name of the dependent parameter is user configurable, the default value is set to NULL.</p> <p>This parameter is made non editable when CanTrcvWakeupByBusUsed is configured as FALSE.</p>

CanTrcv_17_V9251 driver
Table 419 Specification for CanTrcvWakeupSourceRef (continued)

Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: EcuMWakeupSource		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanTrcvWakeupByBusUsed		

6.3.1.5 Container: CanTrcvDemEventParameterRefs

This container contains the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId value. Note: Since CAN transceiver TLE9251V is not supporting production errors this container is not supported. This container is kept only for AUTOSAR schema compatibility and not for traceability.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

6.3.1.5.1 CANTRCV_E_BUS_ERROR

Table 420 Specification for CANTRCV_E_BUS_ERROR

Name	CANTRCV_E_BUS_ERROR		
Description	Reference to the DemEventParameter which will be issued when bus error has occurred. Note: Since CAN transceiver TLE9251V is not supporting production errors this configuration parameter is not supported and made non-editable. This parameter is kept only for AUTOSAR schema compatibility and not for traceability.		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

CanTrcv_17_V9251 driver

6.3.1.6 Container: CanTrcvDioAccess

This container gives CAN transceiver driver information about accessing ports and port pins. In addition relation between CAN transceiver hardware pin names and DIO port access information is given.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

6.3.1.7 Container: CanTrcvDioChannelAccess

This container gives DIO channel access by single CAN transceiver channel.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

6.3.1.7.1 CanTrcvDioSymNameRef

Table 421 Specification for CanTrcvDioSymNameRef

Name	CanTrcvDioSymNameRef		
Description	This parameter gives the symbolic name reference to a configured DIO Port, DIO Channel or DIO Channel Group. Note: CanTrcv_17_V9251 driver supports reference only to a DIO Channel.		
Multiplicity	1..1	Type	EcucChoiceReferenceDef
Range	Reference to Node: DioChannel		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.7.2 CanTrcvHardwareInterfaceName

Table 422 Specification for CanTrcvHardwareInterfaceName

Name	CanTrcvHardwareInterfaceName		
Description	This parameter specifies CAN transceiver hardware interface name. It is typically the name of a CAN transceiver pin. Note: Since CanTrcv_17_V9251 driver uses STB pin of CAN transceiver TLE9251V hardware for mode control, STB is the default name set for this parameter and made non-editable.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		

CanTrcv_17_V9251 driver
Table 422 Specification for CanTrcvHardwareInterfaceName (continued)

Default value	STB		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.8 Container: CanTrcvGeneral

This container gives CAN transceiver driver basic information.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

6.3.1.8.1 CanTrcvDevErrorDetect

Table 423 Specification for CanTrcvDevErrorDetect

Name	CanTrcvDevErrorDetect		
Description	Parameter enables or disables the Default Error Tracer (DET) detection and reporting. Note: By Default this feature is enabled to support debug during integration with application code, must be disabled if successfully integrated to development environment.		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.8.2 CanTrcvGetVersionInfo

Table 424 Specification for CanTrcvGetVersionInfo

Name	CanTrcvGetVersionInfo
-------------	-----------------------

CanTrcv_17_V9251 driver
Table 424 Specification for CanTrcvGetVersionInfo (continued)

Description	Parameter adds or removes the CanTrcv_17_V9251_GetVersionInfo API from the code. Note: The default value of this parameter is set to false to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.8.3 CanTrcvIndex

Table 425 Specification for CanTrcvIndex

Name	CanTrcvIndex		
Description	This parameter specifies the Instance Id of CanTrcv_17_V9251 module. Note: Since only one instance is supported, by default it shall have the Id 0.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.8.4 CanTrcvMainFunctionDiagnosticsPeriod

Table 426 Specification for CanTrcvMainFunctionDiagnosticsPeriod

Name	CanTrcvMainFunctionDiagnosticsPeriod
Description	This parameter describes the period for cyclic call to CanTrcv_MainFunctionDiagnostics. Unit of this parameter is in seconds.

CanTrcv_17_V9251 driver
Table 426 Specification for CanTrcvMainFunctionDiagnosticsPeriod (continued)

	Note: Since CanTrcv_17_V9251_MainFunctionDiagnostics API is not provided by the driver, this parameter is not applicable and made non-editable. This configuration parameter is not used in the code but it is listed for AUTOSAR compatibility.		
Multiplicity	0..1	Type	EcucFloatParamDef
Range	0.001 - 65.535		
Default value	0.001		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.8.5 CanTrcvMainFunctionPeriod
Table 427 Specification for CanTrcvMainFunctionPeriod

Name	CanTrcvMainFunctionPeriod		
Description	This parameter describes the period for cyclic call to CanTrcv_17_V9251_MainFunction. Unit of this parameter is in seconds. Note: Since CAN transceiver TLE9251V do not support polling mode, this parameter is not supported and made non-editable.		
Multiplicity	0..1	Type	EcucFloatParamDef
Range	0.001 - 65.535		
Default value	0.005		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.8.6 CanTrcvTimerType
Table 428 Specification for CanTrcvTimerType

Name	CanTrcvTimerType
Description	This parameter specifies the type of the timer service used in CAN transceiver driver.

CanTrcv_17_V9251 driver
Table 428 Specification for CanTrcvTimerType (continued)

	Note: Default value of this parameter is set to 'None' since McalLib APIs are used to realize the wait time. The parameter is made non-editable.		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	None: No timer type is used. Timer_1us16bit: Specifies 16 bit 1us timer type.		
Default value	None		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.8.7 CanTrcvWaitTime
Table 429 Specification for CanTrcvWaitTime

Name	CanTrcvWaitTime		
Description	This parameter specifies the wait time for transceiver state changes in seconds. Note: The maximum time for mode change by CAN transceiver TLE9251V is 20 us, hence default value is set to 20us and made non editable.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.00020 - 0.000255		
Default value	0.00020		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.8.8 CanTrcvWakeUpSupport
Table 430 Specification for CanTrcvWakeUpSupport

Name	CanTrcvWakeUpSupport
Description	This parameter informs the mode of wake up support .

CanTrcv_17_V9251 driver
Table 430 Specification for CanTrcvWakeUpSupport (continued)

	<p>Note:- Since CAN transceiver TLE9251V wake up functionality will work with ICU module interrupt, CANTRCV_17_V9251_WAKEUP_BY_POLLING and CANTRCV_17_V9251_WAKE_UP_NOT_SUPPORTED options are not supported.</p> <p>Note: A new option CANTRCV_WAKEUP_BY_INTERRUPT is added and set as default value, which is not-editable.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CANTRCV_17_V9251_WAKEUP_BY_INTERRUPT: Wake up by Interrupt.		
Default value	CANTRCV_17_V9251_WAKEUP_BY_INTERRUPT		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.9 Container: CanTrcvPartialNetwork

This container gives CAN transceiver driver information about the configuration of partial networking functionality. Note: Since CAN transceiver TLE9251V is not supporting partial networking, this configuration container is not supported and made non-editable. This configuration container and its parameters are not used in the code but it is listed for AUTOSAR compatibility.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

6.3.1.9.1 CanTrcvBaudRate

Table 431 Specification for CanTrcvBaudRate

Name	CanTrcvBaudRate		
Description	<p>This parameter indicates the CAN bus communication baud rate in kbps.</p> <p>Note: Since CAN transceiver TLE9251V does not control the baud rate this parameter is not supported and made non-editable.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 1000		
Default value	125		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

CanTrcv_17_V9251 driver
Table 431 Specification for CanTrcvBaudRate (continued)

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.9.2 CanTrcvBusErrFlag

Table 432 Specification for CanTrcvBusErrFlag

Name	CanTrcvBusErrFlag		
Description	<p>Indicates if the Bus Error (BUSERR) flag is managed by the BSW. This flag is set if a bus failure is detected by the transceiver.</p> <p>TRUE = Supported by transceiver and managed by BSW.</p> <p>FALSE = Not managed by BSW.</p> <p>Note: Since CAN transceiver TLE9251V is not supporting partial networking, this configuration Parameter is not supported made non-editable.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.9.3 CanTrcvPnCanIdIsExtended

Table 433 Specification for CanTrcvPnCanIdIsExtended

Name	CanTrcvPnCanIdIsExtended		
Description	<p>This parameter indicates whether extended or standard ID is used.</p> <p>TRUE = Extended Can identifier is used</p> <p>FALSE = Standard Can identifier is used</p> <p>Note: Since CAN transceiver TLE9251V is not supporting partial networking, this configuration Parameter is not supported and made non-editable.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		

CanTrcv_17_V9251 driver
Table 433 Specification for CanTrcvPnCanIdIsExtended (continued)

Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.9.4 CanTrcvPnEnabled
Table 434 Specification for CanTrcvPnEnabled

Name	CanTrcvPnEnabled		
Description	<p>This parameter indicates whether the selective wake-up function is enabled or disabled in CAN transceiver TLE9251V hardware.</p> <p>TRUE = Selective wakeup feature is enabled in the transceiver hardware FALSE = Selective wakeup feature is disabled in the transceiver hardware</p> <p>Note: Since CAN transceiver TLE9251V hardware is not supporting partial networking, this configuration parameter is not supported and made non-editable.</p>		
Multiplicity	1..1	Type	EcuCBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.9.5 CanTrcvPnFrameCanId
Table 435 Specification for CanTrcvPnFrameCanId

Name	CanTrcvPnFrameCanId		
Description	<p>This parameter indicates the CAN ID of the Wake-up Frame (WUF).</p> <p>Note: Since CAN transceiver TLE9251V is not supporting partial networking, this configuration parameter is not supported and made non-editable.</p>		

CanTrcv_17_V9251 driver
Table 435 Specification for CanTrcvPnFrameCanId (continued)

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.9.6 CanTrcvPnFrameCanIdMask
Table 436 Specification for CanTrcvPnFrameCanIdMask

Name	CanTrcvPnFrameCanIdMask		
Description	<p>This parameter indicates ID mask for the selective activation of the transceiver. It is used to enable Frame Wake-up (WUF) on a group of IDs.</p> <p>Note: Since CAN transceiver TLE9251V is not supporting partial networking, this configuration Parameter is not supported and made non-editable.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.9.7 CanTrcvPnFrameDlc
Table 437 Specification for CanTrcvPnFrameDlc

Name	CanTrcvPnFrameDlc
Description	<p>This parameter specifies the data length of the Wake-up Frame (WUF).</p> <p>Default value is set to 1 as it is the minimum value supported. Although WUF with DLC=0 is technically possible, it is explicitly not wanted.</p>

CanTrcv_17_V9251 driver
Table 437 Specification for CanTrcvPnFrameDlc (continued)

	Note: Since CAN transceiver TLE9251V is not supporting partial networking, this configuration parameter is not supported and made non-editable.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 8		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.9.8 CanTrcvPowerOnFlag

Table 438 Specification for CanTrcvPowerOnFlag

Name	CanTrcvPowerOnFlag		
Description	<p>This parameter indicates if the Power On Reset (POR) flag is available and is managed by the transceiver.</p> <p>TRUE = Supported by Hardware. FALSE = Not supported by Hardware</p> <p>Note: Since CAN transceiver TLE9251V is not supporting partial networking, this configuration Parameter is not supported and made non-editable.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.10 Container: CanTrcvPnFrameDataMaskSpec

This parameter defines data payload mask to be used on the received payload in order to determine if the transceiver must be woken up by the received Wake-up Frame (WUF). Note: Since CAN transceiver TLE9251V is

CanTrcv_17_V9251 driver

not supporting partial networking, this configuration parameter is not supported and made non-editable. This configuration container and its parameters are not used in the code but it is listed for AUTOSAR compatibility.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

6.3.1.10.1 CanTrcvPnPFrameDataMask

Table 439 Specification for CanTrcvPnPFrameDataMask

Name	CanTrcvPnPFrameDataMask		
Description	<p>This parameter defines the n byte (Byte0 = LSB) of the data payload mask to be used on the received payload in order to determine if the transceiver must be woken up by the received Wake-up Frame (WUF).</p> <p>Note: Since CAN transceiver TLE9251V is not supporting partial networking, this configuration Parameter is not supported and made non-editable.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.1.10.2 CanTrcvPnPFrameDataMaskIndex

Table 440 Specification for CanTrcvPnPFrameDataMaskIndex

Name	CanTrcvPnPFrameDataMaskIndex		
Description	<p>This parameter holds the position n in frame of the data mask-part.</p> <p>Note: Since CAN transceiver TLE9251V is not supporting partial networking, this configuration Parameter is not supported and made non-editable.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL

CanTrcv_17_V9251 driver
Table 440 Specification for CanTrcvPnFrameDataMaskIndex (continued)

Dependency	-
-------------------	---

6.3.1.11 Container: CanTrcvSpiAccess

This container gives CAN transceiver driver information about accessing SPI. If CAN transceiver hardware has no SPI interface, there is no instance of this container. Note: Since CAN transceiver TLE9251V hardware supports only DIO interface, this container is not supported. Note: This configuration container and its sub-containers and parameters are not used in the code but it is listed for AUTOSAR compatibility.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

6.3.1.12 Container: CanTrcvSpiSequence

This container gives CAN transceiver driver information about one SPI sequence. Note: Since CAN transceiver TLE9251V hardware supports only DIO interface, this container is not Supported.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

6.3.1.12.1 CanTrcvSpiAccessSynchronous

Table 441 Specification for CanTrcvSpiAccessSynchronous

Name	CanTrcvSpiAccessSynchronous		
Description	<p>This parameter is used to define whether the access to the SPI sequence is synchronous or asynchronous.</p> <p>true: SPI access is synchronous.</p> <p>false: SPI access is asynchronous.</p> <p>Note: Since CAN transceiver TLE9251V supports only DIO interface, this parameter is not supported and made non-editable.</p>		
Multiplicity	0..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

CanTrcv_17_V9251 driver

6.3.1.12.2 **CanTrcvSpiSequenceName**

Table 442 Specification for CanTrcvSpiSequenceName

Name	CanTrcvSpiSequenceName		
Description	<p>This parameter specifies the reference to a SPI sequence configuration container.</p> <p>Note: Since CAN transceiver TLE9251V hardware supports only DIO interface, this parameter is not Supported and made non-editable.</p>		
Multiplicity	0..*	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: SpiSequence		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

6.3.2 **Functions - Type definitions**

6.3.2.1 **CanTrcv_17_V9251_ConfigType**

Table 443 Specification for CanTrcv_17_V9251_ConfigType

Syntax	CanTrcv_17_V9251_ConfigType
File	CanTrcv_17_V9251.h
Description	This is the type of the external data structure containing the overall initialization data for the CAN transceiver driver and settings affecting all transceivers. Note: Since CanTrcv_17_V9251 module is pre-compile, this type is implemented as void and not used in the driver.
Source	IFX

6.3.3 **Functions - APIs**

6.3.3.1 **CanTrcv_17_V9251_Init**

Table 444 Specification for CanTrcv_17_V9251_Init API

Syntax	void CanTrcv_17_V9251_Init (
---------------	---------------------------------

CanTrcv_17_V9251 driver
Table 444 Specification for CanTrcv_17_V9251_Init API (continued)

	const CanTrcv_17_V9251_ConfigType * const ConfigPtr)	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to driver configuration. Note: Since CanTrcv_17_V9251 is pre compile module, null pointer must be passed as the parameter by caller of this API.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This API initializes all the connected CAN transceivers by setting CAN transceiver hardware to the mode configured by the configuration parameter CanTrcvInitState. The CAN Transceiver initialization status is set at the end of the initialization function execution.</p> <p>Note: Since CanTrcv_17_V9251 module is a pre compile module, NULL_PTR must be passed as the parameter for CanTrcv_17_V9251_Init API.</p>	
Source	AUTOSAR	
Error handling	<p>DET: CANTRCV_17_V9251_E_INIT_FAILED: This error is reported when CanTrcv_17_V9251_Init API is called without NULL_PTR as the parameter.</p> <p>Runtime Errors: None DEM: None Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

6.3.3.2 CanTrcv_17_V9251_SetOpMode
Table 445 Specification for CanTrcv_17_V9251_SetOpMode API

Syntax	Std_ReturnType CanTrcv_17_V9251_SetOpMode (const uint8 Transceiver, const CanTrcv_TrcvModeType OpMode)
---------------	--

CanTrcv_17_V9251 driver
Table 445 Specification for CanTrcv_17_V9251_SetOpMode API (continued)

Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different transceivers	
Parameters (in)	Transceiver OpMode	CAN transceiver to which API call has to be applied. Note: CanTrcv_17_V9251 driver supports 12 channels, so the range of this parameter must be 0 to 11. This parameter contains the desired operating mode. Note: CANTRCV_TRCVMODE_NORMAL and CANTRCV_TRCVMODE_STANDBY modes are supported by CanTrcv_17_V9251 driver.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: CAN Transceiver state have been changed to requested mode. E_NOT_OK: CAN Transceiver state change have failed or the parameter is out of the allowed range. The previous state has not been changed.
Description	This API sets the mode of the CAN transceiver given the value OpMode.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CANTRCV_17_V9251_E_UNINIT: This error will occur when module API service is called without CanTrcv_17_V9251 module initialization.</p> <p>CANTRCV_17_V9251_E_INVALID_TRANSCEIVER: This error will occur when API called with wrong transceiver parameter for the CanTrcv_17_V9251 driver.</p> <p>CANTRCV_17_V9251_E_PARAM_TRCV_OPMODE: This error will occur when API service called with invalid parameter for OpMode.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

CanTrcv_17_V9251 driver**6.3.3.3 CanTrcv_17_V9251_GetOpMode****Table 446 Specification for CanTrcv_17_V9251_GetOpMode API**

Syntax	<pre>Std_ReturnType CanTrcv_17_V9251_GetOpMode (const uint8 Transceiver, CanTrcv_TrcvModeType * const OpMode)</pre>	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. Note: CanTrcv_17_V9251 driver supports 12 channels, so the range of this parameter must be 0 to 11.
Parameters (out)	OpMode	Pointer to operation mode of the applied CAN transceiver.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation mode read successfully. E_NOT_OK: Operation mode was not detected.
Description	This API reads the mode of the CAN transceiver and returns it in the parameter OpMode.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CANTRCV_17_V9251_E_UNINIT: This error will occur when module API service is called without CanTrcv_17_V9251 module initialization.</p> <p>CANTRCV_17_V9251_E_INVALID_TRANSCEIVER: This error will occur when API called with wrong transceiver parameter for the CanTrcv_17_V9251 driver.</p> <p>CANTRCV_17_V9251_E_PARAM_POINTER: This error will occur when API called with Invalid pointer parameter for the CanTrcv_17_V9251 driver.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

CanTrcv_17_V9251 driver**6.3.3.4 CanTrcv_17_V9251_GetBusWuReason****Table 447 Specification for CanTrcv_17_V9251_GetBusWuReason API**

Syntax	<pre>Std_ReturnType CanTrcv_17_V9251_GetBusWuReason (const uint8 Transceiver, CanTrcv_TrcvWakeupReasonType * const reason)</pre>	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. Note: CanTrcv_17_V9251 driver supports 12 channels, so the range of this parameter must be 0 to 11.
Parameters (out)	reason	Pointer to wake up reason of the applied CAN transceiver. Note: Only CANTRCV_WU_POWER_ON, CANTRCV_WU_BY_BUS and CANTRCV_WU_INTERNALLY values are supported by the transceiver hardware
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Transceiver wakeup reason was provided successfully. E_NOT_OK: If no wake up reason is available or if the service request failed due to development errors.
Description	This API reads the wakeup reason for the CAN transceiver and returns it in the parameter reason.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CANTRCV_17_V9251_E_UNINIT: This error will occur when module API service is called without CanTrcv_17_V9251 module initialization.</p> <p>CANTRCV_17_V9251_E_INVALID_TRANSCEIVER: This error will occur when API called with wrong transceiver parameter for the CanTrcv_17_V9251 driver.</p> <p>CANTRCV_17_V9251_E_PARAM_POINTER: This error will occur when API called with Invalid pointer parameter for the CanTrcv_17_V9251 driver.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

CanTrcv_17_V9251 driver**6.3.3.5 CanTrcv_17_V9251_GetVersionInfo****Table 448 Specification for CanTrcv_17_V9251_GetVersionInfo API**

Syntax	<pre>void CanTrcv_17_V9251_GetVersionInfo (Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to version information of CanTrcv_17_V9251 module.
Parameters (in - out)	-	-
Return	void	-
Description	This API reads the version of the CanTrcv_17_V9251 module and returns it in the parameter versionInfo.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CANTRCV_17_V9251_E_PARAM_POINTER: This error will occur when API called with Invalid pointer parameter for the CanTrcv_17_V9251 driver.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	CanTrcvGetVersionInfo	
User hints	None	

6.3.3.6 CanTrcv_17_V9251_SetWakeupMode**Table 449 Specification for CanTrcv_17_V9251_SetWakeupMode API**

Syntax	<pre>Std_ReturnType CanTrcv_17_V9251_SetWakeupMode (const uint8 Transceiver, const CanTrcv_TrcvWakeupModeType TrcvWakeupMode)</pre>
Service ID	0x05

CanTrcv_17_V9251 driver
Table 449 Specification for CanTrcv_17_V9251_SetWakeupMode API (continued)

Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different transceivers	
Parameters (in)	Transceiver TrcvWakeupMode	CAN transceiver to which API call has to be applied. Note: CanTrcv_17_V9251 driver supports 12 channels, so the range of this parameter must be 0 to 11. Requested CAN transceiver wakeup mode. Note: The supported wake up modes is CANTRCV_WUMODE_ENABLE, CANTRCV_WUMODE_DISABLE and CANTRCV_WUMODE_CLEAR.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Wakeup state changed to the requested mode E_NOT_OK: Wakeup state change has failed or the parameter is out of the allowed range. The previous state has not been changed.
Description	<p>This API enables, disables or clears the wake-up events of the CAN transceiver according to parameter TrcvWakeupMode.</p> <p>If parameter TrcvWakeupMode is CANTRCV_WUMODE_ENABLE: wake up event is informed to EcuM.</p> <p>If parameter TrcvWakeupMode is CANTRCV_WUMODE_DISABLE: wake up event is not informed to EcuM and it is stored.</p> <p>If parameter TrcvWakeupMode is CANTRCV_WUMODE_CLEAR: stored pending wake up will be cleared.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CANTRCV_17_V9251_E_INVALID_TRANSCEIVER: This error will occur when API called with wrong transceiver parameter for the CanTrcv_17_V9251 driver.</p> <p>CANTRCV_17_V9251_E_UNINIT: This error will occur when module API service is called without CanTrcv_17_V9251 module initialization.</p> <p>CANTRCV_17_V9251_E_PARAM_TRCV_WAKEUP_MODE: This error will occur when API service called with invalid parameter for TrcvWakeupMode.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

CanTrcv_17_V9251 driver
Table 449 Specification for CanTrcv_17_V9251_SetWakeupMode API (continued)

Configuration dependencies	-
User hints	None

6.3.3.7 CanTrcv_17_V9251_CheckWakeups
Table 450 Specification for CanTrcv_17_V9251_CheckWakeups API

Syntax	Std_ReturnType CanTrcv_17_V9251_CheckWakeups (const uint8 Transceiver)	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. Note: CanTrcv_17_V9251 driver supports 12 channels, so the range of this parameter must be 0 to 11.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : when a valid wake up interrupt is detected E_NOT_OK: when false wake up interrupt is detected or due to DET errors.
Description	This API service is called by underlying CANIF module in case a wake up interrupt is detected. This API validates the wake up by checking the current mode of CAN Transceiver and CanTrcvWakeupByBusUsed configuration parameter, once it gets valid wake up interrupt the mode change of transceiver from standby to normal is taken care in this API. Note: Since CAN transceiver wake up indication on RxD pin is not possible to trace, the wake up validation from hardware is not done.	
Source	AUTOSAR	
Error handling	DET: CANTRCV_17_V9251_E_UNINIT: This error will occur when module API service is called without CanTrcv_17_V9251 module initialization. CANTRCV_17_V9251_E_INVALID_TRANSCEIVER: This error will occur when API called with wrong transceiver parameter for the CanTrcv_17_V9251 driver. Runtime Errors: None DEM: None	

CanTrcv_17_V9251 driver
Table 450 Specification for CanTrcv_17_V9251_CheckWakeups API (continued)

	Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

6.3.4 Notifications and Callbacks

The driver does not support any notification and callback functions.

6.3.5 Scheduled functions

Since the CAN transceiver TLE9251V does not support polling mode, therefore no scheduled functions are available.

6.3.6 Interrupt service routines

No interrupt handlers are available for this driver. Note: The CAN transceiver TLE9251V wake up interrupts are handled by the ICU driver.

6.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

6.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Table 451 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
This error is reported when CanTrcv_17_V9251_Init API is called without NULL_PTR as the parameter.	AUTOSAR	CANTRCV_17_V9251_E_INIT_FAILED=0x27	CanTrcv_17_V9251_Init
This error will occur when API called with wrong transceiver parameter for the CanTrcv_17_V9251 driver.	AUTOSAR	CANTRCV_17_V9251_E_INVALID_TRANSCEIVER=0x01	CanTrcv_17_V9251_CheckWakeups, CanTrcv_17_V9251_SetWakeupsMode, CanTrcv_17_V9251_GetBusWuReason, CanTrcv_17_V9251_GetOpMode, CanTrcv_17_V9251_SetOpMode

CanTrcv_17_V9251 driver
Table 451 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
This error will occur when API called with Invalid pointer parameter for the CanTrcv_17_V9251 driver.	AUTOSAR	CANTRCV_17_V9251_E_PARAM_POINTER=0x02	CanTrcv_17_V9251_GetVersionInfo, CanTrcv_17_V9251_GetBusWuReason, CanTrcv_17_V9251_GetOpMode
This error will occur when API service called with invalid parameter for OpMode.	AUTOSAR	CANTRCV_17_V9251_E_PARAM_TRCV_OPMODE=0x24	CanTrcv_17_V9251_SetOpMode
This error will occur when API service called with invalid parameter for TrcvWakeupMode.	AUTOSAR	CANTRCV_17_V9251_E_PARAM_TRCV_WAKEUP_MODE=0x23	CanTrcv_17_V9251_SetWakeupMode
This error will occur when module API service is called without CanTrcv_17_V9251 module initialization.	AUTOSAR	CANTRCV_17_V9251_E_UNINIT=0x11	CanTrcv_17_V9251_CheckWakeup, CanTrcv_17_V9251_SetWakeupMode, CanTrcv_17_V9251_GetBusWuReason, CanTrcv_17_V9251_GetOpMode, CanTrcv_17_V9251_SetOpMode

6.3.7.2 Production errors

The driver does not report any production errors.

6.3.7.3 Safety errors

The driver does not report any safety errors.

6.3.7.4 Runtime errors

The driver does not report any runtime errors.

6.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

6.3.8.1 Deviations

The section describes the deviations from software specification.

CanTrcv_17_V9251 driver
Table 452 Known deviations

Reference	Deviation
AUTOSAR CAN Transceiver requirement[SWS_CanTrcv_00090]	Since the TLE9251V hardware supports the wake up functionality, NOT_SUPPORTED mode is not available from the CanTrcv_17_V9251 driver.
AUTOSAR CAN Transceiver requirement[SWS_CanTrcv_00091]	Wake-up by polling mode is not supported by the CanTrcv_17_V9251 driver due to hardware limitations. Instead wake-up is supported by the interrupt mode.
AUTOSAR CAN Transceiver requirement [SWS_CanTrcv_00171], SWS_CanTrcv_00172], SWS_CanTrcv_00173]	Since the ICU driver does not depend on Icu_EnableNotification and Icu_DisableNotification for reporting a wake up, these interfaces are not used in the CanTrcv_17_V9251 driver.
AUTOSAR CAN Transceiver requirement[SWS_CanTrcv_00067]	AUTOSAR-specified file structure is modified to avoid the compilation errors and repeated file inclusions.
AUTOSAR CAN Transceiver requirement[SWS_CanTrcv_00148]	CANTRCV_TRCVMODE_SLEEP mode from AUTOSAR SWS is not supported due to hardware limitations.
AUTOSAR CAN Transceiver requirement [SWS_CanTrcv_00228]	The DEM error CANTRCV_E_BUS_ERROR is not supported due to hardware limitations.
AUTOSAR CAN Transceiver requirement [SWS_CanTrcv_00174], [SWS_CanTrcv_00175], [SWS_CanTrcv_00177], [SWS_CanTrcv_00178]	Since the CAN transceiver hardware does not support partial networking, all these requirements are not supported by the driver.

6.3.8.2 Limitations

There are no limitations for the CanTrcv_17_V9251 driver.

6.3.9 Unsupported hardware features

The following hardware feature of the CAN Transceiver TLE9251V is not supported: Forced-receive-only mode.

CanTrcv_17_W9255 driver

7 CanTrcv_17_W9255 driver

7.1 User information

7.1.1 Description

The CAN transceiver is a hardware device, which adapts the signal levels that are used on the CAN bus to the logical (digital) signal levels recognized by the microcontroller. The CAN transceiver driver supports the Infineon TLE9255W hardware. The CAN transceiver driver provides the services for:

- Driver initialization
- Switching of operation modes
- Standard bus wake-up functionality
- CAN partial networking with selective wake-up functionality

The communication between the microcontroller and the CAN transceiver is implemented through the Serial Peripheral Interface (SPI). This communication is synchronous and is configured as full duplex. Multiple CAN transceivers can be connected to the same SPI kernel.

7.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

CanTrcv_17_W9255 driver

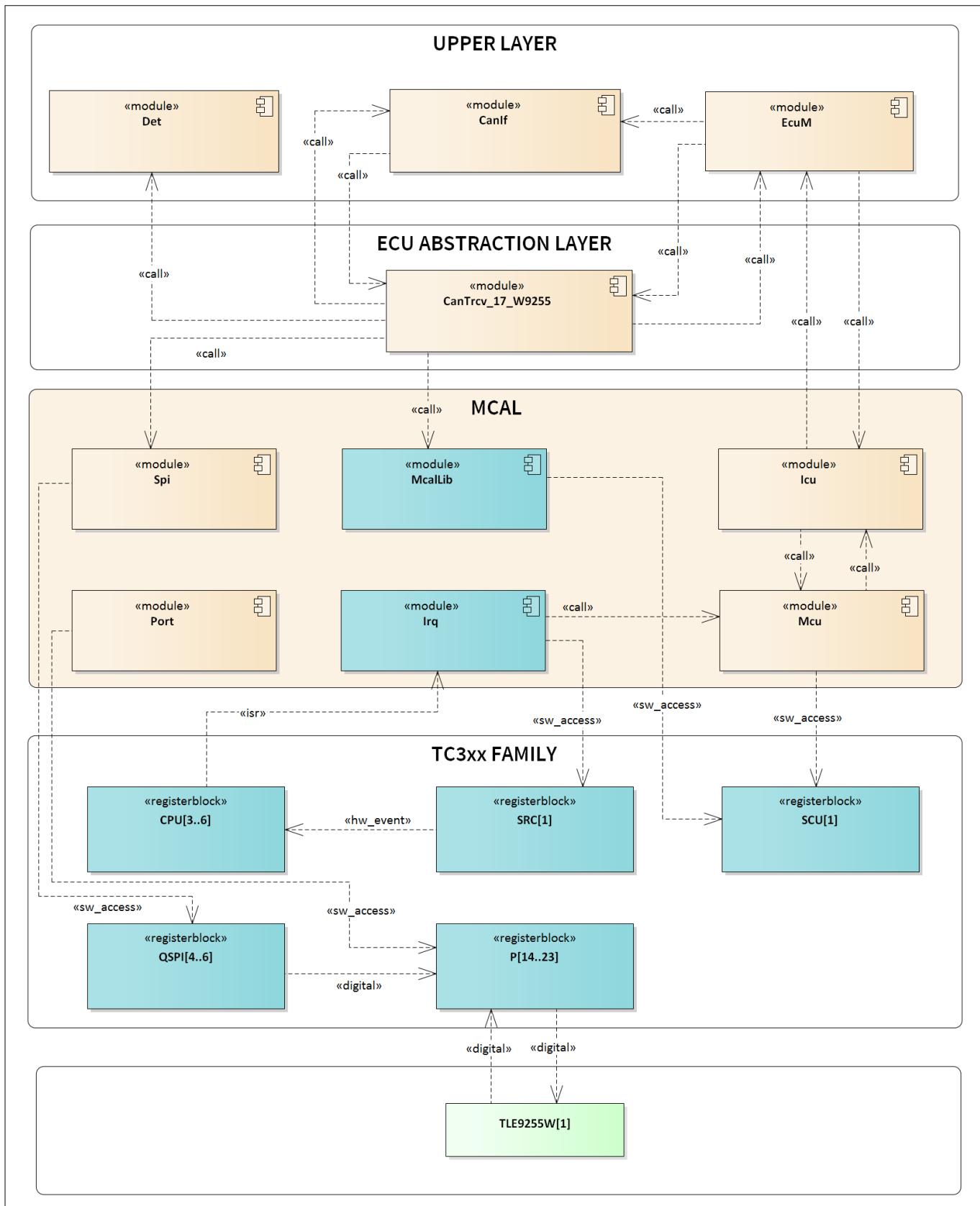


Figure 90

Mapping of hardware-software interfaces

CanTrcv_17_W9255 driver

7.1.2.1 **TLE9255W: primary hardware peripheral**

Hardware functional features

The CAN transceiver driver uses the TLE9255W to provide an interface between the physical CAN bus layer and the CAN protocol controller. The key hardware functional features used by the driver are:

- Interface between CAN controller and CAN physical bus
- CAN Flexible data rate (CAN FD) transmission up to 5 MBit/s
- Supports selective wakeup functionality where the transceiver is woken up by selective wake frames called as wake-up frames (WUFs) when the transceiver is in low power modes
- Wake-up pattern (WUP) detection in all low-power modes
- Local wake-up input
- Wake-up source recognition

The unsupported feature of the TLE9255W is:

- Receive-only mode

Users of the hardware

The CAN transceiver driver exclusively utilizes the TLE9255W module.

Hardware diagnostic features

The hardware diagnostic features used by the driver are:

- The error status register records if any SPI failure is detected or if an invalid SPI command is passed. Both these error scenarios are signaled on the MISO pin. The SPI indicates failures, error counter overflow and synchronization failures to the microcontroller. An invalid SPI command is ignored and the CMD_ERR bit is set and signaled on the MISO pin. Only the microcontroller can reset the CMD_ERR bit. On SPI failure, SPI commands are ignored.
- The SysErr flag in the selective wake status register indicates an error condition in the selective wake unit of the TLE9255W.
- Error counter status register tracks error counter overflow that can occur upon receiving invalid CAN frames.

The unsupported diagnostic features of the TLE9255W are:

- Short-circuit protection
- Undervoltage detection
- Overtemperature warning
- TxD timeout function
- CSN Timeout

Hardware events

The CAN transceiver driver uses the following hardware events from the TLE9255W IP:

Wake-up event : Indication of a valid wake-up event is signaled on the RxD pin and this triggers a mode change.

7.1.2.2 **SCU: dependent hardware peripheral**

Hardware functional features

The CAN transceiver driver depends on the SCU IP for the clock and reset functionalities.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

CanTrcv_17_W9255 driver

The SMU alarms configured for the SCU IP are not monitored by the CAN transceiver driver.

Hardware events

Hardware events from the SCU are not used by the CAN transceiver driver.

7.1.2.3 Port: dependent hardware peripheral

Hardware functional features

The digital signals are routed to the CAN transceiver hardware through the digital port pads. The port pads are configured and enabled through the PORT driver. The CAN transceiver driver depends on PORT driver for configuring the RxD, TxD, MOSI, MISO, CSN, SCLK and WAKE pins of the CAN transceiver hardware.

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Not applicable.

7.1.2.4 SRC: dependent hardware peripheral

Hardware functional features

The CAN transceiver driver depends on the ICU for interrupt handling. The ICU depends on the interrupt router for raising an interrupt to the CPU based on the wake-up events, which indicates wake-up activity on the RxD pin of the transceiver. The RxD pin is connected to the edge detection channel of the ICU.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software.

Hardware diagnostic features

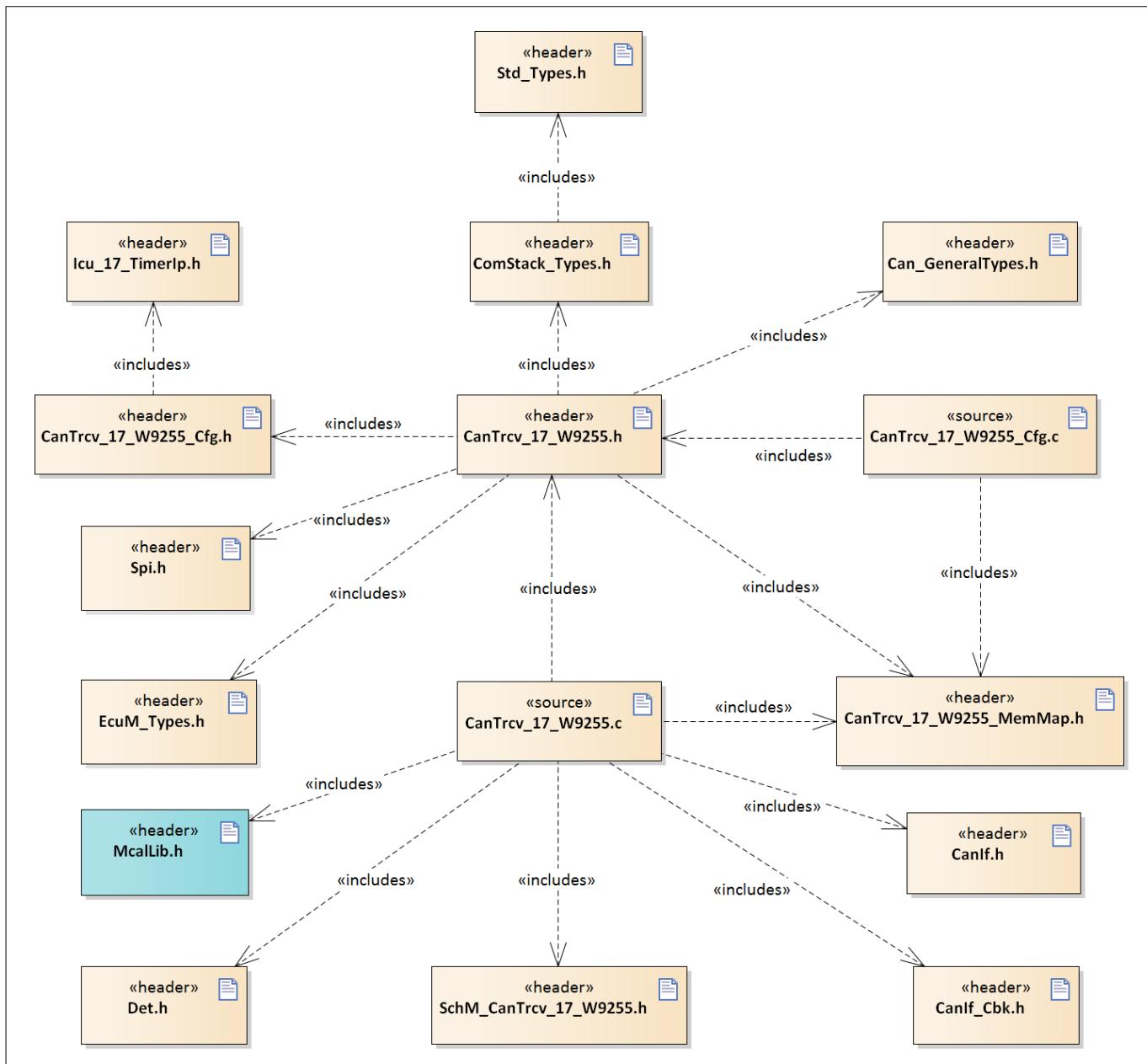
The SMU alarms configured for interrupt router are not monitored by the CAN transceiver driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU. The CAN transceiver driver depends on the ICU driver which provides interrupt handlers as software interfaces, that must be invoked from the ISR.

7.1.3 File structure

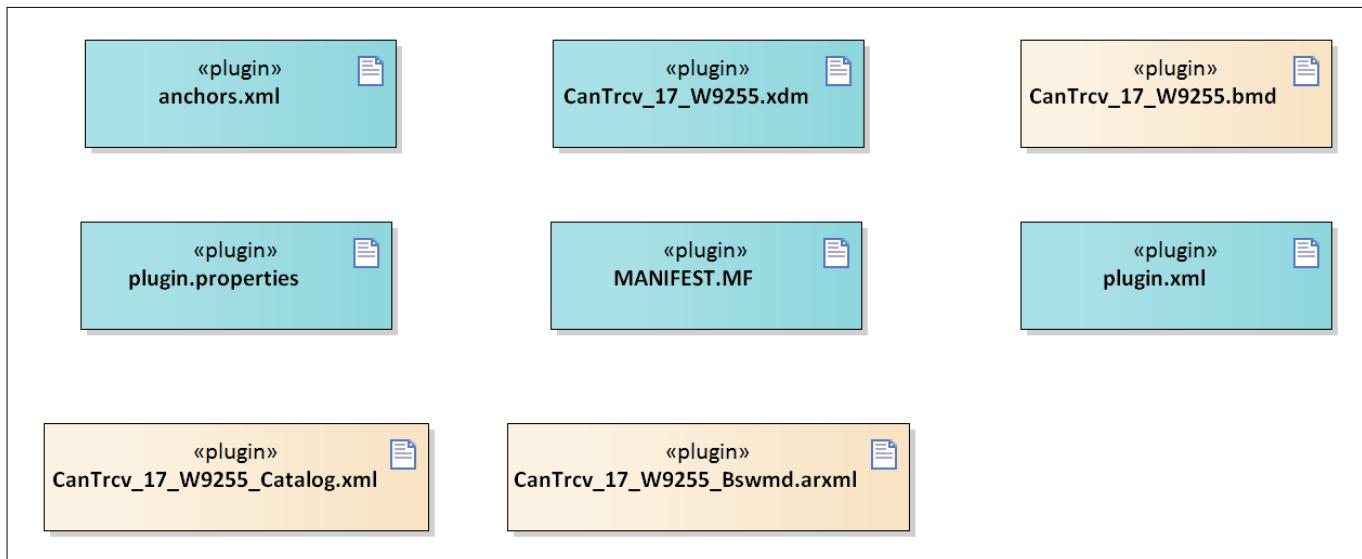
7.1.3.1 C file structure

CanTrcv_17_W9255 driver

Figure 91 **C file structure**
Table 453 **C file structure**

File name	Description
CanIf.h	Header file containing the exported interfaces of CanIf
CanIf_Cbk.h	Header file containing declarations of the CanIf callbacks
CanTrcv_17_W9255.c	File (Static) containing implementation of APIs
CanTrcv_17_W9255.h	Header file (Static) defining prototypes of data structures and APIs
CanTrcv_17_W9255_Cfg. .c	File (Generated) containing definition of the configuration data structures
CanTrcv_17_W9255_Cfg. .h	Header file (Generated) containing constants and pre-processor macros as #defines

CanTrcv_17_W9255 driver
Table 453 C file structure (continued)

File name	Description
CanTrcv_17_W9255_MemMap.h	File (Static) containing the memory section definitions used by the CAN transceiver driver
Can_GeneralTypes.h	Contains all types and constants that are shared among the AUTOSAR CAN modules Can, CanIf and CanTrcv
ComStack_Types.h	Type Definition for Com stack
Det.h	Provides the exported interfaces of Development Error Tracer
EcuM_Types.h	Header file containing type declarations of the EcuM
Icu_17_TimerIp.h	Header file (static) defining prototypes of configuration data structures and APIs
McallLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
SchM_CanTrcv_17_W9255.h	Export header for SchM functions of the CAN transceiver driver
Spi.h	Header file (Static) defining prototypes of data structures and APIs
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

7.1.3.2 Code generator plugin files

Figure 92 Code generator plugin files
Table 454 Code generator plugin files

File name	Description
CanTrcv_17_W9255.bmd	AUTOSAR format XML data model schema file
CanTrcv_17_W9255.xdm	Tresos format XML data model schema file

CanTrcv_17_W9255 driver
Table 454 Code generator plugin files (continued)

File name	Description
CanTrcv_17_W9255_Bsw md.arxml	AUTOSAR format module description file
CanTrcv_17_W9255_Cat alog.xml	AUTOSAR format catalog file as per catalog_V3_0_0.ml.xsd
MANIFEST.MF	Tresos plugin support file containing the metadata for the CAN transceiver driver
anchors.xml	Tresos anchors support file for the CAN transceiver Driver
plugin.properties	Tresos plugin support file for the CAN transceiver driver
plugin.xml	Tresos plugin support file for the CAN Transceiver driver

7.1.4 Integration hints

This section lists the key points that an integrator or user of the CanTrcv_17_W9255 driver must consider.

7.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the CAN transceiver driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of the MCAL, the EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase. Refer to the Notifications and call-backs section for the notification functions called by the transceiver to EcuM.

- **CanIf**

The CanIf module is a part of the AUTOSAR stack that provides upper layers a hardware independent interface to the CAN communication system comprising multiple CAN controllers and CAN transceivers. The `CanIf_Cbk.c` and `CanIf_Cbk.h` files are provided as stub code and needs to be replaced with complete CanIf module during integration phase. Refer to the Notifications and call-backs section for the notification functions called by the transceiver to CanIf.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `CanTrcv_17_W9255_MemMap.h` file. The `CanTrcv_17_W9255_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-

CanTrcv_17_W9255 driver

section macros. The pragmas ensure that the elements are relocated to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

```

*****GLOBAL DATA SECTION *****

#if defined CANTRCV_17_W9255_START_SEC_VAR_CLEARED_QM_LOCAL_32
/* User Pragma here */
#undef CANTRCV_17_W9255_START_SEC_VAR_CLEARED_QM_LOCAL_32
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_STOP_SEC_VAR_CLEARED_QM_LOCAL_32
/* User Pragma here */
#undef CANTRCV_17_W9255_STOP_SEC_VAR_CLEARED_QM_LOCAL_32
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_START_SEC_VAR_CLEARED_QM_LOCAL_16
/* User Pragma here */
#undef CANTRCV_17_W9255_START_SEC_VAR_CLEARED_QM_LOCAL_16
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_STOP_SEC_VAR_CLEARED_QM_LOCAL_16
/* User Pragma here */
#undef CANTRCV_17_W9255_STOP_SEC_VAR_CLEARED_QM_LOCAL_16
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_START_SEC_VAR_CLEARED_QM_LOCAL_8
/* User Pragma here */
#undef CANTRCV_17_W9255_START_SEC_VAR_CLEARED_QM_LOCAL_8
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
/* User Pragma here */
#undef CANTRCV_17_W9255_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_START_SEC_VAR_INIT_QM_LOCAL_8
/* User Pragma here */
#undef CANTRCV_17_W9255_START_SEC_VAR_INIT_QM_LOCAL_8
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_STOP_SEC_VAR_INIT_QM_LOCAL_8
/* User Pragma here */
#undef CANTRCV_17_W9255_STOP_SEC_VAR_INIT_QM_LOCAL_8
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_START_SEC_VAR_INIT_QM_LOCAL_32
/* User Pragma here */
#undef CANTRCV_17_W9255_START_SEC_VAR_INIT_QM_LOCAL_32
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_STOP_SEC_VAR_INIT_QM_LOCAL_32
/* User Pragma here */
#undef CANTRCV_17_W9255_STOP_SEC_VAR_INIT_QM_LOCAL_32
#undef MEMMAP_ERROR

***** CANTRCV_17_W9255 MODULE CONFIG DATA *****

#elif defined CANTRCV_17_W9255_START_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
/* User Pragma here */
#undef CANTRCV_17_W9255_START_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
#undef MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_STOP_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED

```

CanTrcv_17_W9255 driver

```

/* User Pragma here */
#ifndef CANTRCV_17_W9255_STOP_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_START_SEC_CONFIG_DATA_QM_LOCAL_8
/* User Pragma here */
#ifndef CANTRCV_17_W9255_START_SEC_CONFIG_DATA_QM_LOCAL_8
#define MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_STOP_SEC_CONFIG_DATA_QM_LOCAL_8
/* User Pragma here */
#ifndef CANTRCV_17_W9255_STOP_SEC_CONFIG_DATA_QM_LOCAL_8
#define MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_START_SEC_CONFIG_DATA_QM_LOCAL_16
/* User Pragma here */
#ifndef CANTRCV_17_W9255_START_SEC_CONFIG_DATA_QM_LOCAL_16
#define MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_STOP_SEC_CONFIG_DATA_QM_LOCAL_16
/* User Pragma here */
#ifndef CANTRCV_17_W9255_STOP_SEC_CONFIG_DATA_QM_LOCAL_16
#define MEMMAP_ERROR
#endif

***** CANTRCV_17_W9255 MODULE CODE SECTION *****

#elif defined CANTRCV_17_W9255_START_SEC_CODE_QM_LOCAL
/* User Pragma here */
#ifndef CANTRCV_17_W9255_START_SEC_CODE_QM_LOCAL
#define MEMMAP_ERROR
#elif defined CANTRCV_17_W9255_STOP_SEC_CODE_QM_LOCAL
/* User Pragma here */
#ifndef CANTRCV_17_W9255_STOP_SEC_CODE_QM_LOCAL
#define MEMMAP_ERROR
#endif

```

- DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The CAN transceiver driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the CAN transceiver driver must process all the errors reported to the DET module through the `Det_ReportError()` API. The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- DEM**

The DEM module is a part of the AUTOSAR stack that handles all the production errors reported by the BSW modules. The CAN transceiver driver does not report any production errors. The DEM module is not required for the integration of CAN transceiver driver.

- SchM**

The SchM module is a part of the RTE that manages the BSW Scheduler. The CAN transceiver driver uses the exclusive areas defined in the `SchM_CanTrcv_17_W9255.c` file to protect the SFRs and variables from concurrent accesses from different threads. The SchM identified for the CanTrcv_17_W9255 driver is: `SpiStatusUpdate`.

The `SchM_CanTrcv_17_W9255.h` and `SchM_CanTrcv_17_W9255.c` files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the

CanTrcv_17_W9255 driver

SchM functions defined by the CanTrcv_17_W9255 driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM function is shown as follows:

```
***** Sample implementation of SchM_CanTrcv_17_W9255.c *****

void SchM_Enter_CanTrcv_17_W9255_SpiStatusUpdate (void)
{
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}
void SchM_Exit_CanTrcv_17_W9255_SpiStatusUpdate (void)
{
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

The CAN transceiver driver does not report any safety errors.

- **Notifications and callbacks**

The CAN transceiver driver does not implement any notifications. However, the driver reports mode change confirmation, partial networking availability, confirmation of wake-up flags check and clearing of WUF flag indication through notification functions of the CanIf module. The driver also reports wake-up detection through notification functions of the EcuM module.

The driver reports the following notification functions.

`EcuM_SetWakeupEvent(EcuM_WakeupSource)`: notification that a wake-up event is detected

`CanIf_TrcvModeIndication()`: notification for a successful mode transition that was triggered for a transceiver

`CanIf_CheckTrcvWakeFlagIndication()`: notification for successful check of wake-up flags that was triggered for a transceiver

`CanIf_ClearTrcvWufFlagIndication()`: notification that the WUF flag is cleared successfully for the triggered transceiver

`CanIf_ConfirmPnAvailability()`: notification that indicates the triggered transceiver is running in the PN communication mode

- **Operating system (OS)**

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application. The OS files provided by MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

7.1.4.2 Multicore and Resource Manager

The CAN transceiver driver does not support execution on multiple cores simultaneously.

7.1.4.3 MCU support

The CAN transceiver driver is dependent on the MCU driver for the ERU channel allocation and system clock configuration. The initialization of the CAN transceiver driver must be started only after completing the MCU initialization. The following must be considered while configuring the MCU driver in the EB tresos:

Select the `McuHardwareResourceAllocationConf` container and allocate the ERU input and output channels to the ICU driver from the `McuEruAllocationConf` subcontainer.

CanTrcv_17_W9255 driver

The corresponding ERU input and output channels have to be referred in ERUInputConfiguration container in the ICU channel, which is configured for wake-up and edge detection.

7.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the CAN transceiver driver through the PORT configuration and initialize the port pins prior to invoking of CAN transceiver driver initialization. The MISO, MOSI, SCLK, CSN and WAKE pins of CAN transceiver TLE9255W must be configured in the PORT driver configuration.

7.1.4.5 DMA support

The CAN transceiver driver does not use any services provided by the DMA driver.

7.1.4.6 Interrupt connections

The CAN transceiver driver does not use any interrupt source.

CanTrcv_17_W9255 driver

7.1.4.7 Example usage

This chapter describes how the CAN transceiver driver can be configured and how to use different APIs provided by the driver. All the APIs should be provided with valid input parameters. To detect the invalid function parameters, the DET (Development Error Tracer) should be enabled. The behaviour of the APIs is undefined if DET is disabled and wrong parameters are passed.

Configuration of the driver

1. In the MCU driver, configure the system clock and the ERU channels.
2. In the PORT driver, configure port pins referred by the CAN transceiver TLE9255W. For each configured transceiver channel, MISO, MOSI, SCLK, CSN and WAKE pins have to be configured.
3. In the SPI driver, configure the required number of sequences according to the number of channels (external devices) configured in the CAN transceiver. Each transceiver channel must be configured to have one independent sequence with a job and a channel exclusively configured for a transceiver channel.
4. The MCALLIB driver configuration is required for timing services used by the CAN transceiver driver.
5. In the EcuM, configure the wake-up source reference, wake-up source reference for POR and SYSERR.
6. In the CanTrcv_17_W9255 driver, configure for required channels with Normal, Standby or Sleep modes. The `CanTrcvWakeUpByBusUsed` parameter must be enabled for wake-up support for the corresponding channel.
7. In the CanTrcv_17_W9255 driver, the `CanTrcvSpiSequenceName` parameter must be referenced to SPI channel to access the TLE9255W hardware.

In the wake-up by interrupt mode, the following additional configurations are required.

1. In the ICU driver, configure the ICU wake-up capable channel to detect the FALLING EDGE of the CAN transceiver TLE9255W RXD pin, this needs the ERU channel configuration.
2. The IRQ driver configuration is required to configure the interrupt priorities for the interrupts used by the ICU.
3. In the EcuM, configure the wake-up source and same wake-up source must be configured in the CanTrcv_17_W9255 and the ICU configuration.

Refer to the following sample configurations of SPI channel and SPI external device.

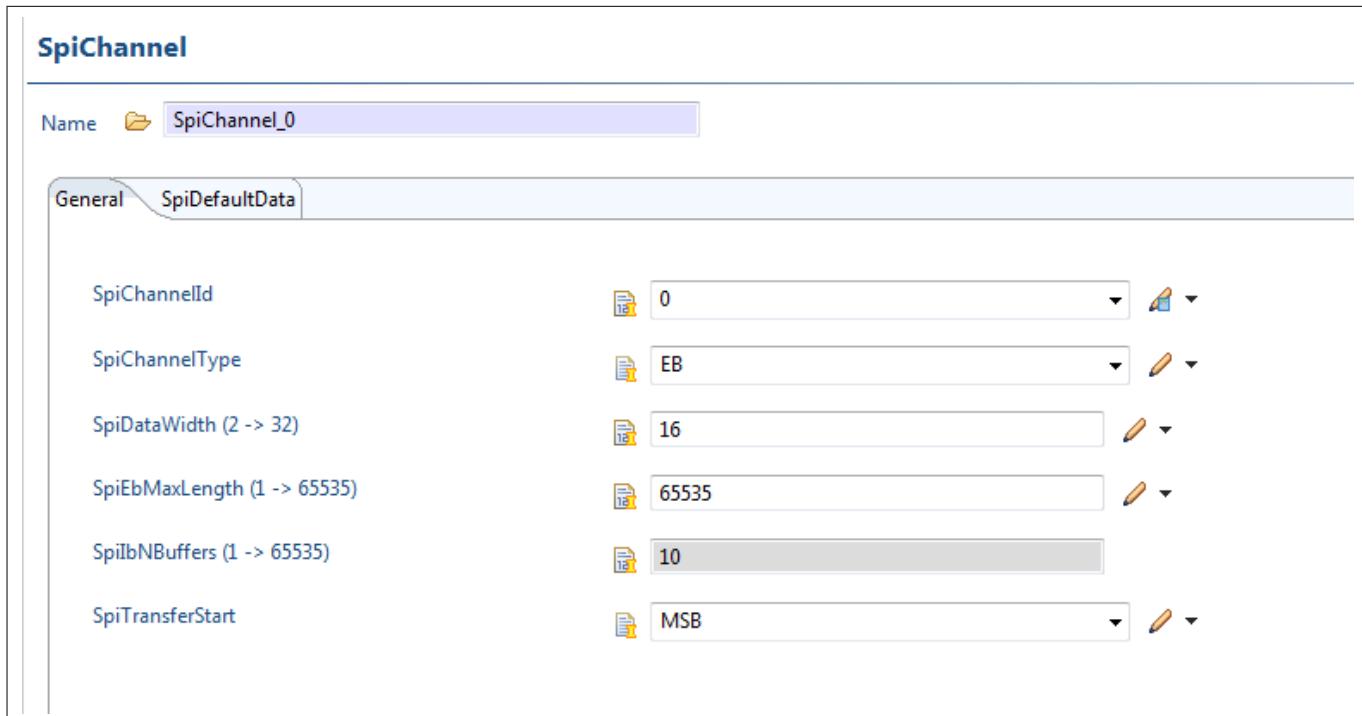
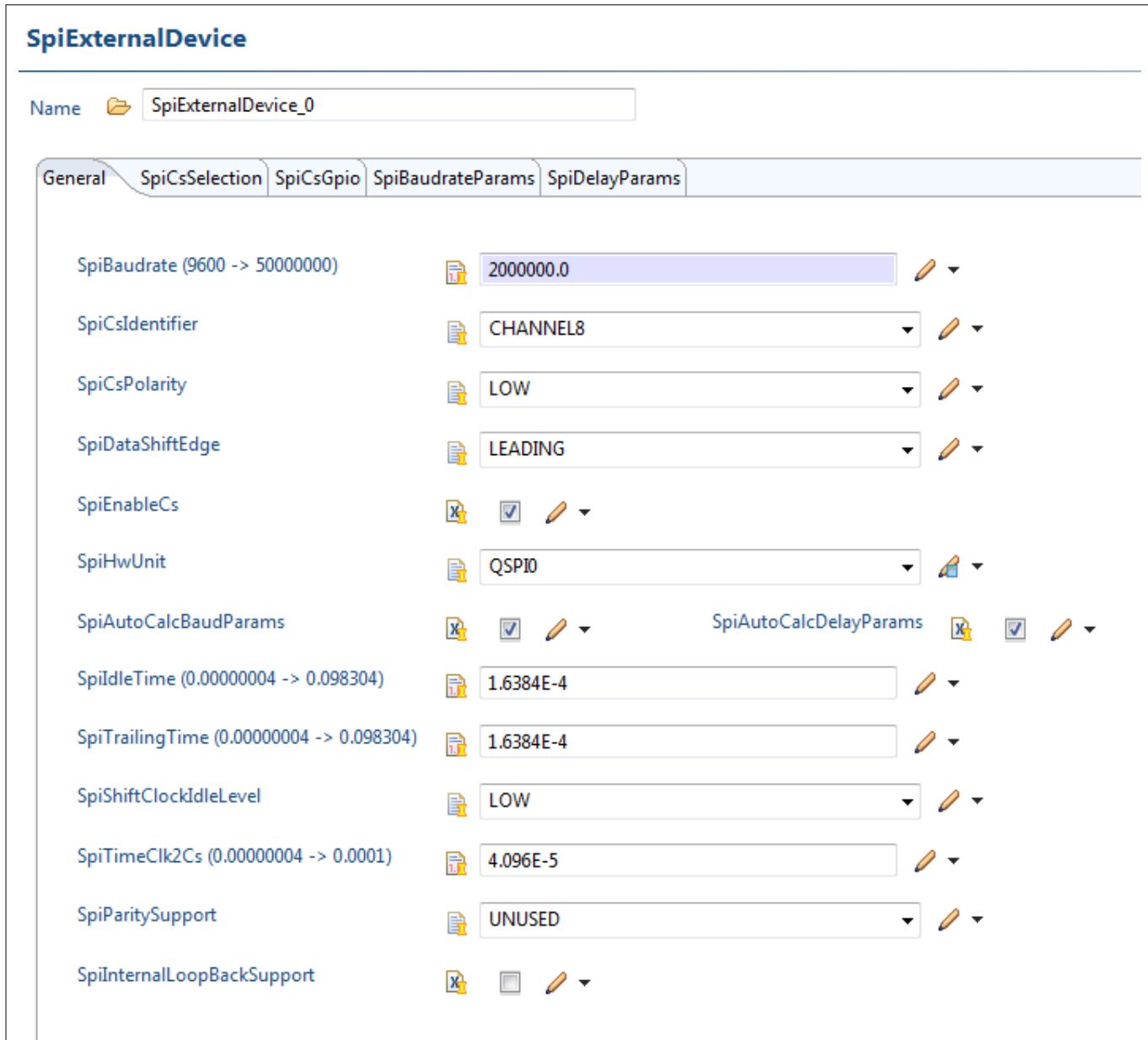


Figure 93 SPI channel configuration

CanTrcv_17_W9255 driver

Figure 94 **SPI external device configuration**
Wake-up by interrupt mode:

The CanTrcv_17_W9255 driver is dependent on the ICU driver for edge detection. The initialization of the CanTrcv_17_W9255 driver must be started only after completion of the ICU initialization. The ICU must be put to sleep mode, and wake-up for the corresponding channel has to be enabled to support the wake-up functionality.

Initialization sequence of CanTrcv_17_W9255 driver:

CanTrcv_17_W9255 driver

The Initialization sequence of the CanTrcv_17_W9255 driver is as follows.

```
/*MCU Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock ();
/* Port Initialization */
Port_Init(&Port_Config);
/* SPI Initialization */
Spi_Init(&Spi_Config);
/*ICU Initialization */
Icu_17_TimerIp_Init(&Icu_17_TimerIp_Config);
/* CanTrcv_17_W9255 Initialization */
CanTrcv_17_W9255_Init(NULL_PTR);
/* Further APIs of CanTrcv_17_W9255 driver can be called now */
```

CAN Transceiver operation mode change:

After the initialization of the CanTrcv_17_W9255 driver, the following sequence can be followed for the mode change operation.

```
/* CanTrcv_17_W9255 mode change operation */
CanTrcv_17_W9255_SetOpMode(0, CANTRCV_TRCVMODE_NORMAL);
```

CAN Transceiver wakeup mode change:

After the initialization of the CanTrcv_17_W9255 driver, the following sequence can be followed for changing the wake-up mode.

```
/* CanTrcv_17_W9255 wake-up mode change */
CanTrcv_17_W9255_SetWakeUpMode(0, CANTRCV_WUMODE_ENABLE);
```

7.1.5 Key architectural considerations

7.1.5.1 Wake-up by interrupt mode

In addition to the wake-up support by the polling mode, the CAN transceiver driver supports the detection of wake-up by the interrupt mode. This can be configured using the CanTrcvWakeUpSupport configuration parameter. In this mode, the RxD pin of the CAN Transceiver hardware is connected to the ERU. Indication of a valid wake-up event is signalled on the RxD pin by the CAN Transceiver. The ICU driver monitors the RxD pin transitions and notifies the EcuM after wake-up detection.

7.1.5.2 User mode support

Since the CAN transceiver driver does not access any AURIX SFRs, the driver does not support the user mode configuration for any of its APIs. Therefore, all APIs of the driver can be executed in the User1 or Supervisor modes.

[cover parentID CANTRCVW9255 = {D0984ABF-D8D3-49bf-9AF1-22CD9DF62F4B}]

CanTrcv_17_W9255 driver**7.1.5.3 CanTrcv_17_W9255_SetOpMode and
 CanTrcv_17_W9255_CheckWakeFlag APIs implemented as
 synchronous**

Since AUTOSAR recommends that the used APIs of the underlying driver (SPI) should be synchronous, therefore synchronous implementation is used for these APIs.

CanTrcv_17_W9255 driver

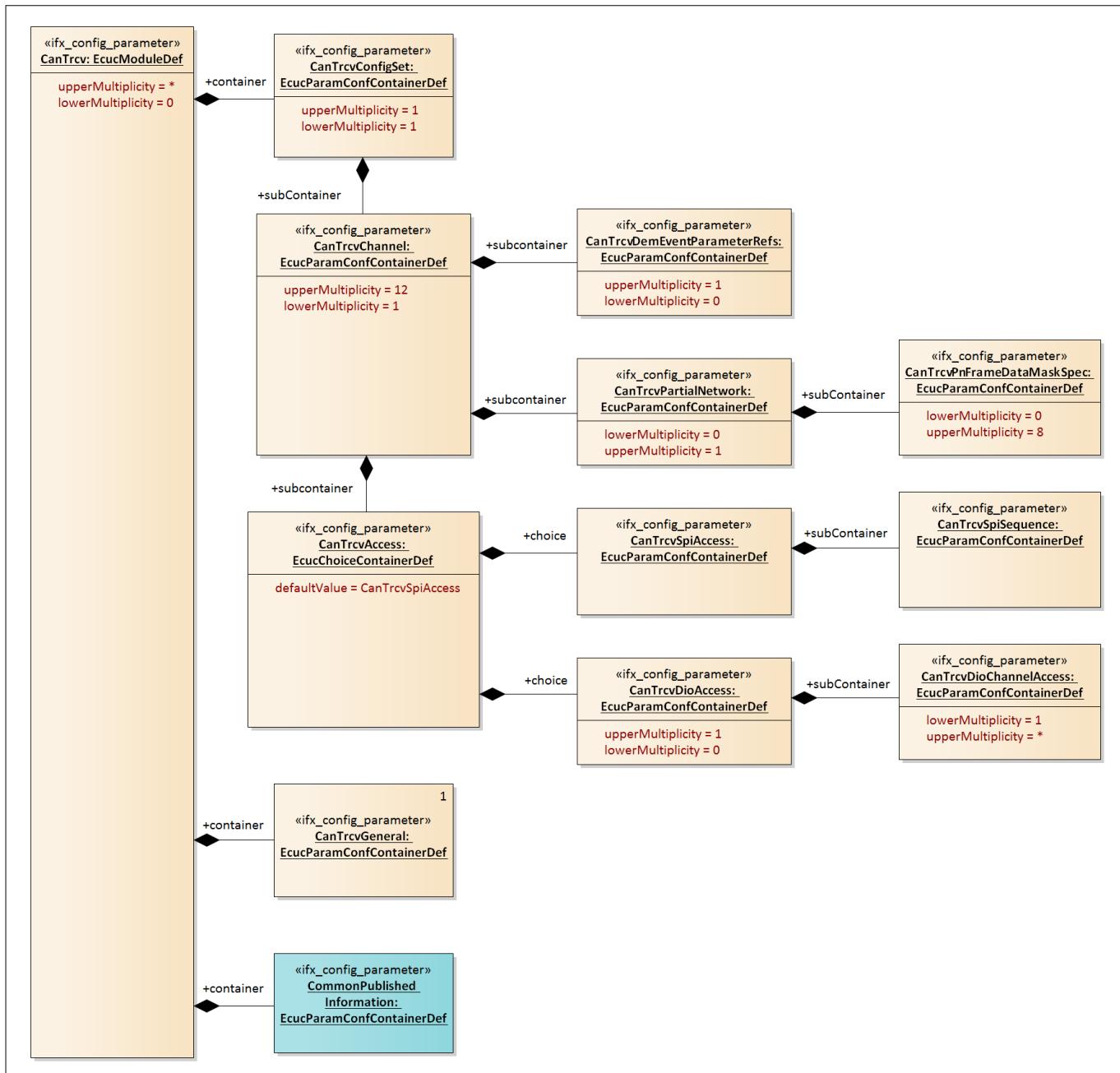
7.2 Assumptions of Use (AoUs)

There are no AoUs for the driver.

CanTrcv_17_W9255 driver

7.3 Reference information

7.3.1 Configuration interfaces


Figure 95 Container hierarchy along with their configuration parameters

7.3.1.1 Container: CanTrcvDemEventParameterRefs

This container contains the references to DemEventParameter elements which shall be invoked using the API `Dem_ReportErrorStatus` in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's `DemEventId` value. Note: Since TLE9255W hardware cannot detect bus failure, this container is not applicable and made non-editable. This configuration container is not used in the code but it is listed for AUTOSAR compatibility.

Post-Build Variant Multiplicity: FALSE

CanTrcv_17_W9255 driver

Multiplicity Configuration Class: Pre-Compile

7.3.1.1.1 CANTRCV_E_BUS_ERROR

Table 455 Specification for CANTRCV_E_BUS_ERROR

Name	CANTRCV_E_BUS_ERROR		
Description	<p>Reference to the DemEventParameter which shall be issued when bus error has occurred.</p> <p>Note: Since TLE9255W hardware cannot detect bus failure, the module does not raise any DEMs. Therefore, this parameter is not applicable and made non-editable. This configuration parameter is not used in the code but it is listed for AUTOSAR compatibility.</p> <p>Since the name of the dependent parameter is user configurable, the default value is set to NULL.</p>		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.2 Container: CommonPublished Information

This container contains the common published information of the TLE9255W CAN Transceiver driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

7.3.1.2.1 ArMajorVersion

Table 456 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	<p>Parameter provides the major version of the AUTOSAR Specification</p> <p>Default value is set to 4, as the CAN Transceiver driver module is following AUTOSAR version 4.2.2.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-

CanTrcv_17_W9255 driver
Table 456 Specification for ArMajorVersion (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.2.2 ArMinorVersion
Table 457 Specification for ArMinorVersion

Name	ArMinorVersion		
Description	Parameter provides the minor version of the AUTOSAR Specification Default value is set to 2, as the CAN Transceiver driver module is following AUTOSAR version 4.2.2.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.2.3 ArPatchVersion
Table 458 Specification for ArPatchVersion

Name	ArPatchVersion		
Description	Parameter provides the patch version of the AUTOSAR Specification Default value is set to 2, as the CAN Transceiver driver module is following AUTOSAR version 4.2.2.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-

CanTrcv_17_W9255 driver
Table 458 Specification for ArPatchVersion (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.2.4 ModuleId
Table 459 Specification for ModuleId

Name	ModuleId		
Description	Parameter provides the Module Id Default value is set to 70, as this is the CAN Transceiver driver module ID		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	70		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.2.5 Release
Table 460 Specification for Release

Name	Release		
Description	Specifies the derive for which the configuration project is created. Default value is derived from the property file and represents the hardware derivative of the micro controller for which the CAN Transceiver driver is being configured.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per hardware derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

CanTrcv_17_W9255 driver
Table 460 Specification for Release (continued)

Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.2.6 SwMajorVersion
Table 461 Specification for SwMajorVersion

Name	SwMajorVersion		
Description	Parameter provides the major version of the Software		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.2.7 SwMinorVersion
Table 462 Specification for SwMinorVersion

Name	SwMinorVersion		
Description	Parameter provides the minor version of the Software		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

CanTrcv_17_W9255 driver**7.3.1.2.8 SwPatchVersion****Table 463 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Parameter provides the patch version of the Software		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.2.9 VendorApiInfix**Table 464 Specification for VendorApiInfix**

Name	VendorApiInfix		
Description	The parameter is used to specify the vendor specific name. Default value is set to W9255, as this is the unique name of the CAN Transceiver driver module provided by IFX.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	W9255		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.2.10 VendorId**Table 465 Specification for VendorId**

Name	VendorId		
Description	Parameter provides the Vendor Id		

CanTrcv_17_W9255 driver
Table 465 Specification for VendorId (continued)

	Default value is set to 17, as this is the IFX vendor ID.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.3 Container: CanTrcv

Configuration of the CAN Transceiver driver module.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

7.3.1.3.1 Config Variant

Table 466 Specification for Config Variant

Name	Config Variant		
Description	Selects the config-variant for the CAN Transceiver module. The default value of this parameter is set to VariantPreCompile as per AUTOSAR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VariantPreCompile: PreCompile Support		
Default value	VariantPreCompile		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

7.3.1.4 Container: CanTrcvChannel

This container gives CAN transceiver driver information about a single CAN transceiver (channel).

Post-Build Variant Multiplicity: FALSE

CanTrcv_17_W9255 driver

Multiplicity Configuration Class: Pre-Compile

7.3.1.4.1 CanTrcvAccess

Table 467 Specification for CanTrcvAccess

Name	CanTrcvAccess		
Description	This container gives CAN Transceiver Driver information about access to a single CAN transceiver. Note: Since TLE9255W hardware supports only SPI interface, the container CanTrcvSpiAccess is set as the default choice.		
Multiplicity	1..1	Type	EcucChoiceContainerDef
Range			
Default value	CanTrcvSpiAccess		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.4.2 CanTrcvChannelId

Table 468 Specification for CanTrcvChannelId

Name	CanTrcvChannelId		
Description	Unique identifier of the CAN Transceiver channel. Note: The channel Id should be less than the number of channels configured. Minimum channel Id is selected as the default value. Note: Range of Channel Id is modified as 0-11 since number of CAN nodes supported in TC3xx is limited to 12.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 11		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU

CanTrcv_17_W9255 driver**Table 468 Specification for CanTrcvChannelId (continued)**

Dependency	-
-------------------	---

7.3.1.4.3 CanTrcvChannelUsed**Table 469 Specification for CanTrcvChannelUsed**

Name	CanTrcvChannelUsed		
Description	This parameter specifies if the respective CAN transceiver channel is enabled or not.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.4.4 CanTrcvControlsPowerSupply**Table 470 Specification for CanTrcvControlsPowerSupply**

Name	CanTrcvControlsPowerSupply		
Description	Indicates if the ECU power supply is controlled by the transceiver. TRUE = Controlled by the transceiver FALSE = Not controlled by the transceiver Note: Since TLE9255W hardware does not control the ECU power supply, this parameter is set FALSE by default and made non-editable.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

CanTrcv_17_W9255 driver
Table 470 Specification for CanTrcvControlsPowerSupply (continued)

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.4.5 CanTrcvHwPnSupport
Table 471 Specification for CanTrcvHwPnSupport

Name	CanTrcvHwPnSupport		
Description	<p>Indicates whether TLE9255W hardware supports the selective wake-up feature.</p> <p>TRUE = Selective wakeup feature is supported by the transceiver</p> <p>FALSE = Selective wakeup feature is not supported by the transceiver</p> <p>Note: Since the wakeup is always supported either by polling or by interrupt, this parameter is not dependent on CanTrcvWakeUpSupport .</p> <p>Note: Since TLE9255W hardware supports PN, this parameter is set TRUE by default and made non-editable.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.4.6 CanTrcvIcuChannelRef
Table 472 Specification for CanTrcvIcuChannelRef

Name	CanTrcvIcuChannelRef		
Description	<p>Reference to the ICU channel for detecting the wakeups.</p> <p>Since the name of the dependent parameter is user configurable, the default value is set to NULL.</p>		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: IcuChannel		
Default value	NULL		

CanTrcv_17_W9255 driver
Table 472 Specification for CanTrcvIcuChannelRef (continued)

Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.4.7 CanTrcvInitState
Table 473 Specification for CanTrcvInitState

Name	CanTrcvInitState		
Description	State of CAN transceiver after call to CanTrcv_17_W9255_Init. Note: Normal mode is set as default mode since the CAN messages can be transmitted and received after driver initialization.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CANTRCV_17_W9255_OP_MODE_NORMAL: Normal operation mode CANTRCV_17_W9255_OP_MODE_SLEEP: Sleep operation mode CANTRCV_17_W9255_OP_MODE_STANDBY: Standby operation mode		
Default value	CANTRCV_17_W9255_OP_MODE_NORMAL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.4.8 CanTrcvMaxBaudrate
Table 474 Specification for CanTrcvMaxBaudrate

Name	CanTrcvMaxBaudrate		
Description	Max baudrate for transceiver hardware type. Unit of the parameter is kbps. The parameter is made non-editable and the default value is set to 5000 as it is the maximum value supported. Note: This parameter is unused by the CAN transceiver driver.		
Multiplicity	1..1	Type	EcucIntegerParamDef

CanTrcv_17_W9255 driver
Table 474 Specification for CanTrcvMaxBaudrate (continued)

Range	0 - 5000		
Default value	5000		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.4.9 CanTrcvPorWakeupSourceRef
Table 475 Specification for CanTrcvPorWakeupSourceRef

Name	CanTrcvPorWakeupSourceRef		
Description	<p>This parameter contains symbolic name reference to specify the wakeup source for this channel that should be used in the calls to EcuM_SetWakeupEvent if POR flag is set in the TLE9255W hardware.</p> <p>Since the name of the dependent parameter is user configurable, the default value is set to NULL.</p> <p>Note: Multiplicity is modified from 0-1 to 1-1 since TLE9255W hardware supports PN.</p>		
Multiplicity	1..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: EcuMWakeupSource		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

7.3.1.4.10 CanTrcvSyserrWakeupSourceRef
Table 476 Specification for CanTrcvSyserrWakeupSourceRef

Name	CanTrcvSyserrWakeupSourceRef		
Description	<p>This parameter contains symbolic name reference to specify the wakeup source for this channel that should be used in the calls to EcuM_SetWakeupEvent if SYSERR flag is set in the TLE9255W hardware.</p>		

CanTrcv_17_W9255 driver**Table 476 Specification for CanTrcvSyserrWakeupSourceRef (continued)**

	Since the name of the dependent parameter is user configurable, the default value is set to NULL. Note: Multiplicity is modified from 0-1 to 1-1 since TLE9255W hardware supports PN.		
Multiplicity	1..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: EcuMWakeupSource		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

7.3.1.4.11 CanTrcvWakeupByBusUsed**Table 477 Specification for CanTrcvWakeupByBusUsed**

Name	CanTrcvWakeupByBusUsed		
Description	<p>Indicates whether wake up by bus functionality is enabled or not.</p> <p>This parameter does not depend on CanTrcvWakeUpSupport since the wake-up by bus functionality is always supported by the transceiver and can be enabled/disabled at channel level.</p> <p>Note: WUP, WUF and LWU events are not reported if this parameter is FALSE.</p> <p>Note: Multiplicity is modified from 0-1 to 1-1 since TLE9255W hardware supports wake-up by bus functionality.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

CanTrcv_17_W9255 driver
7.3.1.4.12 CanTrcvWakeupSourceRef
Table 478 Specification for CanTrcvWakeupSourceRef

Name	CanTrcvWakeupSourceRef		
Description	<p>This parameter contains a reference to the wakeup source for this channel in the EcuM configuration.</p> <p>Implementation Type: reference to EcuM_WakeupSourceType</p> <p>This reference is only needed if CanTrcvWakeupByBusUsed is true.</p> <p>Since the name of the dependent parameter is user configurable, the default value is set to NULL.</p>		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: EcuMWakeupSource		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	CanTrcvWakeupByBusUsed		

7.3.1.5 Container: CanTrcvConfigSet

This container contains the configuration parameters and sub containers of the AUTOSAR CanTrcv_17_W9255 module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

7.3.1.5.1 CanTrcvSPICommRetries
Table 479 Specification for CanTrcvSPICommRetries

Name	CanTrcvSPICommRetries		
Description	<p>Indicates the maximum number of communication retries in case of a failed SPI communication.</p> <p>If configured value is '0', no retry is allowed.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-

CanTrcv_17_W9255 driver
Table 479 Specification for CanTrcvSPICommRetries (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	CanTrcvSpiSequenceName		

7.3.1.5.2 CanTrcvSPICommTimeout
Table 480 Specification for CanTrcvSPICommTimeout

Name	CanTrcvSPICommTimeout		
Description	<p>Indicates the maximum time allowed to the CAN Transceiver for replying to an SPI command. Timeout is configured in milliseconds. Timeout value of '0' means that no specific timeout is to be used by CAN Transceiver and the communication is executed at the best of the SPI hardware capacity.</p> <p>Note: This parameter is made non-editable as synchronous implementation of SPI driver is used.</p> <p>Note: Since this parameter is non-editable, there is no dependency on CanTrcvSpiSequence parameter.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 100		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.6 Container: CanTrcvDioAccess

Container gives CAN transceiver driver information about accessing ports and port pins. In addition relation between CAN transceiver hardware pin names and DIO port access information is given. If CAN transceiver hardware has no DIO interface, there is no instance of this container. Note: Since TLE9255W transceiver hardware has no DIO interface, there is no instance of this container and its parameters. This configuration container and its sub-containers and parameters are not used in the code but it is listed for AUTOSAR compatibility.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

CanTrcv_17_W9255 driver

7.3.1.7 Container: CanTrcvDioChannelAccess

Container gives DIO channel access by single CAN transceiver channel. Note: Since TLE9255W transceiver hardware has no DIO interface, there is no instance of this container.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

7.3.1.7.1 CanTrcvDioSymNameRef

Table 481 Specification for CanTrcvDioSymNameRef

Name	CanTrcvDioSymNameRef		
Description	This parameter gives the reference to a configured DIO channel.		
Multiplicity	1..1	Type	EcucChoiceReferenceDef
Range	Reference to Node: DioChannel		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.7.2 CanTrcvHardwareInterfaceName

Table 482 Specification for CanTrcvHardwareInterfaceName

Name	CanTrcvHardwareInterfaceName		
Description	This parameter specifies CAN transceiver hardware interface name. It is typically the name of a CAN transceiver pin.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

CanTrcv_17_W9255 driver

7.3.1.8 Container: CanTrcvGeneral

This container gives basic information about CAN transceiver driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

7.3.1.8.1 CanTrcvDevErrorDetect

Table 483 Specification for CanTrcvDevErrorDetect

Name	CanTrcvDevErrorDetect		
Description	Parameter enables or disables the Default Error Tracer (DET) detection and reporting. Default value is set to true, as this helps to detect development errors.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.8.2 CanTrcvGetVersionInfo

Table 484 Specification for CanTrcvGetVersionInfo

Name	CanTrcvGetVersionInfo		
Description	Parameter adds or removes the API CanTrcv_17_W9255_GetVersionInfo() from the code. The default value of this parameter is set to false to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

CanTrcv_17_W9255 driver
Table 484 Specification for CanTrcvGetVersionInfo (continued)

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.8.3 CanTrcvIndex
Table 485 Specification for CanTrcvIndex

Name	CanTrcvIndex		
Description	Specifies the Instance Id of this module instance. If only one instance is present it shall have the Id 0. Default value is set to 0 as it is the minimum value supported.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.8.4 CanTrcvMainFunctionDiagnosticsPeriod
Table 486 Specification for CanTrcvMainFunctionDiagnosticsPeriod

Name	CanTrcvMainFunctionDiagnosticsPeriod		
Description	This parameter describes the period for cyclic call to CanTrcv_17_W9255_MainFunctionDiagnostics. Unit of the parameter is seconds. Note: Since CanTrcv_17_W9255_MainFunctionDiagnostics API is not provided by the driver, this parameter is not applicable and made non-editable.		
Multiplicity	0..1	Type	EcucFloatParamDef
Range	0.001 - 65.535		
Default value	0.001		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL

CanTrcv_17_W9255 driver
Table 486 Specification for CanTrcvMainFunctionDiagnosticsPeriod (continued)

Dependency	-
-------------------	---

7.3.1.8.5 CanTrcvMainFunctionPeriod
Table 487 Specification for CanTrcvMainFunctionPeriod

Name	CanTrcvMainFunctionPeriod		
Description	<p>This parameter describes the period for cyclic call to CanTrcv_17_W9255_MainFunction. Unit of the parameter is seconds.</p> <p>It is advisory for all the communication modules to set the default value of this parameter to 0.005 seconds.</p>		
Multiplicity	0..1	Type	EcucFloatParamDef
Range	0.001 - 65.535		
Default value	0.005		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.8.6 CanTrcvTimerType
Table 488 Specification for CanTrcvTimerType

Name	CanTrcvTimerType		
Description	<p>Type of the Time Service Predefined Timer.</p> <p>Default value of this parameter is set to 'None' since McalLib APIs are used to realize the wait time. The parameter is made non-editable.</p>		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	None: No timer configured Timer_1us16bit: 16 bit 1us timer		
Default value	None		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL

CanTrcv_17_W9255 driver
Table 488 Specification for CanTrcvTimerType (continued)

Dependency	-
-------------------	---

7.3.1.8.7 CanTrcvWaitTime
Table 489 Specification for CanTrcvWaitTime

Name	CanTrcvWaitTime		
Description	<p>Wait time for transceiver mode changes. Unit of the parameter is seconds.</p> <p>The minimum and default values are set to 20 micro seconds as it is the worst case wait time needed for a mode change. The parameter is made non-editable.</p> <p>Note: The lower multiplicity of this parameter is set to 1 as the transceiver needs time for mode change.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.000020 - 0.000255		
Default value	0.000020		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.8.8 CanTrcvWakeUpSupport
Table 490 Specification for CanTrcvWakeUpSupport

Name	CanTrcvWakeUpSupport		
Description	<p>Informs whether wake up is supported by polling or interrupt.</p> <p>Note: CANTRCV_17_W9255_WAKEUP_NOT_SUPPORTED is not provided, since the TLE9255W hardware supports wake up functionality.</p> <p>Note: A new option CANTRCV_17_W9255_WAKE_UP_BY_INTERRUPT is added which supports wake-up by interrupt functionality.</p> <p>Note: CanTrcv_17_W9255_MainFunction is available only in the case of wakeup support by polling.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CANTRCV_17_W9255_WAKE_UP_BY_INTERRUPT: Wake up by interrupt CANTRCV_17_W9255_WAKE_UP_BY_POLLING: Wake up by polling		
Default value	CANTRCV_17_W9255_WAKE_UP_BY_POLLING		

CanTrcv_17_W9255 driver
Table 490 Specification for CanTrcvWakeUpSupport (continued)

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.9 Container: CanTrcvPartialNetwork

This container gives CAN transceiver driver information about the configuration of Partial Networking functionality. This configuration container always exists for every channel, since parameter CanTrcvHwPnSupport is always set TRUE and made non-editable.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

7.3.1.9.1 CanTrcvBaudRate

Table 491 Specification for CanTrcvBaudRate

Name	CanTrcvBaudRate		
Description	Baud rate to be set for PN frame wake-up. Unit of the parameter is kbps. TLE9255W hardware supports the following baud rates for the Selective Wake unit: 125 kbit/s, 250 kbit/s, 500 kbit/s and 1Mbit/s. Default value of this parameter is set to 500kbps since CAN standard messages are usually of the same baudrate value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	125 - 1000		
Default value	500		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.9.2 CanTrcvBusErrFlag

Table 492 Specification for CanTrcvBusErrFlag

Name	CanTrcvBusErrFlag
-------------	-------------------

CanTrcv_17_W9255 driver**Table 492 Specification for CanTrcvBusErrFlag (continued)**

Description	Indicates if the Bus Error (BUSERR) flag is managed by the BSW. This flag is set if a bus failure is detected by the transceiver. TRUE = Supported by transceiver and managed by BSW FALSE = Not managed by BSW Note: Since TLE9255W hardware cannot detect bus error, this parameter is not applicable. Hence, this parameter is set FALSE by default and made non-editable. This configuration parameter is not used in the code but it is listed for AUTOSAR compatibility.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.9.3 CanTrcvPnCanIdIsExtended**Table 493 Specification for CanTrcvPnCanIdIsExtended**

Name	CanTrcvPnCanIdIsExtended		
Description	Indicates whether extended or standard CAN Id is used. TRUE = Extended CAN identifier is used FALSE = Standard CAN identifier is used		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

CanTrcv_17_W9255 driver**7.3.1.9.4 CanTrcvPnEnabled****Table 494 Specification for CanTrcvPnEnabled**

Name	CanTrcvPnEnabled		
Description	<p>Indicates whether the selective wake-up functionality is enabled or disabled in CAN Transceiver hardware.</p> <p>TRUE = Selective wakeup feature is enabled in the transceiver hardware FALSE = Selective wakeup feature is disabled in the transceiver hardware</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.9.5 CanTrcvPnFrameCanId**Table 495 Specification for CanTrcvPnFrameCanId**

Name	CanTrcvPnFrameCanId		
Description	<p>CAN ID of the Wake-up Frame</p> <p>Default value is set to 0 as it is the minimum value supported.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

CanTrcv_17_W9255 driver

7.3.1.9.6 **CanTrcvPnPnFrameCanIdMask**

Table 496 Specification for CanTrcvPnPnFrameCanIdMask

Name	CanTrcvPnPnFrameCanIdMask		
Description	ID Mask for the selective activation of the CAN transceiver. It is used to enable WUF on a group of IDs. Default value is set to 4294967295 as it activates WUF only on one ID.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	4294967295		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.9.7 **CanTrcvPnPnFrameDlc**

Table 497 Specification for CanTrcvPnPnFrameDlc

Name	CanTrcvPnPnFrameDlc		
Description	Indicates the Data Length of the WUF. Default value of this parameter is set to 1 as recommended by AUTOSAR. Note: Minimum value of the range is deviated from AUTOSAR requirement and changed to 1, since AUTOSAR SWS states "Although WUF with DLC=0 is technically possible, it is explicitly not wanted" in CanTrcvBaudRate parameter dependency.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 8		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

CanTrcv_17_W9255 driver

7.3.1.9.8 CanTrcvPowerOnFlag

Table 498 Specification for CanTrcvPowerOnFlag

Name	CanTrcvPowerOnFlag		
Description	<p>Indicates if the Power On Reset (POR) flag is available and is managed by the transceiver.</p> <p>TRUE = Supported by Hardware</p> <p>FALSE = Not supported by Hardware</p> <p>Since Power On Reset (POR) flag is available and is managed by the transceiver, this parameter is set TRUE by default and made non-editable.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.10 Container: CanTrcvPnFrameDataMaskSpec

Defines data mask to be used on the received CAN frames in order to determine if the transceiver must be woken up by the received Wake-up Frame. Note: Since the minimum value of CanTrcvPnFrameDlc is 1, at least one data mask needs to be configured if PN is enabled.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

7.3.1.10.1 CanTrcvPnFrameDataMask

Table 499 Specification for CanTrcvPnFrameDataMask

Name	CanTrcvPnFrameDataMask		
Description	<p>Defines the data mask of the WUF at the configured index.</p> <p>Default value is set to 255 as this allows a wide range of data.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	255		
Post-build variant value	TRUE	Post-build variant multiplicity	-

CanTrcv_17_W9255 driver
Table 499 Specification for CanTrcvPnFrameDataMask (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.10.2 CanTrcvPnFrameDataMaskIndex
Table 500 Specification for CanTrcvPnFrameDataMaskIndex

Name	CanTrcvPnFrameDataMaskIndex		
Description	Holds the index of the data mask in the configured WUF. Default value is set to 0 as it is the minimum value supported.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.11 Container: CanTrcvSpiAccess

Container gives CAN transceiver driver information about accessing SPI. Note: Multiplicity is modified from 0-1 to 1-1 since TLE9255W hardware transceiver hardware uses SPI interface for communication.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

7.3.1.12 Container: CanTrcvSpiSequence

Container gives CAN transceiver driver information about one SPI sequence. Note: Multiplicity is modified from 1-* to 1-1 since one sequence is enough for one transceiver channel for communication.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

7.3.1.12.1 CanTrcvSpiAccessSynchronous
Table 501 Specification for CanTrcvSpiAccessSynchronous

Name	CanTrcvSpiAccessSynchronous
-------------	-----------------------------

CanTrcv_17_W9255 driver
Table 501 Specification for CanTrcvSpiAccessSynchronous (continued)

Description	<p>This parameter is used to define whether the access to the SPI sequence is synchronous or asynchronous.</p> <p>TRUE: SPI access is synchronous FALSE: SPI access is asynchronous</p> <p>This parameter is set to true by default and made non-editable as the CAN Transceiver driver always accesses SPI synchronously.</p> <p>Note: Multiplicity is modified from 0-1 to 1-1 since TLE9255W hardware transceiver hardware uses SPI interface for communication.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

7.3.1.12.2 CanTrcvSpiSequenceName
Table 502 Specification for CanTrcvSpiSequenceName

Name	CanTrcvSpiSequenceName		
Description	<p>Reference to a SPI sequence configuration container.</p> <p>Since the name of the dependent parameter is user configurable, the default value is set to NULL.</p> <p>The lower multiplicity of this parameter is set to 1 since TLE9255W transceiver hardware uses SPI interface for communication and needs at least one sequence per channel. The upper multiplicity is set to 1 since one sequence is enough for one transceiver channel for communication.</p>		
Multiplicity	1..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: SpiSequence		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-

CanTrcv_17_W9255 driver
Table 502 Specification for CanTrcvSpiSequenceName (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiSequence		

7.3.2 Functions - Type definitions

7.3.2.1 CanTrcv_17_W9255_ConfigType

Table 503 Specification for CanTrcv_17_W9255_ConfigType

Syntax	CanTrcv_17_W9255_ConfigType
Type	void
File	CanTrcv_17_W9255.h
Range	None
Description	This is the type of the external data structure containing the overall initialization data for the CAN transceiver driver and settings affecting all transceivers. Note: Since CanTrcv_17_W9255 driver is implemented as a Pre-Compile module, this type is implemented as of type void as the module supports single configuration variant.
Source	IFX

7.3.2.2 CanTrcv_17_W9255_PNActivationType

Table 504 Specification for CanTrcv_17_W9255_PNActivationType

Syntax	CanTrcv_17_W9255_PNActivationType	
Type	Enumeration	
File	CanTrcv_17_W9255.h	
Range	0 - CANTRCV_17_W9255_PN_DISABLED 1 - CANTRCV_17_W9255_PN_ENABLED	PN wakeup functionality in CAN Transceiver is disabled. PN wakeup functionality in CAN Transceiver is enabled.
Description	Datatype used for describing whether PN wakeup functionality in the CAN Transceiver is enabled or disabled.	
Source	AUTOSAR	

CanTrcv_17_W9255 driver

7.3.2.3 **CanTrcv_17_W9255_TrsvFlagStateType**

Table 505 Specification for CanTrcv_17_W9255_TrsvFlagStateType

Syntax	CanTrcv_17_W9255_TrsvFlagStateType	
Type	Enumeration	
File	CanTrcv_17_W9255.h	
Range	0 - CANTRCV_17_W9255_FLAG_CLEARED	The flag is cleared in the transceiver hardware.
	1 - CANTRCV_17_W9255_FLAG_SET	The flag is set in the transceiver hardware.
Description	Provides the state of a flag in the transceiver hardware.	
Source	AUTOSAR	

7.3.3 **Functions - APIs**

This section lists all the APIs of the driver.

7.3.3.1 **CanTrcv_17_W9255_Init**

Table 506 Specification for CanTrcv_17_W9255_Init API

Syntax	<pre>void CanTrcv_17_W9255_Init (const CanTrcv_17_W9255_ConfigType * const ConfigPtr)</pre>	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to driver configuration. Note: Since CAN Transceiver is implemented as a pre-compile module, a null pointer shall be passed as the parameter by the caller of this API.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This API initializes all the connected CAN Transceivers. The registers of the TLE9255W hardware are initialized as per the configuration. The CAN Transceiver initialization status is set at the end of the initialization function execution.	
Source	AUTOSAR	
Error handling		

CanTrcv_17_W9255 driver
Table 506 Specification for CanTrcv_17_W9255_Init API (continued)

	<p>DET:</p> <p>CANTRCV_17_W9255_E_NO_TRCV_CONTROL: Error is reported when there is no/incorrect communication to transceiver.</p> <p>CANTRCV_17_W9255_E_INIT_FAILED: Error is reported if initialization of the driver has failed. Since it is a Pre-compile module, the init function expects a NULL pointer to be passed as parameter. This DET is reported if a non-null pointer is passed as a parameter during init.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

7.3.3.2 CanTrcv_17_W9255_SetOpMode
Table 507 Specification for CanTrcv_17_W9255_SetOpMode API

Syntax	<pre>Std_ReturnType CanTrcv_17_W9255_SetOpMode (const uint8 Transceiver, const CanTrcv_TrcvModeType OpMode)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different transceivers	
Parameters (in)	Transceiver OpMode	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11. This parameter contains the desired operating mode
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: If the mode is changed successfully E_NOT_OK: If there is SPI communication failure or a development error occurs.
Description	<p>This API sets the mode of the requested Transceiver to the value OpMode.</p> <p>If PN is enabled, the API checks for POR and SYSERR flags. If POR flag is set, transceiver is reinitialized and if SYSERR flag is set, transceiver is reinitialized for PN functionality.</p>	
Source	AUTOSAR	

CanTrcv_17_W9255 driver
Table 507 Specification for CanTrcv_17_W9255_SetOpMode API (continued)

Error handling	<p>DET:</p> <p>CANTRCV_17_W9255_E_PARAM_TRCV_OPMODE: Error is reported when the API service is called with invalid parameter for OpMode.</p> <p>CANTRCV_17_W9255_E_NO_TRCV_CONTROL: Error is reported when there is no/incorrect communication to transceiver.</p> <p>CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id.</p> <p>CANTRCV_17_W9255_E_TRCV_NOT_STANDBY: Error is reported when the CAN Transceiver is not in Stand-by mode or Sleep mode and has got a request to transit to Sleep mode.</p> <p>CANTRCV_17_W9255_E_UNINIT: Error is reported when the API service is used without initialization.</p> <p>CANTRCV_17_W9255_E_TRCV_NOT_NORMAL: Error is reported when the CAN Transceiver is not in Normal mode or Stand-by mode and has got a request to transit to Stand-by mode.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

7.3.3.3 CanTrcv_17_W9255_GetOpMode

Table 508 Specification for CanTrcv_17_W9255_GetOpMode API

Syntax	<pre>Std_ReturnType CanTrcv_17_W9255_GetOpMode (const uint8 Transceiver, CanTrcv_TrcvModeType * const OpMode)</pre>	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11.
Parameters (out)	OpMode	Pointer to operation mode of the transceiver the API is applied to.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: If the operation mode was detected

CanTrcv_17_W9255 driver
Table 508 Specification for CanTrcv_17_W9255_GetOpMode API (continued)

	E_NOT_OK: If SPI communication failure or a development error occurs
Description	This API gets the mode of the Transceiver and returns it in parameter OpMode.
Source	AUTOSAR
Error handling	<p>DET:</p> <p>CANTRCV_17_W9255_E_UNINIT: Error is reported when the API service is used without initialization.</p> <p>CANTRCV_17_W9255_E_NO_TRCV_CONTROL: Error is reported when there is no/incorrect communication to transceiver.</p> <p>CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id.</p> <p>CANTRCV_17_W9255_E_PARAM_POINTER: Error is reported if API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

7.3.3.4 CanTrcv_17_W9255_GetBusWuReason
Table 509 Specification for CanTrcv_17_W9255_GetBusWuReason API

Syntax	<pre>Std_ReturnType CanTrcv_17_W9255_GetBusWuReason (const uint8 Transceiver, CanTrcv_TrcvWakeupReasonType * const reason)</pre>	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11.
Parameters (out)	reason	Pointer to wake up reason of the transceiver the API is applied to. This parameter can hold the following valid enum values as per CanTrcv_TrcvWakeupReasonType requirement: - CANTRCV_WU_ERROR - CANTRCV_WU_NOT_SUPPORTED

CanTrcv_17_W9255 driver
Table 509 Specification for CanTrcv_17_W9255_GetBusWuReason API (continued)

		- CANTRCV_WU_BY_BUS - CANTRCV_WU_INTERNALLY - CANTRCV_WU_RESET - CANTRCV_WU_POWER_ON - CANTRCV_WU_BY_PIN - CANTRCV_WU_BY_SYSERR
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : If wake up reason was detected E_NOT_OK: If a development error occurs
Description	This API gets the wake-up reason for the requested Transceiver and returns it in the parameter reason. Note: The driver supports the following wake-up reasons: -Wake-up by bus - CANTRCV_WU_BY_BUS -Wake-up by pin - CANTRCV_WU_BY_PIN -Wake-up due to an ECU reset after power on - CANTRCV_WU_POWER_ON -Wake up due to transition to normal mode - CANTRCV_WU_INTERNALLY -Wake-up due to hardware related device failure (SYSERR) - CANTRCV_WU_BY_SYSERR - Wake-up due to an ECU reset - CANTRCV_WU_RESET The driver does not support the following wake-up reasons due to hardware limitations: - Wake up reason not detected due to an error - CANTRCV_WU_ERROR - Information for the wake-up reason not supported - CANTRCV_WU_NOT_SUPPORTED	
Source	AUTOSAR	
Error handling	DET: CANTRCV_17_W9255_E_UNINIT: Error is reported when the API service is used without initialization. CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id. CANTRCV_17_W9255_E_PARAM_POINTER: Error is reported if API is invoked with null-pointer as a parameter. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

CanTrcv_17_W9255 driver**7.3.3.5 CanTrcv_17_W9255_GetVersionInfo****Table 510 Specification for CanTrcv_17_W9255_GetVersionInfo API**

Syntax	<pre>void CanTrcv_17_W9255_GetVersionInfo (Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where to store the version information of the CanTrcv_17_W9255 driver
Parameters (in - out)	-	-
Return	void	-
Description	This API gets the version of the module and returns it in versionInfo.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CANTRCV_17_W9255_E_PARAM_POINTER: Error is reported if API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	CanTrcvGetVersionInfo	
User hints	-	

7.3.3.6 CanTrcv_17_W9255_SetWakeupMode**Table 511 Specification for CanTrcv_17_W9255_SetWakeupMode API**

Syntax	<pre>Std_ReturnType CanTrcv_17_W9255_SetWakeupMode (const uint8 Transceiver, const CanTrcv_TrcvWakeupModeType TrcvWakeupMode)</pre>
Service ID	0x05
Sync/Async	Synchronous

CanTrcv_17_W9255 driver
Table 511 Specification for CanTrcv_17_W9255_SetWakeupMode API (continued)

ASIL Level	QM	
Re-entrancy	Reentrant for different transceivers	
Parameters (in)	Transceiver TrcvWakeupMode	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11. Requested transceiver wakeup mode type
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Wakeup state changed to the requested mode E_NOT_OK: If SPI communication fails, wake-up by bus is disabled or a development error occurs. The previous state has not been changed.
Description	<p>This API enables, disables or clears wake-up events of the Transceiver according to TrcvWakeupMode.</p> <p>Enable mode: The CAN Transceiver driver reports wake-up event during wake-up event detection. Besides, the driver also reports wake-up event if it has a stored wake-up event pending for the addressed transceiver.</p> <p>Disable mode: The wake-up events are disabled on the addressed transceiver. Any wake-up event occurred during this time is stored internally.</p> <p>Clear mode: Stored wake-up event and hardware wake flags are cleared on the addressed transceiver.</p> <p>CANTRCV_17_W9255_E_NO_TRCV_CONTROL DET will be checked only in Clear mode since no communication with the hardware happens in the other two modes.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CANTRCV_17_W9255_E_PARAM_TRCV_WAKEUP_MODE: Error is reported when the API service is called with invalid parameter for TrcvWakeupMode.</p> <p>CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id.</p> <p>CANTRCV_17_W9255_E_UNINIT: Error is reported when the API service is used without initialization.</p> <p>CANTRCV_17_W9255_E_NO_TRCV_CONTROL: Error is reported when there is no/incorrect communication to transceiver.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	

CanTrcv_17_W9255 driver
Table 511 Specification for CanTrcv_17_W9255_SetWakeupMode API (continued)

User hints	-
-------------------	---

7.3.3.7 CanTrcv_17_W9255_CheckWakeups
Table 512 Specification for CanTrcv_17_W9255_CheckWakeups API

Syntax	Std_ReturnType CanTrcv_17_W9255_CheckWakeups (const uint8 Transceiver)	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : If a valid interrupt is detected E_NOT_OK : If SPI communication fails, development error is detected, wake up by bus is disabled for the channel called, or a false interrupt is detected
Description	This service is called by the underlying CanIf, in cases of polling or interrupt mode. This API validates wake-up event on the requested transceiver channel and if true, reports it to EcuM if the wakeup mode is enabled, clears the wake flags on the hardware and changes the mode of the respective channel to Normal.	
Source	AUTOSAR	
Error handling	DET: CANTRCV_17_W9255_E_UNINIT: Error is reported when the API service is used without initialization. CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

CanTrcv_17_W9255 driver**7.3.3.8 CanTrcv_17_W9255_CheckWakeFlag****Table 513 Specification for CanTrcv_17_W9255_CheckWakeFlag API**

Syntax	Std_ReturnType CanTrcv_17_W9255_CheckWakeFlag (const uint8 Transceiver)	
Service ID	0x0e	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: If the request for checking the wakeup flags has been accepted. E_NOT_OK: If the request for checking the wakeup flags has not been accepted, wake up by bus is disabled, development error occurred or if SPI communication fails.
Description	This API checks the status of the wake-up flags from the transceiver hardware and informs the CanIf with the callback notification CanIf_CheckTrcvWakeFlagIndication, that the wake flags of the CAN transceiver with the corresponding Transceiver ID have been checked.	
Source	AUTOSAR	
Error handling	DET: CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

7.3.3.9 CanTrcv_17_W9255_ClearTrcvTimeoutFlag**Table 514 Specification for CanTrcv_17_W9255_ClearTrcvTimeoutFlag API**

Syntax	Std_ReturnType CanTrcv_17_W9255_ClearTrcvTimeoutFlag (
---------------	---

CanTrcv_17_W9255 driver
Table 514 Specification for CanTrcv_17_W9255_ClearTrcvTimeoutFlag API (continued)

	const uint8 Transceiver)	
Service ID	0x0c	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: If the timeout flag is successfully cleared. E_NOT_OK: If SPI communication failure or a development error occurs.
Description	This API clears the status of the timeout flag in the transceiver hardware. Since the configuration parameter CanTrcvHwPnSupport is always TRUE, this API is always available. The timeout flag indicates whether or not the TLE9255W hardware has entered the Selective Sleep Sub-Mode at least once.	
Source	AUTOSAR	
Error handling	DET: CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	CanTrcvHwPnSupport	
User hints	-	

7.3.3.10 CanTrcv_17_W9255_ClearTrcvWufFlag
Table 515 Specification for CanTrcv_17_W9255_ClearTrcvWufFlag API

Syntax	Std_ReturnType CanTrcv_17_W9255_ClearTrcvWufFlag (const uint8 Transceiver)
Service ID	0x0a

CanTrcv_17_W9255 driver
Table 515 Specification for CanTrcv_17_W9255_ClearTrcvWufFlag API (continued)

Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different transceivers	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: If the WUF flag has been cleared. E_NOT_OK: If SPI communication failure or a development error occurs.
Description	This API clears the WUF flag in the transceiver hardware. Since the configuration parameter CanTrcvHwPnSupport is always TRUE, this API is always available.	
Source	AUTOSAR	
Error handling	DET: CANTRCV_17_W9255_E_NO_TRCV_CONTROL: Error is reported when there is no/incorrect communication to transceiver. CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id. CANTRCV_17_W9255_E_UNINIT: Error is reported when the API service is used without initialization. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	CanTrcvHwPnSupport	
User hints	-	

7.3.3.11 CanTrcv_17_W9255_GetTrcvSystemData
Table 516 Specification for CanTrcv_17_W9255_GetTrcvSystemData API

Syntax	Std_ReturnType CanTrcv_17_W9255_GetTrcvSystemData (const uint8 Transceiver, uint32 * const TrcvSysData)
Service ID	0x09
Sync/Async	Synchronous

CanTrcv_17_W9255 driver
Table 516 Specification for CanTrcv_17_W9255_GetTrcvSystemData API (continued)

ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11.
Parameters (out)	TrcvSysData	This parameter holds the selective wake status, error status, transceiver status and wake-up event status information. The first 8 bits of LSB contain the data stored in TRANS_STAT register, the next 8 bits contain the data in SWK_ECNT_STAT register, the next 8 bits depict the data stored in WAKE_STAT register and the last 8 bits contain the data stored in SWK_STAT register
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: If the transceiver status is successfully read. E_NOT_OK: If SPI communication failure or a development error occurs.
Description	This API reads the transceiver status data and returns it through parameter TrcvSysData. Since the configuration parameter CanTrcvHwPnSupport is always TRUE, this API is always available.	
Source	AUTOSAR	
Error handling	DET: CANTRCV_17_W9255_E_UNINIT: Error is reported when the API service is used without initialization. CANTRCV_17_W9255_E_NO_TRCV_CONTROL: Error is reported when there is no/incorrect communication to transceiver. CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id. CANTRCV_17_W9255_E_PARAM_POINTER: Error is reported if API is invoked with null-pointer as a parameter. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	CanTrcvHwPnSupport	
User hints	-	

7.3.3.12 CanTrcv_17_W9255_ReadTrcvSilenceFlag
Table 517 Specification for CanTrcv_17_W9255_ReadTrcvSilenceFlag API

Syntax	Std_ReturnType CanTrcv_17_W9255_ReadTrcvSilenceFlag (
---------------	--

CanTrcv_17_W9255 driver
Table 517 Specification for CanTrcv_17_W9255_ReadTrcvSilenceFlag API (continued)

	<pre> const uint8 Transceiver, CanTrcv_17_W9255_TrccvFlagStateType * const FlagState)</pre>	
Service ID	0x0d	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Transceiver	Pointer to operation mode of the transceiver the API is applied to. This parameter has a valid range of 0-11.
Parameters (out)	FlagState	State of the silence flag
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: If status of the silence flag is successfully read. E_NOT_OK: If status of the silence flag could not be read or a development error occurs.
Description	This API reads the status of the silence flag from the transceiver hardware. Since the configuration parameter CanTrcvHwPnSupport is always TRUE, this API is always available. The silence flag, if set, indicates that there is no communication on the CAN bus for a specified period of time (0.6 - 1.2 seconds). It helps to identify whether or not the TLE9255W hardware is in the Selective Sleep Sub-Mode.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id.</p> <p>CANTRCV_17_W9255_E_PARAM_POINTER: Error is reported if API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	CanTrcvHwPnSupport	
User hints	-	

CanTrcv_17_W9255 driver**7.3.3.13 CanTrcv_17_W9255_ReadTrcvTimeoutFlag****Table 518 Specification for CanTrcv_17_W9255_ReadTrcvTimeoutFlag API**

Syntax	<pre>Std_ReturnType CanTrcv_17_W9255_ReadTrcvTimeoutFlag (const uint8 Transceiver, CanTrcv_17_W9255_TrsvFlagStateType * const FlagState)</pre>	
Service ID	0x0b	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied. This parameter has a valid range of 0-11.
Parameters (out)	FlagState	State of the timeout flag
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: If status of the timeout flag is successfully read. E_NOT_OK: If status of the timeout flag could not be read or a development error occurs.
Description	This API reads the status of the timeout flag from the transceiver hardware. Since the configuration parameter CanTrcvHwPnSupport is always TRUE, this API is always available. The timeout flag indicates whether or not the TLE9255W hardware has entered the Selective Sleep Sub-Mode at least once.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>CANTRCV_17_W9255_E_INVALID_TRANSCEIVER: Error is reported if the API is called with invalid transceiver channel Id.</p> <p>CANTRCV_17_W9255_E_PARAM_POINTER: Error is reported if API is invoked with null-pointer as a parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	CanTrcvHwPnSupport	
User hints	-	

CanTrcv_17_W9255 driver

7.3.3.14 **CanTrcv_17_W9255_SetPNActivationState**

Table 519 Specification for CanTrcv_17_W9255_SetPNActivationState API

Syntax	<pre>Std_ReturnType CanTrcv_17_W9255_SetPNActivationState (const CanTrcv_17_W9255_PNActivationType ActivationState)</pre>	
Service ID	0x0f	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	ActivationState	PN_ENABLED: PN wakeup functionality in CAN Transceiver shall be enabled PN_DISABLED: PN wakeup functionality in CAN Transceiver shall be disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: If PN has been changed to the requested configuration. E_NOT_OK: If the PN configuration change has failed or if an invalid enum value is passed as parameter
Description	This API enables/disables selective wake-up functionality of all those channels which have enabled PN in their configuration.	
Source	AUTOSAR	
Error handling	DET: CANTRCV_17_W9255_E_UNINIT: Error is reported when the API service is used without initialization. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

7.3.4 **Notifications and Callbacks**

The driver does not support any notification and callback functions.

7.3.5 **Scheduled functions**

This section lists all the scheduled functions supported by the driver.

CanTrcv_17_W9255 driver

7.3.5.1 **CanTrcv_17_W9255_MainFunction**

Table 520 Specification for CanTrcv_17_W9255_MainFunction API

Syntax	void CanTrcv_17_W9255_MainFunction (void)	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This API scans all transceiver channels in Stand-by and Sleep modes for wake up events and sets a wake-up event flag to perform these events.</p> <p>Note: The wake-up event flag is polled by CanTrcv_17_W9255_CheckWakeup API in polling mode.</p>	
Source	AUTOSAR	
Error handling	<p>DET: CANTRCV_17_W9255_E_UNINIT: Error is reported when the API service is used without initialization.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	CanTrcvWakeUpSupport	
User hints	-	

7.3.6 **Interrupt service routines**

The driver does not service any interrupts.

Note: The CAN transceiver TLE9255W wakeup interrupts are handled by the ICU driver.

7.3.7 **Error codes classification**

This section explains various error types and their corresponding source APIs.

CanTrcv_17_W9255 driver

7.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Table 521 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
Error is reported if initialization of the driver has failed. Since it is a Pre-compile module, the init function expects a NULL pointer to be passed as parameter. This DET is reported if a non-null pointer is passed as a parameter during init.	AUTOSAR	CANTRCV_17_W9255_E_INIT_FAILED =0x27	CanTrcv_17_W9255_Init
Error is reported if the API is called with invalid transceiver channel Id.	AUTOSAR	CANTRCV_17_W9255_E_INVALID_TRANSCEIVER=0x01	CanTrcv_17_W9255_CheckWakeFlag, CanTrcv_17_W9255_CheckWakeup, CanTrcv_17_W9255_ReadTrcvSilenceFlag, CanTrcv_17_W9255_ClearTrcvTimeoutFlag, CanTrcv_17_W9255_ReadTrcvTimeoutFlag, CanTrcv_17_W9255_ClearTrcvWufFlag, CanTrcv_17_W9255_GetTrcvSystemData, CanTrcv_17_W9255_SetWakeMode, CanTrcv_17_W9255_GetBusWuReason, CanTrcv_17_W9255_GetOpMode, CanTrcv_17_W9255_SetOpMode
Error is reported when there is no/incorrect communication to transceiver.	AUTOSAR	CANTRCV_17_W9255_E_NO_TRCV_CONTROL=0x26	CanTrcv_17_W9255_SetWakeMode, CanTrcv_17_W9255_ClearTrcvWufFlag, CanTrcv_17_W9255_GetTrcvSystemData, CanTrcv_17_W9255_GetOpMode, CanTrcv_17_W9255_SetOpMode, CanTrcv_17_W9255_Init

CanTrcv_17_W9255 driver**Table 521 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
Error is reported if API is invoked with null-pointer as a parameter.	AUTOSAR	CANTRCV_17_W9255_E_PARAM_POINTER=0x02	CanTrcv_17_W9255_GetVersionInfo, CanTrcv_17_W9255_ReadTrcvSilenceFlag, CanTrcv_17_W9255_ReadTrcvTimeoutFlag, CanTrcv_17_W9255_GetTrcvSystemData, CanTrcv_17_W9255_GetBusWuReason, CanTrcv_17_W9255_GetOpMode
Error is reported when the API service is called with invalid parameter for OpMode.	AUTOSAR	CANTRCV_17_W9255_E_PARAM_TRCV_OPMODE=0x24	CanTrcv_17_W9255_SetOpMode
Error is reported when the API service is called with invalid parameter for TrcvWakeupMode.	AUTOSAR	CANTRCV_17_W9255_E_PARAM_TRCV_WAKEUP_MODE=0x23	CanTrcv_17_W9255_SetWakeupMode
Error is reported when the CAN Transceiver is not in Normal mode or Stand-by mode and has got a request to transit to Stand-by mode.	AUTOSAR	CANTRCV_17_W9255_E_TRCV_NOT_NORMAL=0x22	CanTrcv_17_W9255_SetOpMode
Error is reported when the CAN Transceiver is not in Stand-by mode or Sleep mode and has got a request to transit to Sleep mode.	AUTOSAR	CANTRCV_17_W9255_E_TRCV_NOT_STANDBY=0x21	CanTrcv_17_W9255_SetOpMode
Error is reported when the API service is used without initialization.	AUTOSAR	CANTRCV_17_W9255_E_UNINIT=0x1	CanTrcv_17_W9255_MainFunction, CanTrcv_17_W9255_SetPNAactivationState, CanTrcv_17_W9255_CheckWakeup, CanTrcv_17_W9255_ClearTrcvWuffFlag, CanTrcv_17_W9255_GetTrcvSystemData, CanTrcv_17_W9255_SetWakeupMode, CanTrcv_17_W9255_GetBusWuReason,

CanTrcv_17_W9255 driver

Table 521 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
			CanTrcv_17_W9255_GetOpMode, CanTrcv_17_W9255_SetOpMode

7.3.7.2 Production errors

The driver does not report any production errors.

7.3.7.3 Safety errors

The driver does not report any safety errors.

7.3.7.4 Runtime errors

The driver does not report any runtime errors.

7.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

7.3.8.1 Deviations

The section describes the deviations from software specification.

Table 522 Known deviations

Reference	Deviation
AUTOSAR CAN Transceiver requirement[SWS_CanTrcv_00090].	Since the hardware supports the wake-up functionality, the NOT_SUPPORTED mode is not applicable for the CAN transceiver driver.
AUTOSAR CAN Transceiver requirements[SWS_CanTrcv_00171], SWS_CanTrcv_00172], SWS_CanTrcv_00173]	Since the ICU driver does not depend on the Icu_EnableNotification and Icu_DisableNotification APIs for reporting wake-up, these interfaces are not used in the CAN transceiver driver.
.	
AUTOSAR CAN Transceiver requirement[SWS_CanTrcv_00067].	In order to avoid compilation errors and repeated inclusion of files, AUTOSAR specified file structure is modified.
DEM error and CanTrcv_MainFunctionDiagnostics API are not supported.	Since TLE9255W hardware cannot detect bus failure, the CanTrcv_MainFunctionDiagnostics API and DEM error CANTRCV__E_BUS_ERROR is not available by the CAN transceiver driver.

7.3.8.2 Limitations

There are no limitations for the CAN transceiver driver.

CanTrcv_17_W9255 driver

7.3.9 Unsupported hardware features

This section lists all the TLE9255W hardware features that are not supported by the driver. These features are out of scope of the CAN transceiver driver specification.

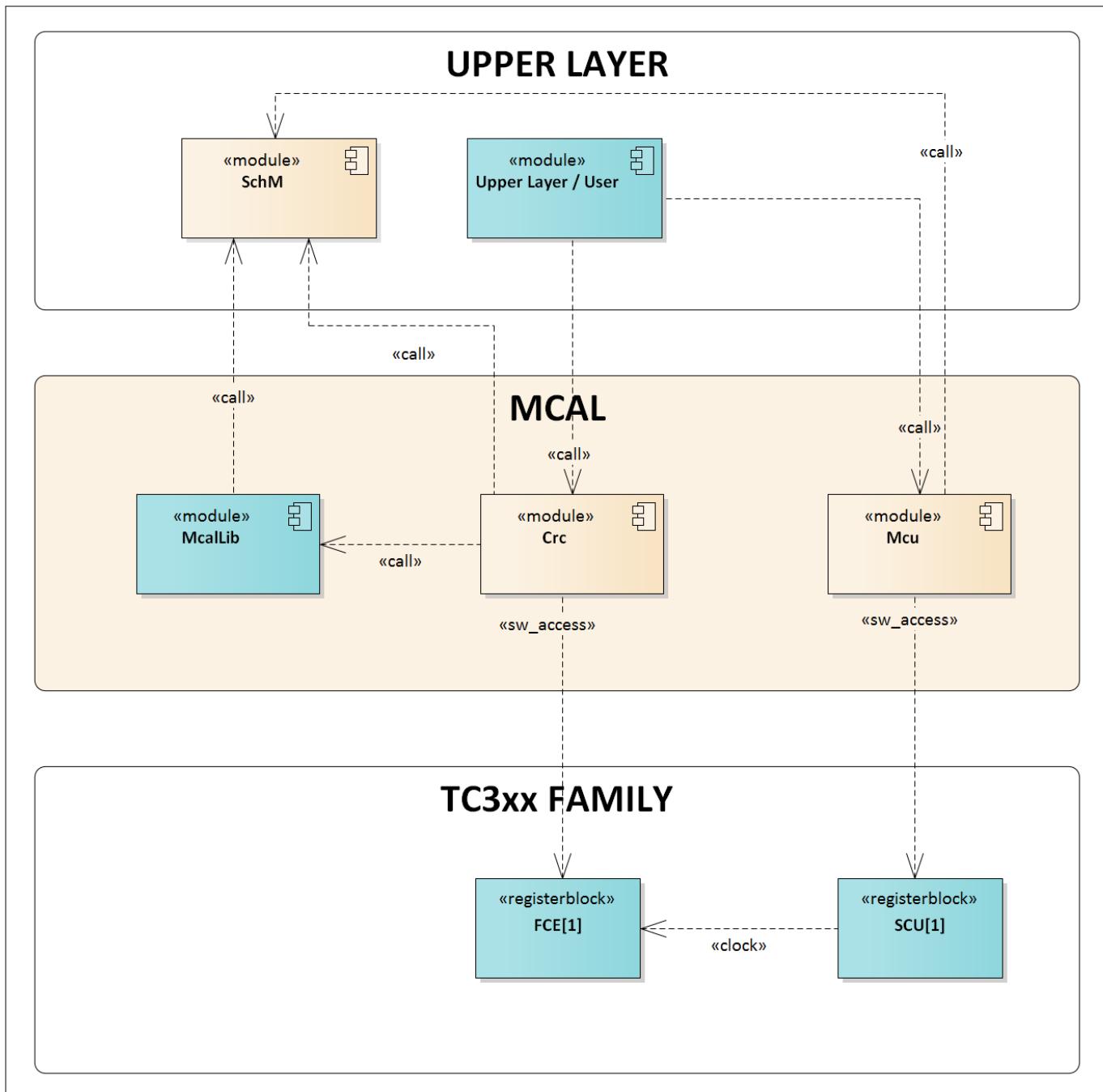
- Receive-only mode
- Fail-safe features namely undervoltage detection, overtemperature warning and TxD timeout function

CRC driver**8 CRC driver****8.1 User information****8.1.1 Description**

The CRC driver provides APIs to configure the CRC functionalities for 8-bit, 16-bit and 32-bit polynomials, prescribed by AUTOSAR. The CRC driver performs CRC calculations by hardware, runtime approach and table-based approach. The CRC driver uses the FCE hardware to perform the CRC calculation on the hardware. The CRC driver is developed as a pre-compile variant.

8.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

CRC driver
Figure 96 **Mapping of hardware-software interfaces**

8.1.2.1 SCU: dependent hardware peripheral

Hardware functional features

The CRC driver depends on the SCU IP for the clock functionality. The driver requires fSPB clock signal for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

CRC driver

The SMU alarms configured for the SCU IP are not monitored by the CRC driver.

Hardware events

Hardware events from the SCU are not used by the CRC driver.

8.1.2.2 FCE: primary hardware peripheral

Hardware functional features

The CRC driver uses the FCE IP for calculating CRC. The key hardware functional feature used by the CRC driver is:

- CRC computation

The unsupported feature of the FCE IP is:

- Automatic signature check

Users of the hardware

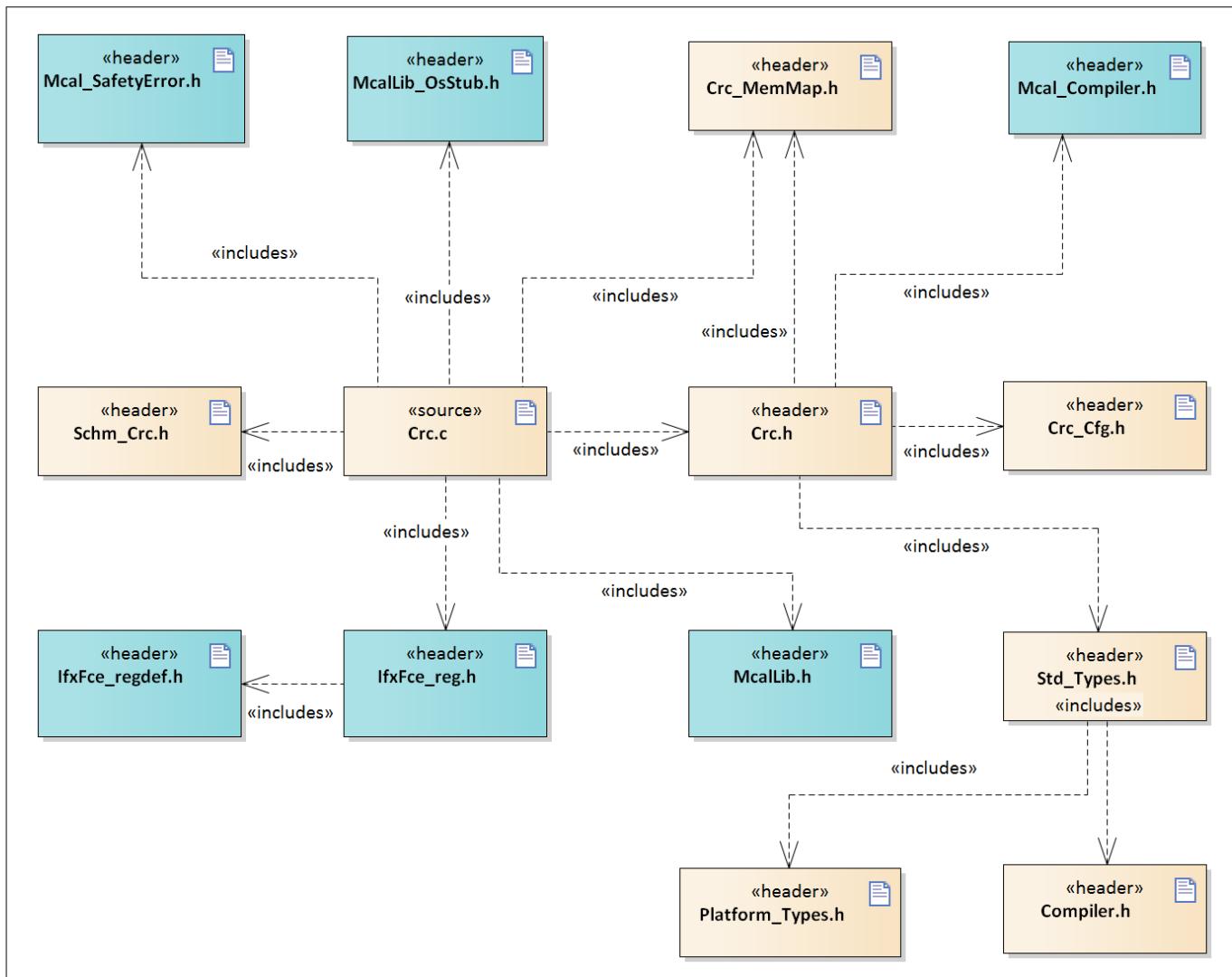
The CRC driver exclusively utilizes the FCE IP.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from the FCE IP like transient error detection or checksum failure are not used by the FCE driver.

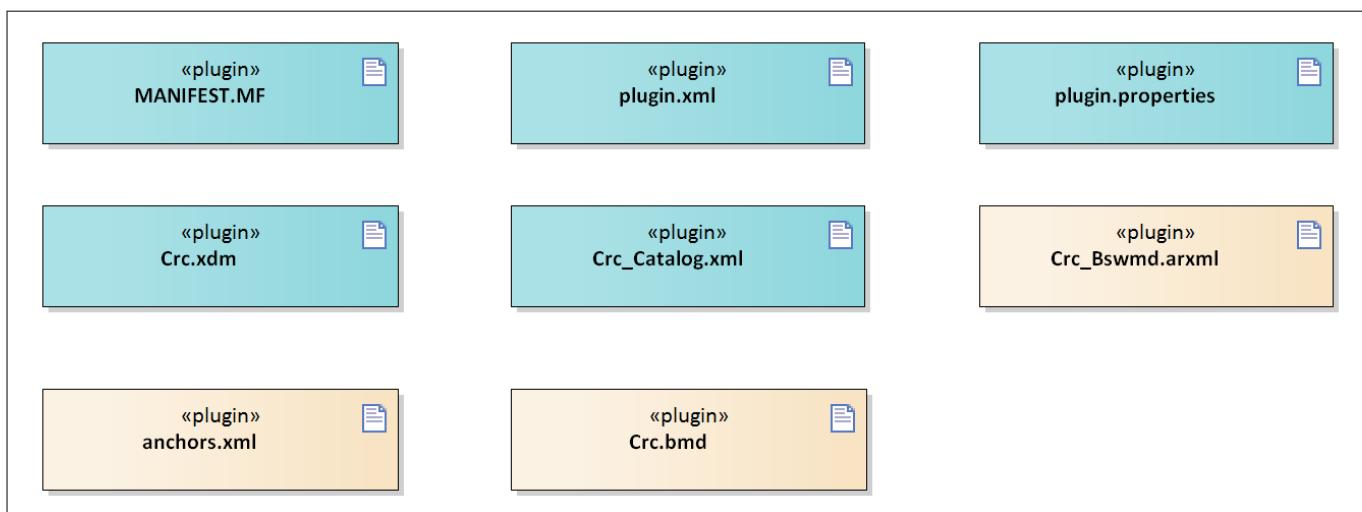
CRC driver**8.1.3 File structure****8.1.3.1 C file structure****Figure 97 C file structure****Table 523 C file structure**

File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Crc.c	Contains the implementation of the CRC feature
Crc.h	Provides the functional prototypes and access to the CRC driver function. This file exports only the necessary interfaces for upper layer.
Crc_Cfg.h	Provides the specific parameters of FCE
Crc_MemMap.h	Maps the CRC registers to their memory location
IfxFce_Reg.h	SFR header file for FCE
IfxFce_RegDef.h	SFR header file for FCE

CRC driver**Table 523 C file structure (continued)**

File name	Description
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore. This shall be included by other drivers to call OS APIs.
Mcal_Compiler.h	Provides abstraction for TriCore™-intrinsic instruction
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
Schm_Crc.h	Export header for Schm functions of CRC driver
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

8.1.3.2 Code generator plugin files

**Figure 98 Code generator plugin files****Table 524 Code generator plugin files**

File name	Description
Crc.bmd	Code template macro file for CRC driver
Crc.xdm	Tresos format XML data model schema file
Crc_Bswmd.arxml	AUTOSAR format module description file
Crc_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the metadata for CRC driver
anchors.xml	AUTOSAR format module description file
plugin.properties	Tresos plugin support file for the CRC driver
plugin.xml	Tresos plugin support file for the CRC driver

CRC driver**8.1.4 Integration hints**

This section lists the modules which are not part of MCAL but required to integrate the CRC driver.

8.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the CRC driver.

- **EcuM**

EcuM module is not required for the integration of the CRC driver.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Crc_MemMap.h` file.

The `Crc_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements

CRC driver

are relocated to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

```
#if defined CRC_START_SEC_CONST_ASIL_B_GLOBAL_8
/*User pragmas here for PFlash*/
#undef CRC_START_SEC_CONST_ASIL_B_GLOBAL_8
#undef MEMMAP_ERROR

#elif defined CRC_STOP_SEC_CONST_ASIL_B_GLOBAL_8
/*User pragmas here for default section*/
#undef CRC_STOP_SEC_CONST_ASIL_B_GLOBAL_8
#undef MEMMAP_ERROR

#elif defined CRC_START_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here for PFlash*/
#undef CRC_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#elif defined CRC_STOP_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here for default section*/
#undef CRC_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#elif defined CRC_START_SEC_VAR_INIT_ASIL_B_GLOBAL_8
/*User pragmas here for non-cached LMU*/
#undef CRC_START_SEC_VAR_INIT_ASIL_B_GLOBAL_8
#undef MEMMAP_ERROR

#elif defined CRC_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_8
/*User pragmas here for default section*/
#undef CRC_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_8
#undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Crc_MemMap.h, wrong pragma command"
#endif
```

- **DET**

DET module is not required for the integration of the CRC driver.

- **DEM**

DEM module is not required for the integration of the CRC driver.

- **SchM**

The SchM module is a part of the RTE that manages the BSW Scheduler. The CRC driver uses the exclusive areas defined in the `SchM_Crc.h` file to protect a variable from concurrent accesses from different threads. The SchM identified for the CRC driver is:

- CriticalSection

The files `SchM_Crc.h` and `SchM_Crc.c` are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the CRC driver as

CRC driver

suspend / resume of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows.

```
***** Sample implementation of SchM_Crc.c *****
#include "Os.h"

void SchM_Enter_Crc_CriticalSection(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Crc_CriticalSection(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety errors**

The CRC driver will report all the detected safety errors through the `Mcal_ReportSafetyError` API .

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError` API is provided in the files

`Mcal_SafetyError.c` and `Mcal_SafetyError.h` as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The CRC driver does not provide any call-backs or notifications.

- **Operating system (OS)**

Enabling and disabling of interrupts must be managed by the OS or the application.

The OS files provided by the MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

8.1.4.2 Multicore and Resource Manager

The CRC driver supports execution of its APIs simultaneously from all CPU cores. FCE hardware channels are not statically configurable to a particular core, that is, FCE hardware channel allocation is not done in the Resource Manager. The FCE hardware channels are dynamically handled in the CRC driver during runtime to ensure the most efficient usage of the resources. The following are the key points to be considered with respect to multicore in the driver:

- Locating constants and variables to correct memory space should be done by the user. Memory sections are marked as GLOBAL (common to all cores). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of the CRC driver is placed under single MemMap section. It can be relocated to any PFlash region.

Data section:

The sections marked as global should be relocated to the non-cached LMU region.

CRC driver**Constants:**

The sections marked as global should be relocated to the PFlash of the master core.

Note: Relocating code, data or constants to a distant memory region would impact execution timings.

8.1.4.3 MCU support

The CRC driver is dependent on MCU driver for clock configuration. The APIs of the CRC driver must be started only after completion of MCU initialization.

8.1.4.4 Port support

CRC driver does not use any services provided by the PORT driver.

8.1.4.5 DMA support

The CRC driver does not use any services provided by the DMA driver.

8.1.4.6 Interrupt connections

The CRC driver does not use any interrupt source.

CRC driver

8.1.4.7 Example usage

Configuration

Each CRC polynomial can be set to any one of the given three modes with the exception of `Crc8H2FMode` which has only two modes.

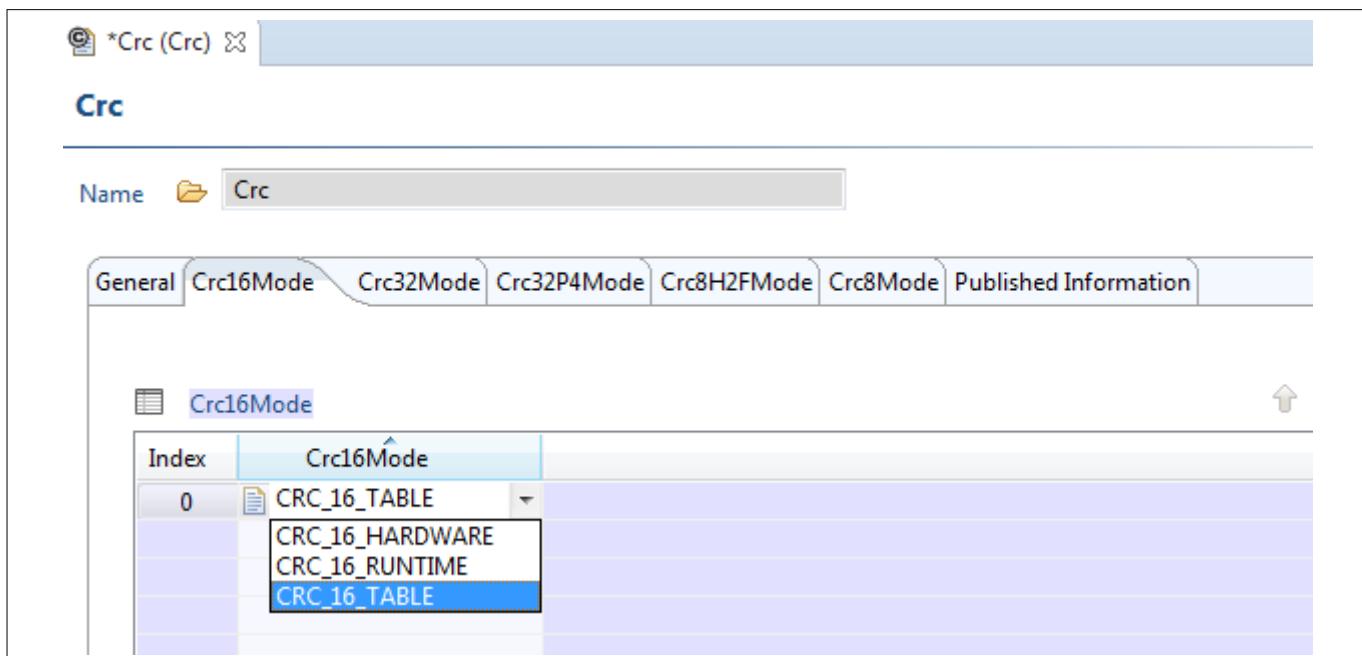


Figure 99 Configuring mode for CRC calculation

If the `CrcSafetyEnable` is ON, the error values should be configured. The default value is zero. If it is OFF, the error value parameters are disabled.

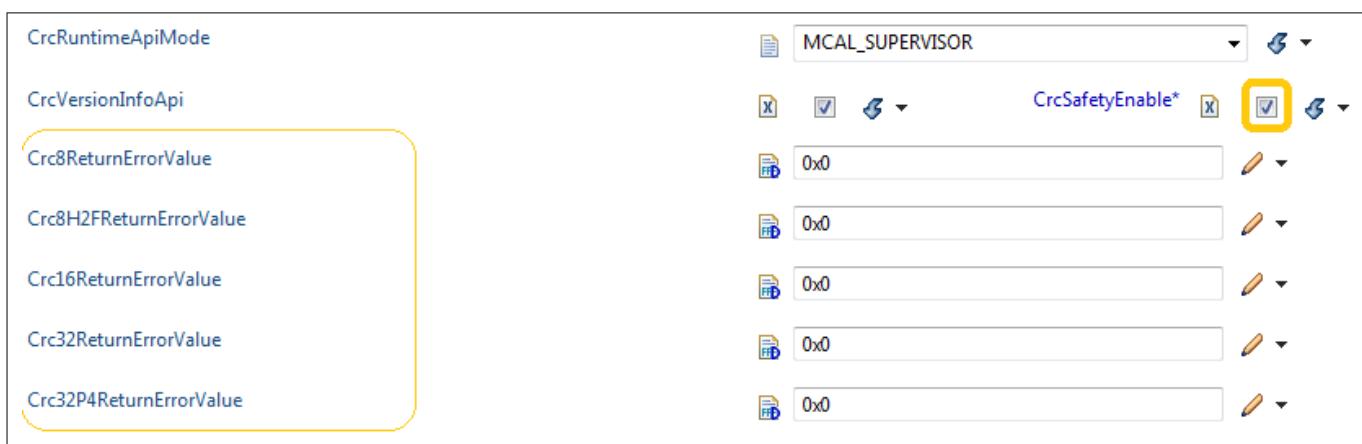


Figure 100 `CrcSafetyEnable` enabled

CRC driver

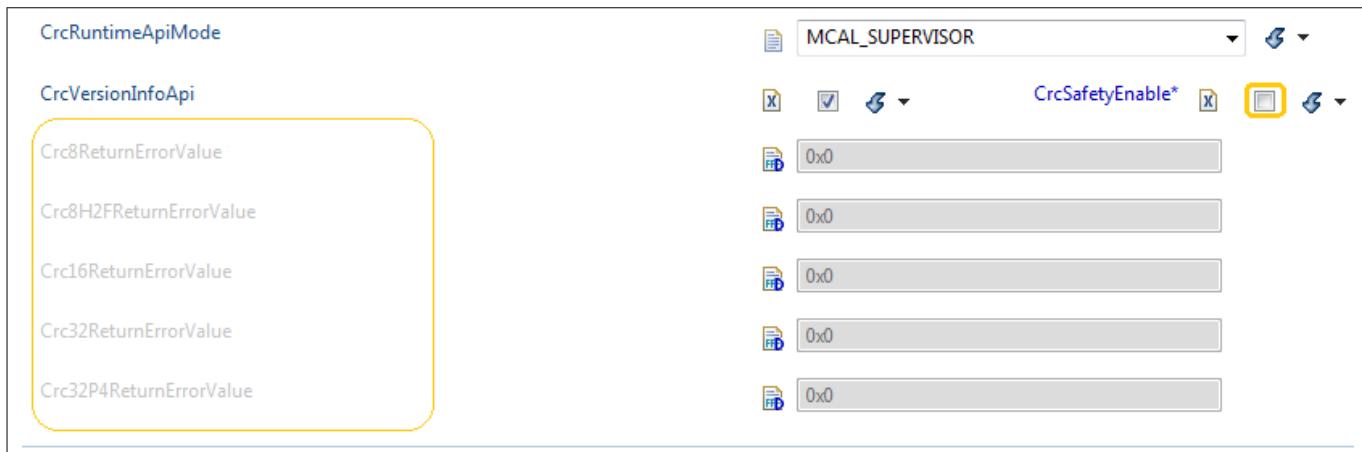


Figure 101 CrcSafetyEnable disabled

Using the APIs

The code snippet shows an example of CRC calculation with 8-bit polynomial (0x1D).

```
/*Calling the CRC8 with polynomial = 0x1D */
Crc8Result = Crc_CalculateCRC8(CrcMessage8, Crc_Length, Crc_StartValue8,
IsFirstCall);
```

The usage of the parameters are as follows:

- `CrcMessage8` is the pointer to the start of the block.
- `Crc_Length` is the size of the block array.
- `Crc_StartValue8` is the start value to be used by algorithm.
- `IsFirstCall` selects the start value for CRC calculation. TRUE will select default initial value for the particular polynomial as start value for CRC calculation. FALSE will select the start value provided as argument as start value for CRC calculation.

8.1.5 Key architectural considerations

8.1.5.1 Dynamic channel handling

The FCE channels are capable of calculating CRC for the polynomials associated with the channel. The channels are allocated for CRC calculation during runtime (based on free channels). FCE channels are not mapped to a specific kernel or core statically. Hence, the Resource Manager is not used for the CRC driver.

CRC driver

8.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Alignment of the data for 16-bit CRC computation using hardware**

User shall ensure that while calculating the 16-bit CRC in the hardware mode using the Crc_CalculateCRC16() API, the input data stream is expected to be aligned to 16-bit word boundary. If not aligned, the remaining 8-bits are assumed to be zero.

[cover parentID CRC={0033DA0F-E032-4df6-96B0-D88AE74DC177}]

- **Alignment of the data for 32-bit CRC computation using hardware**

User shall ensure that while calculating the 32-bit CRC in the hardware mode using the Crc_CalculateCRC32 and Crc_CalculateCRC32P4 APIs, the input data stream is expected to be aligned to 32-bit word boundary. If not aligned, the remaining bits are assumed to be zero.

[cover parentID CRC={6FFA7FDF-A1FF-49e2-8E8D-2EF267554A28}]

- **Error value return**

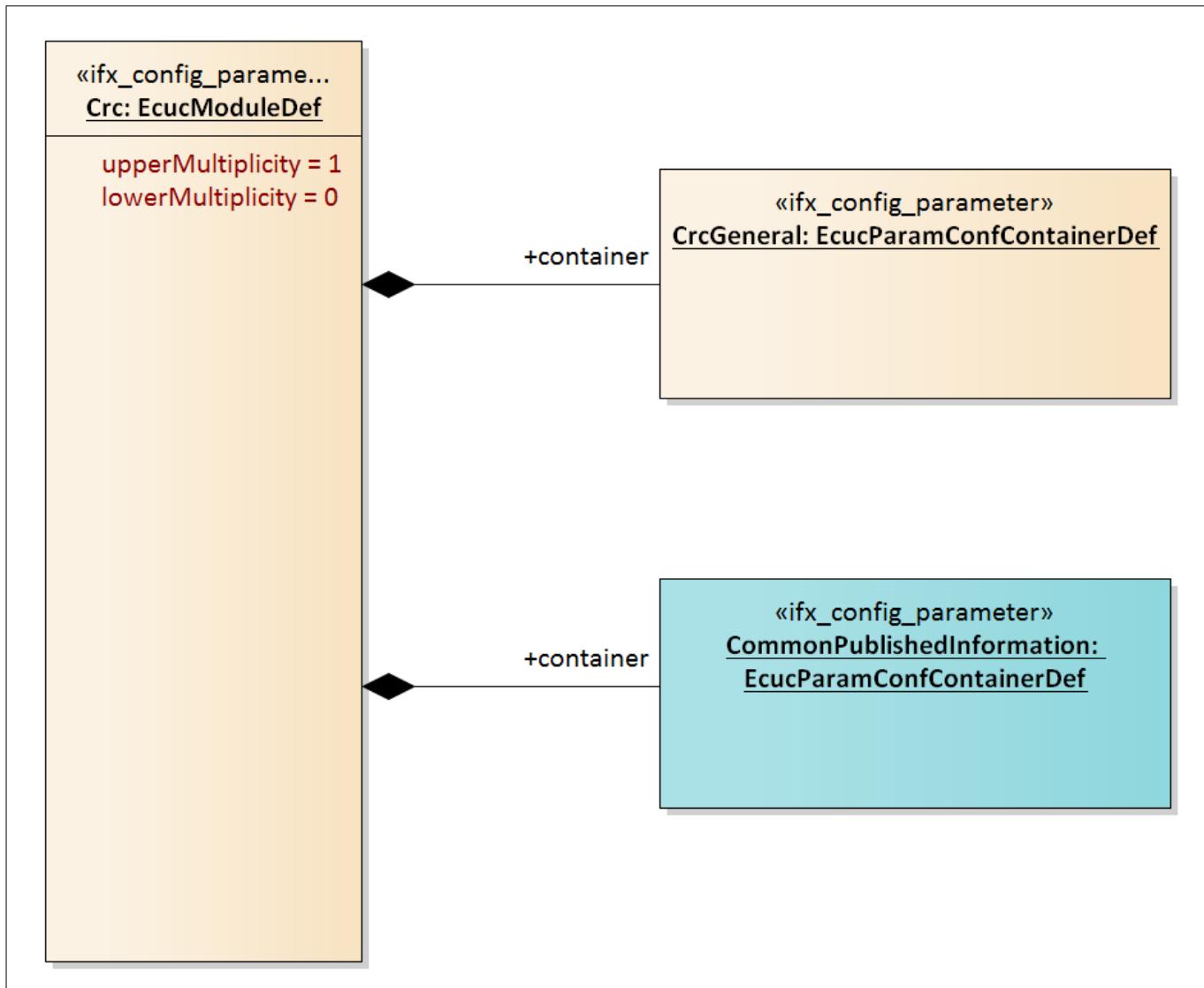
When safety switch is ON, if the result returned by the API is the configured error value and no safety error is reported, then the result is valid.

[cover parentID CRC={6AADA8A0-78EB-4432-A6AD-4862A186D2AD}]

- **FCE register protection**

User shall not modify any FCE-related register.

[cover parentID CRC={68DA99F3-38B9-47b7-BF82-BC9F32BB485B}]

CRC driver**8.3 Reference information****8.3.1 Configuration interfaces****Figure 102 Container hierarchy along with their configuration parameters****8.3.1.1 Container: CommonPublishedInformation**

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

8.3.1.1.1 ARMinorVersion**Table 525 Specification for ARMinorVersion**

Name	ARMinorVersion
Description	Minor version number of the AUTOSAR specification

CRC driver**Table 525 Specification for ARMinorVersion (continued)**

Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.1.2 ARPatchVersion**Table 526 Specification for ARPatchVersion**

Name	ARPatchVersion		
Description	Patch version number of the AUTOSAR specification		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.1.3 ArMajorVersion**Table 527 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Major version number of the AUTOSAR specification		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-

CRC driver**Table 527 Specification for ArMajorVersion (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.1.4 ModuleId**Table 528 Specification for ModuleId**

Name	ModuleId		
Description	This macro gives the Crc driver module ID as described by AUTOSAR		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	201 (0xC9)		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.1.5 SwMajorVersion**Table 529 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Major version number of the driver.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

CRC driver**8.3.1.1.6 SwMinorVersion****Table 530 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Minor version number of the driver.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.1.7 SwPatchVersion**Table 531 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Patch level version number of the driver. The patch version is incremented if the driver is still upwards and downwards compatible (for example, bug fix)		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.1.8 VendorId**Table 532 Specification for VendorId**

Name	VendorId		
Description	Vendor ID of Infineon Technologies		
Multiplicity	1..1	Type	EcuIntegerParamDef

CRC driver**Table 532 Specification for VendorId (continued)**

Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.2 Container: Crc

Configuration of the CRC driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

8.3.1.3 Container: CrcGeneral

General configuration of the CRC driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

8.3.1.3.1 Crc16Mode**Table 533 Specification for Crc16Mode**

Name	Crc16Mode		
Description	Switch to select one of the available CRC 16-bit (CCITT) calculation methods		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	CRC_16_HARDWARE: hardware based CRC16 calculation CRC_16_RUNTIME: runtime based CRC16 calculation CRC_16_TABLE: table based CRC16 calculation		
Default value	CRC_16_TABLE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

CRC driver**8.3.1.3.2 Crc16ReturnValue****Table 534 Specification for Crc16ReturnValue**

Name	Crc16ReturnValue		
Description	Error value to be returned instead of CRC value if safety check is enabled and in case of an incorrect input parameter or in case of a timeout when all channels are busy in hardware mode. Default value of the parameter is the minimum value.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CrcSafetyEnable		

8.3.1.3.3 Crc32Mode**Table 535 Specification for Crc32Mode**

Name	Crc32Mode		
Description	Switch to select one of the available CRC 32-bit (IEEE-802.3 CRC32 ethernet standard) calculation methods		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	CRC_32_HARDWARE: hardware based CRC32 calculation CRC_32_RUNTIME: runtime based CRC32 calculation CRC_32_TABLE: table based CRC32 calculation		
Default value	CRC_32_TABLE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

CRC driver**8.3.1.3.4 Crc32P4Mode****Table 536 Specification for Crc32P4Mode**

Name	Crc32P4Mode		
Description	Switch to select one of the available CRC 32-bit E2E profile 4 calculation methods.		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	CRC_32P4_HARDWARE: hardware based CRC32P4 calculation CRC_32P4_RUNTIME: runtime based CRC32P4 calculation CRC_32P4_TABLE: table based CRC32P4 calculation (default selection)		
Default value	CRC_32P4_TABLE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

8.3.1.3.5 Crc32P4ReturnValue**Table 537 Specification for Crc32P4ReturnValue**

Name	Crc32P4ReturnValue		
Description	Error value to be returned instead of CRC value if safety check is enabled and in case of an incorrect input parameter or in case of a timeout when all channels are busy in hardware mode. Default value of the parameter is the minimum value.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CrcSafetyEnable		

CRC driver**8.3.1.3.6 Crc32ReturnValue****Table 538 Specification for Crc32ReturnValue**

Name	Crc32ReturnValue		
Description	Error value to be returned instead of CRC value if safety check is enabled and in case of an incorrect input parameter or in case of a timeout when all channels are busy in hardware mode. Default value of the parameter is the minimum value.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CrcSafetyEnable		

8.3.1.3.7 Crc8H2FMode**Table 539 Specification for Crc8H2FMode**

Name	Crc8H2FMode		
Description	Switch to select one of the available CRC 8-bit (2Fh polynomial) calculation methods		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	CRC_8H2F_RUNTIME: runtime based CRC8H2F calculation CRC_8H2F_TABLE: table based CRC8H2F calculation		
Default value	CRC_8H2F_TABLE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

8.3.1.3.8 Crc8H2FReturnValue**Table 540 Specification for Crc8H2FReturnValue**

Name	Crc8H2FReturnValue
-------------	--------------------

CRC driver**Table 540 Specification for Crc8H2FReturnErrorValue (continued)**

Description	Error value to be returned instead of CRC value if safety check is enabled and in case of an incorrect input parameter. Default value of the parameter is the minimum value.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CrcSafetyEnable		

8.3.1.3.9 Crc8Mode**Table 541 Specification for Crc8Mode**

Name	Crc8Mode		
Description	Switch to select one of the available CRC 8-bit (SAE J1850) calculation methods		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	CRC_8_HARDWARE: hardware based CRC8 calculation CRC_8_RUNTIME: runtime based CRC8 calculation CRC_8_TABLE: table based CRC8 calculation		
Default value	CRC_8_TABLE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

8.3.1.3.10 Crc8ReturnValue**Table 542 Specification for Crc8ReturnValue**

Name	Crc8ReturnValue
Description	Error value to be returned instead of CRC value if safety check is enabled and in case of an incorrect input parameter or in case of a timeout when all channels are busy in hardware mode. Default value of the parameter is the minimum value.

CRC driver**Table 542 Specification for Crc8ReturnValue (continued)**

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CrcSafetyEnable		

8.3.1.3.11 CrcRuntimeApiMode**Table 543 Specification for CrcRuntimeApiMode**

Name	CrcRuntimeApiMode		
Description	This configuration parameter gives the mode in which the runtime APIs of CRC driver will be used. Since the CRC driver accesses the SFRs, it is more efficient to operate the Crc driver in supervisor mode. Hence, the default mode of operation is supervisor mode.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CRC_MCAL_SUPERVISOR: Operating mode used is Supervisor CRC_MCAL_USER1: Operating mode used is USER1		
Default value	MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.3.12 CrcSafetyEnable**Table 544 Specification for CrcSafetyEnable**

Name	CrcSafetyEnable
Description	Switch to enable/disable the safety check and reporting. TRUE: enables safety check and reporting FALSE: disables safety check and reporting

CRC driver**Table 544 Specification for CrcSafetyEnable (continued)**

	The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.3.13 CrcVersionInfoApi**Table 545 Specification for CrcVersionInfoApi**

Name	CrcVersionInfoApi		
Description	Pre-processor switch to enable / disable the API to read out the driver version information. True: Version info API enabled. False: Version info API disabled. The optional APIs are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.4 Container: CrcPublishedInformation

Post-Build Variant Multiplicity: -

CRC driver

Multiplicity Configuration Class: -

8.3.1.4.1 CrcInitialValue16

Table 546 Specification for CrcInitialValue16

Name	CrcInitialValue16		
Description	Initial Value for this CRC calculation as specified by Autosar		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	65535 - 65535		
Default value	65535		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.4.2 CrcInitialValue32

Table 547 Specification for CrcInitialValue32

Name	CrcInitialValue32		
Description	Initial Value for this CRC calculation as specified by Autosar		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	4294967295 - 4294967295		
Default value	4294967295		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.4.3 CrcInitialValue32P4

Table 548 Specification for CrcInitialValue32P4

Name	CrcInitialValue32P4		
Description	Initial Value for this CRC calculation as specified by Autosar		

CRC driver**Table 548 Specification for CrcInitialValue32P4 (continued)**

Multiplicity	1..1	Type	EcuIntegerParamDef
Range	4294967295 - 4294967295		
Default value	4294967295		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.4.4 CrcInitialValue8**Table 549 Specification for CrcInitialValue8**

Name	CrcInitialValue8		
Description	Initial Value for this CRC calculation as specified by Autosar		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	255 - 255		
Default value	255		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.1.4.5 CrcInitialValue8H2F**Table 550 Specification for CrcInitialValue8H2F**

Name	CrcInitialValue8H2F		
Description	Initial Value for this CRC calculation as specified by Autosar		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	255 - 255		
Default value	255		
Post-build variant value	FALSE	Post-build variant multiplicity	-

CRC driver**Table 550 Specification for CrcInitialValue8H2F (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

8.3.2 Functions - Type definitions**8.3.3 Functions - APIs****8.3.3.1 Crc_CalculateCRC32P4****Table 551 Specification for Crc_CalculateCRC32P4 API**

Syntax	<pre>uint32 Crc_CalculateCRC32P4 (const uint8 * const Crc_DataPtr, const uint32 Crc_Length, const uint32 Crc_StartValue32, const boolean Crc_IsFirstCall)</pre>	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Crc_DataPtr Crc_Length Crc_StartValue32 Crc_IsFirstCall	Pointer to start address of data block to be calculated. Length of data block to be calculated in bytes. Start value when the algorithm starts. TRUE: First call in a sequence or individual CRC calculation. Start from initial value, ignore the parameter Crc_StartValue32P4. FALSE: Subsequent call in a call sequence. The parameter Crc_StartValue32P4 is interpreted to be the return value of the previous function call.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	32-bit result of CRC calculation
Description	This service makes a CRC32 calculation on number of data bytes specified by the API parameter Crc_Length, using the polynomial 0xF4ACFB13. This CRC routine is used by E2E profile 4.	

CRC driver**Table 551 Specification for Crc_CalculateCRC32P4 API (continued)**

	This API supports both single core and multi core operations. Refer the AUTOSAR SWS for further details.
Source	AUTOSAR
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>CRC_E_PARAM_LENGTH: Error ID for zero length check. This is implemented as MCAL safety error.</p> <p>CRC_E_PARAM_POINTER: Error ID for NULLPTR check. This is implemented as MCAL safety error.</p> <p>CRC_E_CHANNEL_TIMEOUT: Error ID for timeout check in hardware mode when waiting for free channels. This is implemented as MCAL safety error.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	Crc32P4Mode

8.3.3.2 Crc_CalculateCRC8H2F**Table 552 Specification for Crc_CalculateCRC8H2F API**

Syntax	<pre>uint8 Crc_CalculateCRC8H2F (const uint8 * const Crc_DataPtr, const uint32 Crc_Length, const uint8 Crc_StartValue8H2F, const boolean Crc_IsFirstCall)</pre>	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Crc_DataPtr Crc_Length Crc_StartValue8H2F Crc_IsFirstCall	<p>Pointer to start address of data block to be calculated.</p> <p>Length of data block to be calculated in bytes.</p> <p>Start value when the algorithm starts.</p> <p>TRUE: First call in a sequence or individual CRC calculation. Start from initial value, ignore the parameter Crc_StartValue8H2F.</p> <p>FALSE: Subsequent call in a call sequence. The parameter Crc_StartValue8H2F is interpreted to be the return value of the previous function call.</p>

CRC driver**Table 552 Specification for Crc_CalculateCRC8H2F API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint8	8-bit result of CRC calculation
Description	<p>This service makes a CRC8 calculation on number of data bytes specified by the API parameter Crc_Length using the polynomial 0x2F.</p> <p>This API supports both single core and multi core operations.</p> <p>Refer the AUTOSAR SWS for further details.</p>	
Source	AUTOSAR	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>CRC_E_PARAM_LENGTH: Error ID for zero length check. This is implemented as MCAL safety error.</p> <p>CRC_E_PARAM_POINTER: Error ID for NULLPTR check. This is implemented as MCAL safety error.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	Crc8H2FMode	
User hints	-	

8.3.3.3 Crc_GetVersionInfo**Table 553 Specification for Crc_GetVersionInfo API**

Syntax	<pre>void Crc_GetVersionInfo (Std_VersionInfoType * const Versioninfo)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	Versioninfo	Pointer where the version information of this driver is stored.
Parameters (in - out)	-	-

CRC driver**Table 553 Specification for Crc_GetVersionInfo API (continued)**

Return	void	-
Description	This service returns the version information of the CRC driver. This API supports both single core and multi core operations.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: CRC_E_PARAM_POINTER: Error ID for NULLPTR check. This is implemented as MCAL safety error. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	CrcVersionInfoApi	
User hints	-	

Crc_CalculateCRC8**Table 554 Specification for Crc_CalculateCRC8 API**

Syntax	<pre>uint8 Crc_CalculateCRC8 (const uint8 * const Crc_DataPtr, const uint32 Crc_Length, const uint8 Crc_StartValue8, const boolean Crc_IsFirstCall)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Crc_DataPtr Crc_Length Crc_StartValue8 Crc_IsFirstCall	Pointer to start address of data block to be calculated. Length of data block to be calculated in bytes. Start value when the algorithm starts. TRUE: First call in a sequence or individual CRC calculation. Start from initial value, ignore the parameter Crc_StartValue8. FALSE: Subsequent call in a call sequence. The parameter Crc_StartValue8 is interpreted to be the return value of the previous function call.
Parameters (out)	-	-

CRC driver**Table 554 Specification for Crc_CalculateCRC8 API (continued)**

Parameters (in - out)	-	-
Return	uint8	8-bit result of CRC calculation
Description	<p>This service makes a CRC8 calculation on number of data bytes specified by the API parameter Crc_Length, with SAE J1850 parameters.</p> <p>This API supports both single core and multi core operations.</p> <p>Refer the AUTOSAR SWS for further details.</p>	
Source	AUTOSAR	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>CRC_E_PARAM_LENGTH: Error ID for zero length check. This is implemented as MCAL safety error.</p> <p>CRC_E_PARAM_POINTER: Error ID for NULLPTR check. This is implemented as MCAL safety error.</p> <p>CRC_E_CHANNEL_TIMEOUT: Error ID for timeout check in hardware mode when waiting for free channels. This is implemented as MCAL safety error.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	Crc8Mode	
User hints	-	

8.3.3.5 Crc_CalculateCRC16**Table 555 Specification for Crc_CalculateCRC16 API**

Syntax	<pre>uint16 Crc_CalculateCRC16 (const uint8 * const Crc_DataPtr, const uint32 Crc_Length, const uint16 Crc_StartValue16, const boolean Crc_IsFirstCall)</pre>	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Crc_DataPtr Crc_Length Crc_StartValue16	Pointer to start address of data block to be calculated. Length of data block to be calculated in bytes. Start value when the algorithm starts.

CRC driver**Table 555 Specification for Crc_CalculateCRC16 API (continued)**

	Crc_IsFirstCall	TRUE: First call in a sequence or individual CRC calculation. Start from initial value, ignore the parameter Crc_StartValue16. FALSE: Subsequent call in a call sequence. The parameter Crc_StartValue16 is interpreted to be the return value of the previous function call.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint16	16-bit result of CRC calculation
Description	<p>This service makes a CRC16 calculation on number of data bytes specified by the API parameter Crc_Length.</p> <p>This API supports both single core and multi core operations.</p> <p>Refer the AUTOSAR SWS for further details.</p>	
Source	AUTOSAR	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>CRC_E_PARAM_LENGTH: Error ID for zero length check. This is implemented as MCAL safety error.</p> <p>CRC_E_PARAM_POINTER: Error ID for NULLPTR check. This is implemented as MCAL safety error.</p> <p>CRC_E_CHANNEL_TIMEOUT: Error ID for timeout check in hardware mode when waiting for free channels. This is implemented as MCAL safety error.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	Crc16Mode	
User hints	-	

8.3.3.6 Crc_CalculateCRC32**Table 556 Specification for Crc_CalculateCRC32 API**

Syntax	<pre>uint32 Crc_CalculateCRC32 (const uint8 * const Crc_DataPtr, const uint32 Crc_Length, const uint32 Crc_StartValue32, const boolean Crc_IsFirstCall)</pre>
Service ID	0x03

CRC driver**Table 556 Specification for Crc_CalculateCRC32 API (continued)**

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Crc_DataPtr Crc_Length Crc_StartValue32 Crc_IsFirstCall	Pointer to start address of data block to be calculated. Length of data block to be calculated in bytes. Start value when the algorithm starts. TRUE: First call in a sequence or individual CRC calculation. Start from initial value, ignore the parameter Crc_StartValue32. FALSE: Subsequent call in a call sequence. The parameter Crc_StartValue32 is interpreted to be the return value of the previous function call.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	32-bit result of CRC calculation
Description	<p>This service makes a CRC32 calculation on number of data bytes specified by the API parameter Crc_Length.</p> <p>This API supports both single core and multi core operations.</p> <p>Refer the AUTOSAR SWS for further details.</p>	
Source	AUTOSAR	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>CRC_E_PARAM_LENGTH: Error ID for zero length check. This is implemented as MCAL safety error.</p> <p>CRC_E_PARAM_POINTER: Error ID for NULLPTR check. This is implemented as MCAL safety error.</p> <p>CRC_E_CHANNEL_TIMEOUT: Error ID for timeout check in hardware mode when waiting for free channels. This is implemented as MCAL safety error.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	Crc32Mode	
User hints	-	

8.3.4 Notifications and Callbacks

The CRC driver does not provide any notifications or callbacks.

CRC driver

8.3.5 Scheduled functions

Not applicable for the CRC driver.

8.3.6 Interrupt service routines

CRC driver does not support any interrupts.

8.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

8.3.7.1 Development errors

The driver does not report any development errors.

8.3.7.2 Production errors

The driver does not report any production errors.

8.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 557 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
Error ID for timeout check in hardware mode when waiting for free channels. This is implemented as MCAL safety error.	IFX	CRC_E_CHANNEL_TIMEOUT=0xCA	Crc_CalculateCRC32, Crc_CalculateCRC16, Crc_CalculateCRC8, Crc_CalculateCRC32P4
Error ID for zero length check. This is implemented as MCAL safety error.	IFX	CRC_E_PARAM_LENGTH=0xC9	Crc_CalculateCRC32, Crc_CalculateCRC16, Crc_CalculateCRC8, Crc_CalculateCRC8H2F, Crc_CalculateCRC32P4
Error ID for NULLPTR check. This is implemented as MCAL safety error.	IFX	CRC_E_PARAM_POINTER=0xC8	Crc_CalculateCRC8H2F, Crc_CalculateCRC32P4, Crc_CalculateCRC8, Crc_CalculateCRC16, Crc_CalculateCRC32, Crc_GetVersionInfo

8.3.7.4 Runtime errors

The driver does not report any runtime errors.

8.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

CRC driver**8.3.8.1 Deviations**

The section describes the deviations from software specification.

Table 558

Reference	Deviation
Autosar SWS: SWS_Crc_00015	For 16-bit and 32-bit CRC calculations in the hardware mode, the input data stream should be aligned to 16-bit and 32-bit word boundaries respectively. If not aligned, the remaining bits are assumed to be zero.
Autosar SWS: SWS_Crc_00016	
Autosar SWS: SWS_Crc_00059	This is also a hardware limitation.

8.3.8.2 Limitations

This section describes the limitations from software specification.

Table 559 Known limitations

Reference	Limitation
Autosar SWS: Ecuc_Crc_00031	Hardware mode is not available for usage in CRC 8-bit computation using 0x2F polynomial due to hardware limitation. However, runtime and table methods are supported.

8.3.9 Unsupported hardware features

The following hardware features are not supported by the CRC driver:

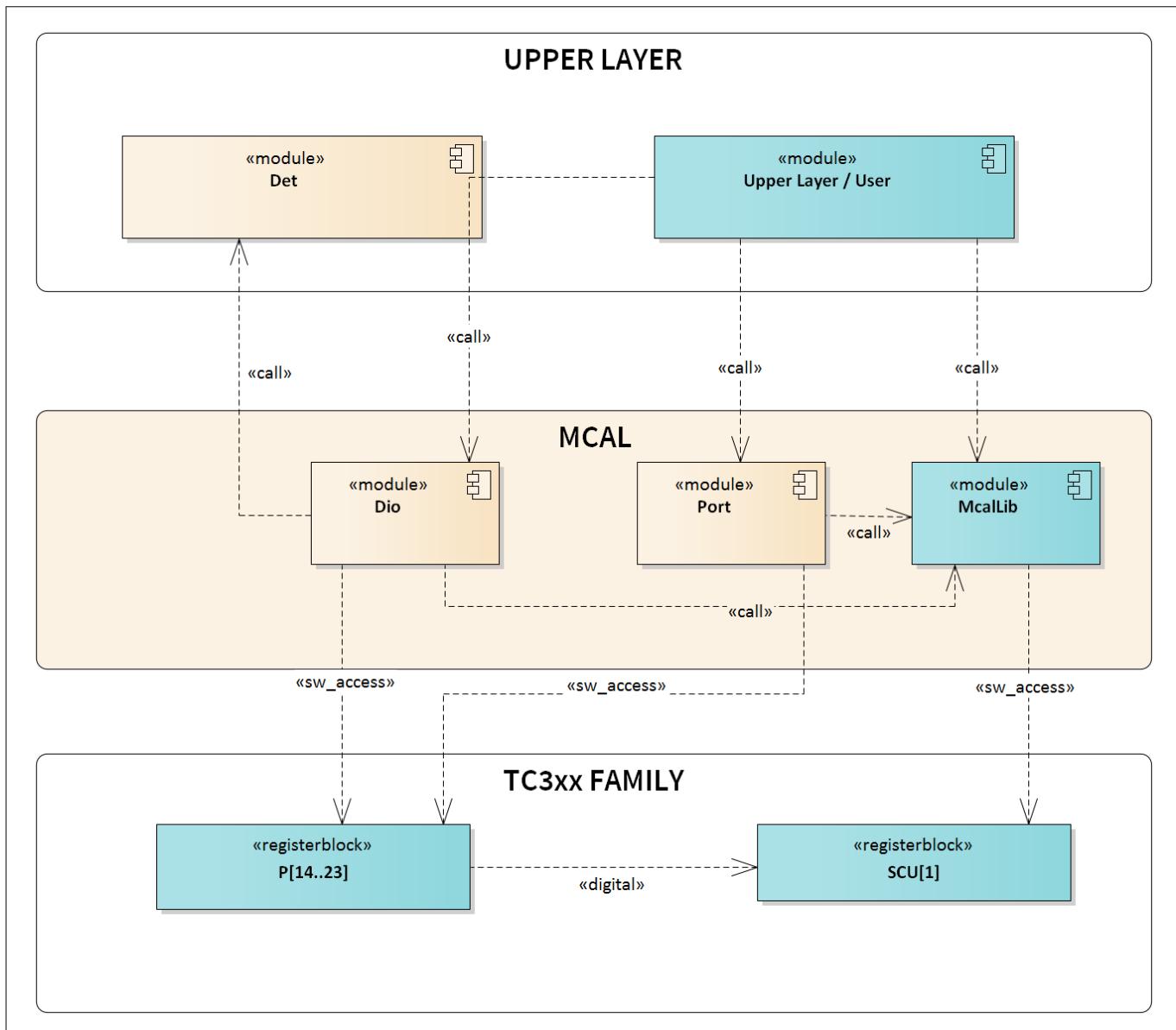
- Interrupt feature of the FCE
- Automatic signature check

DIO driver**9 DIO driver****9.1 User information****9.1.1 Description**

The DIO driver uses the port peripheral. The usage responsibility of the port peripheral is split by AUTOSAR into two modules. The PORT driver configures and sets the properties of port pin. The DIO driver reads or writes to the port pin. The DIO driver provides, port, channel and channel group based read and write access to the internal general purpose IO ports. All read and write services in the DIO driver are unbuffered. Channel refers to individual general purpose IO pin, port refers to DIO channels that are grouped by the hardware, and channel group refers to the formal logical combination of several adjoining DIO channels represented by a logical group. Note that a DIO channel group should belong to one DIO port.

9.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

DIO driver

Figure 103 **Mapping of hardware-software interfaces**

9.1.2.1 Port: primary hardware peripheral

Hardware functional features

The DIO driver is used for read and write access to the internal general purpose IO ports.

The key hardware functional features used by the driver are:

- Set, clear and toggle a portpin through the Pn_OUT and Pn_OMR register

The unsupported features of the DIO (since these are configured by the PORT driver) are:

- LVDS pad control
- Emergency stop
- Function decision control
- Controller selection
- Access enable
- Drive mode

DIO driver**Users of the hardware**

The PORT driver performs the configuration for port pins. The DIO driver performs input and output operation on the configured ports, therefore there is no conflict with the PORT driver. The user shall ensure that the port pins used by the other MCAL drivers are not conflicting with the DIO driver.

Hardware diagnostic features

Not applicable.

Hardware events

Not applicable.

9.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The DIO driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB clock signals for functioning.

Users of the Hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clocktree. To avoid conflict due to simultaneous writes, update to all the ENDINIT protected registers are performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the DIO driver.

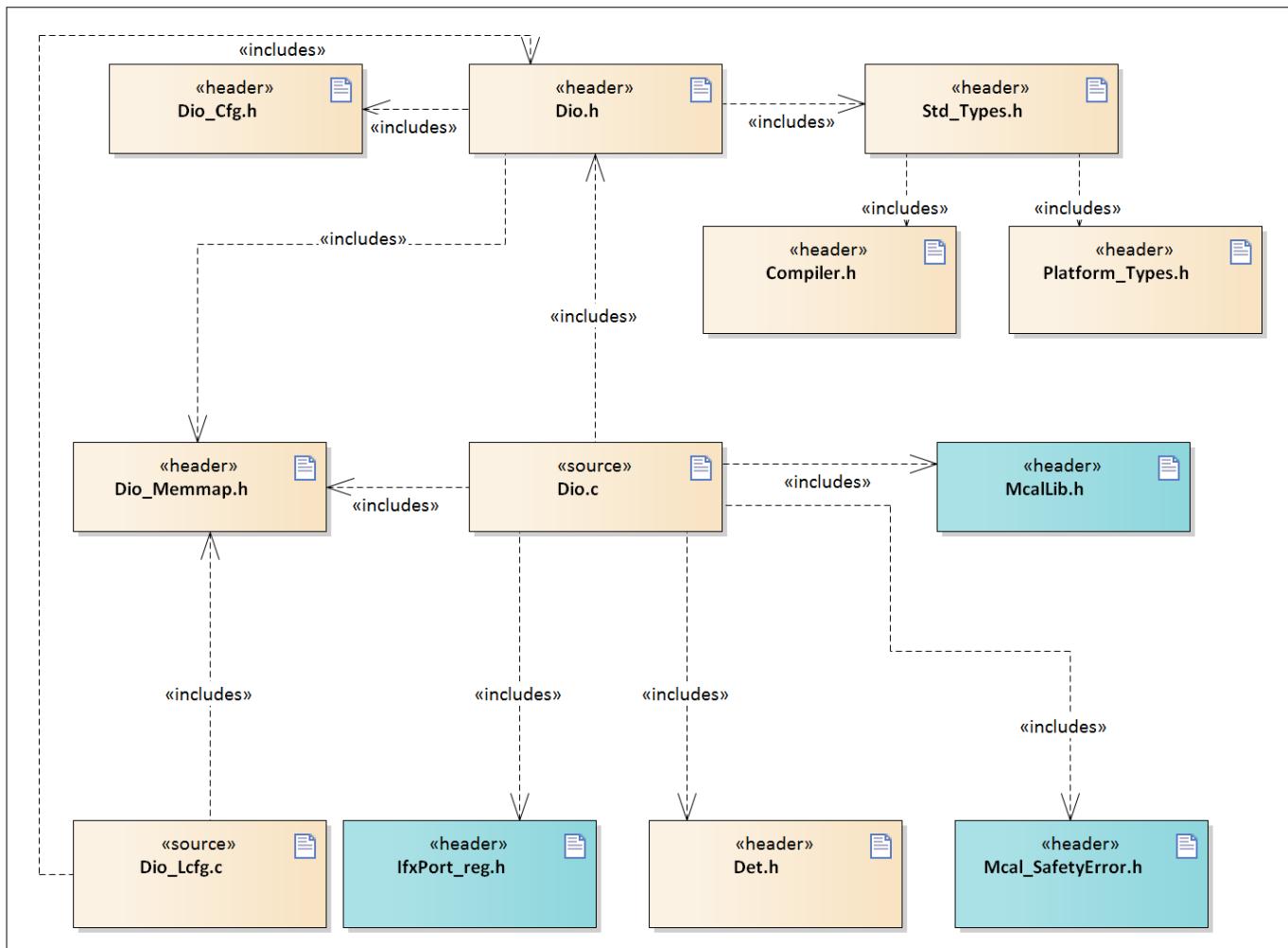
Hardware events

Hardware events from the SCU are not used by the DIO driver.

9.1.3 File structure

9.1.3.1 C file structure

This section provides details of the C files of the DIO driver.

DIO driver

Figure 104 C file structure
Table 560 C file structure

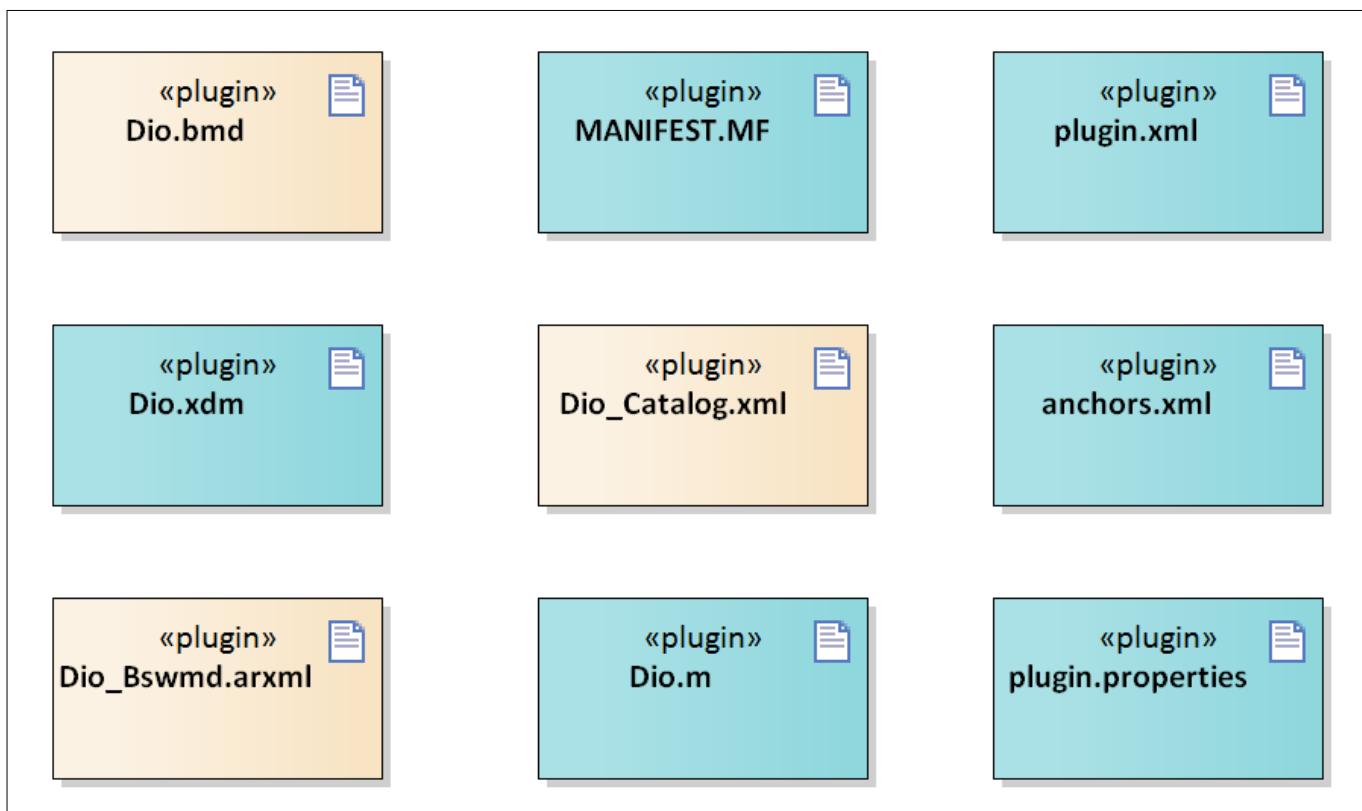
File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Det.h	Provides the exported interfaces of Development Error Tracer
Dio.c	File (Static) containing implementation of APIs
Dio.h	Header file (Static) defining prototypes of data structures and APIs
Dio_Cfg.h	Header file (Generated) containing constants, symbolic names and pre-processor macros.
Dio_Lcfg.c	File (Generated) containing objects to data structures
Dio_Memmap.h	File (Static) containing the memory section definitions used by the DIO driver
IfxPort_reg.h	SFR header file for Port
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR

DIO driver
Table 560 C file structure (continued)

File name	Description
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

9.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the DIO driver.

**Figure 105 Code generator plugin files****Table 561 Code generator plugin files**

File name	Description
Dio.bmd	AUTOSAR format XML data model schema file(for each device)
Dio.m	Code template macro file for DIO driver
Dio.xdm	Tresos format XML data model schema file
Dio_Bswmd.arxml	AUTOSAR format module description file
Dio_Catalog.xml	AUTOSAR format catalog file as per catalog_V3_0_0.ml.xsd
MANIFEST.MF	Tresos plugin support file containing the metadata for DIO driver
anchors.xml	Tresos anchors support file for the DIO driver
plugin.properties	Tresos plugin support file for the DIO driver
plugin.xml	Tresos plugin support file for the DIO driver

DIO driver**9.1.4 Integration hints**

This section lists the key points that an integrator or user of the DIO driver must consider.

9.1.4.1 Integration with AUTOSAR stack

- **EcuM**

The EcuM module is not required for the integrating the DIO driver.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user specific memory regions. To achieve this, all the relocatable elements of the driver are en-capsulated in different memory section macros. These macros are defined in the `Dio_MemMap.h` file.

The `Dio_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements

DIO driver

are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows:

```

/***** CONST DATA -- *****/
#if defined DIO_START_SEC_CONST_ASIL_B_GLOBAL_16
/***** User pragmas here *****/
#undef DIO_START_SEC_CONST_ASIL_B_GLOBAL_16
#undef MEMMAP_ERROR

#elif defined DIO_STOP_SEC_CONST_ASIL_B_GLOBAL_16
/***** User pragmas here *****/
#undef DIO_STOP_SEC_CONST_ASIL_B_GLOBAL_16
#undef MEMMAP_ERROR

/***** CONFIG DATA *****/
#elif defined DIO_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/***** User pragmas here *****/
#undef DIO_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR
#elif defined DIO_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/***** User pragmas here *****/
#undef DIO_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

/***** CODE DATA *****/
#elif defined DIO_START_SEC_CODE_ASIL_B_GLOBAL
/***** User pragmas here *****/
#undef DIO_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#elif defined DIO_STOP_SEC_CODE_ASIL_B_GLOBAL
/***** User pragmas here *****/
#undef DIO_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Dio_MemMap.h, wrong pragma command"
#endif

```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The DIO driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the DIO driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is not required for the integration of the DIO driver.

DIO driver

- **SchM**

The SchM is not required for the integration of the DIO driver.

- **Safety Error**

The DIO driver will report all the detected safety errors through the API `Mcal_ReportSafetyError()`.

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The DIO driver does not provide any callbacks or notifications.

9.1.4.2 Multicore and Resource Manager

The DIO driver supports the multicore functionality. The DIO driver service can be accessed from any core.

9.1.4.3 MCU support

The DIO driver does not use any services provided by the MCU driver.

9.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the DIO driver through the PORT configuration and initialize the port pins prior to invoking the DIO APIs.

9.1.4.5 DMA support

The DIO driver does not use any services provided by the DMA driver.

9.1.4.6 Interrupt connections

The DIO driver does not use any interrupt source.

DIO driver

9.1.4.7 Example usage

DIO driver published symbolic names

The DIO channel and DIO port symbolic names are defined in the `Dio_Cfg.h` (derivative or board specific header file).

Configuration of DIO Channel

The symbolic names for DIO channels is generated as follows. These symbolic names are of type `Dio_ChannelType`.

Example for DIO channel configuration

```
/* User Defined Symbolic Names for the DIO CHANNELS */
#define DioConf_DioChannel_MOTOR_START_STOP (DIO_CHANNEL_0_5)
#define DioConf_DioChannel_MOTOR_DIRECTION (DIO_CHANNEL_0_8)
#define DioConf_DioChannel_CAN_TRCV_ENT0 (DIO_CHANNEL_1_1)
#define DioConf_DioChannel_CAN_TRCV_NSTB0 (DIO_CHANNEL_1_2)
```

Configuration of DIO Port

The symbolic names for DIO port is generated as follows. These symbolic names are of type `Dio_PortType`.

Example for DIO port configuration

```
/* User Defined Symbolic Names for the DIO PORTS */
#define DioConf_DioPort_MOTOR_CTL_PORT (DIO_PORT_0)
#define DioConf_DioPort_CAN_TRCV_PORT (DIO_PORT_1)
```

Configuration of DIO Channel Group

The symbolic names for DIO channel group is generated as follows. These symbolic names are of type `Dio_ChannelGroupType`.

Example for DIO channel group configuration

```
/* User Defined Symbolic Names for the DIO CHANNEL GROUPS */
#define DioConf_DioChannelGroup_MOTOR_CTL_GRP
(&Dio_Config.Dio_ChannelGroupConfigPtr[0])
#define DioConf_DioChannelGroup_CAN_TRCV_GRP
(&Dio_Config.Dio_ChannelGroupConfigPtr[1])
```

Using the APIs

The following code listing shows example calls to different services provided by the DIO driver. This code listing uses symbols as described earlier.

DIO driver

Using of DIO driver services

```

Dio_levelType ChannelVal;
Dio_PortLevelType PortVal;
Dio_PortLevelType ChannelGrpVal;

/* Set level STD_HIGH for port 0 channel 5 */
Dio_WriteChannel(DioConf_DioChannel_MOTOR_START_STOP, STD_HIGH);

/* Read level of port 0 channel 8 */
ChannelVal = Dio_ReadChannel(DioConf_DioChannel_MOTOR_DIRECTION);

/* Write port 1 with all pins set to HIGH */
Dio_WritePort(DioConf_DioPort_CAN_TRCV_PORT, (Dio_PortLevelType) 0x7FFF);

/* Read the level of all the pins of port 0 */
PortVal = Dio_ReadPort(DioConf_DioPort_MOTOR_CTL_PORT);

/* Write to channel group 0 */
Dio_WriteChannelGroup(DioConf_DioChannelGroup_MOTOR_CTL_GRP,
(Dio_PortLevelType) 0xA);

/* Read from channel group 1 */
ChannelGrpVal = Dio_ReadChannelGroup(DioConf_DioChannelGroup_CAN_TRCV_GRP);

```

9.1.5 Key architectural considerations

9.1.5.1 Implementation type

The DIO driver is implemented as Variant Link Time.

9.1.5.2 User mode support

The DIO driver operates both in the User-1 and Supervisor modes without the need of any configuration parameter to configure the behavior.

[cover parentID DIO={1A65EADD-AFD0-4845-B2D2-8257E086DD67}]

[cover parentID DIO={D582CB88-7E55-42fd-A4A6-511FFDD38D19}]

DIO driver

9.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Configuration check**

The user should ensure that the generated configuration is correct against the GUI configuration.

[cover parentID DIO={A4C58AA6-0186-47d1-810A-13AE19E45737}]

- **DIO Flip channel**

Due to the configured pin drive strength and load capacitance connected to the pin, there is a delayed response on the pin to flip. After the call to Dio_FlipChannel() API, the user shall read the pin level using Dio_ReadChannel() API after the necessary delay and confirm the flipped level of the pin. For the delay information refer the datasheet.(Rise / Fall time) Affected APIs: Dio_FlipChannel

[cover parentID DIO={50AE62EA-7A3B-421a-A9F5-1595EAFE62DD}]

- **DIO readonly usage**

The user should ensure that the DIO driver is not used on the analog pins.

[cover parentID DIO={EEBBE858-7E80-40bb-92B2-DA4D61CA9257}]

- **DIO write verification**

The user should perform read operation after each write operation to ensure realization of desired operations.

[cover parentID DIO={A62F0251-C5CC-4b25-B83A-AD9F504F62F6}]

- **Port initcheck**

The DIO driver needs PORT driver to be initialized prior to use of the DIO driver APIs, therefore the Port_InitCheck (AoU) shall be performed by the integrator to check initialization of PORT driver as DIO driver works on pins and ports, which are configured by the PORT driver.

[cover parentID DIO={A2AE117E-4BCF-46c2-9F85-3E871ABDF72F}]

9.3 Reference information

9.3.1 Configuration interfaces

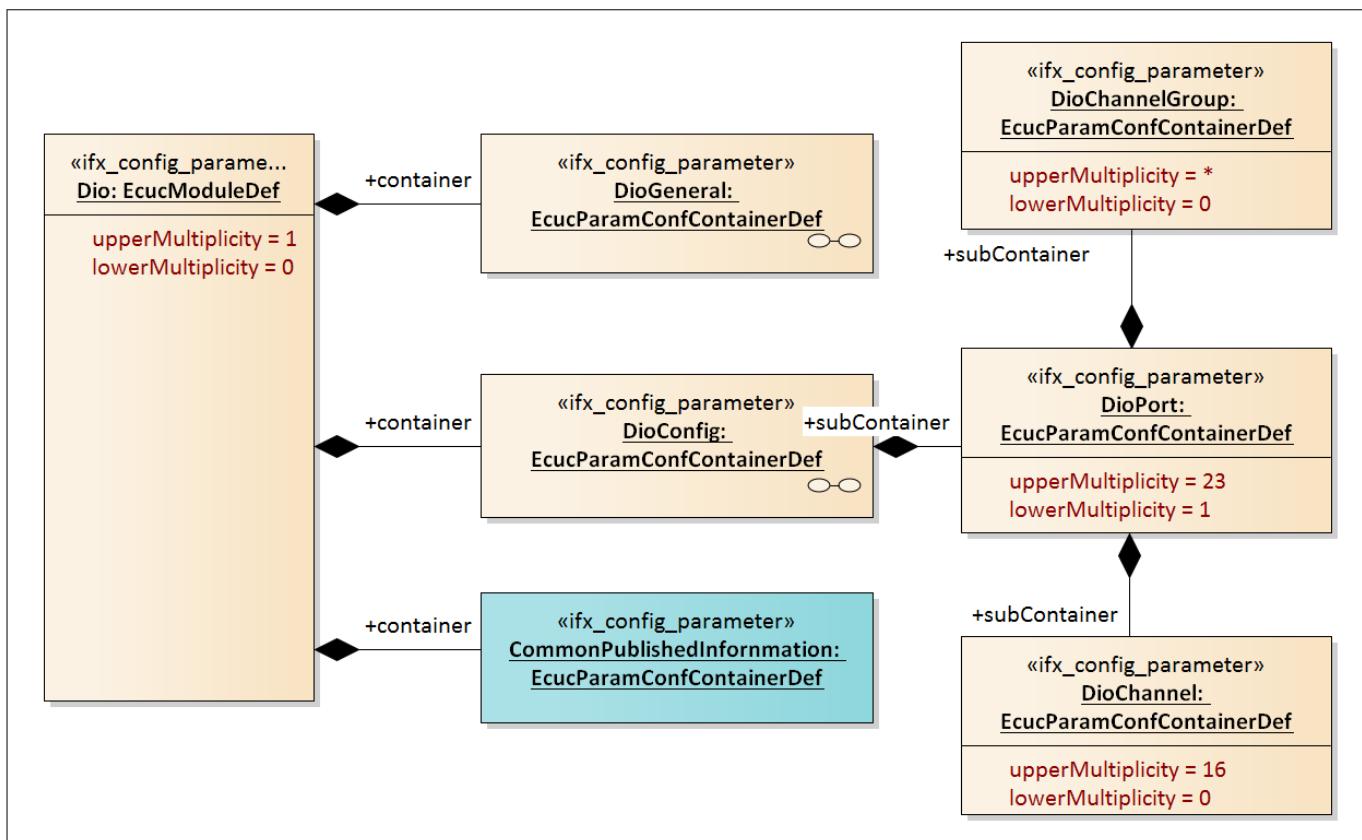


Figure 106 Container hierarchy along with their configuration parameters

9.3.1.1 Container: CommonPublishedInformation

This section describes the information about the module published by the Dio Driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

9.3.1.1.1 ArMajorVersion

Table 562 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	This parameter provides the major version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-

DIO driver**Table 562 Specification for ArMajorVersion (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.1.2 ArMinorVersion**Table 563 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	This parameter provides the minor version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.1.3 ArPatchVersion**Table 564 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	This parameter provides the patch version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

DIO driver**9.3.1.1.4 ModuleId****Table 565 Specification for ModuleId**

Name	ModuleId		
Description	Module ID of DIO		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	120		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.1.5 Release**Table 566 Specification for Release**

Name	Release		
Description	Aurix derivative used for the implementation.		
Multiplicity	1..1	Type	EcuStringParamDef
Range	String		
Default value	As per Hardware derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.1.6 SwMajorVersion**Table 567 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	This parameter provides the major version of the Software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		

DIO driver**Table 567 Specification for SwMajorVersion (continued)**

Default value	As per driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.1.7 SwMinorVersion**Table 568 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	This parameter provides the minor version of the Software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.1.8 SwPatchVersion**Table 569 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	This parameter provides the patch version of the Software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

DIO driver**Table 569 Specification for SwPatchVersion (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.1.9 VendorID**Table 570 Specification for VendorID**

Name	VendorID		
Description	This parameter provides the Vendor Id		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.2 Container: Dio

Configuration of the Dio (Digital IO) module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

9.3.1.2.1 Config Variant**Table 571 Specification for Config Variant**

Name	Config Variant		
Description			
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	Variant LinkTime: Only parameters with "Pre-compile time" and "Link time" are allowed in this variant.		
Default value	Variant LinkTime		
Post-build variant value	FALSE	Post-build variant multiplicity	-

DIO driver
Table 571 Specification for Config Variant (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.3 Container: DioChannel

Configuration of an individual DIO channel.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

9.3.1.3.1 DioChannelId

Table 572 Specification for DioChannelId

Name	DioChannelId		
Description	Channel Id of the DIO channel. This value will be assigned to the symbolic names and consecutive value is calculated for each new channel Id.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 15		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

9.3.1.4 Container: DioChannelGroup

Definition and configuration of DIO channel groups. A channel group represents several adjoining DIO channels represented by a logical group. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the container's short name will be used in the Ecu Configuration Description to specify the symbolic name of the channel group.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Link-Time

DIO driver**9.3.1.4.1 DioChannelGroupIdentification****Table 573 Specification for DioChannelGroupIdentification**

Name	DioChannelGroupIdentification		
Description	<p>The DIO channel group is identified in DIO API by a pointer to a data structure (of type Dio_ChannelGroupType). That data structure contains the channel group information.</p> <p>This parameter contains the code fragment that has to be inserted in the API call to the calling module to get the address of the variable in memory which holds the channel group information.</p>		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

9.3.1.4.2 DioPortMask**Table 574 Specification for DioPortMask**

Name	DioPortMask		
Description	This should be the mask which defines the positions of the channel group. The channels should consist of adjoining bits in the same port. The data type depends on the port width.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Link-Time	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

DIO driver**9.3.1.4.3 DioPortOffset****Table 575 Specification for DioPortOffset**

Name	DioPortOffset		
Description	<p>The position of the Channel Group on the port counted from the LSB. This value can be derived from DioPortMask.</p> <p>Calculation Formula = Position of the first bit of DioPortMask which is set to '1' counted from LSB.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 15		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Link-Time	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

9.3.1.5 Container: DioConfig

This container contains the configuration parameters and sub containers of the AUTOSAR DIO module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

9.3.1.6 Container: DioGeneral

General DIO module configuration parameters.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

9.3.1.6.1 DioDevErrorDetect**Table 576 Specification for DioDevErrorDetect**

Name	DioDevErrorDetect		
Description	<p>Switches the Default Error Tracer detection and notification ON or OFF.</p> <p>True: ON.</p> <p>False: OFF.</p>		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		

DIO driver**Table 576 Specification for DioDevErrorDetect (continued)**

Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

9.3.1.6.2 DioFlipChannelApi**Table 577 Specification for DioFlipChannelApi**

Name	DioFlipChannelApi		
Description	Switch to Adds / Removes the service of Dio_FlipChannel() from the code.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

9.3.1.6.3 DioSafetyEnable**Table 578 Specification for DioSafetyEnable**

Name	DioSafetyEnable		
Description	Switch to enable reporting of safety Errors (Range and plausibility check).		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		

DIO driver**Table 578 Specification for DioSafetyEnable (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

9.3.1.6.4 DioVersionInfoApi**Table 579 Specification for DioVersionInfoApi**

Name	DioVersionInfoApi		
Description	Switch for enabling the API Dio_GetVersionInfo() which returns the version information of the module.		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

9.3.1.7 Container: DioPort

The configuration of individual DIO ports, consisting of channels and possible channel groups. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the container's short name will be used in the Ecu configuration description to specify the symbolic name of the port.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Link-Time

9.3.1.7.1 DioPortId**Table 580 Specification for DioPortId**

Name	DioPortId
-------------	-----------

DIO driver**Table 580 Specification for DioPortId (continued)**

Description	Numeric identifier of the DIO port. Not all MCU ports may be used for DIO, thus there may be gaps in the list of PORTIDs. This value will be assigned to the DIO port symbolic name (i.e. the SHORT-NAME of the DioPort container).		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 41		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

9.3.2 Functions - Type definitions**9.3.2.1 Dio_ChannelGroupType****Table 581 Specification for Dio_ChannelGroupType**

Syntax	Dio_ChannelGroupType	
Type	Structure	
File	Dio.h	
Range	uint16 Mask	This element mask which defines the positions of the channel group. Range: 0x0 - 0xFFFF
	uint8 Offset	This element must be the position of the Channel Group on the port, counted from the LSB. Range: 0 - 15
	Dio_PortType Port	This should be the port on which the Channel group is defined. Range: Refer Data Type
Description	Type for the definition of a channel group, which consists of several adjoining channels within a port.	
Source	AUTOSAR	

9.3.2.2 Dio_ChannelType**Table 582 Specification for Dio_ChannelType**

Syntax	Dio_ChannelType
---------------	-----------------

DIO driver**Table 582 Specification for Dio_ChannelType (continued)**

Type	uint16	
File	Dio.h	
Range	0 to Number of channels available	Number of Channels in a port
Description	Numeric ID of a DIO channel	
Source	AUTOSAR	

9.3.2.3 Dio_ConfigType**Table 583 Specification for Dio_ConfigType**

Syntax	Dio_ConfigType
Type	Structure
File	Dio.h
Description	Defines the type for data structure containing the set of configuration parameters required for initializing the DIO driver.
Source	AUTOSAR

9.3.2.4 Dio_LevelType**Table 584 Specification for Dio_LevelType**

Syntax	Dio_LevelType	
Type	uint8	
File	Dio.h	
Range	0x00	STD_LOW Physical state 0V
	0x01	STD_HIGH Physical state 5V or 3.3V
Description	These are the possible levels a DIO channel can have (input or output)	
Source	AUTOSAR	

9.3.2.5 Dio_PortType**Table 585 Specification for Dio_PortType**

Syntax	Dio_PortType	
Type	uint8	
File	Dio.h	
Range	0 to 41	Number of Dio Ports
Description	Numeric ID of a DIO Port	
Source	AUTOSAR	

DIO driver**9.3.2.6 Dio_PortLevelType****Table 586 Specification for Dio_PortLevelType**

Syntax	Dio_PortLevelType	
Type	uint16	
File	Dio.h	
Range	0x0 – 0xFFFF	It is a type of the value of Dio Port. It inherits the size of the largest port.
Description	It is a type of the value of Dio Port. It inherits the size of the largest port.	
Source		

9.3.3 Functions - APIs

This section lists all the APIs of DIO driver.

9.3.3.1 Dio_FlipChannel**Table 587 Specification for Dio_FlipChannel API**

Syntax	Dio_LevelType Dio_FlipChannel (const Dio_ChannelType ChannelId)	
Service ID	0x11	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	ChannelId	ID of DIO channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Dio_LevelType	The physical level of the corresponding Pin
Description	Service to flip (change from 1 to 0 or from 0 to 1) the level of a channel and return the level of the channel after the flip.	
Source	AUTOSAR	
Error handling	DET: DIO_E_PARAM_INVALID_CHANNEL_ID: Invalid channel name requested. Runtime Errors: None DEM: None Safety Errors: None	

DIO driver**Table 587 Specification for Dio_FlipChannel API (continued)**

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	DioFlipChannelApi
User hints	-

9.3.3.2 Dio_GetVersionInfo**Table 588 Specification for Dio_GetVersionInfo API**

Syntax	void Dio_GetVersionInfo (Std_VersionInfoType * const VersionInfo)	
Service ID	0x12	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfo	Pointer to where to store the version information of this module.
Parameters (in - out)	-	-
Return	void	-
Description	Service to get the version information of this module.	
Source	AUTOSAR	
Error handling	DET: DIO_E_PARAM_POINTER: API service called with a NULL pointer. In case of this error, the API service should return immediately without any further action, besides reporting this development error. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	DioVersionInfoApi	
User hints	-	

DIO driver**9.3.3.3 Dio_ReadChannel****Table 589 Specification for Dio_ReadChannel API**

Syntax	<pre>Dio_LevelType Dio_ReadChannel (const Dio_ChannelType ChannelId)</pre>	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	ChannelId	ID of DIO channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Dio_LevelType	The physical level of the corresponding Pin
Description	Returns the value of the specified DIO channel.	
Source	AUTOSAR	
Error handling	<p>DET: DIO_E_PARAM_INVALID_CHANNEL_ID: Invalid channel name requested.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

9.3.3.4 Dio_ReadChannelGroup**Table 590 Specification for Dio_ReadChannelGroup API**

Syntax	<pre>Dio_PortLevelType Dio_ReadChannelGroup (const Dio_ChannelGroupType * const ChannelGroupIdPtr)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	

DIO driver**Table 590 Specification for Dio_ReadChannelGroup API (continued)**

Re-entrancy	Reentrant	
Parameters (in)	ChannelGroupIdPtr	Pointer to ChannelGroup
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Dio_PortLevelType	Level of a subset of the adjoining bits of a port
Description	This Service reads a subset of the adjoining bits of a port.	
Source	AUTOSAR	
Error handling	DET: DIO_E_PARAM_INVALID_GROUP : Invalid ChannelGroup requested. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

9.3.3.5 Dio_ReadPort**Table 591 Specification for Dio_ReadPort API**

Syntax	Dio_PortLevelType Dio_ReadPort (const Dio_PortType PortId)	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	PortId	ID of DIO Port
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Dio_PortLevelType	Level of all channels of that port
Description	Returns the level of all channels of that port.	

DIO driver**Table 591 Specification for Dio_ReadPort API (continued)**

Source	AUTOSAR
Error handling	<p>DET:</p> <p>DIO_E_PARAM_INVALID_PORT_ID : Invalid port name requested.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

9.3.3.6 Dio_WriteChannel**Table 592 Specification for Dio_WriteChannel API**

Syntax	<pre>void Dio_WriteChannel (const Dio_ChannelType ChannelId, const Dio_LevelType Level)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	ChannelId Level	ID of Dio Channel Value to be written (STD_HIGH / STD_LOW)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to set specified level for a channel	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>DIO_E_PARAM_INVALID_CHANNEL_ID: Invalid channel name requested.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

DIO driver**Table 592 Specification for Dio_WriteChannel API (continued)**

Configuration dependencies	-
User hints	-

9.3.3.7 Dio_WriteChannelGroup**Table 593 Specification for Dio_WriteChannelGroup API**

Syntax	<pre>void Dio_WriteChannelGroup (const Dio_ChannelGroupType * const ChannelGroupIdPtr, const Dio_PortLevelType Level)</pre>	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	ChannelGroupIdPtr Level	Pointer to ChannelGroup Value to be written
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to set a subset of the adjoining bits of a port to a specified level.	
Source	AUTOSAR	
Error handling	<p>DET: DIO_E_PARAM_INVALID_GROUP : Invalid ChannelGroup requested. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

DIO driver**9.3.3.8 Dio_WritePort****Table 594 Specification for Dio_WritePort API**

Syntax	<pre>void Dio_WritePort (const Dio_PortType PortId, const Dio_PortLevelType Level)</pre>	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	PortId Level	ID of DIO Port Value to be written
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to set specified level for Dio Port	
Source	AUTOSAR	
Error handling	DET: DIO_E_PARAM_INVALID_PORT_ID : Invalid port name requested. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

9.3.4 Notifications and Callbacks

The driver does not support any notification and callbacks.

9.3.5 Scheduled functions

The driver does not support any scheduled functions.

9.3.6 Interrupt service routines

Not applicable for the DIO driver.

DIO driver

9.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

9.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Note: *The following error IDs are also reported as safety errors.*

Table 595 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
Invalid port name requested.	AUTOSAR	DIO_E_PARAM_INVALID_PORT_ID =0x14	Dio_WritePort, Dio_ReadPort
Invalid channel name requested.	AUTOSAR	DIO_E_PARAM_INVALID_CHANNEL_ID=0x0A	Dio_FlipChannel, Dio_WriteChannel, Dio_ReadChannel
Invalid ChannelGroup requested.	AUTOSAR	DIO_E_PARAM_INVALID_GROUP =0x1F	Dio_WriteChannelGroup, Dio_ReadChannelGroup
API service called with a NULL pointer. In case of this error, the API service should return immediately without any further action, besides reporting this development error.	AUTOSAR	DIO_E_PARAM_POINTER=0x20	Dio_GetVersionInfo

9.3.7.2 Production errors

The driver does not report any production errors.

9.3.7.3 Safety errors

The driver does not report any safety errors.

9.3.7.4 Runtime errors

The driver does not report any runtime errors.

9.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

DIO driver**9.3.8.1 Deviations****Table 596 Known Deviations**

Reference	Deviation
AUTOSAR_SWS_DIODriver.pdf, AUTOSAR Release 4.2.2: Section 10.1.2 DIO.	The DIO driver is implemented as post-build variant support false, instead of true. Issue is raised via Bugzilla(77125) and confirmed for update in the future AUTOSAR release.
AUTOSAR_SWS_DIODriver.pdf, AUTOSAR Release 4.2.2:ECUC_DIO_00150: DioPortMask.	The parameter DioPortMask is implemented as pre-compile instead of link time. The parameter DioPortMask is used for generating derived macros. Therefore, this parameter is implicitly converted to pre-compile.

9.3.8.2 Limitations

Not applicable for DIO.

9.3.9 Unsupported hardware features

The DIO driver only provides discrete I/O functionality. Therefore following features of PORT peripheral are not supported by the DIO driver:

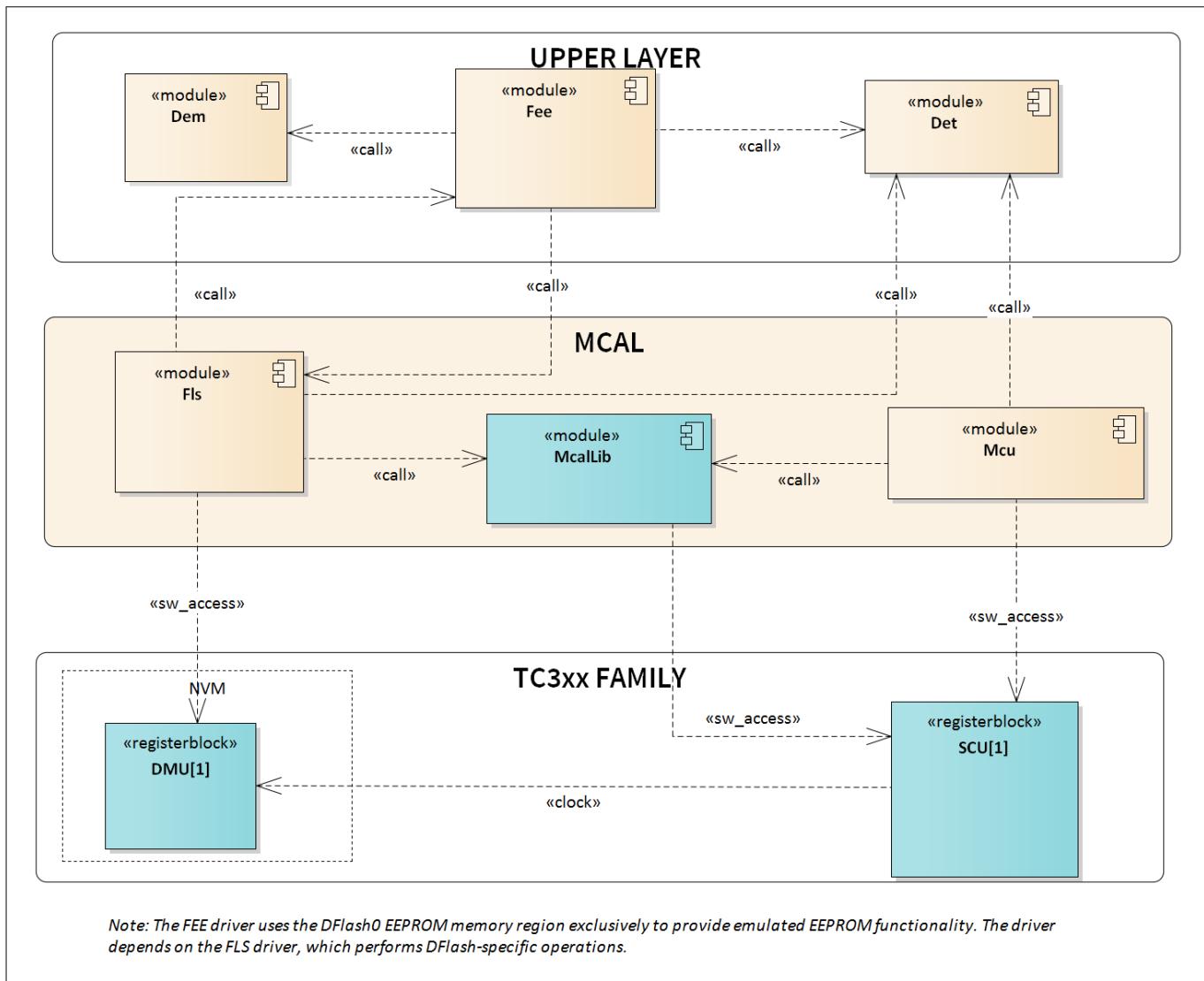
- LVDS pad control
- Port emergency stop
- Port pad drive modes
- Port pin function decision control
- Port pin controller select

FEE driver**10 FEE driver****10.1 User information****10.1.1 Description**

The FEE driver provides Flash EEPROM emulation as per AUTOSAR through standard services and well-defined configuration. Additionally, customer specific features such as virgin Flash handling, quasi-static (QS) data block support, un-configured data block support, erase-suspend resume are also made available. In view of the second generation of AURIX™ hardware, the DFlash0 EEPROM memory region is exclusively used by the FEE driver to provide emulated EEPROM functionality. The DFlash-specific operations such as erase, read and write are implemented in the FLS driver. The DFlash1 is not used by the FEE. DFlash1 is reserved for HSM. The FEE driver is delivered as a Post-Build variant as FEE functionality is not only available for the run time application but also for the boot code.

10.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

FEE driver

Figure 107 Mapping of hardware-software interfaces
10.1.3 File structure
10.1.3.1 C File Structure

FEE driver

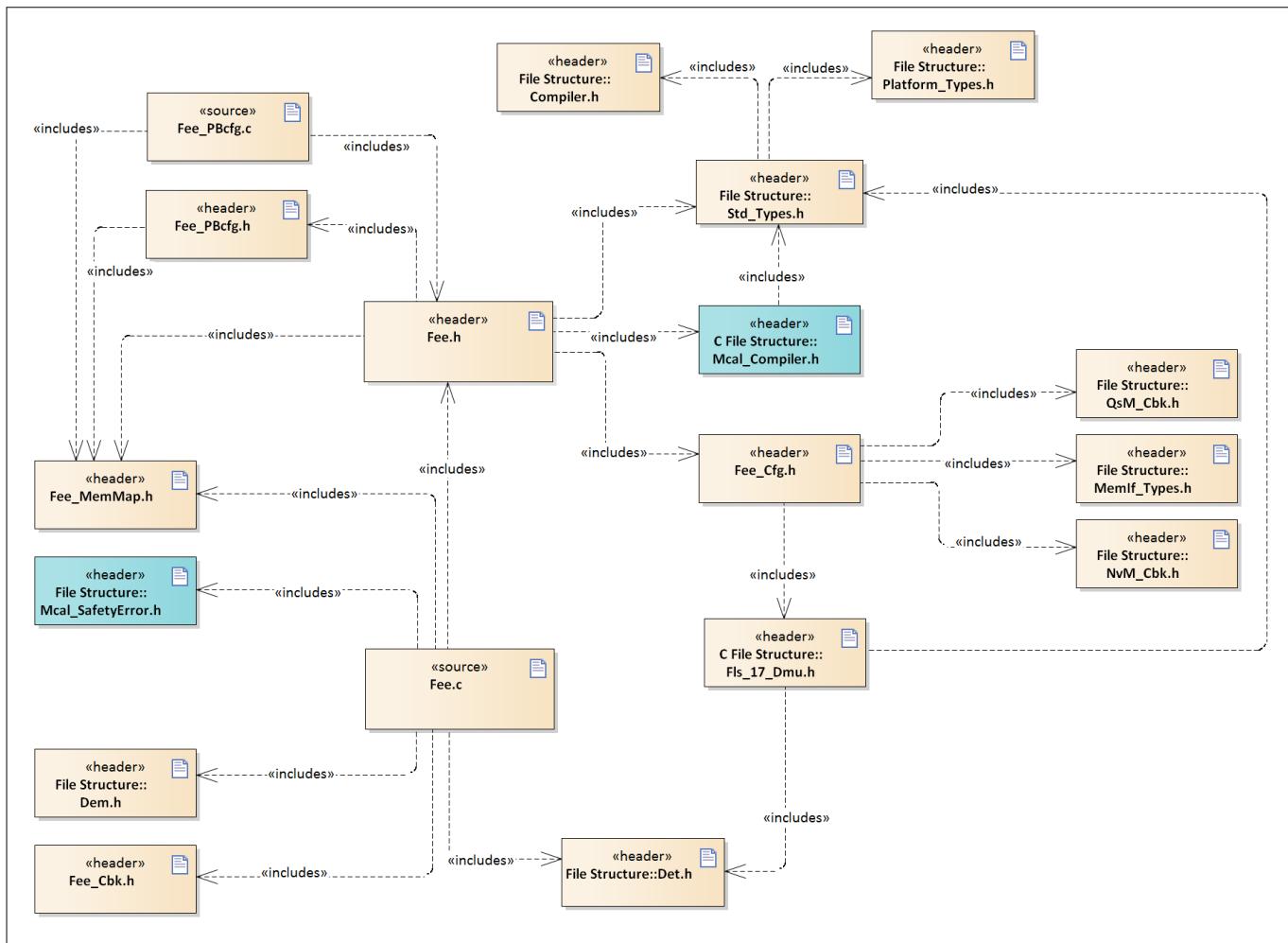


Figure 108 C File Structure

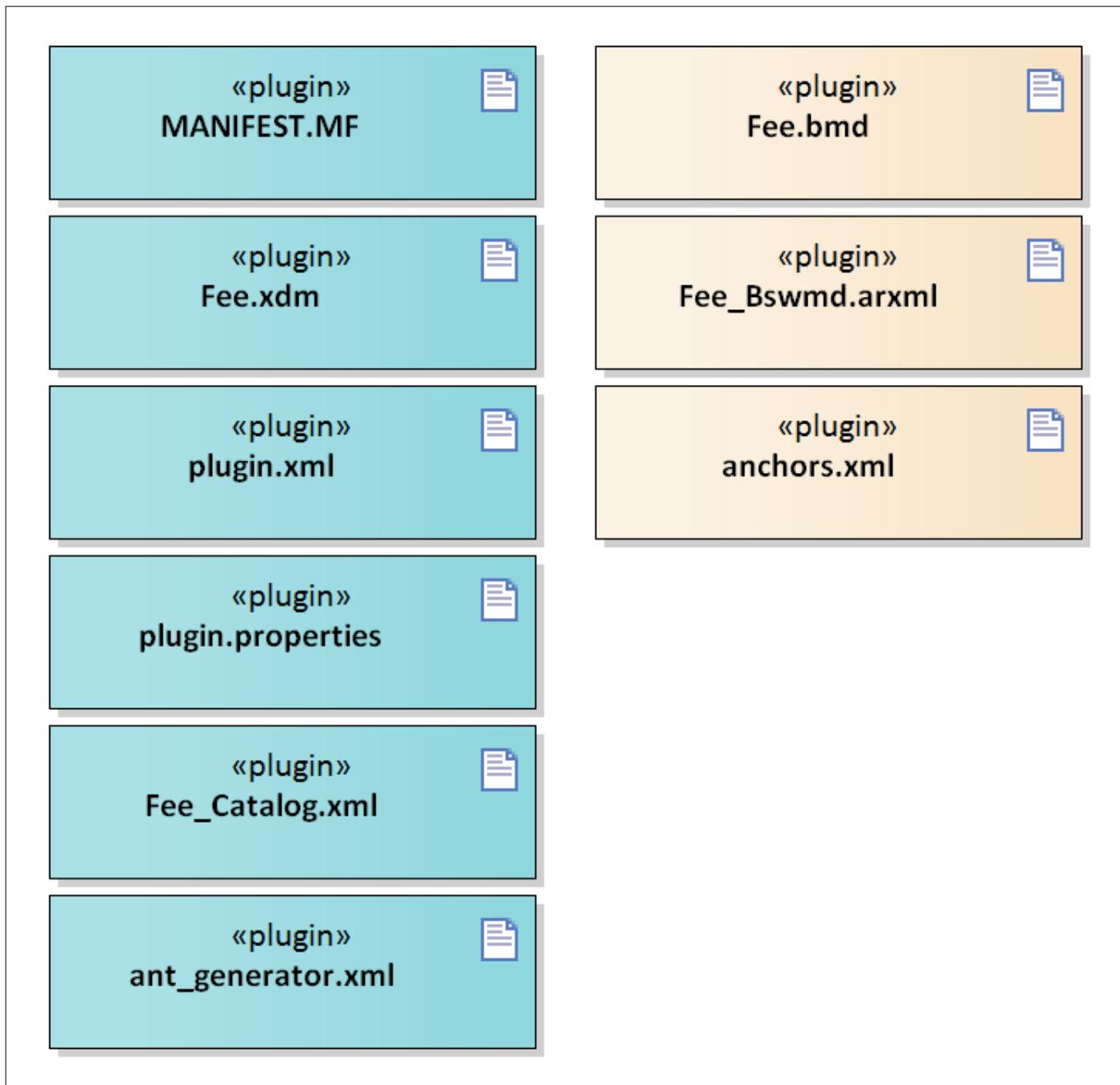
Table 597 C File Structure

File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Dem.h	Provides the exported interfaces of Diagnostic Event Manager
Det.h	Provides the exported interfaces of Development Error Tracer
Fee.c	Contains the functionality of the FEE driver
Fee.h	This header file exports the functionality of the FEE driver
Fee_Cbk.h	This header file contains the declarations of callback functions provided by the FEE driver
Fee_Cfg.h	Contains pre-compile configuration data of the FEE driver
Fee_MemMap.h	File containing the memory section definitions used by the FEE driver
Fee_PBcfg.c	Contains configuration data of the FEE driver
Fee_PBcfg.h	File (Generated) containing declaration of the post-build configuration data structures

FEE driver**Table 597 C File Structure (continued)**

File name	Description
Fls_17_Dmu.h	This header file exports macros, type definitions, interrupt service routine and function prototypes for the Flash driver
Mcal_Compiler.h	Provides abstraction for TriCore-intrinsic instruction
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
MemIf_Types.h	Header file containing the type declaration of MemIf
NvM_Cbk.h	Call back header file for NvM
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
QsM_Cbk.h	Interface file that provides the callback function prototypes to be used by FEE driver.
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

10.1.3.2 Code Generator Plugin Files

FEE driver**Figure 109 Code Generator Plugin Files****Table 598 Code Generator Plugin Files**

File name	Description
Fee.bmd	AUTOSAR format XML data model schema file (for each device)
Fee.xdm	Tresos format XML data model schema file
Fee_Bswmd.arxml	AUTOSAR format module description file
Fee_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the metadata for FEE driver
anchors.xml	Tresos anchors support file for the FEE driver

FEE driver
Table 598 Code Generator Plugin Files (continued)

File name	Description
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point
plugin.properties	Tresos plugin support file for the FEE driver
plugin.xml	Tresos plugin support file for the FEE driver

10.1.4 Integration hints

This section lists the key points that an integrator or user of the FEE driver must consider.

10.1.4.1 Integration with AUTOSAR stack

This section lists the modules which are not part of MCAL but required to integrate the FEE driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase. While integrating, the EcuM module can initialize the FEE driver.

- **FLS**

The FEE driver depends on the FLS driver for operation. Therefore, the Infineon FLS driver is required to be configured to operate with the Infineon FEE driver.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows re-location of text, variables, constants and configuration data to user specific memory regions. In order to achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the Fee_MemMap.h.

The file Fee_MemMap.h is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is depicted below.

FEE driver

```

#if defined FEE_START_SEC_VAR_INIT_ASIL_B_GLOBAL_UNSPECIFIED
/* User pragmas here */
#undef FEE_START_SEC_VAR_INIT_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

#elif defined FEE_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_UNSPECIFIED
/* User pragmas here */
#undef FEE_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

#elif defined FEE_START_SEC_CODE_ASIL_B_GLOBAL
/* User pragmas here */
#undef FEE_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#elif defined FEE_STOP_SEC_CODE_ASIL_B_GLOBAL
/* User pragmas here */
#undef FEE_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Fee_MemMap.h, wrong pragma command"
#endif

```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the Basic Software modules. The FEE driver reports all the development errors to the DET module through the API `Det_ReportError()`. The user of the FEE driver must process all the errors reported to the DET module through the API `Det_ReportError()`.

The files `Det.h` and `Det.c` are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is a part of the AUTOSAR stack that handles all the production errors reported by the Basic Software modules. The FEE driver reports all the production errors to the DEM modules through the API `Dem_ReportErrorStatus()`. The user of the FEE driver must process all the production errors (fail / pass) reported to the DEM module through the API `Dem_ReportErrorStatus()`.

The files `Dem.h` and `Dem.c` are provided in the MCAL package as a stub code and needs to be replaced with a complete DEM module during the integration phase.

- **SchM**

SchM is not required for the integration of FEE driver.

- **Safety error**

The FEE driver will report all the detected safety errors through the API `Mcal_ReportSafetyError()`.

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The API `Mcal_ReportSafetyError()` is provided in the files `Mcal_SafetyError.c` and `Mcal_SafetyError.h` as a stub code, and must be updated by the integrator to handle the reported errors.

FEE driver

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The FEE driver implements callback functions invoked from the FLS driver. These functions are to be configured at the time of configuring the FLS driver.

The FEE driver reports the completion of jobs and errors through configurable notification functions. These functions can be configured by the user while configuring the FEE module in Tresos.

- **Operating system (OS)**

The FEE driver does not use any operating system service.

10.1.4.2 Multicore and Resource Manager

FEE driver does not support execution on multiple cores in parallel.

10.1.4.3 MCU support

FEE driver does not use any services provided by the MCU driver.

10.1.4.4 Port support

The FEE driver does not use any services provided by the Port driver.

10.1.4.5 DMA support

The FEE driver does not use any services provided by the DMA driver.

10.1.4.6 Interrupt connections

The FEE driver does not use any interrupt source.

FEE driver

10.1.4.7 Example usage

Configuration of the driver

The FEE driver could be configured in the following three modes:

FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA, FEE_DOUBLE_SECTOR_DATA_ONLY or
FEE_QUASI_STATIC_DATA_ONLY.

For configuration, the user needs to set FeeBlockTypeConfigured appropriately. FeeBlockTypeConfigured could be found under general tab in the EB tresos configuration for FEE. For example, if the user intends to use both double-sector as well as Quasi-Static (QS) data, then FeeBlockTypeConfigured should be set as FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA.

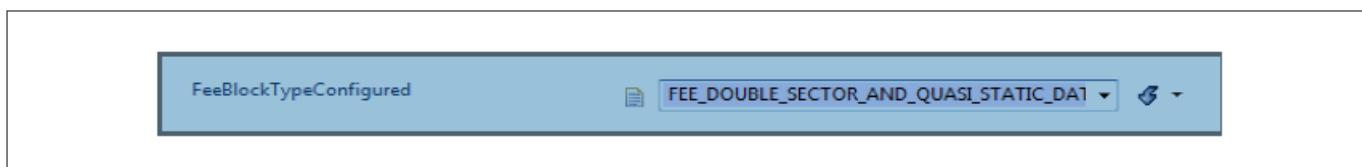


Figure 110 Configuration of FeeBlockTypeConfigured

Note:

- For only normal data (double sector), FeeBlockTypeConfigured should be set as FEE_DOUBLE_SECTOR_DATA_ONLY.
- For only QS configuration FeeBlockTypeConfigured should be set as FEE_QUASI_STATIC_DATA_ONLY.

Depending upon the configuration chosen by the user, the corresponding data blocks should be configured in FeeBlockConfiguration. For example, if FeeBlockTypeConfigured is set as

FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA, then in the FeeBlockConfiguration section both the normal double sector and QS blocks need to be configured.

Fee																																												
Name Fee																																												
General FeeBlockConfiguration FeeNvmJobEndNotification FeeNvmJobErrorNotification FeeQsJobEndNotification FeeQsJobErrorNotification FeeDemEventParameterRefs Published Information																																												
FeeBlockConfiguration																																												
<table border="1"> <thead> <tr> <th>In...</th> <th>Name</th> <th>FeeBlockNumber</th> <th>FeeQsBlockInst...</th> <th>FeeQsBlockAdd...</th> <th>FeeQuasiStaticMa...</th> <th>FeeBlockSize</th> <th>FeeIm...</th> <th>FeeNu... @ FeeDeviceIndex</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>FeeBlockConfiguration_0</td> <td>1</td> <td>0</td> <td>0x0</td> <td><input checked="" type="checkbox"/></td> <td>8192</td> <td><input checked="" type="checkbox"/></td> <td>0 @ /Fls/Fls/FlsGeneral</td> </tr> <tr> <td>1</td> <td>FeeBlockConfiguration_1</td> <td>2</td> <td></td> <td>0xc000</td> <td><input checked="" type="checkbox"/></td> <td>12288</td> <td><input checked="" type="checkbox"/></td> <td>0 @ /Fls/Fls/FlsGeneral</td> </tr> <tr> <td>2</td> <td>FeeBlockConfiguration_2</td> <td>3</td> <td>0</td> <td>0xf000</td> <td><input checked="" type="checkbox"/></td> <td>4096</td> <td><input checked="" type="checkbox"/></td> <td>0 @ /Fls/Fls/FlsGeneral</td> </tr> </tbody> </table>									In...	Name	FeeBlockNumber	FeeQsBlockInst...	FeeQsBlockAdd...	FeeQuasiStaticMa...	FeeBlockSize	FeeIm...	FeeNu... @ FeeDeviceIndex	0	FeeBlockConfiguration_0	1	0	0x0	<input checked="" type="checkbox"/>	8192	<input checked="" type="checkbox"/>	0 @ /Fls/Fls/FlsGeneral	1	FeeBlockConfiguration_1	2		0xc000	<input checked="" type="checkbox"/>	12288	<input checked="" type="checkbox"/>	0 @ /Fls/Fls/FlsGeneral	2	FeeBlockConfiguration_2	3	0	0xf000	<input checked="" type="checkbox"/>	4096	<input checked="" type="checkbox"/>	0 @ /Fls/Fls/FlsGeneral
In...	Name	FeeBlockNumber	FeeQsBlockInst...	FeeQsBlockAdd...	FeeQuasiStaticMa...	FeeBlockSize	FeeIm...	FeeNu... @ FeeDeviceIndex																																				
0	FeeBlockConfiguration_0	1	0	0x0	<input checked="" type="checkbox"/>	8192	<input checked="" type="checkbox"/>	0 @ /Fls/Fls/FlsGeneral																																				
1	FeeBlockConfiguration_1	2		0xc000	<input checked="" type="checkbox"/>	12288	<input checked="" type="checkbox"/>	0 @ /Fls/Fls/FlsGeneral																																				
2	FeeBlockConfiguration_2	3	0	0xf000	<input checked="" type="checkbox"/>	4096	<input checked="" type="checkbox"/>	0 @ /Fls/Fls/FlsGeneral																																				

Figure 111 FeeBlockConfiguration

Note that FeeQsBlockAddress is applicable only for QS blocks. In the above example, it is 0xc000 because this address is configured in FLS. This address should not overlap with the normal double-sector size addresses. Further, while configuring the QS block address (FeeQsBlockAddress) and QS block sizes (FeeBlockSize), the user must take care to fill the values for each QS data block properly in a way that they do not breach the QS sector size configured in FLS. The QS addresses should not overlap each other as well and should be contiguous.

- If FeeBlockTypeConfigured is configured as FEE_DOUBLE_SECTOR_DATA_ONLY then only the normal double-sector data blocks should be configured.
- If FeeBlockTypeConfigured is configured as FEE_QUASI_STATIC_DATA_ONLY is selected then only the QS data blocks should be configured.

FEE driver

The configuration, corresponding to that done in FEE, must be done in FLS as well. This could be done as follows:

1. Go to FLS configuration and configure the sectors for normal double sector and QS.
2. Mention the sector size for each.
3. Give the start address for both the sectors appropriately. Note that the QS sector address should be after the sectors of the normal double-sector address.

For example, in FEE, each sector of the normal double sector is of 0x60000 or 393216 bytes (384 KB) in size, giving a total of 768 KB or 786432 bytes in size for both the sectors (double sector), therefore the value of QS address is 0xC0000 (786432), which is after the addresses used for the double sector. The total size of the QS sector is therefore, 0x40000, which is the remaining size after the sector size for both the sectors of the normal double sector is allocated.

The calculation for the QS sector start address could be as follows:

- QS Sector start address = Sector size of one sector for double sector algorithm X 2
- As per the above-mentioned example:
 - Sector size of one sector for double sector algorithm = 0x60000
 - QS sector start address = $2 \times 0x60000 = 0xC0000$

The calculation for the QS sector size, in case the normal double sectors are also being used, could be summarized as follows:

- QS sector size = Total Flash size - (Sector size of one sector for double sector algorithm X 2)
- As per Figure:
 - Total Flash size = 0xfa000
 - Sector size of one sector for double-sector algorithm = 0x60000
 - QS sector = $0xfa000 - (2 \times 0x60000) = 0x3a000$

Note that it is responsibility of the user to configure the sectors judiciously in the manner as described above.

FlsSector					
FlsSector ↑ ↓					
Index	Name	FlsNumberOfSectors	FlsPageSize	FlsSectorSize	FlsSectorStartaddress
0	FlsSector_0	2	8	0x60000	0x0
1	FlsSector_1	1	8	0x3a000	0xc000

Figure 112 FLS sectors configuration

Note:

- FlsSectorStartaddress for QS should not overlap with NVM blocks.
- If FeeBlockTypeConfigured is configured as FEE_DOUBLE_SECTOR_DATA_ONLY then only one FlsSector with a value of 2 for FlsNumberOfSectors should be configured.
- If FeeBlockTypeConfigured is configured as FEE_QUASI_STATIC_DATA_ONLY then only one FlsSector with a value of 1 for FlsNumberOfSectors. So, in the above case, FlsSector_0 with a value of 1 should be configured.
- If FeeBlockTypeConfigured is configured as FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA then the order of sectors in FLS configuration should the FlsSector_0 should be configured with 2 sectors and then FlsSector_1 should be configured with a value of 1 for the number of sectors.

Initialization of FEE driver

As part of application initialization task, initialize the FLS and FEE drivers by calling the following APIs.

FEE driver

```
#include "Fee.h"
extern const Fls_17_Dmu_ConfigType Fls_17_Dmu_Config;
extern const Fee_ConfigType Fee_Config;
Fls_17_Dmu_Init(&Fls_17_Dmu_Config); /* taken from Fls_17_Dmu_PBcfg.c */
Fee_Init(&Fee_Config); /* taken from Fee_PBcfg.c */
```

This completes the initialization sequence.

FEE operation

For runtime FEE services, Fee_MainFunction is the scheduling function provided by the FEE driver. This function along with the scheduling function of FLS - Fls_17_Dmu_MainFunction () should be called periodically so that it can process the jobs. This API is a service for performing the processing of the Fee_Read (), Fee_Write (), etc. So, the main periodic task of the application should include the following.

Fls_17_Dmu_MainFunction ();

Fee_MainFunction ();

After performing any FEE operation like Fee_Read (), Fee_Write (), the following main functions should be called as follows:

```
while (Fee_GetStatus () != MEMIF_IDLE)
{
    Fls_17_Dmu_MainFunction ();
    Fee_MainFunction ();
}
```

Configuration of QS blocks

When user chooses to configure Quasi-Static data the following points should be considered:

- FeeQsBlockInstances of the FEE block should be 0 if it is declared as one of the multiple QS instances of another FEE block.
- FEE block with multiple QS instances should have same FeeBlockSize for all the block instances.
- FeeQsBlockAddress should not overlap with the previously configured FEE QS block instance.
- FeeQsBlockAddress should be contiguous for all the QS block instances configured as a part of FEE block having multiple QS instances.
- FeeBlockNumber should be contiguous for all the block instances(for the QS block having multiple instances).
- Number of QS block instances configured as a part of FEE block having multiple QS instances, should be same as FeeQsBlockInstances set for the FEE block configured.

FEE driver

Fee										
Name Fee										
General FeeBlockConfiguration FeeNvmJobEndNotification FeeNvmJobErrorNotification FeeQsJobEndNotification FeeQsJobErrorNotification FeeDemEventParameterRefs Published Information										
FeeBlockConfiguration										
Index	Name	FeeBlockNu...	FeeQsBlockInsta...	FeeQsBlockAddr...	FeeQuasiStaticMana...	FeeBlockSize	FeeI...	FeeNumb...	FeeDeviceIndex	
0	FeeBlockConfiguration_00	130	2	8192	<input checked="" type="checkbox"/>	8192	<input type="checkbox"/>	0	/Fls/Fls/FlsGeneral	
1	FeeBlockConfiguration_01	131	0	16384	<input checked="" type="checkbox"/>	8192	<input type="checkbox"/>	0	/Fls/Fls/FlsGeneral	
2	FeeBlockConfiguration_02	19	3	24576	<input checked="" type="checkbox"/>	4096	<input type="checkbox"/>	0	/Fls/Fls/FlsGeneral	
3	FeeBlockConfiguration_03	20	0	28672	<input checked="" type="checkbox"/>	4096	<input type="checkbox"/>	0	/Fls/Fls/FlsGeneral	
4	FeeBlockConfiguration_04	21	0	32768	<input checked="" type="checkbox"/>	4096	<input type="checkbox"/>	0	/Fls/Fls/FlsGeneral	
5	FeeBlockConfiguration_05	100	1	36864	<input checked="" type="checkbox"/>	8192	<input type="checkbox"/>	0	/Fls/Fls/FlsGeneral	

Figure 113 Sample QS block configuration

Key points to consider

FEE and FLS dependency

The user has to take care to link the Infineon FEE with the FLS module that is configured (FLS configuration parameter: FlsIfxFeeUse) specifically to support Infineon implementation of the FEE. This is required because the FLS module implements additional non-Autosar APIs for use by Infineon FEE and these are available only when FLS is configured to support the Infineon FEE.

Writing blocks close to GC threshold

When writing a new data block, the size of the previous data blocks present in a WL is added to the size of the new incoming block and the result is compared with the threshold. If the threshold is likely to be breached, then the new incoming block will be attempted to be written on the next consecutive word-line. If it is determined that even in this scenario the threshold will be breached then the GC is triggered. The consequence of this decision is that a few pages of the Flash memory close to the threshold is unutilized and the GC may seem to be triggered earlier than expected. The occurrence of this behaviour is dependent on the size of the blocks already present in the Flash close to the threshold and the size of the block that is requested to be written.

FEE_E_GC_TRIG DEM

During GC, if the total size of the blocks to be copied is greater than the available space in the sector (threshold is breached), then FEE_E_GC_TRIG DEM will be triggered and an illegal state notification will be raised. The user must make sure that the block sizes and threshold are configured judiciously.

Data pointer for Fee_Read and Fee_Write API

Data pointer passed in read and write API need to be memory aligned(word aligned).

Quasi Static data blocks

Quasi Static data blocks are big data blocks (multiple of 4K) that are infrequently updated over the life time of ECU. The NVM cannot be easily adapted to handle quasi static data. So, all standard NVM blocks is handled via NVRAM Manager. However, quasi static data is to be handled by quasi static manager.

Quasi Static manager is implemented by the user to manage the quasi-static data. Quasi Static data blocks are read and written using FEE's read and write APIs. There are other APIs provided by FEE, for example, Fee_17_EraseQuasiStaticData, that are meant exclusively for QS data. Please refer the API chapter for further information on APIs applicable for Quasi Static data.

FEE driver

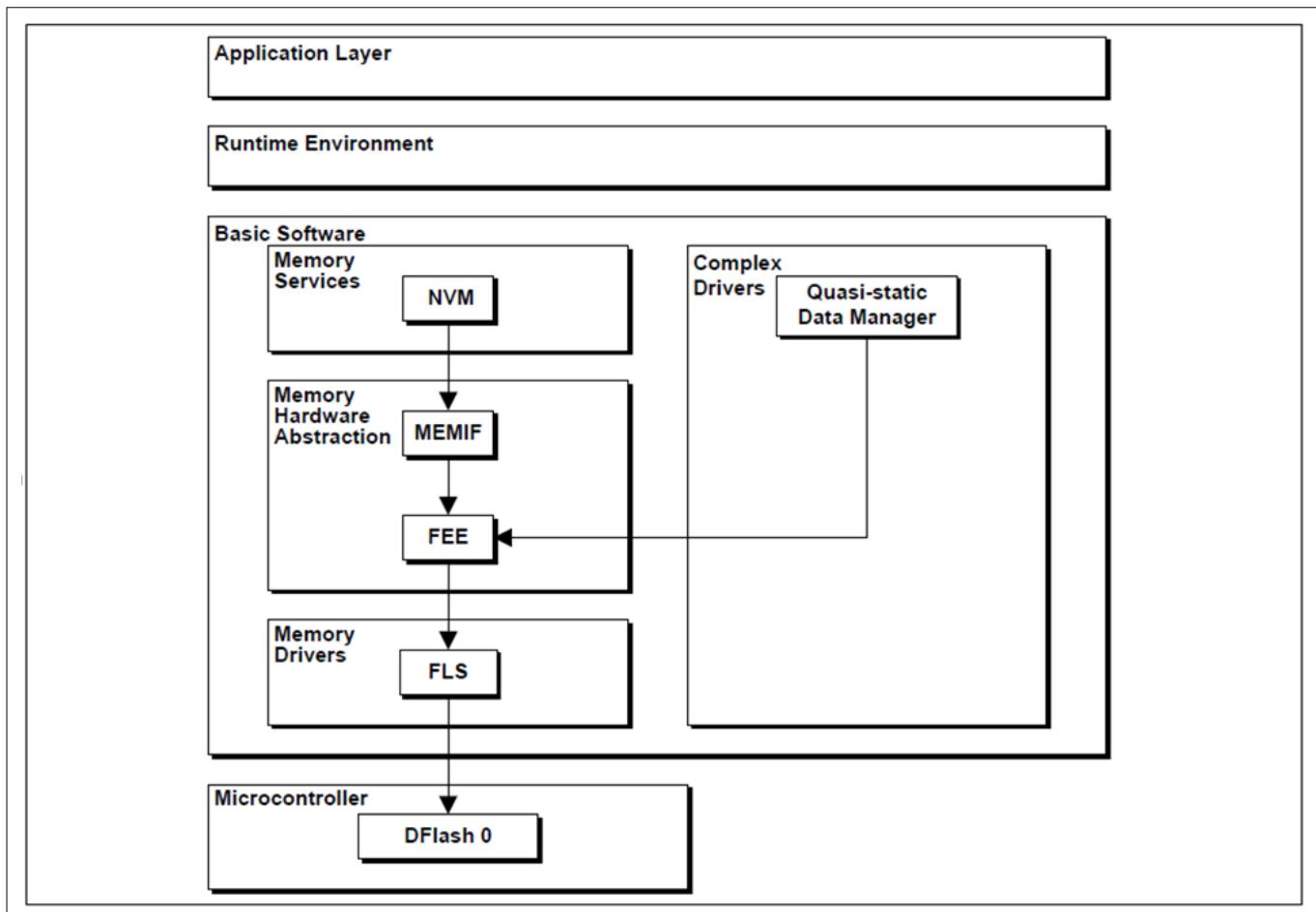


Figure 114 Handling DFlash0 by NVM and Quasi-static Manager

Quasi Static Block configuration: For Quasi Static block configuration please refers to section configuration interface.

DFlash Configuration: User can configure DFlash in one of three different configurations using configuration parameter “FeeBlockTypeConfigured”.

FeeBlockTypeConfigured = FEE_DOUBLE_SECTOR_DATA_ONLY

Only NVM data is present

FEE driver

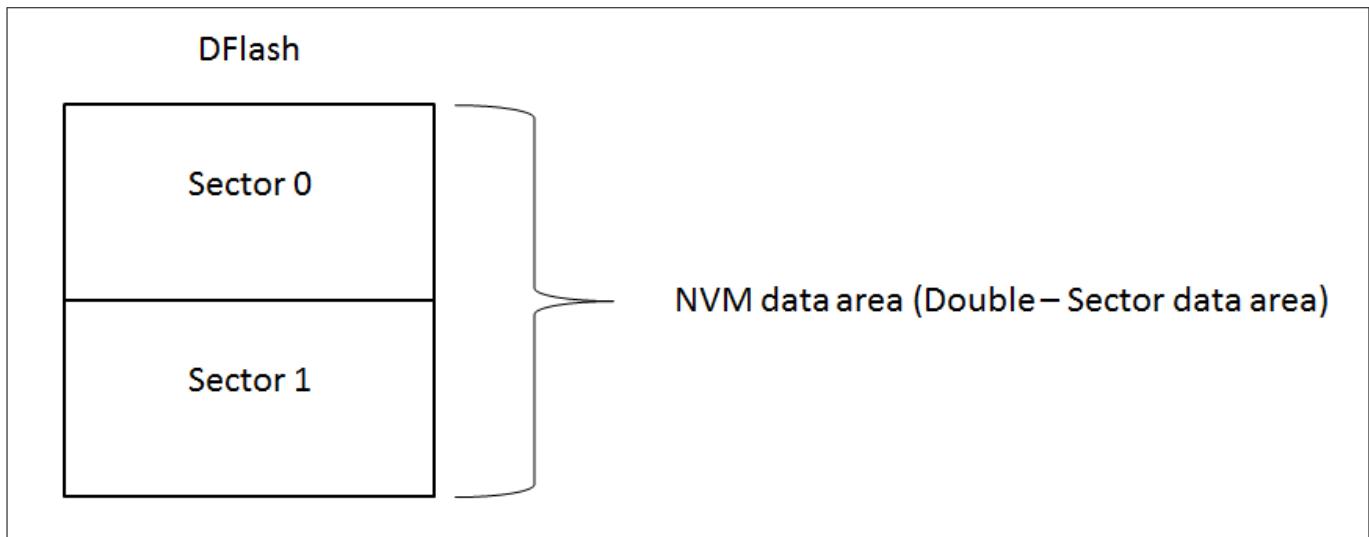


Figure 115 Only NVM data

FeeBlockTypeConfigured = FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA

NVM and Quasi static data is present

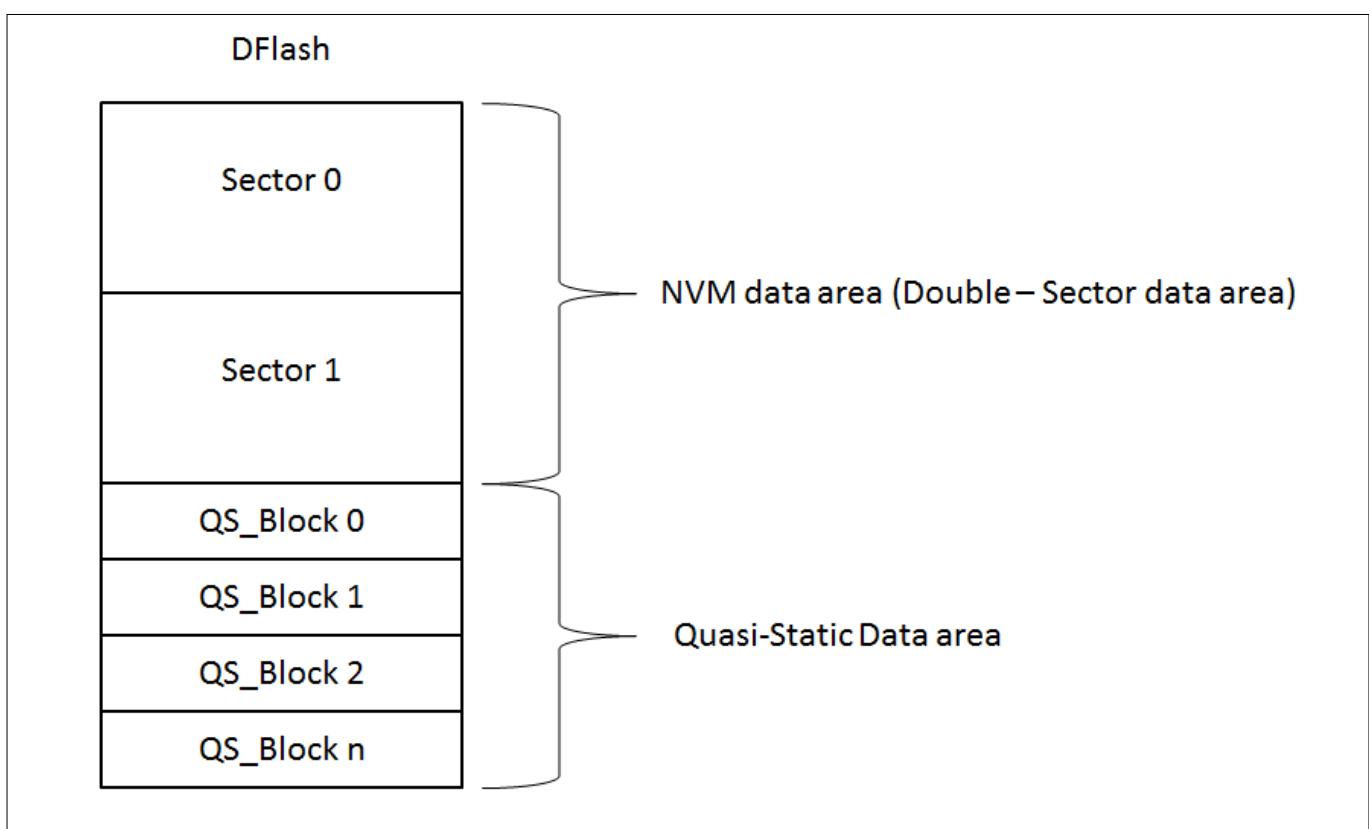


Figure 116 NVM and Quasi static data

FeeBlockTypeConfigured = FEE_QUASI_STATIC_DATA_ONLY

Only Quasi static data is present

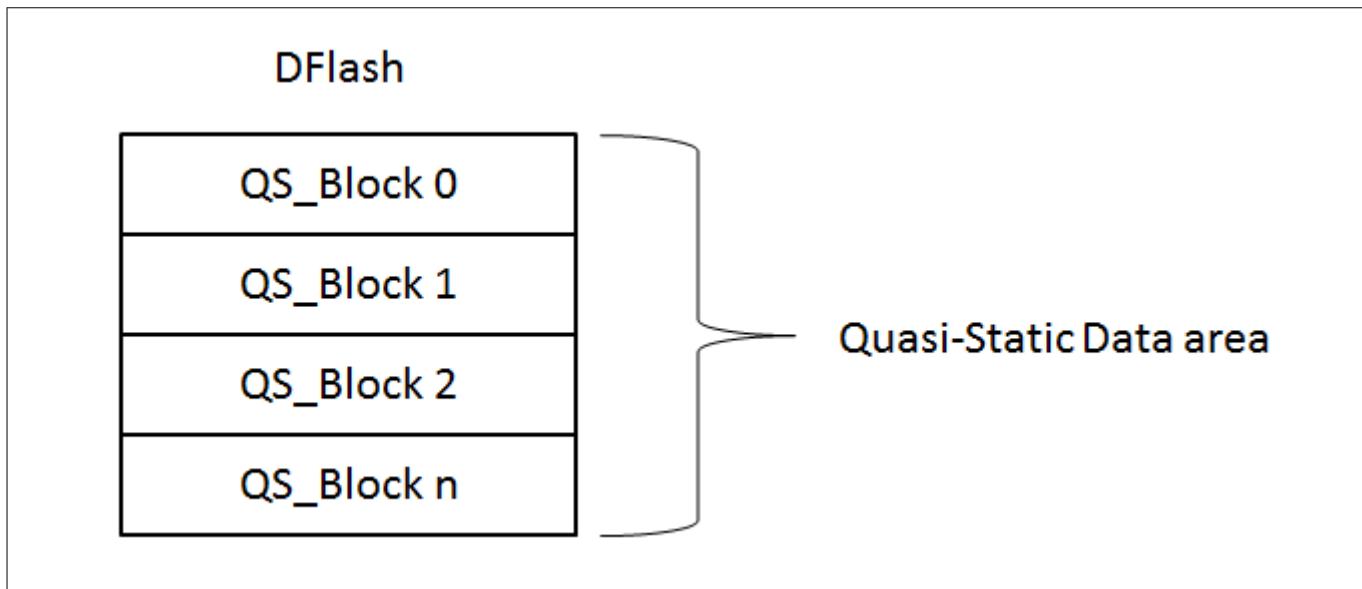


Figure 117 Quasi static data

10.1.5 Key architectural considerations

The key architectural considerations are as follows:

- The FEE driver is platform independent and does not implement any interrupt service routines.
- For the double-sector algorithm, the area of the DFlash0 is divided into two sectors.
- At a given time, the double-sector algorithm will fill one of the sectors with data blocks and data will be updated by simply appending/writing the new data block to the free space, available at the end of the previous written block. When the sector being used becomes filled to a certain threshold, the process of garbage collection is triggered. The garbage collection process copies the most recent copies of the data blocks to the other sector and then erases the previously filled sector. This will be repeated when the second sector gets filled to a threshold.
- As data is frequently updated, word-line/bit-line shorts may occur in the double-sector DFlash range during erase and write.
- After any erase operation in the double-sector algorithm, word-line failures will be detected and handled. The design is scalable to handle a maximum of two word-line failures.
- Quasi-Static data handling: The standard EEPROM emulation algorithm is not efficient for relatively big data blocks with a low update rate (500 erase/write cycles over the entire QS data area). Therefore, in addition to the standard EEPROM emulation algorithm, a special algorithm for infrequently updated data blocks (Quasi-Static data) in a special DFlash area has been provided.

10.1.5.1 Double-sector algorithm is used

The algorithm used for EEPROM emulation with DFlash is Double-Sector algorithm.

10.1.5.2 Quasi-static data algorithm is used

In addition to the standard double-sector EEPROM emulation algorithm, a special algorithm for infrequently updated data blocks (called Quasi-Static data) is made available.

FEE driver

10.1.5.3 Post build variant

In FEE module the configuration parameters namely FeeQsBlockInstances, FeeQsBlockAddress, FeeBlockSize, FeeImmediateData and FeeNumberOfWriteCycles are implemented as post build parameters.

10.1.5.4 Callback notification when erase verify error occurs

The Fee_17_JobEraseErrorNotification function shall perform error handling operations and subsequently call the user configured QS job error notification routine of the upper layer module if configured.

10.1.5.5 Notifications to upper layer

Job end, job error, illegal state notifications are given through different interfaces for NVM and QS separately.

10.1.5.6 Handling ECC errors during GC

Configured and un-configured blocks containing ECC errors will be dropped during garbage collection. No notification is provided when such instances occur.

Note: When an ECC error occurs, not much can be done to retrieve the data block. To generate an illegal state notification might be too harsh as ECC errors might go away after the next erase cycle. Providing a notification without informing the user about the block ID that was dropped does not seem helpful. FEEÃ¢â€žâ¢s notification interfaces do not allow passing of parameters. When the user makes a request to read a configured block that was dropped during GC due to an ECC error, the job will end with a job result MEMIF_BLOCK_INCONSISTENT.

10.1.5.7 Ongoing erase cannot be canceled

Ongoing erase, be it in NVM (during GC) or QS cannot be canceled.

10.1.5.8 DET check for Fee_JobErrorNotification and Fee_JobEndNotification

For the Fee_JobErrorNotification and Fee_JobEndNotification APIs, the FEE_E_UNINIT DET is detected and reported when these APIs are called before initializing the FEE module.

10.1.5.9 Init check

To help protect against corruption of FEE global data structures, in addition to checking if the provided configuration pointer is not NULL, a CRC based protection mechanism is implemented to protect the global data structure. The CRC is determined at the end of Fee_Init API and again as part of Fee_17_InitCheck API. The two values are compared to determine if the global data was corrupted after Fee_Init was called. For this to work correctly, it is assumed that the Fee_17_InitCheck will be called after Fee_Init API and before any other Fee API or the Fee_Mainfunction API is called. Fee_17_InitCheck API should be called before Fee_MainFunction API because it modifies the global data structure when it runs asynchronous operations needed for preparing the FEE to service user requests.

10.1.5.10 Handling of invalid sectors

The FeeEraseAllEnable configurable parameter is provided, for the user, to allow erasing sectors when the sectors are found to be invalid.

FEE driver

10.1.5.11 Behavior of Fee_17_EraseQuasiStatic data for QS blocks with multiple instances

The behavior of the Fee_17_EraseQuasiStaticData API is illustrated as follows.

The screenshot shows a software interface titled 'Fee' with a sub-tab 'Fee'. The main window displays a table of 'FeeBlockConfiguration' entries. The columns are labeled: Index, Name, FeeBlockNumb..., FeeQsBlockInsta..., FeeQsBlockAddr..., FeeQuasiStaticManag..., FeeBlockSize, FeeIndex, FeeNumb..., and FeeDeviceIndex. The data in the table is as follows:

Index	Name	FeeBlockNumb...	FeeQsBlockInsta...	FeeQsBlockAddr...	FeeQuasiStaticManag...	FeeBlockSize	FeeIndex	FeeNumb...	FeeDeviceIndex
0	FeeBlockConfiguration_00	130	2	8192	☒	8192	☒	0	/Fls/Fls/FlsGeneral
1	FeeBlockConfiguration_01	131	0	16384	☒	8192	☒	0	/Fls/Fls/FlsGeneral
2	FeeBlockConfiguration_02	19	3	24576	☒	4096	☒	0	/Fls/Fls/FlsGeneral
3	FeeBlockConfiguration_03	20	0	28672	☒	4096	☒	0	/Fls/Fls/FlsGeneral
4	FeeBlockConfiguration_04	21	0	32768	☒	4096	☒	0	/Fls/Fls/FlsGeneral
5	FeeBlockConfiguration_05	100	1	36864	☒	8192	☒	0	/Fls/Fls/FlsGeneral

Figure 118 Sample QS block configuration

For the above-mentioned configurations, the Fee_17_EraseQuasiStaticData API would behave as follows:

Fee_17_EraseQuasiStaticData(19, 3): Blocks 19, 20, 21 will be erased

Fee_17_EraseQuasiStaticData(19, 1): Block 19 will be erased

Fee_17_EraseQuasiStaticData(19, 2): Blocks 19, 20 will be erased

Fee_17_EraseQuasiStaticData(21, 1): Block 21 will be erased

Fee_17_EraseQuasiStaticData(20, 2): Request rejected with INVALID_BLOCK_INSTANCES error will be raised

Fee_17_EraseQuasiStaticData(21, 0): Request rejected with INVALID_BLOCK_INSTANCES error will be raised

10.1.5.12 Behavior of illegal state notifications

If FeeVirginFlashIllegalState is set to TRUE, FEE invokes the illegal state notification upon detection of virgin Flash. Three cases arise: either the double sector area is in the virgin state or the QS area is in the virgin state or both areas are in the virgin state. If the area of the Flash used by the double-sector algorithm is found to be in the virgin state, the illegal state notification function configured in FeeNvmIllegalStateNotification is invoked and the illegal state notification function configured in FeeQsIllegalStateNotification will not be invoked irrespective of whether QS is in virgin or not. If the area used by the double-sector algorithm is not in the virgin state but the area used for Quasi-Static data is in the virgin state, then only the notification function configured in FeeQsIllegalStateNotification will be invoked. The user application should process the notification functions considering this behavior.

10.1.5.13 User mode support

FEE driver is a platform (device/hardware) independent module. It does not access any SFR directly.

[cover parentID FEE={1A65EADD-AFD0-4845-B2D2-8257E086DD67}]

FEE driver

10.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Association of QS job end notifications**

If the erase-suspend feature is enabled (FeeUseEraseSuspend=TRUE) and if a read or write request is made to a QS block when there is an ongoing QS erase operation, then it is assumed that the user shall associate the first job end notification with the completed QS read or write request and the second job end notification shall be associated with the completed erase operation.

[cover parentID FEE={A98061B0-8442-437f-938B-ADA60082DD87}]

- **Calling init-check API**

It is assumed that the Fee_17_InitCheck() API shall be called after calling Fee_Init and before any other FEE API or the Fee_Mainfunction is called.

[cover parentID FEE={38B90F4D-4D06-4acc-9076-F19EE0CE40F1}]

- **Cancelling write and invalidate requests**

It is assumed that the ongoing user-initiated write and invalidate block requests are not canceled using the Fee_Cancel() API. It is advisable that the module status be ascertained by making a call to the Fee_GetStatus() API and a new request be made only after the module status reaches MEMIF_IDLE.

[cover parentID FEE={6FC8C9A5-45A4-4d50-B7F7-07111FAC7677}]

- **Erase all when stateless**

When FeeEraseAllEnable = TRUE, and during initialization, it is found the Flash does not contain state pages, the area of the Flash used for the normal data (NVM) will be erased without any indication to the user. It is assumed that the user is aware of this behavior.

[cover parentID FEE={351399FF-6B79-4ba2-922C-B8B87E9D76BA}]

- **Erasing Immediate blocks**

Due to the way the double sector algorithm works, an immediate block cannot be erased. Therefore, AUTOSAR-defined Fee_EraseImmediateBlock API shall be implemented as a dummy function that always returns E_NOT_OK. It is assumed that the user will handle the implications of this behavior.

[cover parentID FEE={D0E18D9D-6DC2-4420-AFE2-F71830D81D22}]

- **FEE and pre-emption**

It is assumed that a call to an FEE service shall not be made so as to pre-empt another ongoing FEE service. This means that the FEE calls cannot be made from separate executing tasks in a manner that one call can pre-empt another. A call to a FEE service made from an interrupt context is also prohibited if an ongoing FEE service is interrupted. The FEE services are non-reentrant and when an FEE service is pre-empted by another FEE service, internal data structures and state machines of the FEE become inconsistent.

[cover parentID FEE={8D3A9F16-281E-4073-B93E-0A0DB5BB5AA0}]

- **FEE single core execution**

It is assumed that the FEE software is called in the run-time application phase as well as in the boot phase from one core only.

[cover parentID FEE={BA2D7D53-75A9-4919-87BA-7F28A21A1493}]

- **Handling illegal state notification**

When an illegal state notification is generated by the FEE it means that the FEE module can no longer be used until the system is reset. Therefore, it is assumed that the user shall not make any further requests to the FEE after an illegal state notification has been received.

[cover parentID FEE={8BF148A6-BD68-4566-BFAB-B642C1355690}]

- **High priority QS write requests**

FEE driver

Before requesting a high priority write to a QS block, the Fee_17_CancelAll() API should be called to cancel all ongoing operations. If the GC is ongoing and the erase-suspend feature (FeeUseEraseSuspend) is configured as FALSE, then the high-priority write request will be serviced after the GC is completed.

[cover parentID FEE={21C52084-8CDF-4266-AE87-F3687BD0778F}]

- **QS block configuration**

It is assumed that the user shall adhere to the following rules while configuring the QS data blocks:

- Block addresses shall not overlap with the address of any other block.
- Block size for the all instances of the block with multiple instances shall be same.
- Block number shall be contiguous for the blocks having multiple instances.
- Block instance number shall be 1 for blocks that do not have multiple instances.
- Block instance number shall be zero if it is an instance of a block with multiple instances.

[cover parentID FEE={6AB71A5D-A820-47ec-A9A6-3D2A67031E11}]

- **QS writes on pre-erased blocks**

When a QS block write is requested, the FLS performs a program verify error check by verifying the state of the PVER bit. If the PVER reports an error, the FEE causes a job error notification to be generated. One situation where this could occur is when the user application attempts to write to a QS block that was not erased. The FEE does not manage the erase of QS blocks. The user application is assumed to perform this. It is assumed that the write to a QS block will be requested on a pre-erased QS block. The state of the QS block may be determined by calling the Fee_17_GetQuasiStaticBlockInfo() API. If it is determined that the block is invalid, the user may choose to erase the block.

[cover parentID FEE={26D6A684-6C95-4087-A7BF-97B04B0B8830}]

- **Re-enabling GC**

In order to re-enable the GC, it is assumed that the user shall first call Fee_Cancel to cancel any pending write job and then call Fee_17_EnableGCStart.

[cover parentID FEE={E1DE636E-5623-4874-87CB-52B8274BF4A0}]

- **Re-trying a QS request**

If a read (Fee_Read) or write (Fee_Write) request made to a quasi-static data block is returned with E_NOT_OK, it is assumed that the user application shall retry the request in a subsequent raster (after one cycles of the FEE main function). If the erase-suspend feature is enabled and when the erase is ongoing in the hardware and when the FEE requests to suspend the erase to perform a QS write, a corner case arises when the erase operation has just finished in the hardware, but the FLS has not yet processed the event. In this case, the attempt to suspend the erase will fail, causing the QS block write operation to be rejected with E_NOT_OK. This situation is momentary and it is assumed that the user application will re-attempt the QS write request in a subsequent raster.

[cover parentID FEE={90C082AB-1FE4-420a-80CF-A9A6E3473515}]

- **Requesting cancel from interrupt context**

It is assumed that Fee_Cancel and Fee_17_CancelAll are not called from an interrupt context. As FEE services are asynchronous and non-re-entrant, calling Fee_Cancel or Fee_17_CancelAll from an interrupt context causes problems if the main task is already executing an FEE service. Fee_Cancel or Fee_17_CancelAll should also not be called in a manner that can cause it to pre-empt another FEE service.

[cover parentID FEE={D92B66E8-4B6F-4167-94D0-957C43D11A34}]

- **Retrying canceled jobs**

It is assumed that canceled jobs will be re-submitted by the application at some later point in time, after the QS block write is completed.

[cover parentID FEE={3F79DA18-B6E0-43e1-942C-9753B2CBAC97}]

- **Sensing mode configuration**

FEE driver

It is assumed that the Flash will be operated in the single ended sensing mode.

[cover parentID FEE={32FE1DAF-A76D-4146-80EE-0E5EED506DB0}]

- **User data protection**

User data is not protected by a CRC. It is assumed that the CRC forms a part of user data block and the user application will take the responsibility for checking the correctness of its data.

[cover parentID FEE={207D5CD1-185A-424e-A9B9-6A7EF6F7214D}]

- **Virgin state of Flash**

When in the virgin state, it is assumed that the user shall ensure that the entire area of DFlash0 allocated for the Double-Sector algorithm and Quasi-Static data is erased and contains no data.

[cover parentID FEE={2DE71581-F37F-4fd7-8271-06562F2A6647}]

FEE driver

10.3 Reference information

10.3.1 Configuration interfaces

FEE driver

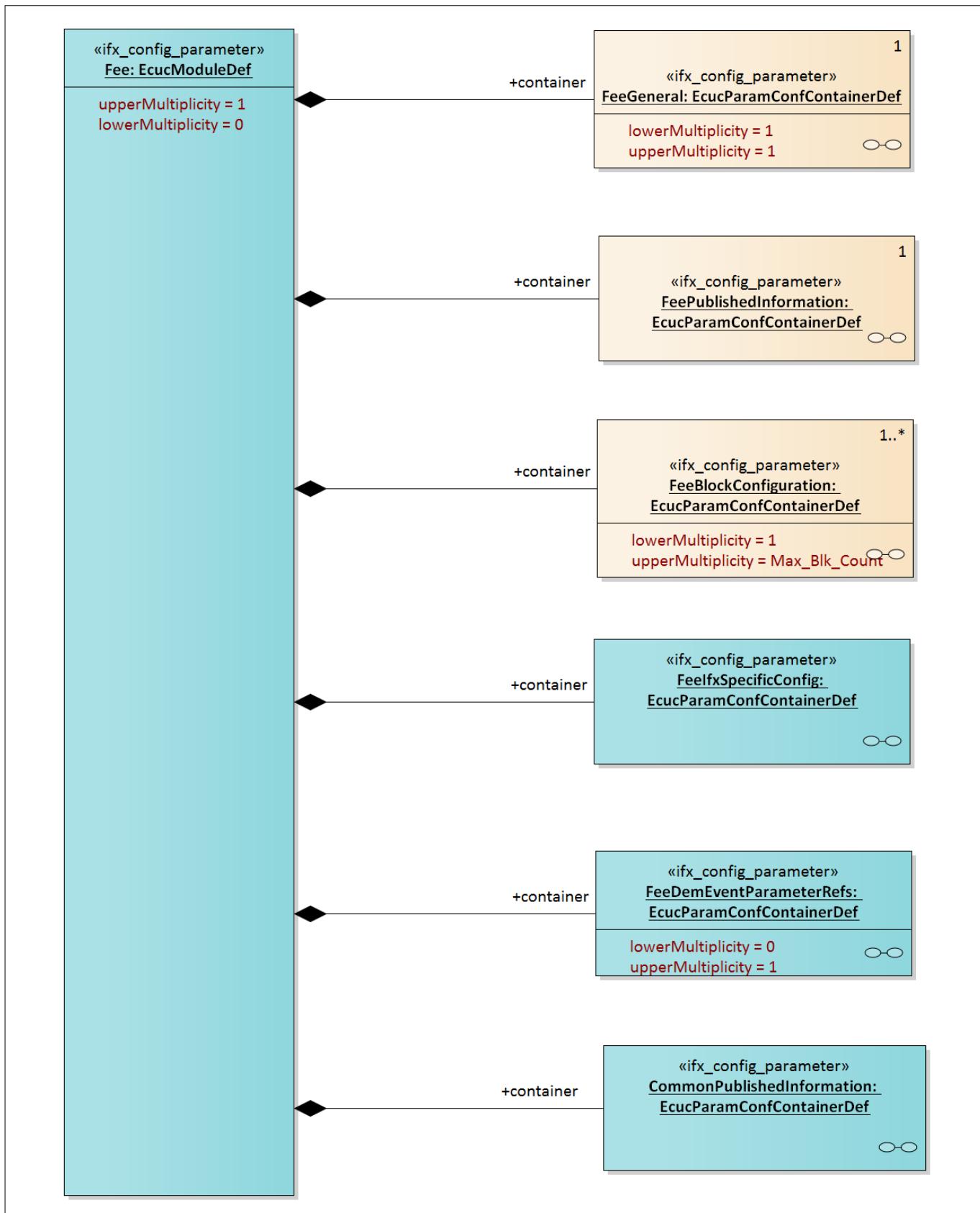


Figure 119 Container hierarchy along with their configuration parameters

10.3.1.1 Container: CommonPublishedInformation

FEE driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

10.3.1.1.1 ArMajorVersion**Table 599 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Major version number of AUTOSAR specification on which the driver implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.1.2 ArMinorVersion**Table 600 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	Minor version number of AUTOSAR specification on which the driver implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

FEE driver**10.3.1.1.3 ArPatchVersion****Table 601 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	Patch version number of AUTOSAR specification on which the driver implementation is based on.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.1.4 ModuleId**Table 602 Specification for ModuleId**

Name	ModuleId		
Description	Provides the FEE driver module ID as described by AUTOSAR : Wp1.1.2 Basic Software Module List		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	21		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.1.5 Release**Table 603 Specification for Release**

Name	Release		
Description	Specifies the derivative for which the configuration project is created.		
Multiplicity	1..1	Type	EcucStringParamDef

FEE driver**Table 603 Specification for Release (continued)**

Range	String		
Default value	As per the hardware derivative		
Post-build variant value		Post-build variant multiplicity	-
Value configuration class		Multiplicity configuration class	-
Origin	IFX	Scope	
Dependency	-		

10.3.1.1.6 SwMajorVersion**Table 604 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Major version number of the vendor specific implementation of the driver. The numbering is vendor specific.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.1.7 SwMinorVersion**Table 605 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Minor version number of the vendor specific implementation of the driver. The numbering is vendor specific. MINOR_VERSION is incremented if the driver is still downwards compatible (For example, new functionality added)		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		

FEE driver**Table 605 Specification for SwMinorVersion (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.1.8 SwPatchVersion**Table 606 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Patch level version number of the vendor specific implementation of the driver. The numbering is vendor specific. The PATCH_VERSION is incremented if the driver is still upwards and downwards compatible (for example, bug fixed)		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.1.9 VendorId**Table 607 Specification for VendorId**

Name	VendorId		
Description	Vendor Id of the supplier		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-

FEE driver
Table 607 Specification for VendorId (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.2 Container: Fee

This container holds the configurations of the FEE (Flash EEPROM Emulation) module.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

10.3.1.3 Container: FeeBlockConfiguration

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

10.3.1.3.1 FeeBlockNumber

Table 608 Specification for FeeBlockNumber

Name	FeeBlockNumber		
Description	<p>Specifies the block number of the logical block.</p> <p>The value of this parameter should be unique across configurations.</p> <p>Note: To find the physical address of the block, the FEE algorithm uses the cache table, that is, the physical address of the block is not calculated based on the block number using any static relation as suggested by AUTOSAR. Also for QS data the address is not directly calculated from the block number. Therefore, the block number can be any unique arbitrary number different from 0x0001 and 0xFFFF.</p> <p>Since the minimum possible value for the block number is 1, therefore the default value is set as 1.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 65534		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

FEE driver**10.3.1.3.2 FeeBlockSize****Table 609 Specification for FeeBlockSize**

Name	FeeBlockSize		
Description	<p>Specifies the size of the logical block.</p> <p>Note:</p> <p>[a] For each of the NVM logical block, the FEE algorithm appends 8 bytes of header information at the beginning of the block and 1 page (8 bytes) of marker information at the end of the block. In between the header and the marker, each page (8 bytes) of the DFlash0 contains 1 byte of FEE internal information and 7 bytes of data of the logical block.</p> <p>Therefore, the total size (in bytes) occupied by the logical block in the DFlash0 is:</p> <p>If ((FeeBlockSize % 7) == 0), total size= ((FeeBlockSize / 7) * 8) + 16</p> <p>If ((FeeBlockSize % 7) != 0), total size= (((FeeBlockSize / 7) + 1) * 8) + 16</p> <p>Arithmetic of type integer is used in the calculation.</p> <p>[b] For QS blocks, the size should be an integer multiple of 4Kb and this also includes the block overhead.</p> <p>Since the minimum possible value for the block size is 1 therefore, the default value is set as 1.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 65535		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

10.3.1.3.3 FeeDeviceIndex**Table 610 Specification for FeeDeviceIndex**

Name	FeeDeviceIndex		
Description	<p>Specifies the device index (handle). This information is needed by the NVRAM manager to direct a request to the memory abstraction layer (MemIf) to address a certain logical block.</p> <p>Since the name of the dependent container is user configurable, the default value is kept as NULL.</p>		
Multiplicity	1..1	Type	EcuReferenceDef
Range	Reference to Node: FlsGeneral		
Default value	NULL		

FEE driver**Table 610 Specification for FeeDeviceIndex (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

10.3.1.3.4 FeeImmediateData**Table 611 Specification for FeeImmediateData**

Name	FeeImmediateData		
Description	<p>Specifies the type (priority) of the logical block.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or Double-Sector and Quasi-Static.</p> <p>The parameter is set to FALSE for normal data and TRUE for immediate data.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.3.5 FeeNumberOfWriteCycles**Table 612 Specification for FeeNumberOfWriteCycles**

Name	FeeNumberOfWriteCycles
Description	<p>Defines the block write cycle count of a particular logical block. It denotes the maximum number of times a particular logical block can be written.</p> <p>However, the value 0 denotes that this block is not bound by any limit and can be written as many times as desired by the user.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or Double-Sector and Quasi-Static.</p>

FEE driver**Table 612 Specification for FeeNumberOfWriteCycles (continued)**

	The minimum value is chosen as the default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.3.6 FeeQsBlockAddress**Table 613 Specification for FeeQsBlockAddress**

Name	FeeQsBlockAddress		
Description	<p>Specifies the address of The QS block in the Flash. In case of NVM blocks, this parameter is not used. The address should be 4K aligned.</p> <p>It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both.</p> <p>minimum value is chosen as the default value.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - Fls Range		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.3.7 FeeQsBlockInstances**Table 614 Specification for FeeQsBlockInstances**

Name	FeeQsBlockInstances		
Description	Specifies the number of block instances in case of parent QS block. Multiple QS blocks can exist together and each of the QS block can have more than one instance. This parameter is		

FEE driver**Table 614 Specification for FeeQsBlockInstances (continued)**

	<p>used for configuring the number of instances within one QS block. For example, if there are 4 instances of QS block with id = 0x400, 0x401, 0x402 and 0x403 then the number of instances for block id = 0x400 is 4. Also, for the rest of the blocks with id = 0x401, 0x402 and 0x403, the number of instances = 0. The block numbers for the instances should be configured sequentially as illustrated.</p> <p>It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both.</p> <p>The minimum value is chosen as the default value.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - (Fls Total Size) / 4K		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.3.8 FeeQuasiStaticManager**Table 615 Specification for FeeQuasiStaticManager**

Name	FeeQuasiStaticManager		
Description	<p>Specifies the user of the configured block.</p> <p>It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both.</p> <p>The default block configuration is double sector data , therefore FeeQuasiStaticManager parameter is disabled by default.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

FEE driver**10.3.1.4 Container: FeeDemEventParameterRefs**

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

10.3.1.4.1 FEE_E_GC_ERASE**Table 616 Specification for FEE_E_GC_ERASE**

Name	FEE_E_GC_ERASE		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that is passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: error while executing erase operation during GC.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.4.2 FEE_E_GC_INIT**Table 617 Specification for FEE_E_GC_INIT**

Name	FEE_E_GC_INIT		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that is passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: error during the Init GC activity.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		

FEE driver**Table 617 Specification for FEE_E_GC_INIT (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.4.3 FEE_E_GC_READ**Table 618 Specification for FEE_E_GC_READ**

Name	FEE_E_GC_READ		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that is passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: error while executing the read operation during GC.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.4.4 FEE_E_GC_TRIG**Table 619 Specification for FEE_E_GC_TRIG**

Name	FEE_E_GC_TRIG		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that is passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: error due to insufficient space in the new sector for copying data blocks or state block.</p>		

FEE driver**Table 619 Specification for FEE_E_GC_TRIGGER (continued)**

	It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.4.5 FEE_E_GC_WRITE**Table 620 Specification for FEE_E_GC_WRITE**

Name	FEE_E_GC_WRITE		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that is passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: error while executing the write operation during GC.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

FEE driver**10.3.1.4.6 FEE_E_INVALIDATE****Table 621 Specification for FEE_E_INVALIDATE**

Name	FEE_E_INVALIDATE		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that is passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: error while executing the user-triggered invalidate request.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.4.7 FEE_E_READ**Table 622 Specification for FEE_E_READ**

Name	FEE_E_READ		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that is passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: uncorrectable ECC error while executing the user read request.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL

FEE driver**Table 622 Specification for FEE_E_READ (continued)**

Dependency	FeeBlockTypeConfigured
-------------------	------------------------

10.3.1.4.8 FEE_E_UNCONFIG_BLK_EXCEEDED**Table 623 Specification for FEE_E_UNCONFIG_BLK_EXCEEDED**

Name	FEE_E_UNCONFIG_BLK_EXCEEDED		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that is passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: error due to exceeding the unconfigured number of blocks. Refer to the FeeMaxBlockCount configuration parameter.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.4.9 FEE_E_WRITE**Table 624 Specification for FEE_E_WRITE**

Name	FEE_E_WRITE		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that is passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: error while executing the user write request.</p> <p>It is applicable when FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		

FEE driver**Table 624 Specification for FEE_E_WRITE (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.4.10 FEE_E_WRITE_CYCLES_EXHAUSTED**Table 625 Specification for FEE_E_WRITE_CYCLES_EXHAUSTED**

Name	FEE_E_WRITE_CYCLES_EXHAUSTED		
Description	<p>Existence of this parameter decides if the FEE driver would raise this particular DEM. The short name of the referenced DEM parameter is used as the symbol that passed to the Dem_ReportEventStatus() function while reporting the error.</p> <p>Error description: error due to exceeding the configured limit of the write cycles for the given block.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.5 Container: FeeGeneral

Container for general parameters. These parameters are not specific to a block.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

FEE driver**10.3.1.5.1 FeeBlockTypeConfigured****Table 626 Specification for FeeBlockTypeConfigured**

Name	FeeBlockTypeConfigured		
Description	<p>Pre-processor switch to indicate the type of block configuration out of 3 allowed classifications, that is, whether only NVM data, or NVM and Quasi-Static data or only Quasi-Static data.</p> <p>Default value is set to Double sector data only as it is the most common configuration used.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA : FEE_DOUBLE_SECTOR_DATA_ONLY : FEE_QUASI_STATIC_DATA_ONLY :		
Default value	FEE_DOUBLE_SECTOR_DATA_ONLY		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.5.2 FeeDevErrorDetect**Table 627 Specification for FeeDevErrorDetect**

Name	FeeDevErrorDetect		
Description	<p>Switches the DET detection and notification ON or OFF.</p> <p>TRUE: enabled (ON).</p> <p>FALSE: disabled (OFF).</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

FEE driver**Table 627 Specification for FeeDevErrorDetect (continued)**

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.5.3 FeeInitCheckApi**Table 628 Specification for FeeInitCheckApi**

Name	FeeInitCheckApi		
Description	<p>Enables the user to avail the functionality to check if FEE initialization is correct.</p> <p>True: Fee_17_InitCheck() API is enabled for use.</p> <p>False: Fee_17_InitCheck() API is disabled for use.</p> <p>The default value is set to FALSE for the optional feature to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.5.4 FeeMainFunctionPeriod**Table 629 Specification for FeeMainFunctionPeriod**

Name	FeeMainFunctionPeriod		
Description	<p>Period between successive calls to the main function (in seconds).</p> <p>10ms is the widely used function period therefore, it is kept as the default value.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.001 - 1s		
Default value	10ms		
Post-build variant value	FALSE	Post-build variant multiplicity	-

FEE driver**Table 629 Specification for FeeMainFunctionPeriod (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

10.3.1.5.5 FeeNvmJobEndNotification**Table 630 Specification for FeeNvmJobEndNotification**

Name	FeeNvmJobEndNotification		
Description	<p>Mapped to the job end notification routine provided by the upper layer module (NvM_JobEndNotification).</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p> <p>By default, the optional notification is disabled to minimize the executable code size.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.5.6 FeeNvmJobErrorNotification**Table 631 Specification for FeeNvmJobErrorNotification**

Name	FeeNvmJobErrorNotification		
Description	<p>Mapped to the job error notification routine provided by the upper layer module (NvM_JobErrorNotification).</p> <p>It is applicable when FEE module is configured as Double-Sector only or Double-Sector and Quasi-Static.</p> <p>The optional notification is disabled by default to minimize the executable code size.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		

FEE driver**Table 631 Specification for FeeNvmJobErrorNotification (continued)**

Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.5.7 FeePollingMode**Table 632 Specification for FeePollingMode**

Name	FeePollingMode		
Description	Pre-processor switch to enable or disable the polling mode for this module. This parameter is set to FALSE and is non-editable. TRUE: Polling mode enabled, callback functions (provided to the FLS module) are disabled. FALSE: Polling mode disabled, callback functions (provided to the FLS module) are enabled. Infineon implementation of FEE works only in the non-polling mode. Therefore the default value is set to FALSE and is non-editable.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

10.3.1.5.8 FeeQsJobEndNotification**Table 633 Specification for FeeQsJobEndNotification**

Name	FeeQsJobEndNotification
Description	Mapped to the job end notification routine for the completed QS jobs provided by the upper layer module that uses the FEE services for QS blocks. The configured function is invoked by the FEE when a QS job completes successfully.

FEE driver**Table 633 Specification for FeeQsJobEndNotification (continued)**

	<p>The user is expected to provide a function of type void FeeQsJobEndNotification (void) if a notification for a completed QS job is required. If no notification is required, this parameter should be configured as NULL_PTR. The notification function provided is expected to be synchronous.</p> <p>This parameter is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both.</p> <p>By default the optional notification is disabled by default to minimize the executable code size.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.5.9 FeeQsJobErrorNotification**Table 634 Specification for FeeQsJobErrorNotification**

Name	FeeQsJobErrorNotification		
Description	<p>Mapped to the job error notification routine for the failed QS jobs provided by the upper layer module that uses the FEE services for QS blocks. The configured function is invoked by the FEE when a QS job fails to complete successfully.</p> <p>The user is expected to provide a function of type void FeeQsJobErrorNotification (void) if a notification for a failed QS job is required. If no notification is required, this parameter should be configured as NULL_PTR.</p> <p>The notification function provided is expected to be synchronous.</p> <p>It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both.</p> <p>By default the optional notification is disabled to minimize the executable code size.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE

FEE driver**Table 634 Specification for FeeQsJobErrorNotification (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.5.10 FeeSafetyEnable**Table 635 Specification for FeeSafetyEnable**

Name	FeeSafetyEnable		
Description	<p>Enables the user to avail the functionality to detect and report the safety errors.</p> <p>By default the detection of safety related errors is enabled to ensure that safety issues are addressed during the product lifecycle.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.5.11 FeeSetModeSupported**Table 636 Specification for FeeSetModeSupported**

Name	FeeSetModeSupported
Description	<p>Compiler switch to enable or disable the SetMode functionality of the FEE module.</p> <p>TRUE: setMode functionality supported</p> <p>FALSE: setMode functionality not supported</p> <p>Note:</p> <p>This configuration setting should be consistent with that of all underlying Flash device drivers (configuration parameter FlsSetModeApi).</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>

FEE driver**Table 636 Specification for FeeSetModeSupported (continued)**

	The default value is set to disable for the optional feature to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

10.3.1.5.12 FeeVersionInfoApi**Table 637 Specification for FeeVersionInfoApi**

Name	FeeVersionInfoApi		
Description	Pre-processor switch to enable or disable the API to read the version information of the module. TRUE: Version information API enabled. FALSE: Version information API disabled. The default value is set to disable for the optional feature to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

FEE driver**10.3.1.5.13 FeeVirtualPageSize****Table 638 Specification for FeeVirtualPageSize**

Name	FeeVirtualPageSize		
Description	<p>Size in bytes to which logical blocks will be aligned.</p> <p>The value of this parameter is 8 and contrary to AUTOSAR this value cannot be changed and is fixed to 8.</p> <p>According to the hardware the page size is 8, therefore, the default value of FeeVirtualPageSize is 8.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	8 - 8		
Default value	8		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

10.3.1.6 Container: FeelfxSpecificConfig

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

10.3.1.6.1 FeeCancelAllApi**Table 639 Specification for FeeCancelAllApi**

Name	FeeCancelAllApi		
Description	<p>Pre-processor switch to enable or disable the Fee_17_CancelAll API.</p> <p>It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both.</p> <p>By default, the Fee_17_CancelAll() API is disabled to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

FEE driver**Table 639 Specification for FeeCancelAllApi (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.2 FeeEraseAllEnable**Table 640 Specification for FeeEraseAllEnable**

Name	FeeEraseAllEnable		
Description	<p>Allows the user to configure if the sectors should be erased when FEE identifies an invalid sector state during initialization. When configured to TRUE, both the FEE sectors are erased. In such a case, the FEE resume to normal state but the previous data (if any) cannot be recovered.</p> <p>When configured to FALSE, the FEE settles in the illegal state and cannot continue to operate. However, data is kept intact in the DFlash and may be recovered.</p> <p>Note: This is applicable only for the NVM section where double sector is used and not applicable for QS block region.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p> <p>By default FeeEraseAllEnable is disabled to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.3 FeeGcRestart**Table 641 Specification for FeeGcRestart**

Name	FeeGcRestart
Description	Specifies the GC restarting point.

FEE driver**Table 641 Specification for FeeGcRestart (continued)**

	<p>It is applicable when the FEE module is configured as Double-Sector only or Double-Sector and Quasi-Static both.</p> <p>Note: Determination of NVM sector states, QS blocks dirty state and QS Flash virgin state is performed independent of this parameter during initialization.</p> <p>FEE_GC_RESTART_INIT: the following operations are started after FEE initialization is completed:</p> <ol style="list-style-type: none"> 1. Init GC 2. Cache build 3. QS dirty/virgin handling <p>FEE_GC_RESTART_WRITE: the following operations are started when user job is requested:</p> <ol style="list-style-type: none"> 1. Init GC 2. Cache build 3. QS dirty/virgin handling 		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	FEE_GC_RESTART_INIT: FEE_GC_RESTART_WRITE:		
Default value	FEE_GC_RESTART_INIT		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.4 FeeGetCycleCountApi**Table 642 Specification for FeeGetCycleCountApi**

Name	FeeGetCycleCountApi		
Description	<p>Pre-processor switch to enable or disable the Fee_17_GetCycleCount() API.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p> <p>By default, the get cycle count is disabled to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

FEE driver**Table 642 Specification for FeeGetCycleCountApi (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.5 FeeGetPrevDataApi**Table 643 Specification for FeeGetPrevDataApi**

Name	FeeGetPrevDataApi		
Description	Pre-processor switch to enable or disable the Fee_17_GetPrevData() API. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both. By default, the Fee_17_GetPrevData() API is disabled to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.6 FeeMaxBlockCount**Table 644 Specification for FeeMaxBlockCount**

Name	FeeMaxBlockCount
Description	Specifies the total number of blocks throughout all configurations. Note that the blocks which are shared across configurations are counted only once. This parameter decides the cache table size at pre-compile time (cache table size = FeeMaxBlockCount +10. The size of the cache table is increased by 10 as a safety margin). The cache table holds the information about configured, un-configured and QS blocks. Configuration of this parameter is carried out carefully. Higher value implies higher RAM area consumption for the cache table. The lower value would result in the loss of un-configured

FEE driver**Table 644 Specification for FeeMaxBlockCount (continued)**

	<p>block(s) during GC (if FeeUnConfigBlkOverflowHandle is FEE_CONTINUE) or illegal state of FEE (if FeeUnConfigBlkOverflowHandle is FEE_STOP_AT_GC).</p> <p>Example 1: Configuration sets having mutually exclusive blocks:</p> <p>Configuration A: Number of blocks configured is 10.</p> <p>Configuration B: Number of blocks configured is 25.</p> <p>Then, FeeMaxBlockCount = 35.</p> <p>Example 2: Five blocks are shared/used by both Configuration A and Configuration B:</p> <p>Configuration A: Number of blocks configured is 10.</p> <p>Configuration B: Number of blocks configured is 25.</p> <p>Then, FeeMaxBlockCount = 30 (that is, shared blocks are counted only once).</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - Fee.NumberOfBlocks		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.7 FeeMaxBytesPerCycle**Table 645 Specification for FeeMaxBytesPerCycle**

Name	FeeMaxBytesPerCycle		
Description	<p>Specifies the maximum number of data bytes that are processed in one FEE main function call during the read, write and compare operations. The size is inclusive of the block overhead like header, consecutive page id, and so on.</p> <p>It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p>		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	FEE_MAX_BYTES_128: FEE_MAX_BYTES_256: FEE_MAX_BYTES_512: FEE_MAX_BYTES_64:		
Default value	FEE_MAX_BYTES_64		
Post-build variant value	FALSE	Post-build variant multiplicity	-

FEE driver**Table 645 Specification for FeeMaxBytesPerCycle (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.8 FeeNvmIllegalStateNotification**Table 646 Specification for FeeNvmIllegalStateNotification**

Name	FeeNvmIllegalStateNotification		
Description	<p>This parameter is a pointer to a notification API which is called when the FEE (NVM part) reaches an Illegal state. Illegal state means that the FEE is not able to proceed and the user should perform a power-on reset.</p> <p>NVM illegal notification can also be raised due to hardware errors during internal activities of FEE such as GC, initialisation of GC.</p> <p>It is applicable when FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.</p> <p>By default, the optional notification is disabled to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.9 FeeQsHardenErrorNotification**Table 647 Specification for FeeQsHardenErrorNotification**

Name	FeeQsHardenErrorNotification		
Description	<p>Mapped to the hardening error notification routine provided by the upper layer module that uses the FEE services for the QS blocks. The configured function is invoked by the FEE when an error is encountered while performing a hardening operation.</p> <p>The user is expected to provide a function of type void FeeQsHardenErrorNotification (void) when a notification for a failed hardening operation is required. If no notification is required, this parameter should be configured as NULL_PTR.</p>		

FEE driver**Table 647 Specification for FeeQsHardenErrorNotification (continued)**

	<p>The notification function provided is expected to be synchronous.</p> <p>This parameter is applicable when the FEE module is configured as Double-Sector and Quasi-Static.</p> <p>By default, the optional notification is disabled to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.10 FeeQsIllegalStateNotification**Table 648 Specification for FeeQsIllegalStateNotification**

Name	FeeQsIllegalStateNotification		
Description	<p>This parameter is a pointer to a notification API which is called when the FEE (QS part) reaches an Illegal state.</p> <p>Illegal notification is raised when the QS area happens to be in the virgin state and FeeVirginFlashIllegalState is not set.</p> <p>It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both.</p> <p>By default, the optional notification is disabled to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

FEE driver**10.3.1.6.11 FeeStateVarStructure****Table 649 Specification for FeeStateVarStructure**

Name	FeeStateVarStructure		
Description	This parameter is a pointer to a structure which would contain all the global variables of the FEE driver. Using this, the user can allocate the space for the variables at his best to avoid any possible linking problems.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node:		
Default value	Fee_StateVar		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.12 FeeThresholdValue**Table 650 Specification for FeeThresholdValue**

Name	FeeThresholdValue		
Description	Describes the threshold value (in bytes before the end of the FEE sector) for triggering garbage collection/sector change. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - Fee Threshold Size		
Default value	200		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.13 FeeUnConfigBlkOverflowHandle**Table 651 Specification for FeeUnConfigBlkOverflowHandle**

Name	FeeUnConfigBlkOverflowHandle
-------------	------------------------------

FEE driver**Table 651 Specification for FeeUnConfigBlkOverflowHandle (continued)**

Description	Specifies the behavior of the FEE driver when the cache table overflow occurs, that is, insufficient space in the cache table due to wrongly configured value of FeeMaxBlockCount (more number of blocks were detected during cache build, which cannot be accommodated in the cache table). FEE_CONTINUE: un-configured blocks which could not be accommodated in the cache table are lost after the GC. The FEE continues as expected for the currently active configuration. FEE_STOP_AT_GC: During the GC, the FEE enters a pseudo illegal state where only read operation is allowed but write is not allowed. Note: If FeeUnConfigBlock is set to FEE_UNCONFIG_BLOCK_IGNORE then this parameter is irrelevant. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	FEE_CONTINUE: FEE_STOP_AT_GC:		
Default value	FEE_CONTINUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.14 FeeUnConfigBlock**Table 652 Specification for FeeUnConfigBlock**

Name	FeeUnConfigBlock		
Description	Specifies whether unconfigured blocks should be copied to the new sector or ignored during the GC. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	FEE_UNCONFIG_BLOCK_IGNORE: FEE_UNCONFIG_BLOCK_KEEP:		
Default value	FEE_UNCONFIG_BLOCK_IGNORE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

FEE driver**Table 652 Specification for FeeUnConfigBlock (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.15 FeeUseEraseSuspend**Table 653 Specification for FeeUseEraseSuspend**

Name	FeeUseEraseSuspend		
Description	<p>Specifies the usage of the erase-suspend feature provided by the hardware. If it is configured as TRUE, then user read, write and invalidate requests are serviced during the erase operation of the GC or QS block. If it is configured as FALSE, then user requests during erase operations are not accepted.</p> <p>By default, the erase-suspend feature is disabled to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FeeBlockTypeConfigured		

10.3.1.6.16 FeeVirginFlashIllegalState**Table 654 Specification for FeeVirginFlashIllegalState**

Name	FeeVirginFlashIllegalState		
Description	<p>Allows the user to configure the behavior of the FEE upon detection of virgin Flash (DFlash0). It can either be configured to call the configured illegal state notification function or to program the state blocks and perform normal operation.</p> <p>Values:</p> <p>FALSE: upon detection of virgin Flash (DFlash0) - FEE programs the initial state blocks and is available for user requests.</p> <p>TRUE: upon detection of virgin Flash (DFlash0) - FEE reaches illegal state and calls the configured illegal state notification function, if configured.</p>		

FEE driver**Table 654 Specification for FeeVirginFlashIllegalState (continued)**

	By default, the FeeVirginFlashIllegalState is disabled to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

10.3.1.7 Container: FeePublishedInformation

This container holds additional published parameters not covered by the CommonPublishedInformation container. Note that these parameters do not have any configuration class setting, because they are published information.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

10.3.1.7.1 FeeBlockOverhead**Table 655 Specification for FeeBlockOverhead**

Name	FeeBlockOverhead		
Description	Management overhead per logical block in bytes. [a] Block management overhead per logical NVM block in bytes is given by: (i) If ((FeeBlockSize % 7) == 0) then, ((FeeBlockSize / 7) * 8) + 16 - FeeBlockSize (ii) If ((FeeBlockSize % 7) != 0) then, (((FeeBlockSize / 7) + 1) * 8) + 16 - FeeBlockSize Note: Integer arithmetic is used in the calculation. [b] Block management overhead per logical QS block in bytes is minimum 36 bytes and can be more depending on the block size as minimum size allowed is 4k bytes Note: If the management overhead depends on the block size or block location, a formula has to be provided that allows the configurator to calculate the management overhead correctly. Since the default FEE block size is 1 therefore, 17 is taken as the block overhead by default.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	17 - 17		

FEE driver**Table 655 Specification for FeeBlockOverhead (continued)**

Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

10.3.1.7.2 FeePageOverhead**Table 656 Specification for FeePageOverhead**

Name	FeePageOverhead		
Description	<p>Management overhead per page in bytes.</p> <p>This value is applicable only for the pages containing logical block data bytes for NVM (that is, not applicable for header and marker). For QS blocks, minimum block size to be configured is sector size which is 4k and this is not relevant.</p> <p>Note: If the management overhead depends on the block size or block location, a formula has to be provided that allows the configurator to calculate the management overhead correctly.</p> <p>Overhead per page is 1 byte which is set as the default value.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 1		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

10.3.2 Functions - Type definitions**10.3.2.1 Fee_BlockType****Table 657 Specification for Fee_BlockType**

Syntax	Fee_BlockType
---------------	---------------

FEE driver**Table 657 Specification for Fee_BlockType (continued)**

Type	Structure	
File	Fee.h	
Range	unsigned_int CycleCountLimit : 24	FEE Block Cycle Count Range – [0.....16777215]
	unsigned_int FeeImmediateData : 8	Determine FEE Data FEE_NORMAL_DATA or FEE_IMMEDIATE_DATA Range: 0 - FEE_NORMAL_DATA 1 - FEE_IMMEDIATE_DATA
	unsigned_int BlockNumber : 16	FEE logical block number Range – [1.....65534]
	unsigned_int Size : 16	Size of the logical block Range – [0.....65535]
	unsigned_int Address : 32	FEE block address for quasi blocks only. This parameter is used for quasi feature only. Range – [0.....max size (device specific)]
	unsigned_int Instances : 16	FEE block instances for quasi blocks only. This parameter is used for quasi feature only. Range – [0.....256]
	unsigned_int FeeUser : 8	FEE user type determination FEE_NVM_USER or FEE_QUASI_STATIC_USER Range: 0 - FEE_NVM_USER 1 - FEE_QUASI_STATIC_USER
Description	This type definition contains the types for the logical block configuration data	
Source	IFX	

10.3.2.2 Fee_ConfigType**Table 658 Specification for Fee_ConfigType**

Syntax	Fee_ConfigType	
Type	Structure	
File	Fee.h	
Range	-	
Description	Configuration data structure of the FEE module. Elements of this structure are defined during the design phase	
Source	IFX	

FEE driver**10.3.2.3 Fee_DataType****Table 659 Specification for Fee_DataType**

Syntax	Fee_DataType
Type	Enumeration
File	Fee.h
Range	1 - FEE_IMMEDIATE_DATA 0 - FEE_NORMAL_DATA
Description	This type definition contains enumerations for the logical block types. FEE NVM block type: Normal Type OR Immediate Type
Source	IFX

10.3.2.4 Fee_NotifyFunctionPtrType**Table 660 Specification for Fee_NotifyFunctionPtrType**

Syntax	Fee_NotifyFunctionPtrType
Type	Pointer to a function of type void Function_Name (void)
File	Fee.h
Description	Defines the function pointer type for the call back functions (job completion, job failure, illegal state).
Source	IFX

10.3.2.5 Fee_PageType**Table 661 Specification for Fee_PageType**

Syntax	Fee_PageType
Type	uint16
File	Fee.h
Range	uint16
Description	Number of DFlash pages read/written
Source	IFX

10.3.2.6 Fee_QsBlock_StateType**Table 662 Specification for Fee_QsBlock_StateType**

Syntax	Fee_QsBlock_StateType
Type	Enumeration
File	Fee.h

FEE driver**Table 662 Specification for Fee_QsBlock_StateType (continued)**

Range	0 - FEE_QS_PROG_STATE_ERASE_STARTED	Internal state to indicate an erase job has been initialized.
	1 - FEE_QS_PROG_STATE_DESTROY	Internal state to indicate programmed state to be destroyed
	2 - FEE_QS_PROG_STATE_ERASE_COMPLETE	Internal state to indicate erase is completed
	3 - FEE_QS_PROG_STATE_WRITE_COMPLETE	Internal state to indicate write is complete
	4 - FEE_QS_PROG_STATE_WRITE_STARTED	Internal state to indicate write is started
	5 - FEE_QS_START_ERASE	Internal state to indicate an erase is requested
	6 - FEE_QS_START_BCC_WRITE	Internal state to indicate a write of the BCC
	7 - FEE_QS_START_BLOCK_WRITE	Internal state to indicate a start of the writing operation of the block
	8 - FEE_QS_ERASE_COMPLETE	Internal state to indicate Erase complete and erase complete state is properly set
	9 - FEE_QS_DIRTY_ERASE	Internal state to indicate Erase complete and erase complete state is properly set
	10 - FEE_QS_WRITE_COMPLETE	Internal state to indicate write complete and write state is properly set
	11 - FEE_QS_DIRTY_WRITE	Internal state to indicate write complete and write state is incomplete and nonzero
	12 - FEE_QS_ERASE_STARTED	Internal state to indicate erase started
	13 - FEE_QS_WRITE_STARTED	Internal state to indicate write started state is set and block partially written
	14 - FEE_QS_DESTROY	State in which QS marker pages are written with 0xFF.
	25 - FEE_QS_INVALID	All states are invalid
Description	This type definition contains enumerations for QS Block State type	
Source	IFX	

10.3.2.7 Fee_QuasiStaticBlockInfoType**Table 663 Specification for Fee_QuasiStaticBlockInfoType**

Syntax	Fee_QuasiStaticBlockInfoType
Type	Structure
File	Fee.h

FEE driver**Table 663 Specification for Fee_QuasiStaticBlockInfoType (continued)**

Range	uint16 Bcc	Block Cycle count Range - [0.....65535]
	Fee_QsBlock_StateType State	Block state Range - Refer to the enum Fee_QsBlock_StateType for range.
Description	This type definition contains the types for holding the block information about Block cycle count and Block state (for Quasi blocks only).	
Source	IFX	

10.3.2.8 Fee_StateDataType**Table 664 Specification for Fee_StateDataType**

Syntax	Fee_StateDataType	
Type	Structure	
File	Fee.h	
Range	To be elaborated in Design	
Description	This type definition contains the types for holding the FEE driver status data.	
Source	IFX	

10.3.2.9 Fee_UserType**Table 665 Specification for Fee_UserType**

Syntax	Fee_UserType	
Type	Enumeration	
File	Fee.h	
Range	0 - FEE_NVM_USER	
	1 - FEE_QUASI_STATIC_USER	
Description	FEE feature type selection: NVM OR Quasi	
Source	IFX	

10.3.3 Functions - APIs**10.3.3.1 Fee_17_CancelAll****Table 666 Specification for Fee_17_CancelAll API**

Syntax	void Fee_17_CancelAll (void)	
---------------	---	--

FEE driver**Table 666 Specification for Fee_17_CancelAll API (continued)**

Service ID	0x28	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Service to cancel any ongoing internal/user read or write job. However, the ongoing erase in the hardware cannot be cancelled. This API is expected to be called before a high priority QS data write is requested. When a user requested job (pending or ongoing) is cancelled, a job error notification is generated. If an internal operation such as garbage collection is cancelled, then no job error notification is generated.</p> <p>Note:</p> <ol style="list-style-type: none"> 1) This API is not available when FEE is configured to support only Double Sector data (that is NVM data only). 2) This API is not available if FeeCancelAllApi is disabled (FALSE). 3) This API will not cancel module initialization related activities and no safety error(FEE_E_INVALID_CANCEL) will be raised. 	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FEE_SE_UNINIT: API service is called when the module is not initialized</p> <p>FEE_SE_INVALID_CANCEL: API service is called while no job is pending</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FeeBlockTypeConfigured,FeeCancelAllApi	
User hints	-	

10.3.3.2 Fee_17_DisableGcStart**Table 667 Specification for Fee_17_DisableGcStart API**

Syntax	void Fee_17_DisableGcStart (
---------------	---------------------------------

FEE driver**Table 667 Specification for Fee_17_DisableGcStart API (continued)**

	void)	
Service ID	0x22	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The user can call this API to prevent the GC from being started in case the threshold is crossed in the active FEE sector. This API does not stop an ongoing GC but only prevents the GC from being triggered by the write/invalidate request issued by the user.</p> <p>Note: This API is applicable only for Double-Sector data.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FEE_SE_UNINIT: API service is called when the module is not initialized</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FeeBlockTypeConfigured	
User hints	-	

10.3.3.3 Fee_17_EnableGcStart**Table 668 Specification for Fee_17_EnableGcStart API**

Syntax	void Fee_17_EnableGcStart (void)
Service ID	0x21
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Non Reentrant

FEE driver**Table 668 Specification for Fee_17_EnableGcStart API (continued)**

Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This service allows enabling the trigger of GC, if GC is disabled earlier by calling the Fee_17_DisableGcStart() API.</p> <p>After this API is called, if the sector is filled up to the threshold level and additional write / invalidate request is issued, then GC is initiated.</p> <p>Note: This API is applicable only for the Double-Sector data.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FEE_SE_UNINIT: API service is called when the module is not initialized</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FeeBlockTypeConfigured	
User hints	-	

10.3.3.4 Fee_17_EraseQuasiStaticData**Table 669 Specification for Fee_17_EraseQuasiStaticData API**

Syntax	<pre>Std_ReturnType Fee_17_EraseQuasiStaticData (const uint16 BlockNumber, const uint16 Instances)</pre>	
Service ID	0x25	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber Instances	Logical block number Number of block instances
Parameters (out)	-	

FEE driver**Table 669 Specification for Fee_17_EraseQuasiStaticData API (continued)**

Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the requested job has been accepted by the module. E_NOT_OK: the requested job has not been accepted by the module because the request is either pending or the QS is not initialized or the GC is running or the block is not found or the given block instance is incorrect
Description	Service to request an erase job for one or multiple consecutive instances of a Quasi-Static data block Note: This API is applicable only for the QS data block type.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: FEE_SE_INVALID_BLOCK_NO: API service is called with an invalid block number FEE_SE_BUSY: API service is called while the module is busy processing a user request FEE_SE_INVALID_BLOCK_INSTANCES: API service is called with invalid block instances FEE_SE_UNINIT: API service is called when the module is not initialized <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FeeBlockTypeConfigured	
User hints	-	

10.3.3.5 Fee_17_GetCycleCount**Table 670 Specification for Fee_17_GetCycleCount API**

Syntax	<pre>Std_ReturnType Fee_17_GetCycleCount (const uint16 BlockNumber, uint32 * const CountPtr)</pre>	
Service ID	0x20	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber	Logical block number

FEE driver**Table 670 Specification for Fee_17_GetCycleCount API (continued)**

Parameters (out)	CountPtr	Pointer to the variable to which the cycle count is to be updated
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the cycle count was read and returned successfully. E_NOT_OK: the function could not read the cycle count as FEE was busy or a read error occurred or the cache update was not complete or GC was ongoing.
Description		When called for a configured non-QS BlockNumber, the write cycle count of the given block is returned. When called with BlockNumber = 0, this routine delivers the FEE sector erase cycle count. Note: This API is not applicable for the Quasi-Static data blocks.
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: FEE_SE_INVALID_BLOCK_NO: API service is called with an invalid block number FEE_SE_PARAM_POINTER: API service is called with an invalid data pointer FEE_SE_BUSY: API service is called while the module is busy processing a user request FEE_SE_UNINIT: API service is called when the module is not initialized <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FeeGetCycleCountApi,FeeBlockTypeConfigured	
User hints	-	

10.3.3.6 Fee_17_GetPrevData**Table 671 Specification for Fee_17_GetPrevData API**

Syntax	Std_ReturnType Fee_17_GetPrevData (const uint16 BlockNumber, const uint16 BlockOffset, uint8 * const DataBufferPtr, const uint16 Length)
Service ID	0x23
Sync/Async	Asynchronous
ASIL Level	B
Re-entrancy	Non Reentrant

FEE driver**Table 671 Specification for Fee_17_GetPrevData API (continued)**

Parameters (in)	BlockNumber BlockOffset Length	Logical block number Address offset within the block Number of bytes to be read
Parameters (out)	DataBufferPtr	Pointer to data buffer
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the read job is accepted E_NOT_OK: the read job is not accepted as there is a pending request (module is busy).
Description	<p>This API reads one preceding occurrence of data (that is most recent one in history) of the given block. This API accepts the request and updates the FEE internal variables. However the actual reading of the data is done by the Fee_MainFunction after the cache is built.</p> <p>Note:</p> <ol style="list-style-type: none"> After GC only latest copy of the blocks is available. So, if this API is called immediately after GC then it will read the latest copy of the given block. This is a Non-Autosar API and is applicable only for non-QS blocks. 	
Source	IFX	
Error handling	<p>DET: None Runtime Errors: None DEM: None Safety Errors: FEE_SE_PARAM_POINTER: API service is called with an invalid data pointer FEE_SE_INVALID_BLOCK_NO: API service is called with an invalid block number FEE_SE_UNINIT: API service is called when the module is not initialized FEE_SE_INVALID_BLOCK_OFS : API service is called with the invalid block offset FEE_SE_INVALID_BLOCK_LEN: API service is called with an invalid length information FEE_SE_BUSY: API service is called while the module is busy processing a user request</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FeeGetPrevDataApi	
User hints	-	

10.3.3.7 Fee_17_GetQuasiStaticBlockInfo**Table 672 Specification for Fee_17_GetQuasiStaticBlockInfo API**

Syntax	Std_ReturnType Fee_17_GetQuasiStaticBlockInfo (const uint16 BlockNumber,
---------------	---

FEE driver**Table 672 Specification for Fee_17_GetQuasiStaticBlockInfo API (continued)**

	Fee_QuasiStaticBlockInfoType * const BlockInfoPtr)	
Service ID	0x26	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber	Logical block number
Parameters (out)	BlockInfoPtr	Constant pointer to the BlockInfo structure
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : QS block information is read successfully E_NOT_OK : QS block information is not available
Description	<p>Service to read the Block State and the Block Cycle Counter of the given Quasi-Static data block instance.</p> <p>It is expected that before a QS read, write or erase operation is requested, this service is called to know the status of the QS block instance. This service provides an opportunity to ascertain if a block instance was erased before a write request is made, or, if a previous block instance write was interrupted before a new read request is made.</p> <p>Note: The QS block state values returned by this API are :</p> <ul style="list-style-type: none"> FEE_QS_ERASE_STARTED : If the block erase was started FEE_QS_ERASE_COMPLETE: If the block erase was completed FEE_QS_WRITE_STARTED : If the write to the block started FEE_QS_WRITE_COMPLETE: If the writ to the block completed FEE_QS_INVALID : If the block was invalid 	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <ul style="list-style-type: none"> FEE_SE_UNINIT: API service is called when the module is not initialized FEE_SE_PARAM_POINTER: API service is called with an invalid data pointer FEE_SE_INVALID_BLOCK_NO: API service is called with an invalid block number FEE_SE_BUSY: API service is called while the module is busy processing a user request <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FeeBlockTypeConfigured	
User hints	-	

FEE driver**10.3.3.8 Fee_17_GetQuasiStaticJobResult****Table 673 Specification for Fee_17_GetQuasiStaticJobResult API**

Syntax	MemIf_JobResultType Fee_17_GetQuasiStaticJobResult (void)	
Service ID	0x27	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	MemIf_JobResultType	<p>MEMIF_JOB_OK : The last job has been finished successfully and there is no job pending</p> <p>MEMIF_JOB_PENDING : The last job is waiting for execution or currently being executed</p> <p>MEMIF_JOB_CANCELED : The last job has been cancelled</p> <p>MEMIF_JOB_FAILED:</p> <ol style="list-style-type: none"> 1. The FEE driver has not been initialized (Fee_Init not called) 2. The last read/write/erase job failed.
Description	<p>Service to query the result of the last accepted job issued by the QS Manager</p> <p>This API is applicable only if the FEE is configured to support the Quasi-Static data blocks.</p> <p>Note: This API is applicable only for the QS data block type.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FEE_SE_UNINIT: API service is called when the module is not initialized</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FeeBlockTypeConfigured	
User hints	-	

FEE driver**10.3.3.9 Fee_17_InitCheck****Table 674 Specification for Fee_17_InitCheck API**

Syntax	<pre>Std_ReturnType Fee_17_InitCheck (const Fee_ConfigType * const ConfigPtr)</pre>	
Service ID	0x30	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the selected Configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Module is initialized properly. E_NOT_OK: Module is not initialized properly due to - Fee_CfgPtr is NULL - Fee_CfgPtr is not matching with given ConfigPtr. - Fee is not yet completely initialized.
Description	<p>This function verifies the initialization of the FEE driver. Note: The application should follow the following calling sequence: 1. Call Fee_Init 2. Call Fee_17_InitCheck Note: The Fee_17_InitCheck() API should be called before any other FEE API or Fee_MainFunction is called.</p>	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FeeInitCheckApi	
User hints	-	

FEE driver**10.3.3.10 Fee_Cancel****Table 675 Specification for Fee_Cancel API**

Syntax	<pre>void Fee_Cancel (void)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Service to call the cancel function of the underlying flash driver.</p> <p>When an ongoing user requested read job is canceled, the Infineon specific FLS service to cancel non-erase jobs is called.</p> <p>For an ongoing user requested write, invalidate block, there is no cancel operation initiated with the FLS driver. Canceling ongoing user requested write and invalidate operations that are already ongoing cause internal data structures to become inconsistent and therefore these are not canceled.</p> <p>If a user requested read, write or invalidate job is not started and is held in pending state due an ongoing internal operation, these jobs can be canceled by the Fee_Cancel API.</p> <p>Ongoing internal read, write and erase operations are not cancelled.</p> <p>Fee_Cancel should not be called for QS data. The service to cancel all ongoing jobs should be used instead.</p> <p>Note: This API is applicable only for double sector data (NVM) block.</p> <p>Note: There are deviations taken for Aurix2G with respect to this AUTOSAR API.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FEE_E_UNINIT : API service is called when the module is not initialized</p> <p>FEE_E_INVALID_CANCEL: API service is called while no job is pending</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

FEE driver**Table 675 Specification for Fee_Cancel API (continued)**

Configuration dependencies	-
User hints	-

10.3.3.11 Fee_EraseImmediateBlock**Table 676 Specification for Fee_EraseImmediateBlock API**

Syntax	Std_ReturnType Fee_EraseImmediateBlock (const uint16 BlockNumber)	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber	Logical Block Number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	-
Description	Service to erase an immediate logical block. Since the double-sector algorithm is used with the threshold limit for triggering GC, write requests of immediate block during GC can be accommodated within the pre-erased threshold area of the active FEE sector. Hence, this API is implemented as an empty function returning E_NOT_OK always.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

FEE driver**10.3.3.12 Fee_GetJobResult****Table 677 Specification for Fee_GetJobResult API**

Syntax	MemIf_JobResultType Fee_GetJobResult (void)	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	MemIf_JobResultType	<p>MEMIF_JOB_OK: the last job has been finished successfully and there is no pending job pending</p> <p>MEMIF_JOB_PENDING: the last job is waiting for execution or is being executed currently. This can happen in the following cases:</p> <ol style="list-style-type: none"> 1. When there is a pending request waiting to be executed (GC/Init on-going) 2. When there is a request being executed <p>MEMIF_JOB_CANCELED : The last job has been cancelled</p> <p>MEMIF_JOB_FAILED :</p> <ul style="list-style-type: none"> - The FEE driver has not been initialized (Fee_Init not called) - The last read/write/invalidate job failed. <p>MEMIF_BLOCK_INCONSISTENT :</p> <ul style="list-style-type: none"> - The requested block is inconsistent, it may contain corrupted data. - The requested block to be read is present in the configuration but is not written in DFlash yet <p>MEMIF_BLOCK_INVALID : The requested block has been invalidated; the requested read operation cannot be performed</p>
Description	Service to query the result of the last accepted job issued by the NVM.	
Source	AUTOSAR	
Error handling	DET: FEE_E_UNINIT : API service is called when the module is not initialized Runtime Errors: None	

FEE driver**Table 677 Specification for Fee_GetJobResult API (continued)**

	DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

10.3.3.13 Fee_GetStatus**Table 678 Specification for Fee_GetStatus API**

Syntax	MemIf_StatusType Fee_GetStatus (void)	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	MemIf_StatusType	<p>MEMIF_UNINIT: the FEE driver has not been initialized (Fee_Init not called)</p> <p>MEMIF_IDLE: the FEE driver and the underlying Flash driver are currently idle. This can happen in the following cases:</p> <ul style="list-style-type: none"> - When there is no pending request and GC is idle - When there is no pending request and GC cannot be started because GC restart point is FEE_GC_RESTART_WRITE - When there is no pending request and GC has entered a fail state - When there is no pending request and Init GC has entered a fail state - When the write has entered a fail state <p>MEMIF_BUSY: the FEE driver is currently busy processing a request.</p> <p>MEMIF_BUSY_INTERNAL: the FEE module is busy with internal management operations</p>

FEE driver**Table 678 Specification for Fee_GetStatus API (continued)**

Description	Service to return the status of the driver. Though GC is considered as an internal management operation, the driver status is maintained as MEMIF_BUSY until the GC copy is over (this is for normal block). During GC erase the module state becomes MEMIF_BUSY_INTERNAL. In the case of immediate block the module status is MEMIF_BUSY_INTERNAL, the moment GC is triggered.
Source	AUTOSAR
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

10.3.3.14 Fee_GetVersionInfo**Table 679 Specification for Fee_GetVersionInfo API**

Syntax	<pre>void Fee_GetVersionInfo (Std_VersionInfoType * const VersionInfoPtr)</pre>	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer to the standard version information structure.
Parameters (in - out)	-	-
Return	void	-
Description	Service to return the version information of the FEE module. <i>Note: This API is applicable if the FeeVersionInfoApi is enabled.</i>	
Source	AUTOSAR	
Error handling	DET: FEE_E_PARAM_POINTER: API service is called with an invalid data pointer Runtime Errors: None	

FEE driver**Table 679 Specification for Fee_GetVersionInfo API (continued)**

	<p>DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	FeeVersionInfoApi
User hints	-

10.3.3.15 Fee_Init**Table 680 Specification for Fee_Init API**

Syntax	<pre>void Fee_Init (const Fee_ConfigType * const ConfigPtr)</pre>	
Service ID	0x00	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the selected configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to initialize the FEE module.	
Source	AUTOSAR	
Error handling	<p>DET: FEE_E_PARAM_POINTER: API service is called with an invalid data pointer Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

FEE driver**10.3.3.16 Fee_InvalidateBlock****Table 681 Specification for Fee_InvalidateBlock API**

Syntax	<pre>Std_ReturnType Fee_InvalidateBlock (const uint16 BlockNumber)</pre>	
Service ID	0x07	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber	Logical block number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the requested job has been accepted by the module. E_NOT_OK: the requested job has not been accepted by the module.
Description	<p>Service to invalidate a logical block. This service initiates a write job to mark the block as invalid, make the module status as MEMIF_BUSY and set the job result to MEMIF_JOB_PENDING.</p> <p>This API is only available for non-QS data blocks. For QS data blocks, the service to erase QS data blocks is to be used.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FEE_E_UNINIT : API service is called when the module is not initialized</p> <p>FEE_E_INVALID_BLOCK_NO: API service is called with an invalid block number</p> <p>FEE_E_BUSY: API service is called while the module is busy processing a user request</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

FEE driver**10.3.3.17 Fee_Read****Table 682 Specification for Fee_Read API**

Syntax	<pre>Std_ReturnType Fee_Read (const uint16 BlockNumber, const uint16 BlockOffset, uint8 * const DataBufferPtr, const uint16 Length)</pre>	
Service ID	0x02	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber BlockOffset Length	Logical block number. Read offset inside the block. Number of bytes to read
Parameters (out)	DataBufferPtr	Pointer to data buffer
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the requested job has been accepted by the module. E_NOT_OK: the requested job has not been accepted by the module.
Description	This service initiates the read job for a QS or a non-QS block. If the length to be read is zero, then the function returns E_NOT_OK and no DET is raised.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FEE_E_UNINIT : API service is called when the module is not initialized</p> <p>FEE_E_INVALID_BLOCK_NO: API service is called with an invalid block number</p> <p>FEE_E_INVALID_BLOCK_OFS : API service is called with an invalid block offset</p> <p>FEE_E_PARAM_POINTER: API service is called with an invalid data pointer</p> <p>FEE_E_INVALID_BLOCK_LEN: API service is called with an invalid length information</p> <p>FEE_E_BUSY: API service is called while the module is busy processing a user request</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

FEE driver**10.3.3.18 Fee_SetMode****Table 683 Specification for Fee_SetMode API**

Syntax	<pre>void Fee_SetMode (const MemIf_ModeType Mode)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Mode	Desired mode for the underlying flash driver. The mode value can be MEMIF_MODE_SLOW or MEMIF_MODE_FAST
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Service to call the Fls_17_Dmu_SetMode function of the underlying Flash driver.</p> <p>Note: This API is applicable if FeeSetModeSupport is enabled and if the FEE is configured to support the double-sector (NVM) data. If the Mode parameter passed to this function is other than MEMIF_MODE_SLOW or MEMIF_MODE_FAST then the error is detected and reported by the FLS module.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FEE_E_UNINIT : API service is called when the module is not initialized</p> <p>FEE_E_BUSY: API service is called while the module is busy processing a user request</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FeeSetModeSupported	
User hints	-	

10.3.3.19 Fee_Write**Table 684 Specification for Fee_Write API**

Syntax	Std_ReturnType Fee_Write
	(

FEE driver**Table 684 Specification for Fee_Write API (continued)**

	<pre> const uint16 BlockNumber, const uint8 * const DataBufferPtr)</pre>	
Service ID	0x03	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber DataBufferPtr	Logical block number Pointer to data buffer
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the requested job has been accepted by the module. E_NOT_OK: the requested job has not been accepted by the module.
Description	This service initiates the write to the given block number.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FEE_E_UNINIT: API service is called when the module is not initialized</p> <p>FEE_E_INVALID_BLOCK_NO: API service is called with an invalid block number</p> <p>FEE_E_PARAM_POINTER: API service is called with an invalid data pointer</p> <p>FEE_E_BUSY: API service is called while the module is busy processing a user request</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

10.3.4 Notifications and Callbacks**10.3.4.1 Fee_17_IllegalStateNotification****Table 685 Specification for Fee_17_IllegalStateNotification API**

Syntax	void Fee_17_IllegalStateNotification (
---------------	---

FEE driver**Table 685 Specification for Fee_17_IllegalStateNotification API (continued)**

	void)	
Service ID	0x24	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This notification function is called by the underlying FLS driver when FLS driver reaches an illegal state.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: Fee_SE_UNINIT: API service is called when the module is not initialized <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

10.3.4.2 Fee_17_JobEraseErrorNotification**Table 686 Specification for Fee_17_JobEraseErrorNotification API**

Syntax	void Fee_17_JobEraseErrorNotification (void)	
Service ID	0x29	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-

FEE driver**Table 686 Specification for Fee_17_JobEraseErrorNotification API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to report to the FEE module the failure of an erase operation when EVER (Erase Verify) error occurred.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: FEE_SE_UNINIT: API service is called when the module is not initialized <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

10.3.4.3 Fee_17_JobProgErrorNotification**Table 687 Specification for Fee_17_JobProgErrorNotification API**

Syntax	<pre>void Fee_17_JobProgErrorNotification (void)</pre>	
Service ID	0x31	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to report to the FEE module when the Program Verify Error occurred while programming/writing.	
Source	IFX	

FEE driver**Table 687 Specification for Fee_17_JobProgErrorNotification API (continued)**

Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: FEE_SE_UNINIT: API service is called when the module is not initialized <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

10.3.4.4 Fee_JobEndNotification**Table 688 Specification for Fee_JobEndNotification API**

Syntax	void Fee_JobEndNotification (void)	
Service ID	0x10	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to report to FEE module about the successful end of an asynchronous operation performed by the underlying Flash driver.	
Source	AUTOSAR	
Error handling	DET: FEE_E_UNINIT : API service is called when the module is not initialized Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	

FEE driver**Table 688 Specification for Fee_JobEndNotification API (continued)**

User hints	-
------------	---

10.3.4.5 Fee_JobErrorNotification**Table 689 Specification for Fee_JobErrorNotification API**

Syntax	void Fee_JobErrorNotification (void)	
Service ID	0x11	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to report to the FEE module that the underlying flash driver (FLS) failed to perform an asynchronous operation.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FEE_E_UNINIT : API service is called when the module is not initialized</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>FEE_E_READ: Failure during the Block read</p> <p>FEE_E_WRITE: Failure during the Block write</p> <p>FEE_E_INVALIDATE: Failure during the Block invalidate</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

FEE driver**10.3.5 Scheduled functions****10.3.5.1 Fee_MainFunction****Table 690 Specification for Fee_MainFunction API**

Syntax	<pre>void Fee_MainFunction (void)</pre>	
Service ID	0x12	
Sync/Async	NA	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The scheduled function helps to drive asynchronous jobs- read, write, erase and internal management jobs like garbage collection.	
Source	AUTOSAR	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM:</p> <ul style="list-style-type: none"> FEE_E_READ: Failure during the Block read FEE_E_INVALIDATE: Failure during the Block invalidate FEE_E_GC_WRITE: Failure during the GC write FEE_E_UNCONFIG_BLK_EXCEEDED: Unconfigured Block count limit reached FEE_E_WRITE: Failure during the Block write FEE_E_WRITE_CYCLES_EXHAUSTED: Block maximum write count exceeded FEE_E_GC_ERASE: Failure during the GC erase FEE_E_GC_INIT: Failure in the GC during initialization FEE_E_GC_READ: Failure during the GC read FEE_E_GC_TRIG: GC triggering GC <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

FEE driver

10.3.6 Interrupt service routines

The FEE driver does not service any interrupts.

10.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

10.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Note: *The following error IDs are also reported as safety errors.*

Table 691 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
API service is called when the module is not initialized.	AUTOSAR	FEE_E_UNINIT =0x01	Fee_JobEndNotification, Fee_JobErrorNotification, Fee_Write, Fee_SetMode, Fee_Read, Fee_InvalidateBlock, Fee_GetJobResult, Fee_Cancel
API service is called with an invalid block number.	AUTOSAR	FEE_E_INVALID_BLOCK_NO=0x02	Fee_Read, Fee_Write, Fee_InvalidateBlock
API service is called with an invalid block offset.	AUTOSAR	FEE_E_INVALID_BLOCK_OFS =0x03	Fee_Read
API service is called with an invalid data pointer.	AUTOSAR	FEE_E_PARAM_POINTER=0x04	Fee_Read, Fee_Init, Fee_GetVersionInfo, Fee_Write
API service is called with an invalid length information.	AUTOSAR	FEE_E_INVALID_BLOCK_LEN=0x05	Fee_Read
API service is called while the module is busy processing a user request.	AUTOSAR	FEE_E_BUSY=0x06	Fee_Write, Fee_Read, Fee_SetMode, Fee_InvalidateBlock
API service is called while no job is pending.	AUTOSAR	FEE_E_INVALID_CANCEL=0x08	Fee_Cancel

10.3.7.2 Production errors

The following table lists all the production errors reported by the driver.

Table 692 Description of production errors reported

Description	Source	Error code and value	Applicable APIs
Failure during the GC erase.	IFX	FEE_E_GC_ERASE=Assigned by DEM	Fee_MainFunction

FEE driver**Table 692 Description of production errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
Failure in the GC during initialization.	IFX	FEE_E_GC_INIT=Assigned by DEM	Fee_MainFunction
Failure during the GC read.	IFX	FEE_E_GC_READ=Assigned by DEM	Fee_MainFunction
GC triggering GC.	IFX	FEE_E_GC_TRIG=Assigned by DEM	Fee_MainFunction
Failure during the GC write.	IFX	FEE_E_GC_WRITE=Assigned by DEM	Fee_MainFunction
Failure during the Block invalidate.	IFX	FEE_E_INVALIDATE=Assigned by DEM	Fee_JobErrorNotification, Fee_MainFunction
Failure during the Block read.	IFX	FEE_E_READ=Assigned by DEM	Fee_JobErrorNotification, Fee_MainFunction
Unconfigured Block count limit reached.	IFX	FEE_E_UNCONFIG_BLK_EXCEEDED=Assigned by DEM	Fee_MainFunction
Failure during the Block write.	IFX	FEE_E_WRITE=Assigned by DEM	Fee_JobErrorNotification, Fee_MainFunction
Block maximum write count exceeded.	IFX	FEE_E_WRITE_CYCLES_EXHAUSTED=Assigned by DEM	Fee_MainFunction

10.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 693 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
API service is called when the module is not initialized.	IFX	FEE_SE_UNINIT=0x01	Fee_17_JobProgErrorNotification, Fee_17_EraseQuasiStaticData, Fee_17_GetCycleCount, Fee_17_JobEraseErrorNotification, Fee_17_IllegalStateNotification, Fee_17_GetQuasiStaticJobResult, Fee_17_GetQuasiStaticBlockInfo, Fee_17_GetPrevData, Fee_17_EnableGcStart, Fee_17_DisableGcStart, Fee_17_CancelAll
API service is called with an invalid block number.	IFX	FEE_SE_INVALID_BLOCK_NO=0x02	Fee_17_GetQuasiStaticBlockInfo, Fee_17_GetPrevData,

FEE driver
Table 693 Description of safety errors reported (continued)

Description	Source	Error code and value	Applicable APIs
			Fee_17_GetCycleCount, Fee_17_EraseQuasiStatic Data
API service is called with the invalid block offset.	IFX	FEE_SE_INVALID_BLOCK_OFS=0x03	Fee_17_GetPrevData
API service is called with an invalid data pointer.	IFX	FEE_SE_PARAM_POINTER=0x04	Fee_17_GetCycleCount, Fee_17_GetQuasiStaticBl ockInfo, Fee_17_GetPrevData
API service is called with an invalid length information.	IFX	FEE_SE_INVALID_BLOCK_LEN=0x05	Fee_17_GetPrevData
API service is called while the module is busy processing a user request.	IFX	FEE_SE_BUSY=0x06	Fee_17_GetPrevData, Fee_17_GetQuasiStaticBl ockInfo, Fee_17_EraseQuasiStatic Data, Fee_17_GetCycleCount
API service is called while no job is pending.	IFX	FEE_SE_INVALID_CANCEL=0x08	Fee_17_CancelAll
API service is called with invalid block instances.	IFX	FEE_SE_INVALID_BLOCK_INSTANCES =0x20	Fee_17_EraseQuasiStatic Data

10.3.7.4 Runtime errors

The driver does not report any runtime errors.

10.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

10.3.8.1 Deviations

The section describes the deviations from software specification.

Table 694 Known deviations

Reference	Deviation
FEE as a precompile module	According to AUTOSAR, the FEE driver should be implemented as a pre-compile variant module. However, the Infineon FEE driver is implemented as a post-build variant.
FeeImmedateData	According to AUTOSAR, FeeImmedateData should be implemented as pre-compile variant. However, this configuration parameter is implemented as a post-build variant.

FEE driver
Table 694 Known deviations (continued)

Reference	Deviation
FeeNumberOfWriteCycles	According to AUTOSAR, FeeNumberOfWriteCycles should be implemented as pre-compile variant. However, this configuration parameter is implemented as a post-build variant.
FeeBlockSize	According to AUTOSAR, FeeBlockSize should be implemented as pre-compile variant. However, this configuration parameter is implemented as a post-build variant.

10.3.8.2 Limitations

The following are the limitations for the FEE driver.

FEE double-sector algorithm

The following need to be noted with respect to the various FEE functionalities:

- **NVM write request during on going QS erase**

If erase-suspend feature is enabled and if a QS block erase is on going, an NVM write request is not allowed. The API Fee_Write/ Fee_InvalidateBlock request made to a non-QS block will return E_NOT_OK in this situation. The reason is that the next NVM write could possibly trigger a GC and the erase as part of the GC cannot be handled by the hardware because a new erase cannot be triggered as there is an already suspended erase request (QS).

[cover parentID FEE={956375E0-AFCC-4ea5-A508-7385754F54BD}]

- **Behavior of Fee_17_CancelAll()**

An on going erase be it in NVM (during GC) or QS cannot be cancelled. The Fee_17_CancelAll() API has no effect when called immediately after API Fee_Init() before the execution of scheduled function (time-consuming operations are executed here). Therefore, it is advised to check the FEE module state and issue subsequent requests to FEE only after the module reaches the idle state.

[cover parentID FEE={4BCD67F5-4630-4a81-99AC-CE8DB2367320}]

- **Block Cycle Count overflow**

If Fee write cycle limit (FeeNumberOfWriteCycles) for the block is configured to zero then Fee writes are allowed even when the block cycle count rolls over the limit of 2^24.

[cover parentID FEE={04AF6E7C-1E1A-4373-BEEE-43BD0CF0D380}]

- **Cache build time**

When there are a large number of very small NVM blocks, the cache build can take a lot of time. The effect is more noticeable when the cache build is triggered on the next user read or write request. These requests could experience delayed service.

Example:

If project has following configuration

Device = TC39X (1 MB DFLASH)

NVM sector size = 512 KB

Number of NVM block = 21792 (block size is 1 byte each)

FeeGcRestart = FEE_GC_RESTART_WRITE

FeeBlockTypeConfigured = FEE_DOUBLE_SECTOR_DATA_ONLY

and all the block are written in NVM sector then execution of first request(read/write) after initialization of FEE will be delayed by 167.86 ms approximately.

FEE driver

[cover parentID FEE={B682DB86-FC22-4e54-B8CF-0D570A26C607}]

- **Check and hardening time**

The check and hardening process checks and hardens 4% of the area of flash memory allocated for QS data. In this area, if there are more pages that need hardening, then the time taken by the check and hardening process will increase, resulting in increased execution time peak.

Example :

If project has following configuration

Device = TC39X (1 MB DFLASH)

NVM sector = 4 KB

QS sector = 1024-8= 1016 KB

4% of QS sector = 40 KB (approximately)

FeeBlockTypeConfigured = FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA then the time taken by check and hardening process will 6.34 ms approximately.

[cover parentID FEE={B0F8B663-696C-4456-8690-B6195FB7E929}]

- **Handling more than 2 word line failures**

FEE is designed to handle up to two WL failures. After two WL failures, the Flash is considered to have gone bad and an illegal state notification is raised.

[cover parentID FEE={16A0FBC9-1BA5-4dbf-9EA0-55A858FA5163}]

- **Handling rare state block combinations**

State block combinations indicating (dirty valid, dirty erased), (valid, dirty valid) and (dirty erase, valid) are considered as rare cases. These situations can occur if the state blocks reside on a failed WL or due to aging. In these cases, repair is attempted and if repair fails, the state block is written to the next free WL and if this state block write fails, then illegal state notification is generated leading to loss of data. In these rare cases, FEE is not robust up to 2 WL failures.

Two state pages in different sectors getting corrupted to indicate a particular invalid state combination AND a repair failing AND a write to the next+1 WL also failing - is expected to be an extremely rare case that the FEE does not handle.

[cover parentID FEE={8CBC5576-1AE0-4f99-BB26-D8B156C08615}]

- **Handling user requests during check and hardening**

When the check and hardening process is in progress, user read and write requests will not be accepted if the FEE module state is MEMIF_BUSY. User read write requests will be accepted if the module state is MEMIF_BUSY_INTERNAL but will be serviced with delay. The request will be serviced after the check and hardening process completes. A high priority QS write may be requested after calling Fee_17_CancelAll API.

[cover parentID FEE={E9C7680C-4309-4e2e-84E6-969423B8C5DE}]

- **No notification for dropped blocks**

The garbage collection process copies the latest instance of the data blocks by referring to the information present in the cache. If while reading a block during the copy phase, an un-correctable ECC error is encountered, then the block is dropped and it is not copied to the other sector. No notification is provided. The GC continues with the remaining blocks.

This is a limitation mainly arising from the fact that notification interfaces cannot pass the block ID of the dropped block.

[cover parentID FEE={01D4D541-E4A4-445b-B6A1-1D2D5EA145FF}]

10.3.9 Unsupported hardware features

Not applicable for the FEE driver.

Fls_17_Dmu driver**11 Fls_17_Dmu driver****11.1 User information****11.1.1 Description**

The FLS driver offers well-defined configuration and standard services as per AUTOSAR for the initialization, read, write and erase of DFlash 0. Apart from this there are some non-AUTOSAR services provided as well for example Fls_17_Dmu_CompareWordsSync, Fls_17_Dmu_CancelNonEraseJobs, Fls_17_Dmu_VerifyErase, Fls_17_Dmu_VerifySectorErase, Fls_17_Dmu_GetNotifCaller and so on. User gets an encapsulated access to the underlying DFlash0 through the FLS driver. The scope of the FLS driver is limited only to the DFlash0 Bank. The module is delivered as Post-Build variant.

11.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

Fls_17_Dmu driver

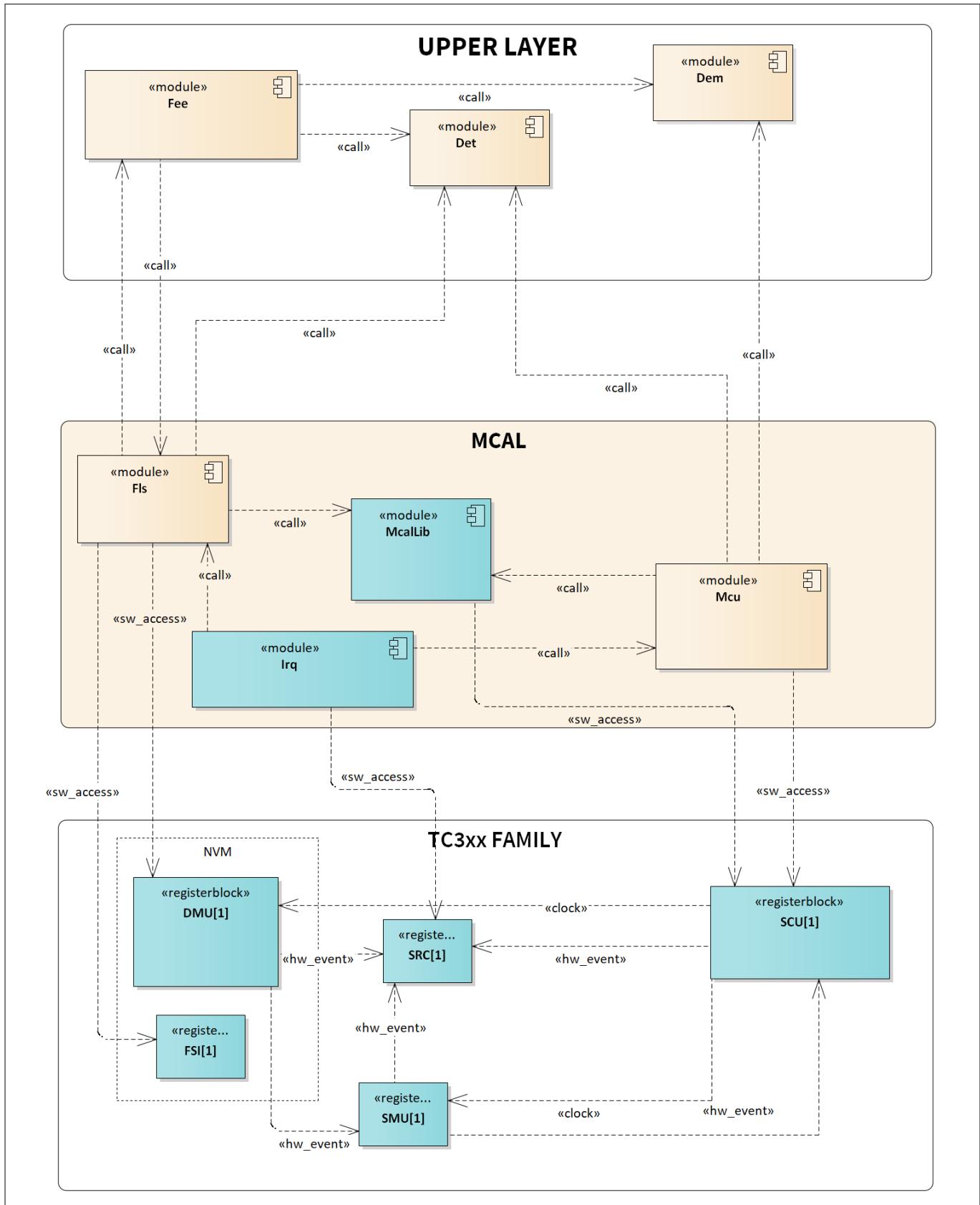


Figure 120 Mapping of hardware-software interfaces

Fls_17_Dmu driver

11.1.2.1 DMU-DFlash0: primary hardware peripheral

Hardware functional features

The FLS driver uses the DMU for operations such as read, write, suspend, resume, user content count(hardening) and erase DFlash0 memory. The key hardware functional features used by the driver are:

- Single ended sensing mode support for DFlash 0
- Writing and erasing DFlash 0:
 - 8 bytes page programming and 32 bytes burst programming
 - Erase by multi-sector erase commands
- Suspend, resume for erase operation
- Interrupt service requests for end of busy (EOBM bit) for erase and write operations in hardware

The unsupported features of the DMU are:

- Compliment sensing mode for DFlash 0
- ECC error reporting to safety management unit (SMU)
- Suspend, resume for write operation

Users of the hardware

The FLSLOADER and FLS drivers utilize the DMU IP module. FLS is used during runtime and FLSLOADER is used during the boot. Hence, the access to the DMU registers is not concurrent

Hardware diagnostic features

- The ECC is used for error detection. Dynamic correction of single, double and triple-bit errors and detection of quad-bit errors
- The SMU alarms configured for the DMU are not monitored by the FLS driver

Hardware events

The following hardware events notified by SFR flags are used in FLS driver:

- Error flags are raised upon occurrence of errors in programming, erasing, reading or erase suspend / resume operation
- Erase verify error (EVER): This flag is set by the erase commands when there is an erase verification error
- Program verify error (PVER): This flag is set by the program commands when there is a program verification error
- Protection error (PROER): This flag is set by the hardware when write or erase command executed on protected memory section
- Operation Error (OPER): This flag is set by the hardware when Flash standard interface (FSI) encounters any error
- Sequence Error (SQER): This flag is set by the hardware when improper DMU command sequences are executed
- End of busy(EOBM): This flag enables the interrupt to report the end of erase and program operations

11.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The FLS driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSRI, fFSI and fSPB clock signals for functioning. The system clock is set up through the MCU driver. It is mandatory for the user to set up an appropriate system clock.

Users of the hardware

Fls_17_Dmu driver

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the FLS driver.

Hardware events

Hardware events from the SCU are not used by the FLS driver.

11.1.2.3 SRC: dependent hardware peripherals

Hardware functional features

The FLS driver depends on the interrupt router for raising an interrupt to the CPU based on the end of busy event, which indicates the end or finish of the ongoing erase or write job in the HW.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software. Interrupt mode is available when the FLS is used without the Infineon FEE. Infineon FEE supports only the polling mode, therefore, the interrupt mode is not available with the Infineon FEE.

Hardware diagnostic features

The SMU alarms configured for interrupt router are not monitored by the FLS driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU. The FLS driver provides interrupt handlers as software interfaces, which must be invoked from the ISR. The following hardware events/interrupts are notified for DMU DFlash0:

- Programming completion through end of busy (EOB)
- Erase completion through end of busy (EOB)

Fls_17_Dmu driver

11.1.3 File structure

11.1.3.1 C File Structure

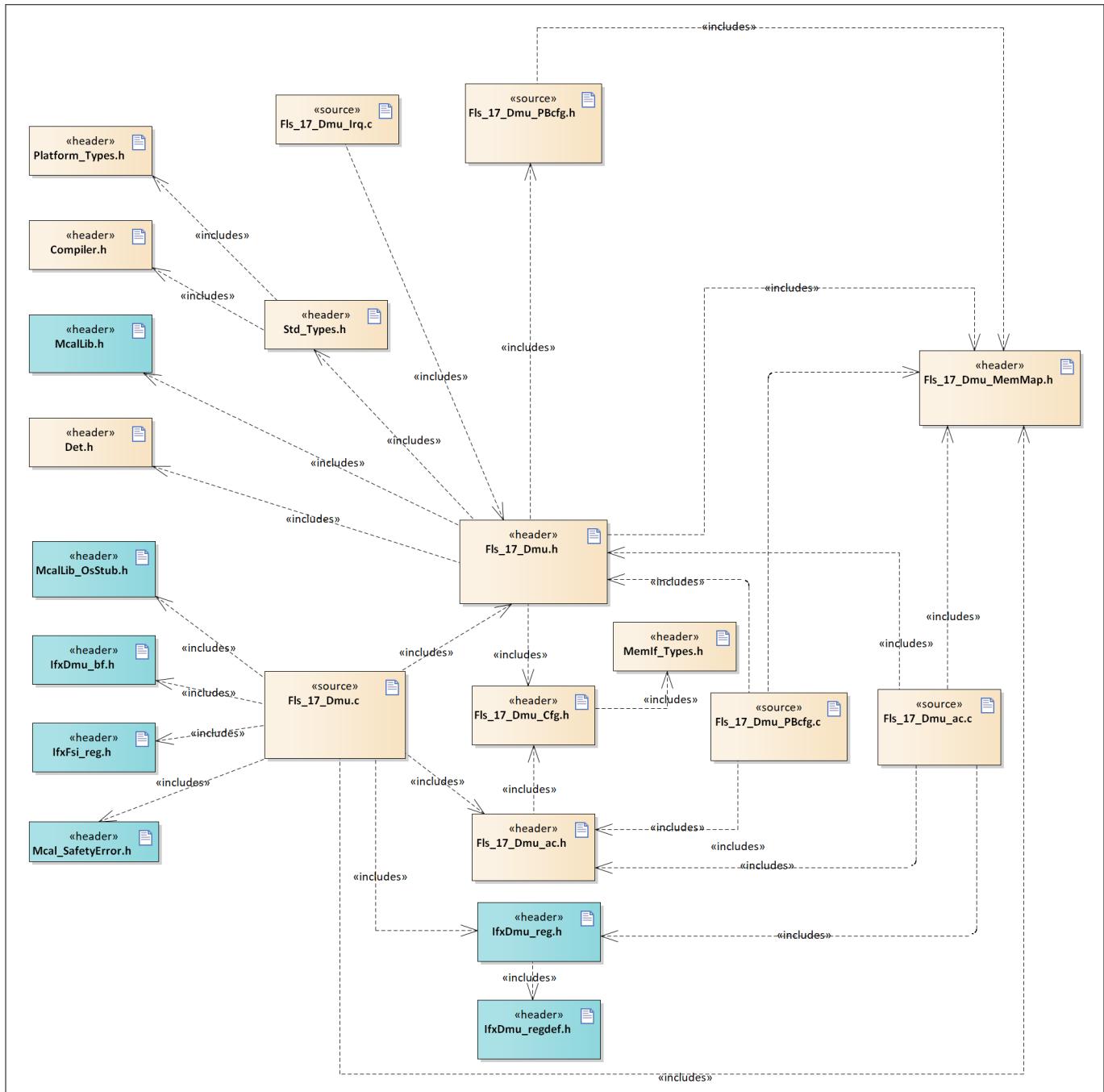


Figure 121 C File Structure

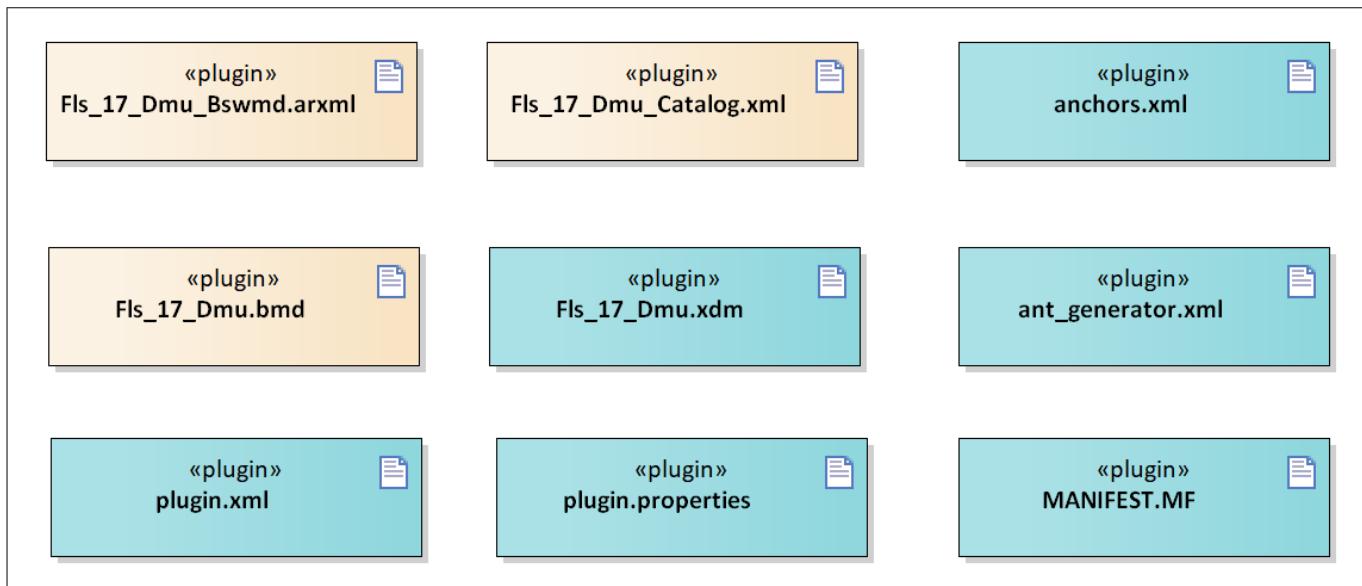
Table 695 C File Structure

File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Det.h	Provides the exported interfaces of Development Error Tracer

Fls_17_Dmu driver
Table 695 C File Structure (continued)

File name	Description
Fls_17_Dmu.c	This file contains functionality of the FLS driver. Version checks are also done in this file.
Fls_17_Dmu.h	This header file exports macros, type definitions, interrupt service routine and function prototypes for the Flash driver
Fls_17_Dmu_Cfg.h	Contains driver pre-compile configuration parameters Contain definitions for all pre-compile time configuration parameters defined as pre-processor directive which are specified for BSW module
Fls_17_Dmu_Irq.c	Interrupt handler file for FLS
Fls_17_Dmu_MemMap.h	File containing the memory section definitions used by the FLS driver
Fls_17_Dmu_PBcfg.c	Contains driver post-build configuration parameters
Fls_17_Dmu_PBcfg.h	File (generated) containing declaration of the post-build configuration data structures
Fls_17_Dmu_ac.c	Command cycles for Flash operations
Fls_17_Dmu_ac.h	Header file for macros used by Fls_17_Dmu_ac.c
IfxDmu_bf.h	SFR header file for Dmu
IfxDmu_reg.h	SFR header file for Dmu
IfxDmu_RegDef.h	SFR header file for Dmu
IfxFsi_Reg.h	SFR header file for FSI
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore. This shall be included by other drivers to call OS APIs.
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
MemIf_Types.h	Header file containing the type declaration of MemIf
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

11.1.3.2 Code Generator Plugin Files

Fls_17_Dmu driver

Figure 122 Code Generator Plugin Files
Table 696 Code Generator Plugin Files

File name	Description
Fls_17_Dmu.bmd	AUTOSAR format XML data model schema file (for each device)
Fls_17_Dmu.xdm	Tresos format XML data model schema file
Fls_17_Dmu_Bswmd.arxml	AUTOSAR format module description file
Fls_17_Dmu_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the meta-data for FLS driver
anchors.xml	Tresos anchors support file for the FLS driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point.
plugin.properties	Tresos plugin support file for the FLS driver
plugin.xml	Tresos plugin support file for the FLS driver

11.1.4 Integration hints

This section lists the key points that an integrator or user of the FLS driver must consider.

11.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the FLS driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of the MCAL, the EcuM is used for initialization and de-initialization of the

Fls_17_Dmu driver

software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the file `Fls_17_Dmu_MemMap.h`.

The `Fls_17_Dmu_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

```
#if defined FLS_17_DMU_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
/*User pragma here*/
#define FLS_17_DMU_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined FLS_17_DMU_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
/*User pragma here*/
#define FLS_17_DMU_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined FLS_17_DMU_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
/*User pragma here*/
#define FLS_17_DMU_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
#define MEMMAP_ERROR
#elif defined FLS_17_DMU_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
/*User pragma here*/
#define FLS_17_DMU_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
#define MEMMAP_ERROR
#elif defined FLS_17_DMU_START_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
/*User pragma here*/
#define FLS_17_DMU_START_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined FLS_17_DMU_STOP_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
/*User pragma here*/
#define FLS_17_DMU_STOP_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined FLS_17_DMU_START_SEC_CODE_ASIL_B_LOCAL
/*User pragma here*/
#define FLS_17_DMU_START_SEC_CODE_ASIL_B_LOCAL
#define MEMMAP_ERROR
#elif defined FLS_17_DMU_STOP_SEC_CODE_ASIL_B_LOCAL
/*User pragma here*/
#define FLS_17_DMU_STOP_SEC_CODE_ASIL_B_LOCAL
#define MEMMAP_ERROR
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW. The FLS driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the FLS driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

Fls_17_Dmu driver

The Det.h and Det.c files are provided in the MCAL package as a stub code and need to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is not required for the integration of FLS driver.

- **SchM**

The SchM is not required for the integration of FLS driver.

- **Safety error**

The FLS driver will report all the detected safety errors through the Mcal_ReportSafetyError() API.

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The Mcal_ReportSafetyError() API is provided in the Mcal_SafetyError.c and Mcal_SafetyError.h files as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The FLS driver does not implement any notifications. However, the FLS driver reports the job end and error through notification function. These notification functions can be configured by the user in the EB Tresos.

- **Operating system(OS)**

The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application. The OS files provided by MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

11.1.4.2 Multicore and Resource Manager

The FLS driver does not support execution on multiple cores.

11.1.4.3 MCU support

The FLS driver is dependent on the MCU driver for the clock configuration. The initialization of the FLS driver must be started only after completing the MCU initialization.

11.1.4.4 Port support

The FLS driver does not use any services provided by the PORT driver.

11.1.4.5 DMA support

The FLS driver does not use any services provided by the DMA driver.

11.1.4.6 Interrupt connections

The following events can trigger an interrupt service request to the Interrupt Router (IR)

- End of BUSY(EOBM): if DMU_HF_EER.EOBM = 1B and one of the DMU_HF_STATUS flags D0BUSY, D1BUSY or PFlash flags transitions from 1 to 0 then an interrupt service request is triggered (for example wake-up, erase sequences or program sequences)
- Operation Error (OPER): if DMU_HF_EER.OPERM = 1B and DMU_HF_ERRSR.OPER flag is set
- Protection Error (PROER): if DMU_HF_EER.PROERM = 1B and DMU_HF_ERRSR.PROER flag is set

Fls_17_Dmu driver

- Sequence Error (SQER): if DMU_HF_EER.SQERM = 1B and DMU_HF_ERRSR.SQER flag is set
- Program Verify Error (PVER): if DMU_HF_EER.PVERM = 1B and DMU_HF_ERRSR.PVER flag is set
- Erase Verify Error (EVER): if DMU_HF_EER.EVERM = 1B and DMU_HF_ERRSR.EVER flag is set

The event that triggered the interrupt can be determined from the DMU_HF_STATUS and DMU_HF_ERRSR registers. An interrupt event must be triggered when the event appears again and the corresponding status flag is still set. The FLS driver enables and uses only EOBM interrupt. Other interrupt mentioned are not used by FLS driver. End of BUSY interrupts are only generated after completion of start-up. The following diagram depicts the interrupt connections of DMU data Flash:

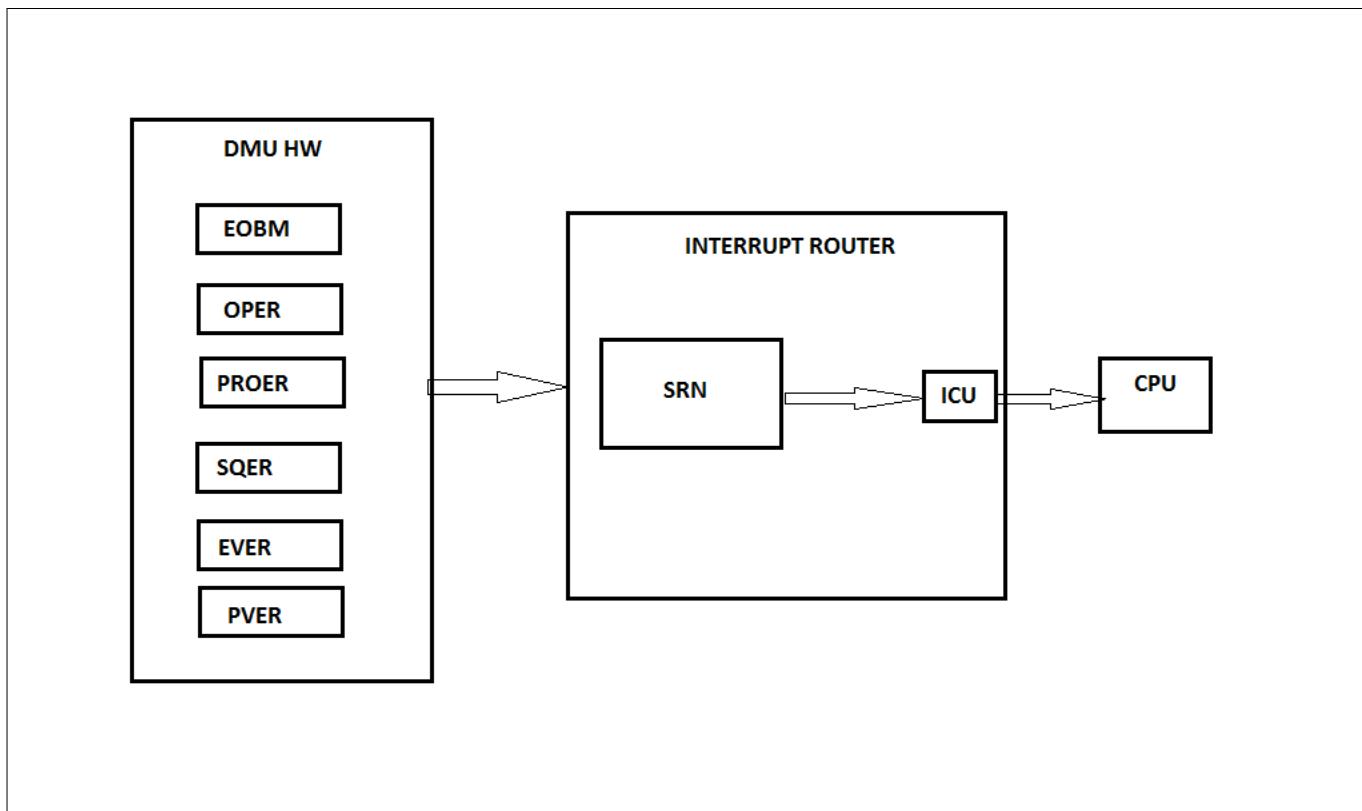


Figure 123 Interrupt mode

Invoking of interrupt handlers provided by the driver must be done by the user. If used with the Infineon FEE, user should ensure to use the FLS driver is in polled mode. A sample invocation of FLS driver interrupt handler is as follows:

```

ISR(DMUHOST_ISR)
/* Enable Global Interrupts */
{
    ENABLE();

    /* Call to Flash Interrupt function */
    Fls_17_Dmu_Isr();
}

```

Fls_17_Dmu driver

11.1.4.7 Example usage

This section explains an example usage scenario of the FLS driver for a nominal case. Applications usually adopt and modify the configuration and usage sequence as per their use-case.

Configuration of the driver

The configuration of the driver involves the following steps.

1. Configuration of the System Clock fSYS. This configuration is done using the MCU driver
2. Configuration of the FLS driver: The Flash driver is delivered as a post-build. The configuration of sectors should be done in the FlsSectorList container.

The FlsSector within the FlsSectorList container requires the following parameter: FlsNumberOfSectors(number of sectors), FlsSectorSize(sector size) and the FlsSectorStartAddress(Start/Begin address of the sector).

Note: This also has a dependency on whether the IFX FEE has been used or not.

Initialization of Flash driver

The following code snippet shows the steps involved in the initialization of the Flash driver.

```
#include "Std_Types.h"
#include "Mcu.h"
#include "Fls_17_Dmu.h"
#include "Irq.h"

extern const Mcu_ConfigType Mcu_Config;
extern const Fls_17_Dmu_ConfigType Fls_17_Dmu_Config;

/*Initialization of MCU*/
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Initialization of flash module */
Fls_17_Dmu_Init(&Fls_17_Dmu_Config);
#if FLS_USE_INTERRUPTS == ON
/* Configure FLS Module Interrupt Priority.
Use only for FLS INTERRUPT Mode. */
IrqDmu_Init();
#endif
```

Flash operations

Fls_17_Dmu_MainFunction() is the only scheduled function provided by the FLS driver. This function should be called periodically, so that it can process the jobs without hardware interrupt support. This API is a service for performing the processing of the Flash read, write, erase and compare jobs. The timeout monitoring of erase or write operations is done based on the Fls_17_Dmu_MainFunction() cycle time. Timeout monitoring is not done for read or compare as the read times are considerably small to monitor through Fls_17_Dmu_MainFunction() cycle time.

In case, the Fls_17_Dmu_MainFunction() is called in a continuous loop as - while(1) or at a rate faster than FlsCallCycle then unintentional timeouts may occur. For such cases, the user needs to disable the timeout monitoring using Fls_17_Dmu_ControlTimeoutDet() API. Once disabled, the timeouts will not be monitored at all. The safety and DET errors related to timeout will not be triggered as well.

Fls_17_Dmu driver

The code snippet shows an example of the steps involved in erasing, writing and reading a data Flash bank after initialization of the Flash.

```
#define FLS_17_DMU_NVMSECTOR0_STARTADDRESS (0xaf000000U)
#define FLS_17_DMU_NVMSECTOR_SIZE (0x20000U)
#define FLS_17_DMU_PAGE_SIZE (8U)
.....
uint8 Test_ProgData[2 * FLS_PAGESIZE]; /*write buffer*/
uint8 Test_ReadData[2 * FLS_PAGESIZE]; /*read buffer*/
...

/*Demo erase with timeout enabled*/
Std_ReturnType Fls_DemoErase(void)
{
/* Erase DFLASH BANK 0 */
ReturnVal =
Fls_17_Dmu_Erase(FLS_17_DMU_NVMSECTOR0_STARTADDRESS, FLS_17_DMU_NVMSECTOR_SIZE);

/*If erase job scheduled properly*/
if(ReturnVal == E_OK)
{
/* Poll till Erase completed */
while(Fls_17_Dmu_GetStatus() != MEMIF_IDLE)
{
Delay(); /*Delay should correspond to the FlsCallCycle value configured by
the user*/

Fls_17_Dmu_MainFunction();
}
if(Fls_17_Dmu_GetJobResult() != MEMIF_JOB_OK)
{
ReturnVal = E_NOT_OK;
}
}
}

/*Demo erase with timeout disabled as the Fls_17_Dmu_MainFunction() is
called at a rate faster than FlsCallCycle */
Std_ReturnType Fls_DemoErase(void)
{
Fls_17_Dmu_ControlTimeoutDet(0); /* 0 = disables timeout*/

/* Erase DFLASH BANK 0 */
ReturnVal =
Fls_17_Dmu_Erase(FLS_17_DMU_NVMSECTOR0_STARTADDRESS, FLS_17_DMU_NVMSECTOR_SIZE);

/*If erase job scheduled properly*/
if(ReturnVal == E_OK)
{
/* Poll till Erase completed */
while(Fls_17_Dmu_GetStatus() != MEMIF_IDLE)
```

Fls_17_Dmu driver

```

{
  Fls_17_Dmu_MainFunction();
}
if(Fls_17_Dmu_GetJobResult() != MEMIF_JOB_OK)
{
  ReturnVal = E_NOT_OK;
}
}

/*Demo write with timeout enabled*/
Std_ReturnType Fls_DemoWrite(void)
{
/* Assuming the write bufferTest_ProgData is already filled with some data
Write first 2 pages of DFLASH BANK 0.*/
  ReturnVal = Fls_17_Dmu_Write(Test_ProgData, FLS_BANK0_ADDR,
(2*FLS_PAGESIZE));

/*If the write job scheduled properly*/
if(ReturnVal == E_OK)
{
/* Poll till Erase completed */
  while(Fls_17_Dmu_GetStatus() != MEMIF_IDLE)
  {
    Delay(); /*Delay should correspond to the FlsCallCycle value configured by
the user*/
    Fls_17_Dmu_MainFunction();
  }
  if(Fls_17_Dmu_GetJobResult() != MEMIF_JOB_OK)
  {
    ReturnVal = E_NOT_OK;
  }
}
}

/*Demo write with timeout disabled as the Fls_17_Dmu_MainFunction() is
called at a rate faster than FlsCallCycle */
Std_ReturnType Fls_DemoWrite(void)
{
  Fls_17_Dmu_ControlTimeoutDet(0); /* 0 = disables timeout*/

/* Assuming the write bufferTest_ProgData is already filled with some data
Write first 2 pages of DFLASH BANK 0.*/
  ReturnVal = Fls_17_Dmu_Write(Test_ProgData, FLS_BANK0_ADDR,
(2*FLS_PAGESIZE));

/*If the write job scheduled properly*/
if(ReturnVal == E_OK)
{
/* Poll till write completed */
  while(Fls_17_Dmu_GetStatus() != MEMIF_IDLE)
}
}

```

Fls_17_Dmu driver

```
{  
    Fls_17_Dmu_MainFunction();  
}  
  
if(Fls_17_Dmu_GetJobResult() != MEMIF_JOB_OK)  
{  
    ReturnVal = E_NOT_OK;  
}  
}  
}  
  
Std_ReturnType Fls_DemoRead(void)  
{  
/* Read the first two pages */  
    ReturnVal =  
Fls_17_Dmu_Read(FLS_17_DMU_NVMSECTOR0_STARTADDRESS, Test_ReadData, (2 *  
FLS_PAGESIZE));  
  
    if(ReturnVal == E_OK)  
    {  
        while(Fls_17_Dmu_GetStatus() != MEMIF_IDLE)  
        {  
/* Wait till Write is completed */  
        Fls_17_Dmu_MainFunction();  
        }  
  
        if(Fls_17_Dmu_GetJobResult() != MEMIF_JOB_OK)  
        {  
            ReturnVal = E_NOT_OK;  
        }  
    }  
}
```

11.1.5 Key architectural considerations

There are no key architectural considerations for the driver.

Fls_17_Dmu driver

11.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Check for initialization**

The integrator shall ensure that proper initialization is done by calling the Fls_17_Dmu_Init() API before invoking any other service of the FLS driver.

[cover parentID FLS={78C52790-FD02-4374-ABC5-1E94933BAAAA}]

- **FLS initialization and Initcheck**

The integrator shall verify the correctness of initialization by calling the Fls_17_Dmu_InitCheck() API after the initialization is completed by the Fls_17_Dmu_Init() API. The Fls_17_Dmu_InitCheck() API checks if the initialized fixed global SFRs and fixed global variables of the FLS driver are initialized according to the configuration.

[cover parentID FLS={4E3B5CD0-694B-410c-A6B1-EDEAE53603CB}]

- **Working of suspend in standalone mode**

When FLS driver is used in standalone mode, the Fls_17_Dmu_SuspendErase() API shall be invoked by the application only when the previous job requested was an erase operation. This is to ensure that any operation other than erase is not suspended unintentionally.

[cover parentID FLS={F516B301-0F41-4864-B0E7-F92DAABC0EEA}]

- **Clock set-up**

Clocks are not set up by the FLS driver. The integrator shall ensure that the clocks needed for the flash operations on DFLASH0 are correctly set up using the MCU driver.

[cover parentID FLS={F34583B5-3E53-4bf2-8CCC-E64FC399B03B}]

- **Non-reentrant APIs**

The FLS driver's APIs are non-reentrant and therefore, the integrator shall ensure that multiple invocation of the FLS API(s) does not occur from different contexts, threads or cores.

[cover parentID FLS={71BD2EA3-E26F-44e3-ADCB-C2F0D64080D2}]

- **Non-usage of DFlash1**

When the FLS driver is being used for operations on the DFlash0, the integrator shall ensure that the DFlash1 is not used independently by any other driver, except for the HSM operations.

[cover parentID FLS={8CC1F5A8-581B-4c82-8364-12E90AF1E1DA}]

- **Using FLS for DFlash1 operation**

The integrator shall not use the FLS driver to perform operations on the DFlash1 hardware.

[cover parentID FLS={8D39DF0E-A919-4762-838D-4B9E9A90650C}]

- **ADER and bus access error behavior**

For bus access monitoring over SRI, the following errors are reported: - SRI access address phase error: If an ECC error occurs during the address phase of an SRI access, then the DMU_HF_ERRSR.ADER bit will be set and an error will be signaled to the SMU. The SRI access will terminate with an error. This error shall not be handled in the FLS driver and shall be handled by the user. - SRI access write data phase error: If an ECC error occurs on the data phase of an SRI write access, the an error will be signaled to the SMU. This error shall not be handled in the FLS driver and shall be handled by the user.

[cover parentID FLS={D5F895FF-AD5F-4337-88E3-B5FC8116ADFF}]

- **Write address**

The integrator shall ensure the correctness of the TargetAddress for write operation and also ensure that this address is not protected against writes by the Flash driver.

[cover parentID FLS={4E1CFF64-D76E-440a-8B68-1EC5C9E9B28E}]

- **Access to FLS SFRs from CPU core**

Fls_17_Dmu driver

Integrator shall ensure that the FLS driver is invoked from the CPU core that has access to the FLS SFR(s).

[cover parentID FLS={8120AD04-68B1-4eff-AE48-AD14FD6CCD14}]

- **Correctness of config pointer**

The user shall ensure that the config pointer passed is correct.

[cover parentID FLS={95C5FF4A-CBB7-4c18-A117-B754402C4D2C}]

- **Correctness of DFlash0 size configuration**

The integrator shall ensure that the total size of DFlash0 in the hardware is greater than or equal to the size of the data flash (DFlash0) mentioned in the configuration.

[cover parentID FLS={EF72308E-CE84-46eb-9B83-D79951DB6D74}]

- **Invocation of Fls_17_Dmu_GetNotifCaller() API**

The integrator shall ensure that Fls_17_Dmu_GetNotifCaller() is called only from inside the callback notification functions invoked by the FLS. The Fls_17_Dmu_GetNotifCaller() is needed to identify the notification so that the caller can take appropriate action.

[cover parentID FLS={BE3D0479-4FB5-48e3-B995-10FE8AC2E49B}]

- **No multicore support**

Integrator shall ensure that all the FLS services are executed from one core only. The FLS does not support multicore capability.

[cover parentID FLS={49337170-0313-49f8-91CC-EE972E4A91FA}]

- **Precaution during read operation**

The integrator shall ensure that the source address given for read is not protected against reads.

[cover parentID FLS={F30D701D-9250-4fa0-A700-C8AB627D30A5}]

Fls_17_Dmu driver

11.3 Reference information

11.3.1 Configuration interfaces

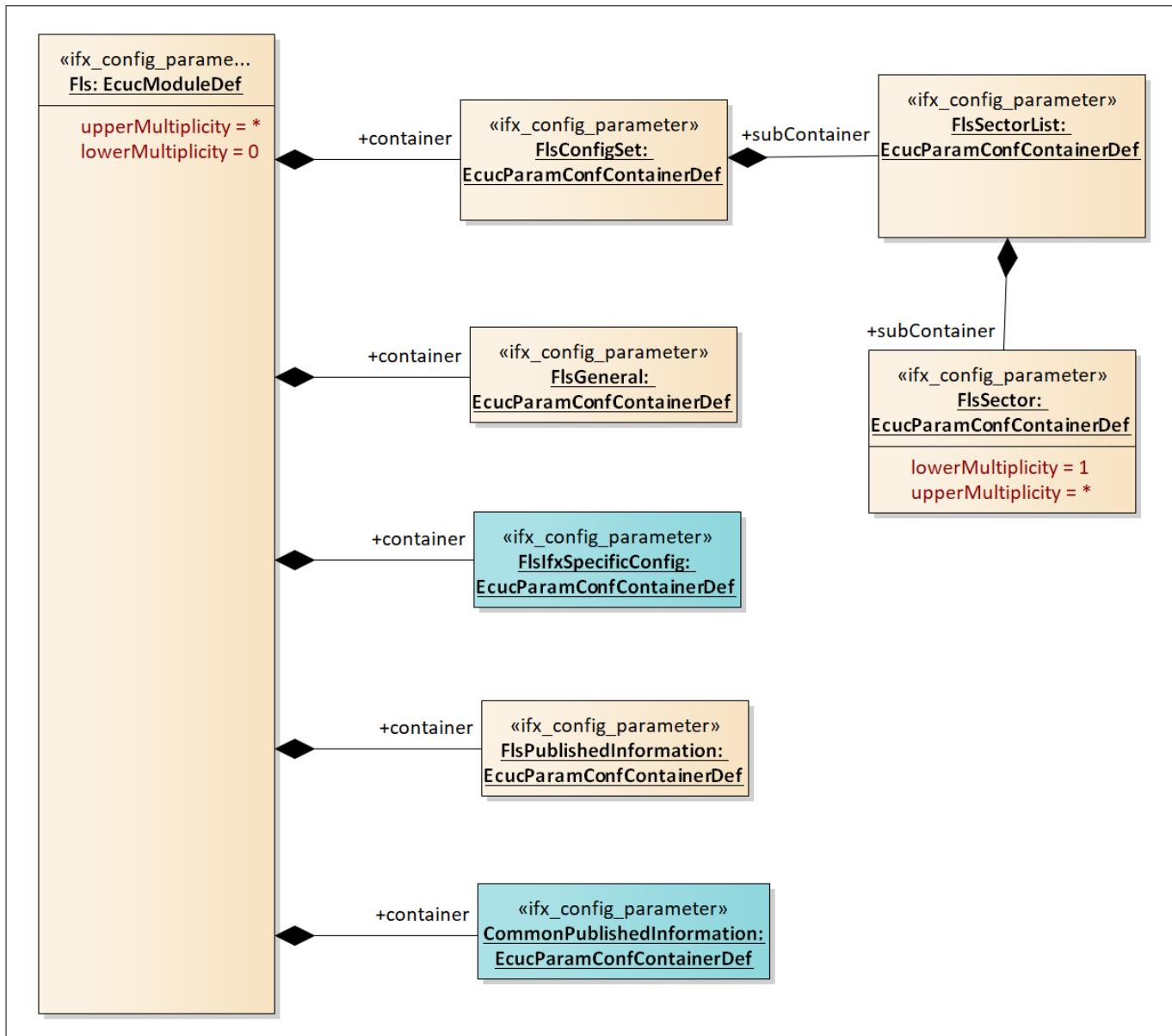


Figure 124 Container hierarchy along with their configuration parameters

11.3.1.1 Container: CommonPublishedInformation

This section describes the information about the module published by the FLS driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

Fls_17_Dmu driver**11.3.1.1.1 ArMajorVersion****Table 697 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Major version number of AUTOSAR specification on which the driver implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.1.2 ArMinorVersion**Table 698 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	Minor version number of AUTOSAR specification on which the driver implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.1.3 ArPatchVersion**Table 699 Specification for ArPatchVersion**

Name	ArPatchVersion	
Description	Patch version number of AUTOSAR specification on which the driver implementation is based on.	

Fls_17_Dmu driver**Table 699 Specification for ArPatchVersion (continued)**

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.1.4 ModuleId**Table 700 Specification for ModuleId**

Name	ModuleId		
Description	Provides the module ID of the flash driver module ID as described by AUTOSAR : Wp1.1.2 Basic Software Module List		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	92		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.1.5 Release**Table 701 Specification for Release**

Name	Release		
Description	Indicates the AURIX™ device derivative used for the implementation.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per the configuration		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Fls_17_Dmu driver
Table 701 Specification for Release (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.1.6 SwMajorVersion
Table 702 Specification for SwMajorVersion

Name	SwMajorVersion		
Description	Major version number of the vendor specific implementation of the driver. The numbering is vendor specific. The MAJOR_VERSION is incremented if the driver is not downwards compatible any more (for example, existing API changed)		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver version.		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.1.7 SwMinorVersion
Table 703 Specification for SwMinorVersion

Name	SwMinorVersion		
Description	Minor version number of the vendor specific implementation of the driver. The numbering is vendor specific. MINOR_VERSION is incremented if the driver is still downwards compatible (example new functionality added)		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver version.		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Fls_17_Dmu driver
Table 703 Specification for SwMinorVersion (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.1.8 SwPatchVersion
Table 704 Specification for SwPatchVersion

Name	SwPatchVersion		
Description	Patch level version number of the vendor specific implementation of the driver. The numbering is vendor specific. The PATCH_VERSION is incremented if the driver is still upwards and downwards compatible (for example, bug fix).		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	As per driver version.		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.1.9 VendorApiInfix
Table 705 Specification for VendorApiInfix

Name	VendorApiInfix		
Description	VendorApiInfix of the IFX		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	Dmu		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

Fls_17_Dmu driver
Table 705 Specification for VendorApiInfix (continued)

Dependency	-
-------------------	---

11.3.1.1.10 VendorId
Table 706 Specification for VendorId

Name	VendorId		
Description	Vendor Id of the supplier		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.2 Container: Fls

This container holds the configuration of the FLS (internal or external) driver module. The multiplicity describes the number of Flash drivers present, therefore, there will be one container for each Flash driver in the ECUC template. When no Flash driver is present, the multiplicity is 0.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: -

11.3.1.3 Container: FlsConfigSet

This container is for the runtime configuration parameters of the Flash driver. Implementation Type:
Fls_17_Dmu_ConfigType.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

11.3.1.3.1 FlsAcErase
Table 707 Specification for FlsAcErase

Name	FlsAcErase
Description	<p>FlsAcLoadOnJobStart = TRUE This parameter specifies the RAM address from which the erase command cycles are copied and executed.</p> <p>FlsAcLoadOnJobStart = FALSE</p>

Fls_17_Dmu driver
Table 707 Specification for FlsAcErase (continued)

	The value entered here is ignored and the location of the erase command cycles in the PFlash is used. Note: This parameter is not used and hence not supported. In TC3xx Pflash & Dflash can be read in parallel and hence there is no need to load Dflash access code into RAM		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.3.2 FlsAcWrite
Table 708 Specification for FlsAcWrite

Name	FlsAcWrite		
Description	FlsAcLoadOnJobStart = TRUE This parameter specifies the RAM address to which the write command cycles are copied and executed. FlsAcLoadOnJobStart = FALSE The value entered here is ignored and the location of the write command cycles in the program flash is used. Note: This parameter is not used and hence not supported. In TC3xx Pflash & Dflash can be read in parallel and hence there is no need to load Dflash access code into RAM.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Fls_17_Dmu driver
11.3.1.3.3 FlsCallCycle
Table 709 Specification for FlsCallCycle

Name	FlsCallCycle		
Description	Cycle time of calls of the main function for the Flash driver(in seconds). This parameter is used in the timeout monitoring for the write/erase jobs. A value of 10 ms is selected as default assuming that this duration would be a reasonable frequency to check the status of scheduled user jobs.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0001 - 1		
Default value	0.01		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.3.4 FlsDefaultMode
Table 710 Specification for FlsDefaultMode

Name	FlsDefaultMode		
Description	This parameter is the default read mode of the data flash(DFLASH0) on the device after initialization. The default value has been selected assuming that a read in MEMIF_MODE_SLOW mode(32 bytes) would be reasonable for the user.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MEMIF_MODE_FAST: driver is working in the fast(burst) mode. MEMIF_MODE_SLOW: driver is working in the slow mode.		
Default value	MEMIF_MODE_SLOW		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Fls_17_Dmu driver**11.3.1.3.5 FlsEraseVerifyErrNotif****Table 711 Specification for FlsEraseVerifyErrNotif**

Name	FlsEraseVerifyErrNotif		
Description	User defined notification function pointer of type 'void fn_name (void)'. This notification function is called by the FLS driver for giving notification of the EVER bit error during the erase job. If the FlsEraseVerifyErrNotif is configured as NULL, the notification functions are not called. Note: This parameter is valid only if the Infineon FEE is used and should be configured as Fee_17_JobEraseErrorNotification. The Fee_17_JobEraseErrorNotification is the name of the Infineon FEE erase verification error notification function and therefore has been given as the default value. If the Infineon FEE is not used, then this parameter is not supported.		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	Fee_17_JobEraseErrorNotification		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	IFX	Scope	LOCAL
Dependency	FlsIfxFeeUse		

11.3.1.3.6 FlsJobEndNotification**Table 712 Specification for FlsJobEndNotification**

Name	FlsJobEndNotification		
Description	User defined notification function pointer of type void fn_name (void). This notification function is called by the FLS driver on successful completion of the job. If the FlsJobEndNotification is configured as NULL, the notification functions are not called. If the Infineon FEE is used, it should be configured as Fee_JobEndNotification. Assuming the usage is with Infineon FEE, the default value has been set as Fee_JobEndNotification. Note: The integrator or user has to verify the function address if numerical value is provided.		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	Fee_JobEndNotification		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE

Fls_17_Dmu driver**Table 712 Specification for FlsJobEndNotification (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.3.7 FlsJobErrorNotification**Table 713 Specification for FlsJobErrorNotification**

Name	FlsJobErrorNotification		
Description	User defined notification function pointer of type void fn_name (void). This notification function is called by the FLS driver on cancellation of the job or a failure in executing the job. If the FlsJobErrorNotification is configured as NULL, the notification functions is not called. If the Infineon FEE is used, it should be configured as Fee_JobErrorNotification. Assuming the usage with Infineon FEE, the default value has been given as Fee_JobErrorNotification. Note: The integrator/user has to verify the function address if numerical value is provided.		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	Fee_JobErrorNotification		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.3.8 FlsMaxReadFastMode**Table 714 Specification for FlsMaxReadFastMode**

Name	FlsMaxReadFastMode		
Description	The maximum number of bytes to read in one cycle of the job processing of the Flash driver in fast mode. This configuration of this parameter will affect Compare and Blank check operation as well. Note: The value configured for FlsMaxReadFastMode should be more than the value configured for FlsMaxReadNormalMode. Therefore, the default value has been set assuming		

Fls_17_Dmu driver**Table 714 Specification for FlsMaxReadFastMode (continued)**

	a word aligned read address from data flash(DFLASH0) and more than the value of FlsMaxReadNormalMode.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - FlsTotalSize		
Default value	64		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	FlsTotalSize		

11.3.1.3.9 FlsMaxReadNormalMode**Table 715 Specification for FlsMaxReadNormalMode**

Name	FlsMaxReadNormalMode		
Description	<p>The maximum number of bytes to read in one cycle of the job processing of the Flash driver in normal mode. This configuration of this parameter will affect Compare and Blank check operation as well.</p> <p>The default value has been given assuming the read address from DFLASH0 is word aligned and is less than the value of FlsMaxReadFastMode.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - FlsTotalSize		
Default value	32		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	FlsTotalSize		

11.3.1.3.10 FlsMaxWriteFastMode**Table 716 Specification for FlsMaxWriteFastMode**

Name	FlsMaxWriteFastMode
-------------	---------------------

Fls_17_Dmu driver**Table 716 Specification for FlsMaxWriteFastMode (continued)**

Description	The maximum number of bytes to write in one cycle of the job processing of the Flash driver. In Aurix the write can be either page write (1 page = 8 bytes) or burst write (4 pages = 32 bytes). This parameter is not supported as the burst mode for write is used by default and if the length of data to be written is less than or equal to 24 bytes (that is less than or equal to 4 pages) then page write is used for these remaining bytes.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	32 - 32		
Default value	32		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.3.11 FlsMaxWriteNormalMode**Table 717 Specification for FlsMaxWriteNormalMode**

Name	FlsMaxWriteNormalMode		
Description	The maximum number of bytes to write in one cycle of the job processing of the Flash driver. In Aurix the write can be either page write (1 page = 8 bytes) or burst write (4 pages = 32 bytes). This parameter is not supported as the burst mode for write is used by default and if the length of data to be written is less than or equal to 24 bytes (that is less than or equal to 4 pages) then page write is used for these remaining bytes.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	32 - 32		
Default value	32		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Fls_17_Dmu driver**11.3.1.3.12 FlsProgVerifyErrNotif****Table 718 Specification for FlsProgVerifyErrNotif**

Name	FlsProgVerifyErrNotif		
Description	User defined notification function pointer of type void fn_name (void). This notification function is called by the FLS Driver for giving notification of the PVER error during write/programming job. If the FlsProgVerifyErrNotif is configured as NULL then the notification functions is not called. Note: This parameter is valid only if IFX FEE is used and should be configured as Fee_17_JobProgErrorNotification. The Fee_17_JobProgErrorNotification() is the Infineon FEE programming error notification and therefor has been given as the default value. If Infineon FEE is not used, then this parameter is not supported.		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	Fee_17_JobProgErrorNotification		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	IFX	Scope	LOCAL
Dependency	FlsIfxFeeUse		

11.3.1.3.13 FlsProtection**Table 719 Specification for FlsProtection**

Name	FlsProtection		
Description	This parameter is not supported as the protection is best handled by the FlsLoader. This parameter is unused and hence disabled.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Fls_17_Dmu driver**11.3.1.3.14 FlsWaitStateErrorCorrection****Table 720 Specification for FlsWaitStateErrorCorrection**

Name	FlsWaitStateErrorCorrection		
Description	<p>Defines wait state configuration for error correction.</p> <p>Minimum value for the ECC cycles : Ceiling(tDFECC * fFSI)</p> <p>The wait cycles to be programmed in the DMU_HF_DWAIT register is ECC cycles - 1.</p> <p>For example, if the tDFECC = 20 ns, with fFSI = 100 MHz. The number of error correction cycles equals 2 therefore program values are:</p> <p>DMU_HF_DWAIT.ECC = 1</p> <p>So for error correction cycles of 2, the value to be entered here is 1.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	FLS_17_DMU_WAIT_STATE_ERRORCORRECTION0 - FLS_17_DMU_WAIT_STATE_ERRORCORRECTION7		
Default value	FLS_17_DMU_WAIT_STATE_ERRORCORRECTION1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.3.15 FlsWaitStateRead**Table 721 Specification for FlsWaitStateRead**

Name	FlsWaitStateRead		
Description	<p>Defines wait state configuration for read access.</p> <p>Minimum value for the DFlash0 read cycles : Ceiling (tDF * fFSI)</p> <p>The wait cycles to be programmed in the DMU_HF_DWAIT register is DFLASH read cycles - 1.</p> <p>For example, if the tDF = 100 ns and fFSI = 100 MHz. The number of DFlash read cycles equals 10, therefore program values are:</p> <p>DMU_HF_DWAIT.RFLASH = 9</p> <p>So for read cycles of 10, the value to be entered here is 9.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	FLS_17_DMU_WAIT_STATE_READACCESS0 - FLS_17_DMU_WAIT_STATE_READACCESS255		
Default value	FLS_17_DMU_WAIT_STATE_READACCESS9		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Fls_17_Dmu driver
Table 721 Specification for FlsWaitStateRead (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.4 Container: FlsExternalDriver

This container is present for external Flash drivers only. Internal Flash drivers do not use the parameter listed in this container, hence its multiplicity is 0 for internal drivers. Note: This is not supported as external drivers are not supported.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

11.3.1.4.1 FlsSpiReference
Table 722 Specification for FlsSpiReference

Name	FlsSpiReference		
Description	Reference to SPI sequence (required for external Flash drivers). Note: This is not supported as external drivers are not supported.		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5 Container: FlsGeneral

This container holds the for general parameters of the FLS driver. These parameters are always pre-compile.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

Fls_17_Dmu driver**11.3.1.5.1 FlsAcLoadOnJobStart****Table 723 Specification for FlsAcLoadOnJobStart**

Name	FlsAcLoadOnJobStart		
Description	<p>If this macro is enabled, then the erase access code is loaded in the RAM during Fls_17_Dmu_Erase() API call and unloaded after the completion or cancellation of the job.</p> <p>Similarly, the write access code is loaded in the RAM during the Fls_17_Dmu_Write() API call and unloaded after the completion or cancellation of the job.</p> <p>If this Macro is disabled then the write and erase access code of the FLS driver are executed from the program flash.</p> <p>Note: This parameter shall be non-editable. The FLS driver access code executes from program flash(PFlash). In TC3xx Pflash Dflash can be read in parallel and hence there is no need to load Dflash access code into RAM.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.2 FlsBaseAddress**Table 724 Specification for FlsBaseAddress**

Name	FlsBaseAddress		
Description	<p>The Flash memory start address (also see SWS_Fls_00208 and SWS_Fls_00209).</p> <p>This parameter defines the lower boundary for the read / write / erase/compare and blank check jobs.</p> <p>This parameter is fixed and not editable.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	Op== - Based on the target device		
Default value	0xAF000000		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Fls_17_Dmu driver**Table 724 Specification for FlsBaseAddress (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.3 FlsBlankCheckApi**Table 725 Specification for FlsBlankCheckApi**

Name	FlsBlankCheckApi		
Description	This parameter is used to enable/disable the Fls_17_Dmu_BankCheck() API.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.4 FlsCancelApi**Table 726 Specification for FlsCancelApi**

Name	FlsCancelApi		
Description	This parameter is used to enable/disable the Fls_17_Dmu_Cancel() API		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Fls_17_Dmu driver**Table 726 Specification for FlsCancelApi (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.5 FlsCompareApi**Table 727 Specification for FlsCompareApi**

Name	FlsCompareApi		
Description	<p>This parameter is used to enable/disable the Fls_17_Dmu_Compare() API.</p> <p>Note: The default value set as FALSE for the optional features to minimize the code size for execution.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.6 FlsDevErrorDetect**Table 728 Specification for FlsDevErrorDetect**

Name	FlsDevErrorDetect		
Description	<p>Pre-processor switch for enabling the development error detection and reporting.</p> <p>When this parameter is enabled, the parameters in each API are validated.</p> <p>Note: The default value is set as TRUE to ensure that the error detection is enabled and relevant issues are handled during product life cycle.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		

Fls_17_Dmu driver**Table 728 Specification for FlsDevErrorDetect (continued)**

Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.7 FlsDriverIndex**Table 729 Specification for FlsDriverIndex**

Name	FlsDriverIndex		
Description	This parameter is used to assign an index to the FLS driver. Note: The default value is set to minimum.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 254		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

11.3.1.5.8 FlsGetJobResultApi**Table 730 Specification for FlsGetJobResultApi**

Name	FlsGetJobResultApi		
Description	This parameter is used to enable/disable the Fls_17_Dmu_GetJobResult() API.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Fls_17_Dmu driver**Table 730 Specification for FlsGetJobResultApi (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.9 FlsGetStatusApi**Table 731 Specification for FlsGetStatusApi**

Name	FlsGetStatusApi		
Description	This parameter is used to enable/disable the Fls_17_Dmu_GetStatus() API.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.10 FlsIfxFeeUse**Table 732 Specification for FlsIfxFeeUse**

Name	FlsIfxFeeUse		
Description	This parameter is used to enable/disable the use of Infineon FEE specific APIs. The default value is set assuming that FLS driver is used with Infineon FEE.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Fls_17_Dmu driver**Table 732 Specification for FlsIfxFeeUse (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin		Scope	LOCAL
Dependency	-		

11.3.1.5.11 FlsInitApiMode**Table 733 Specification for FlsInitApiMode**

Name	FlsInitApiMode		
Description	<p>This parameter is used for configuring the 'User' or 'Supervisor' mode for initialization in the FLS driver.</p> <p>Note: By default access level of all the APIs is set to supervisor so that, there is no dependency on the OS functions to write into the access protected SFRs.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>FLS_17_DMU_MCAL_SUPERVISOR: mode used is SUPERVISOR</p> <p>FLS_17_DMU_MCAL_USER1: operating mode used is USER1</p>		
Default value	FLS_17_DMU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsRuntimeApiMode		

11.3.1.5.12 FlsInitCheckApi**Table 734 Specification for FlsInitCheckApi**

Name	FlsInitCheckApi		
Description	<p>Switch to enable the safety check for initialization using Fls_17_Dmu_InitCheck() API.</p> <p>Note: The default value is set to FALSE for the optional features to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		

Fls_17_Dmu driver**Table 734 Specification for FlsInitCheckApi (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.5.13 FlsRunTimeErrorDetect**Table 735 Specification for FlsRunTimeErrorDetect**

Name	FlsRunTimeErrorDetect		
Description	The activation of the runtime errors is configurable (ON / OFF) at the pre-compile time. Note: The default value is set as TRUE to ensure that the error detection is enabled and relevant issues are handled during product life cycle.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.5.14 FlsRuntimeApiMode**Table 736 Specification for FlsRuntimeApiMode**

Name	FlsRuntimeApiMode		
Description	This configuration parameter gives the mode in which the runtime API is used. Note: By default access level of all the APIs is set to supervisor so that, there is no dependency on the OS functions to write into the access protected SFRs.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	FLS_17_DMU_MCAL_SUPERVISOR: The mode used is SUPERVISOR FLS_17_DMU_MCAL_USER1: operating mode used is USER1		

Fls_17_Dmu driver**Table 736 Specification for FlsRuntimeApiMode (continued)**

Default value	FLS_17_DMU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.5.15 FlsSafetyEnable**Table 737 Specification for FlsSafetyEnable**

Name	FlsSafetyEnable		
Description	This parameter is used to enable/disable the safety notifications for the FLS module. Note: The default value is set to TRUE to ensure that the safety issues are addressed.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.5.16 FlsSetModeApi**Table 738 Specification for FlsSetModeApi**

Name	FlsSetModeApi		
Description	This parameter is used to enable/disable the Fls_17_Dmu_SetMode() API. Note: The default value is set to FALSE for the optional feature to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		

Fls_17_Dmu driver**Table 738 Specification for FlsSetModeApi (continued)**

Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.17 FlsTotalSize**Table 739 Specification for FlsTotalSize**

Name	FlsTotalSize		
Description	<p>This parameter is used to calculate the upper boundary for the read/write/erase/compare and blank check jobs.</p> <p>Entire DFlash 0 area is used only by FEE because the unused area (other than the area used for EEPROM emulation) in DFLASH0 data flash becomes unusable for any other purpose as it would incur too many disturbances from the cycled EEPROM area.</p> <p>If the FEE operates in the double sector mode only, then the minimum size should be 8kb, because the logical sector size of the DFLASH0 data flash is 4kb and FEE needs minimum 2 sectors (double sector algorithm), the minimum value that can be configured for this configuration parameter is limited to 8kb.</p> <p>If the FEE operates in quasi only mode then the minimum size would be 4kb as per the minimum logical block size of the DFLASH0 data flash.</p> <p>If the FEE operates in both double sector and quasi state, the minimum value to be used for quasi would be 4kb and the remaining would be divided into 2 sectors of equal size.</p> <p>Similarly, if the minimum size for double sector(8kb) is used, when both double sector and quasi has to operate, then the remaining area could be used for quasi, in multiples of 4kb.</p> <p>Note: While configuring this parameter, user has to take care of the total DFLASH0 size available on a variant.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	4096 - 1048576		
Default value	Based on Target Device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Fls_17_Dmu driver**11.3.1.5.18 FlsUseInterrupts****Table 740 Specification for FlsUseInterrupts**

Name	FlsUseInterrupts		
Description	Pre-processor switch for enabling the interrupts. Note: This parameter is disabled if the Infineon FEE is used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.5.19 FlsVersionInfoApi**Table 741 Specification for FlsVersionInfoApi**

Name	FlsVersionInfoApi		
Description	This parameter is used to enable/disable the Fls_17_Dmu_GetVersionInfo() API.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.6 Container: FlsIfxSpecificConfig

This container lists all the Infineon specific pre-compile configuration parameters

Post-Build Variant Multiplicity: -

Fls_17_Dmu driver

Multiplicity Configuration Class: -

11.3.1.6.1 FlsEraseSuspendTimeout

Table 742 Specification for FlsEraseSuspendTimeout

Name	FlsEraseSuspendTimeout		
Description	Timeout parameter for the erase suspend feature (number of loops).		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	12000 - 65535		
Default value	12000		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsUseEraseSuspend		

11.3.1.6.2 FlsIllegalStateNotification

Table 743 Specification for FlsIllegalStateNotification

Name	FlsIllegalStateNotification		
Description	This parameter is a pointer to a notification function, which is called when the FLS driver reaches an illegal state. The illegal state here signifies that the FLS driver is not able to proceed. No more FLS request is triggered. In such a case, system reset is recommended.		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Fls_17_Dmu driver
11.3.1.6.3 FlsStateVarStruct
Table 744 Specification for FlsStateVarStruct

Name	FlsStateVarStruct		
Description	This parameter is used to provide the name of the structure containing the entire global variables specific to the Flash driver.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	FlsStateVar		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.6.4 FlsUseEraseSuspend
Table 745 Specification for FlsUseEraseSuspend

Name	FlsUseEraseSuspend		
Description	Compile switch to enable or disable the FLS erase suspend and erase resume features. STD_ON : FLS suspend/resume feature for erase is enabled STD_OFF: FLS suspend/resume feature for erase is disabled		
Multiplicity	0..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	Post-Build
Origin	IFX	Scope	LOCAL
Dependency	-		

11.3.1.7 Container: FlsPublishedInformation

Additional published parameters not covered by CommonPublishedInformation container. Note that these parameters do not have any configuration class setting, because they are published information.

Fls_17_Dmu driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

11.3.1.7.1 FlsAcLocationErase
Table 746 Specification for FlsAcLocationErase

Name	FlsAcLocationErase		
Description	<p>Location of the erase command cycles. This parameter is not used, instead use the FlsAcErase parameter.</p> <p>Note : This parameter is not applicable as the flash driver access code executes from program flash. Therefore, this is not supported.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.7.2 FlsAcLocationWrite
Table 747 Specification for FlsAcLocationWrite

Name	FlsAcLocationWrite		
Description	<p>Location of the write command cycles. This parameter is not used, instead use the FlsAcWrite parameter.</p> <p>Note: This parameter is not relevant as flash driver access code executes from program flash. Therefore, this is not supported</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Fls_17_Dmu driver**11.3.1.7.3 FlsAcSizeErase****Table 748 Specification for FlsAcSizeErase**

Name	FlsAcSizeErase		
Description	Number of bytes in the RAM needed for the erase Flash access code. Note: This parameter is not relevant as the flash driver access code executes from program flash. Therefore, this is not supported.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.7.4 FlsAcSizeWrite**Table 749 Specification for FlsAcSizeWrite**

Name	FlsAcSizeWrite		
Description	Number of bytes in the RAM needed for the write Flash access code. Note: This parameter is not relevant as the Flash access code executes from program flash. Therefore, this is not supported.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.7.5 FlsEraseTime**Table 750 Specification for FlsEraseTime**

Name	FlsEraseTime
-------------	--------------

Fls_17_Dmu driver
Table 750 Specification for FlsEraseTime (continued)

Description	Maximum time to erase one logical sector in microseconds. The default value is given as per datasheet.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 4294967295.0		
Default value	1500000		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.7.6 FlsErasedValue
Table 751 Specification for FlsErasedValue

Name	FlsErasedValue		
Description	The contents of an erased Flash memory cell. The default value is selected as 0 as this is the value on DFLASH0 after erase.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.7.7 FlsExpectedHwId
Table 752 Specification for FlsExpectedHwId

Name	FlsExpectedHwId		
Description	Unique identifier of the hardware device that is expected by the driver (the device for which the driver has been implemented). Note: This parameter is not used as it is applicable only for external flash drivers.		
Multiplicity	1..1	Type	EcucStringParamDef

Fls_17_Dmu driver**Table 752 Specification for FlsExpectedHwId (continued)**

Range	String		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.7.8 FlsSpecifiedEraseCycles**Table 753 Specification for FlsSpecifiedEraseCycles**

Name	FlsSpecifiedEraseCycles		
Description	Number of erase cycles specified for the Flash device (usually given in the device data sheet). The default value is selected based on the datasheet.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	125000		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.7.9 FlsWriteTime**Table 754 Specification for FlsWriteTime**

Name	FlsWriteTime		
Description	Maximum time for one write operation, in microseconds, that is, burst write (32 bytes). The default value has been given based on the target parameter of the hardware DFLASH0 for burst write.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 4294967295.0		
Default value	140		

Fls_17_Dmu driver
Table 754 Specification for FlsWriteTime (continued)

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.8 Container: FlsSector

This container contains configuration description of a flashable sector.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

11.3.1.8.1 FlsNumberOfSectors
Table 755 Specification for FlsNumberOfSectors

Name	FlsNumberOfSectors		
Description	Number of continuous sectors with identical values for FlsSectorSize and FlsPageSize(in bytes). The FlsSectorStartAddress parameter denotes the start address of the first sector. Note: The maximum and the default value for this parameter is '2' as it is used with the double sector algorithm.		
Multiplicity	1..1	Type	EcuclIntegerParamDef
Range	1 - 2		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.8.2 FlsPageSize
Table 756 Specification for FlsPageSize

Name	FlsPageSize
Description	Size of one FLS age in bytes. This parameter is fixed, therefore, not configurable.

Fls_17_Dmu driver
Table 756 Specification for FlsPageSize (continued)

Multiplicity	1..1	Type	EcclIntegerParamDef
Range	8 - 8		
Default value	8		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

11.3.1.8.3 FlsSectorSize
Table 757 Specification for FlsSectorSize

Name	FlsSectorSize		
Description	Size of the FLS Sector (in bytes). For double sector data, this parameter will be the size of one of the sectors. If no quasi-static data is used, then the value of this parameter will typically be half of FlsTotalSize and should be in the multiple of 4 Kb. For quasi-static data this contains the quasi region and should be in the multiple of 4K bytes. If both double sector and quasi-static data are used then two containers should be used to specify the sector size appropriately such that the total size is justified. For example, the minimum size for quasi would be 4 Kb and the rest could be dedicated for using the double sector algorithm. For more details, refer to FlsNumberOfSectors.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	4096 - 1048576		
Default value	DFLASH0 total size divided by 2 (DFLASH0 total size varies)		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	FlsTotalSize		

11.3.1.8.4 FlsSectorStartaddress
Table 758 Specification for FlsSectorStartaddress

Name	FlsSectorStartaddress
-------------	-----------------------

Fls_17_Dmu driver
Table 758 Specification for FlsSectorStartaddress (continued)

Description	Start address of the DFlash0 sector. Implementation Type: Fls_17_Dmu_AddressType.		
Multiplicity	1..1	Type	EcuUIntegerParamDef
Range	0 - 1048576		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	FlsTotalSize		

11.3.1.9 Container: FlsSectorList

List of flashable sectors and pages.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

11.3.2 Functions - Type definitions
11.3.2.1 Fls_17_Dmu_AddressType
Table 759 Specification for Fls_17_Dmu_AddressType

Syntax	Fls_17_Dmu_AddressType	
Type	uint32	
File	Fls_17_Dmu.h	
Range	0 – 4294967295	Size depends on target platform and DFLASH0 data flash memory on the flash device.
Description	Used as an address offset from the configured Flash base address to access a certain Flash memory area. Note: The Fls_17_Dmu_AddressType type has the lower limit as 0 and the FLS base address is always added to it to arrive at the correct address.	
Source	AUTOSAR	

11.3.2.2 Fls_17_Dmu_ConfigType
Table 760 Specification for Fls_17_Dmu_ConfigType

Syntax	Fls_17_Dmu_ConfigType
---------------	-----------------------

Fls_17_Dmu driver
Table 760 Specification for Fls_17_Dmu_ConfigType (continued)

Type	Structure	
File	Fls_17_Dmu.h	
Range	HW dependent structure	Structure to hold the Flash driver configuration set. The contents of the initialization data structure are specific to the Flash memory hardware.
Description	A pointer to such a structure is provided to the Flash driver initialization routine for configuration of the driver and Flash memory hardware.	
Source	AUTOSAR	

11.3.2.3 Fls_17_Dmu_HardenType
Table 761 Specification for Fls_17_Dmu_HardenType

Syntax	Fls_17_Dmu_HardenType	
Type	uint32	
File	Fls_17_Dmu.h	
Range	0 - FLS_17_DMU_HARDENCHK_NOTREQD	Hardening not required
	2 - FLS_17_DMU_HARDENCHK_ERROR	Hardening failed due to some error.
	1 - FLS_17_DMU_HARDENCHK_REQRD	Hardening required
Source	IFX	

11.3.2.4 Fls_17_Dmu_Job_Type
Table 762 Specification for Fls_17_Dmu_Job_Type

Syntax	Fls_17_Dmu_Job_Type	
Type	uint8	
File	Fls_17_Dmu.h	
Range	0 - FLS_NO_JOB	No notification was called
	1 - FLS_WRITE_JOB	Notification for the write job
	2 - FLS_ERASE_JOB	Notification for the erase job
	3 - FLS_READ_JOB	Notification for the read job
	4 - FLS_COMPARE_JOB	Notification for the compare job
	6 - FLS_CANCEL_JOB	Notification for the canceled job
	9 - FLS_BLANKCHECK_JOB	Notification for the blank check
Description	This is an Infineon specific API and not listed in the SWS. Return values of the Fls_17_Dmu_GetNotifCaller() function.	
Source	IFX	

Fls_17_Dmu driver**11.3.2.5 Fls_17_Dmu_LengthType****Table 763 Specification for Fls_17_Dmu_LengthType**

Syntax	Fls_17_Dmu_LengthType	
Type	uint32	
File	Fls_17_Dmu.h	
Range	0 – 4294967295	Should be the same type as Fls_AddressType because of arithmetic operations. Size depends on the target platform and the DFLASH0 data flash memory on the device.
Description	Specifies the number of bytes to read/write/erase/compare.	
Source	AUTOSAR	

11.3.2.6 Fls_17_Dmu_NotifyFunctionPtrType**Table 764 Specification for Fls_17_Dmu_NotifyFunctionPtrType**

Syntax	Fls_17_Dmu_NotifyFunctionPtrType
Type	Pointer to a function of type void Function_Name (void)
File	Fls_17_Dmu.h
Description	Function pointer type for callback functions. Used for job end, job error and illegal functions.
Source	IFX

11.3.3 Functions - APIs

This section describes the syntax and the logic of the API functions.

11.3.3.1 Fls_17_Dmu_BankCheck**Table 765 Specification for Fls_17_Dmu_BankCheck API**

Syntax	<pre>Std_ReturnType Fls_17_Dmu_BankCheck (const Fls_17_Dmu_AddressType TargetAddress, const Fls_17_Dmu_LengthType Length)</pre>	
Service ID	0x0A	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	TargetAddress	Address in the DFlash0 data flash memory from which the blank check should be started.

Fls_17_Dmu driver**Table 765 Specification for Fls_17_Dmu_BankCheck API (continued)**

	Length	Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Number of bytes to be checked for erase pattern. Min.: 1 Max.: FLS_17_DMU_TOTAL_SIZE - TargetAddress
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: request for blank checking has been accepted by the module E_NOT_OK: request for blank checking has not been accepted by the module
Description	The Fls_17_Dmu_BankCheck should verify, whether a given memory area has been erased but not (yet) programmed. The function should limit the maximum number of checked Flash cells per main function cycle to the configured value FlsMaxReadNormalMode or FlsMaxReadFastMode, respectively.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver.</p> <p>FLS_17_DMU_E_BUSY: Reported when the any FLS driver API service is called while the driver is still busy executing previous operation.</p> <p>FLS_17_DMU_E_PARAM_LENGTH: Reported when the FLS driver API service is called with wrong length.</p> <p>FLS_17_DMU_E_PARAM_ADDRESS: Reported when the FLS driver API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FlsBlankCheckApi	
User hints	-	

Fls_17_Dmu driver**11.3.3.2 Fls_17_Dmu_Cancel****Table 766 Specification for Fls_17_Dmu_Cancel API**

Syntax	void Fls_17_Dmu_Cancel (void)	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	void
Description	Cancels an ongoing job. Note: Fls_17_Dmu_Cancel() shall not be invoked from interrupt context.	
Source	AUTOSAR	
Error handling	DET: FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FlsCancelApi	
User hints	-	

11.3.3.3 Fls_17_Dmu_CancelNonEraseJobs**Table 767 Specification for Fls_17_Dmu_CancelNonEraseJobs API**

Syntax	void Fls_17_Dmu_CancelNonEraseJobs (void)
Service ID	0x23
Sync/Async	Synchronous

Fls_17_Dmu driver**Table 767 Specification for Fls_17_Dmu_CancelNonEraseJobs API (continued)**

ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	void
Description	<p>This is an Infineon specific API and not listed in the SWS.</p> <p>Service for canceling the ongoing flash jobs except the erase job. This function aborts the pending jobs (except the erase job), so that directly after returning from this function, a new job can be accepted by the driver.</p> <p>The function resets the internal job processing variables of the driver(such as address, length and data pointer) and sets the driver state to idle.</p> <p>The routine sets the job result to MEMIF_JOB_CANCELED, if the job result currently has the following value: MEMIF_JOB_PENDING. Otherwise, it leaves the job result unchanged.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FlsIfxFeeUse	
User hints	-	

Fls_17_Dmu_Compare**Table 768 Specification for Fls_17_Dmu_Compare API**

Syntax	Std_ReturnType Fls_17_Dmu_Compare (const Fls_17_Dmu_AddressType SourceAddress, const uint8 * const TargetAddressPtr, const Fls_17_Dmu_LengthType Length)
Service ID	0x08
Sync/Async	Asynchronous
ASIL Level	B
Re-entrancy	Non reentrant

Fls_17_Dmu driver**Table 768 Specification for Fls_17_Dmu_Compare API (continued)**

Parameters (in)	SourceAddress TargetAddressPtr Length	Source address in the DFLASH0 data flash memory. This address offset is added to the data flash memory base address. Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Pointer to the target data buffer Number of bytes to compare Min.: 1 Max.: FLS_17_DMU_TOTAL_SIZE - SourceAddress
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: compare command is accepted E_NOT_OK: compare command is not accepted
Description	Compares the contents of an area of the DFLASH0 data flash memory with that of an application data buffer.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FLS_17_DMU_E_PARAM_ADDRESS: Reported when the FLS driver API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned.</p> <p>FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver.</p> <p>FLS_17_DMU_E_PARAM_LENGTH: Reported when the FLS driver API service is called with wrong length.</p> <p>FLS_17_DMU_E_BUSY: Reported when the any FLS driver API service is called while the driver is still busy executing previous operation.</p> <p>FLS_17_DMU_E_PARAM_DATA: Reported when the FLS driver API service is called with the value of source/target address as NULL pointer.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FlsCompareApi	
User hints	-	

Fls_17_Dmu driver

11.3.3.5 Fls_17_Dmu_CompareWordsSync

Table 769 Specification for Fls_17_Dmu_CompareWordsSync API

Syntax	<pre>Std_ReturnType Fls_17_Dmu_CompareWordsSync (const Fls_17_Dmu_AddressType SourceAddress, const uint32 * const TargetAddressPtr, const uint32 Length)</pre>	
Service ID	0x22	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	SourceAddress TargetAddressPtr Length	Source address in the DFLASH0 data flash memory. This address offset is added to the DFLASH0 data flash memory base address. Pointer to the target data buffer. Number of words to be compared. It takes the value from 1 to DFLASH0 total size.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: compare is successful E_NOT_OK: compare is not successful
Description	<p>This is an IFX specific API and not listed in the SWS.</p> <p>It is a service for comparing the contents on the DFLASH0 data flash memory synchronously.</p> <p>Note: The range check is performed only when 'FlsSafetyEnable' is enabled.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors:</p> <p>FLS_17_DMU_E_COMPARE_FAILED: Reported when the compare operation fails.</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FLS_17_DMU_SE_PARAM_ADDRESS: Reported when the API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned.</p> <p>FLS_17_DMU_SE_PARAM_LENGTH: Reported when the API service is called with wrong length.</p> <p>FLS_17_DMU_SE_PARAM_DATA: Reported when the API service is called, with the source/target address as NULL pointer.</p> <p>FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.</p>	

Fls_17_Dmu driver**Table 769 Specification for Fls_17_Dmu_CompareWordsSync API (continued)**

	FLS_17_DMU_SE_BUSY: This safety error is raised when the API service is called while the FLS driver is still busy. <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	FlsIfxFeeUse
User hints	-

Fls_17_Dmu_ControlTimeoutDet**Table 770 Specification for Fls_17_Dmu_ControlTimeoutDet API**

Syntax	void Fls_17_Dmu_ControlTimeoutDet (const uint8 Param)	
Service ID	0x27	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	Param	= 0 for disable = 1 for re-enable
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	void
Description	This function disables and re-enables the detection/reporting of the timeout DET for the write and erase operations. This is an Infineon specific API.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: FLS_17_DMU_SE_PARAM_INVLD: This safety error is reported when the parameter passed as argument of the function is not valid. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

Fls_17_Dmu driver**11.3.3.7 Fls_17_Dmu_Erase****Table 771 Specification for Fls_17_Dmu_Erase API**

Syntax	<pre>Std_ReturnType Fls_17_Dmu_Erase (const Fls_17_Dmu_AddressType TargetAddress, const Fls_17_Dmu_LengthType Length)</pre>	
Service ID	0x01	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	TargetAddress Length	Target address in the DFLASH0 data flash memory. This address offset is added to the DFLASH0 data flash memory base address. Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Number of bytes to erase Min.: 1 Max.: FLS_17_DMU_TOTAL_SIZE - TargetAddress
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: erase command accepted E_NOT_OK: erase command not accepted
Description	This API is a service for erasing one or more complete Flash sectors.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FLS_17_DMU_E_PARAM_ADDRESS: Reported when the FLS driver API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned.</p> <p>FLS_17_DMU_E_PARAM_LENGTH: Reported when the FLS driver API service is called with wrong length.</p> <p>FLS_17_DMU_E_BUSY: Reported when any FLS driver API service is called while the driver is still busy executing previous operation.</p> <p>FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver.</p> <p>Runtime Errors:</p> <p>FLS_17_DMU_E_ERASE_FAILED: Reported when the erase operation on DFLASH0 fails.</p> <p>DEM: None</p> <p>Safety Errors:</p>	

Fls_17_Dmu driver
Table 771 Specification for Fls_17_Dmu_Erase API (continued)

	<p>FLS_17_DMU_SE_ILLGL_OPERTN: This safety error is raised when the erase operation is suspended and a new erase operation is initiated.</p> <p>FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

11.3.3.8 Fls_17_Dmu_GetJobResult
Table 772 Specification for Fls_17_Dmu_GetJobResult API

Syntax	MemIf_JobResultType Fls_17_Dmu_GetJobResult (void)	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	MemIf_JobResultType	The result of the last job
Description	<p>Returns the result of the last job.</p> <p><i>Note: When the Infineon FEE is present, for the Fls_17_Dmu_CompareWordsSync(), Fls_17_Dmu_ReadWordsSync(), Fls_17_Dmu_VerifyErase() and Fls_17_dmu_verifySectorErase() APIs, the job result is not updated. Therefore, the job result returned for the mentioned APIs are of the previous jobs.</i></p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

Fls_17_Dmu driver
Table 772 Specification for Fls_17_Dmu_GetJobResult API (continued)

Configuration dependencies	FlsGetJobResultApi
User hints	-

11.3.3.9 Fls_17_Dmu_GetNotifCaller
Table 773 Specification for Fls_17_Dmu_GetNotifCaller API

Syntax	Fls_17_Dmu_Job_Type Fls_17_Dmu_GetNotifCaller (void)	
Service ID	0x29	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Fls_17_Dmu_Job_Type	FLS job that raised the notification
Description	Returns the FLS job that raised the notification. It should be called only from the callback notification functions of the upper layers. This is an Infineon specific API and not listed in the SWS.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FlsIfxFeeUse	
User hints	-	

Fls_17_Dmu driver**11.3.3.10 Fls_17_Dmu_GetOperStatus****Table 774 Specification for Fls_17_Dmu_GetOperStatus API**

Syntax	Std_ReturnType Fls_17_Dmu_GetOperStatus (void)	
Service ID	0x26	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: no OPER error E_NOT_OK: OPER error occurred
Description	This is an Infineon specific API and not listed in the SWS. Returns whether the OPER error had occurred or not.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

11.3.3.11 Fls_17_Dmu_GetStatus**Table 775 Specification for Fls_17_Dmu_GetStatus API**

Syntax	MemIf_StatusType Fls_17_Dmu_GetStatus (void)
Service ID	0x04
Sync/Async	Synchronous
ASIL Level	B

Fls_17_Dmu driver**Table 775 Specification for Fls_17_Dmu_GetStatus API (continued)**

Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	MemIf_StatusType	The state of the driver
Description	Returns the driver state.	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FlsGetStatusApi	
User hints	-	

11.3.3.12 Fls_17_Dmu_GetVersionInfo**Table 776 Specification for Fls_17_Dmu_GetVersionInfo API**

Syntax	<pre>void Fls_17_Dmu_GetVersionInfo (Std_VersionInfoType * const VersionInfoPtr)</pre>	
Service ID	0x10	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer to where to store the version information of this module.
Parameters (in - out)	-	-
Return	void	-
Description	Returns the version information of this module.	
Source	AUTOSAR	

Fls_17_Dmu driver
Table 776 Specification for Fls_17_Dmu_GetVersionInfo API (continued)

Error handling	<p>DET:</p> <p>FLS_17_DMU_E_PARAM_POINTER: Reported when the FLS driver's Fls_17_Dmu_GetVersionInfo() API service is called with a NULL pointer as argument.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	FlsVersionInfoApi
User hints	-

11.3.3.13 Fls_17_Dmu_Init
Table 777 Specification for Fls_17_Dmu_Init API

Syntax	<pre>void Fls_17_Dmu_Init (const Fls_17_Dmu_ConfigType * const ConfigPtr)</pre>	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	ConfigPtr	Pointer to the FLS driver configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	void
Description	Initializes the Flash driver.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FLS_17_DMU_E_PARAM_CONFIG: Reported when the FLS driver API service is called with a wrong parameter.</p> <p>FLS_17_DMU_E_BUSY: Reported when the any FLS driver API service is called while the driver is still busy executing previous operation.</p> <p>Runtime Errors:</p> <p>FLS_17_DMU_E_INIT_FAILED: This runtime error is reported if OPER error is detected during initialization.</p>	

Fls_17_Dmu driver
Table 777 Specification for Fls_17_Dmu_Init API (continued)

	<p>DEM: None</p> <p>Safety Errors:</p> <p>FLS_17_DMU_SE_INIT_FAILED: This safety error is reported when the FLS erase operation is suspended and Fls_17_Dmu_Init() is invoked or the DFLASH0 emulation mode is not set to single ended sensing mode.</p> <p>FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

11.3.3.14 Fls_17_Dmu_InitCheck
Table 778 Specification for Fls_17_Dmu_InitCheck API

Syntax	<pre>Std_ReturnType Fls_17_Dmu_InitCheck (const Fls_17_Dmu_ConfigType ConfigPtr)</pre>	
Service ID	0x2B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	ConfigPtr	
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: if initialization comparison is success E_NOT_OK: if initialization comparison fails
Description	This API checks the initialization values.	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

Fls_17_Dmu driver
Table 778 Specification for Fls_17_Dmu_InitCheck API (continued)

Configuration dependencies	FlsInitCheckApi
User hints	-

11.3.3.15 Fls_17_Dmu_IsHardeningRequired
Table 779 Specification for Fls_17_Dmu_IsHardeningRequired API

Syntax	<pre>Fls_17_Dmu_HardenType Fls_17_Dmu_IsHardeningRequired (const Fls_17_Dmu_AddressType TargetAddress, const uint8 AlignChk)</pre>	
Service ID	0x28	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress AlignChk	Target address in Flash memory. This address offset is added to the Flash memory base address This parameter signifies whether the hardening is to be done at the page level or WL level. The following are the values which will be used for indication: - hardening is done at the page level if the value of this parameter is: FLS_17_DMU_PAGE_HARDEN(0x55) - hardening is done at the 'Word-line level' if the value of this parameter is: FLS_17_DMU_WORDLINE_HARDEN(0xAA)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Fls_17_Dmu_HardenType	0 - FLS_17_DMU_HARDEN_NOTREQRD: Hardening is not required. 1 - FLS_17_DMU_HARDEN_REQRD: Hardening is required. 2 - FLS_17_DMU_HARDEN_ERROR: Hardening failed due to error.
Description	<p>This is an Infineon specific API and not listed in the SWS.</p> <p>The function checks whether the contents of the DFLASH0 data flash memory at requested Page or WL address need hardening or not.</p> <p>Note: It checks for the complete given page or word address depending on the value passed in the 'Alignchk' parameter.</p>	
Source	IFX	
Error handling	DET: None	

Fls_17_Dmu driver
Table 779 Specification for Fls_17_Dmu_IsHardeningRequired API (continued)

	<p>Runtime Errors:</p> <p>FLS_17_DMU_E_HARDENCHK_FAIL: This is reported when the hardening check fails due to the hardware error.</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FLS_17_DMU_SE_PARAM_INVLD: This safety error is reported when the parameter passed as argument of the function is not valid.</p> <p>FLS_17_DMU_SE_PARAM_ADDRESS: Reported when the API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned.</p> <p>FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.</p> <p>FLS_17_DMU_SE_BUSY: This safety error is raised when the API service is called while the FLS driver is still busy.</p> <p>FLS_17_DMU_SE_HW_TIMEOUT: This safety error is raised when the wait time for the execution of the suspend/resume operation expires.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	FlsIfxFeeUse
User hints	-

11.3.3.16 Fls_17_Dmu_Read
Table 780 Specification for Fls_17_Dmu_Read API

Syntax	<pre>Std_ReturnType Fls_17_Dmu_Read (const Fls_17_Dmu_AddressType SourceAddress, uint8 * const TargetAddressPtr, const Fls_17_Dmu_LengthType Length)</pre>	
Service ID	0x07	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	SourceAddress Length	Source address in the DFlash0 data flash memory. This address offset will be added to the DFlash0 data flash memory base address. Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Number of bytes to read Min.: 1

Fls_17_Dmu driver**Table 780 Specification for Fls_17_Dmu_Read API (continued)**

		Max.: FLS_17_DMU_TOTAL_SIZE - SourceAddress
Parameters (out)	TargetAddressPtr	Pointer to the target data buffer
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: read command has been accepted E_NOT_OK: read command has not been accepted
Description	Reads from flash memory.	
Source	AUTOSAR	
Error handling	DET: FLS_17_DMU_E_PARAM_ADDRESS: Reported when the FLS driver API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned. FLS_17_DMU_E_PARAM_LENGTH: Reported when the FLS driver API service is called with wrong length. FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver. FLS_17_DMU_E_BUSY: Reported when the any FLS driver API service is called while the driver is still busy executing previous operation. FLS_17_DMU_E_PARAM_DATA: Reported when the FLS driver API service is called with the value of source/target address as NULL pointer. Runtime Errors: None DEM: None Safety Errors: FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

11.3.3.17 Fls_17_Dmu_ReadWordsSync**Table 781 Specification for Fls_17_Dmu_ReadWordsSync API**

Syntax	<pre>Std_ReturnType Fls_17_Dmu_ReadWordsSync (const Fls_17_Dmu_AddressType SourceAddress, uint32 * const TargetAddressPtr, const uint32 Length)</pre>
---------------	---

Fls_17_Dmu driver**Table 781 Specification for Fls_17_Dmu_ReadWordsSync API (continued)**

Service ID	0x21	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	SourceAddress Length	Source address in the DFLASH0 data flash memory. This address offset is added to the DFLASH0 base address. Number of words to be read. It takes the value from 1 to DFLASH0 data flash size.
Parameters (out)	TargetAddressPtr	Pointer to target data buffer
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: read command is accepted E_NOT_OK: read command is not accepted
Description	<p>This is an Infineon specific API and not listed in the SWS.</p> <p>It is a service to read synchronously from the DFLASH0 data flash memory.</p> <p>Note: The range check is performed for the input parameters only when the 'FlsSafetyEnable' configuration parameter is enabled.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors:</p> <p>FLS_17_DMU_E_READ_FAILED: Reported when the read operation on DFLASH0 fails.</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FLS_17_DMU_SE_PARAM_LENGTH: Reported when the API service is called with wrong length.</p> <p>FLS_17_DMU_SE_PARAM_ADDRESS: Reported when the API service is called with the target source address that is out of the range or when the passed address is not sector or page aligned.</p> <p>FLS_17_DMU_SE_PARAM_DATA: Reported when the API service is called, with the source/target address as NULL pointer.</p> <p>FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.</p> <p>FLS_17_DMU_SE_BUSY: This safety error is raised when the API service is called while the FLS driver is still busy.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FlsIfxFeeUse	
User hints	-	

Fls_17_Dmu driver**11.3.3.18 Fls_17_Dmu_ResumeErase****Table 782 Specification for Fls_17_Dmu_ResumeErase API**

Syntax	Std_ReturnType Fls_17_Dmu_ResumeErase (void)	
Service ID	0x2A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Erase resume command was accepted and passed or Erase was not suspended when this API was called E_NOT_OK: Erase resume command was not accepted or failed
Description	This is an IFX specific API and not listed in the SWS. It is a service for resuming a suspended erase of a sector.	
Source	IFX	
Error handling	DET: None Runtime Errors: FLS_17_DMU_E_RESUME_FAIL: This is reported when the resume of the erase operation fails due to the hardware error. DEM: None Safety Errors: FLS_17_DMU_SE_HW_TIMEOUT: This safety error is raised when the wait time for the execution of the suspend/resume operation expires. FLS_17_DMU_SE_BUSY: This safety error is raised when the API service is called while the FLS driver is still busy. FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FlsUseEraseSuspend	
User hints	-	

Fls_17_Dmu driver**11.3.3.19 Fls_17_Dmu_SetMode****Table 783 Specification for Fls_17_Dmu_SetMode API**

Syntax	<pre>void Fls_17_Dmu_SetMode (const MemIf_ModeType Mode)</pre>	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	Mode	MEMIF_MODE_SLOW: slow read access MEMIF_MODE_FAST: fast read access
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	void
Description	Sets the flash operation mode of the driver.	
Source	AUTOSAR	
Error handling	<p>DET: FLS_17_DMU_E_BUSY: Reported when any FLS driver API service is called while the driver is still busy executing previous operation. FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: FLS_17_DMU_SE_PARAM_INVLD: This safety error is reported when the parameter passed as argument of the function is not valid.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	FlsSetModeApi	
User hints	-	

11.3.3.20 Fls_17_Dmu_SuspendErase**Table 784 Specification for Fls_17_Dmu_SuspendErase API**

Syntax	Std_ReturnType Fls_17_Dmu_SuspendErase
	(

Fls_17_Dmu driver
Table 784 Specification for Fls_17_Dmu_SuspendErase API (continued)

	void)	
Service ID	0x25	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Erase suspend successful or erase is already suspended E_NOT_OK: erase suspend failed or this API is called when erase is not on-going
Description	This is an IFX specific API and not listed in the SWS. It is a service for suspending an ongoing erase of a sector.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: FLS_17_DMU_SE_SUSPNDERASE_FAIL: This safety error is raised when the suspend error(ERR) in the suspend register(HF_SUSPEND) is set. FLS_17_DMU_SE_HW_TIMEOUT: This safety error is raised when the wait time for the execution of the suspend/resume operation expires. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FlsUseEraseSuspend	
User hints	-	

11.3.3.21 Fls_17_Dmu_VerifyErase
Table 785 Specification for Fls_17_Dmu_VerifyErase API

Syntax	Std_ReturnType Fls_17_Dmu_VerifyErase (const Fls_17_Dmu_AddressType TargetAddress, uint32 * const UnerasedWordlineAddressPtr, uint8 * const UnerasedWordlineCountPtr)
---------------	--

Fls_17_Dmu driver**Table 785 Specification for Fls_17_Dmu_VerifyErase API (continued)**

Service ID	0x24	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress	Target offset address in the DFLASH0 data flash memory. This address offset is added to the DFLASH0 data flash memory base address. The input value for TargetAddress can only be the start address of either of the sectors used by the Infineon FEE double sector algorithm.
Parameters (out)	UnerasedWordlineAddress Ptr UnerasedWordlineCountPt r	Pointer to the first un-eraseable WL address. Pointer to the un-eraseable WL count.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Erase verification command was accepted and passed E_NOT_OK: Erase verification command was not accepted or failed with more than two un-erasable WL
Description	This is an Infineon specific API and not listed in the SWS. It is a synchronous service to verify the erase operation performed on one of the sector out of two sectors(as per double sector algorithm used by Infineon FEE).	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: FLS_17_DMU_SE_PARAM_ADDRESS: Reported when the API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned. FLS_17_DMU_SE_PARAM_DATA: Reported when the API service is called, with the source/target address as NULL pointer. FLS_17_DMU_SE_BUSY: This safety error is raised when the API service is called while the FLS driver is still busy. FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FlsIfxFeeUse	
User hints	-	

Fls_17_Dmu driver**11.3.3.22 Fls_17_Dmu_VerifySectorErase****Table 786 Specification for Fls_17_Dmu_VerifySectorErase API**

Syntax	<pre>Std_ReturnType Fls_17_Dmu_VerifySectorErase (const Fls_17_Dmu_AddressType TargetAddress, uint32 * const UnerasedWordlineAddressPtr, uint8 * const UnerasedWordlineCountPtr, const uint8 Sector)</pre>	
Service ID	0x2C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress Sector	Target offset address in the DFLASH0 data flash memory. This parameter is an address offset and is added to the DFLASH0 data flash memory base address. The TargetAddress can only be the value of the start address of either of the sectors of the Infineon FEE double sector algorithm. Logical sub sector number (of the corresponding NVM sector) to be verified
Parameters (out)	UnerasedWordlineAddress Ptr UnerasedWordlineCountPt r	Pointer to the first un-erased WL address. Pointer to the un-erased WL count.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Erase verification operation is accepted and passed. E_NOT_OK: Erase verification operation is not accepted or failed with more than two un-erasable WLs.
Description	This is an Infineon specific API and not listed in the SWS. It is a service for verifying the erase of a logical sub sector synchronously.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: FLS_17_DMU_SE_PARAM_ADDRESS: Reported when the API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned. FLS_17_DMU_SE_PARAM_DATA: Reported when the API service is called, with the source/target address as NULL pointer. FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.	

Fls_17_Dmu driver
Table 786 Specification for Fls_17_Dmu_VerifySectorErase API (continued)

	<p>FLS_17_DMU_SE_BUSY: This safety error is raised when the API service is called while the FLS driver is still busy.</p> <p>FLS_17_DMU_SE_PARAM_INVLD: This safety error is reported when the parameter passed as argument of the function is not valid.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	FlsIfxFeeUse
User hints	-

11.3.3.23 Fls_17_Dmu_Write
Table 787 Specification for Fls_17_Dmu_Write API

Syntax	<pre>Std_ReturnType Fls_17_Dmu_Write (const Fls_17_Dmu_AddressType TargetAddress, const uint8 * const SourceAddressPtr, const Fls_17_Dmu_LengthType Length)</pre>	
Service ID	0x02	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non reentrant	
Parameters (in)	TargetAddress SourceAddressPtr Length	Target address in the DFlash0 hardware memory. This address offset is be added to the DFlash0 base address. Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Pointer to the source data buffer. Number of bytes to write Min.: 1 Max.: FLS_17_DMU_TOTAL_SIZE - TargetAddress
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: write operation accepted E_NOT_OK: write operation not accepted
Description	Writes one or more complete flash pages.	
Source	AUTOSAR	
Error handling	DET:	

Fls_17_Dmu driver
Table 787 Specification for Fls_17_Dmu_Write API (continued)

	<p>FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver.</p> <p>FLS_17_DMU_E_PARAM_LENGTH: Reported when the FLS driver API service is called with wrong length.</p> <p>FLS_17_DMU_E_PARAM_ADDRESS: Reported when the FLS driver API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned.</p> <p>FLS_17_DMU_E_BUSY: Reported when the any FLS driver API service is called while the driver is still busy executing previous operation.</p> <p>FLS_17_DMU_E_PARAM_DATA: Reported when the FLS driver API service is called with the value of source/target address as NULL pointer.</p> <p>FLS_17_DMU_E_VERIFY_ERASE_FAILED: Reported when the erase verification(blank check) fails. This DET is detected and notified only when the Infineon FEE is not used.</p> <p>Note: When the development error or safety error detection is ON, the erase verification during interrupt mode is done by means EVER bit in the HF_ERRSR register of DMU, as the EVER bit provides a much more precise result of the erase verification for the erased cells in the hardware.</p> <p>Runtime Errors:</p> <p>FLS_17_DMU_E_WRITE_FAILED: Reported when write operation on DFLASH0 fails.</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

11.3.4 Notifications and Callbacks

The FLS driver does not implement any notifications. However, it does report the completion and error status of a job through notification callback functions of the FEE.

11.3.5 Scheduled functions

11.3.5.1 Fls_17_Dmu_MainFunction

Table 788 Specification for Fls_17_Dmu_MainFunction API

Syntax	<pre>void Fls_17_Dmu_MainFunction (void)</pre>
---------------	---

Fls_17_Dmu driver**Table 788 Specification for Fls_17_Dmu_MainFunction API (continued)**

Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This API is a service for performing the read, write, erase, compare and blank check jobs on the DFLASH0 hardware.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver.</p> <p>FLS_17_DMU_E_VERIFY_WRITE_FAILED: Reported when the write verification (compare) fails. This DET is detected and notified only when the Infineon FEE is not used.</p> <p>FLS_17_DMU_E_TIMEOUT: Reported when the timeout limit is exceeded during the execution of an FLS driver's API.</p> <p>FLS_17_DMU_E_VERIFY_ERASE_FAILED: Reported when the erase verification(blank check) fails. This DET is detected and notified only when the Infineon FEE is not used.</p> <p>Note: When the development error or safety error detection is ON, the erase verification during interrupt mode is done by means EVER bit in the HF_ERRSR register of DMU, as the EVER bit provides a much more precise result of the erase verification for the erased cells in the hardware.</p> <p>Runtime Errors:</p> <p>FLS_17_DMU_E_ERASE_FAILED: Reported when the erase operation on DFLASH0 fails.</p> <p>FLS_17_DMU_E_WRITE_FAILED: Reported when write operation on DFLASH0 fails.</p> <p>FLS_17_DMU_E_COMPARE_FAILED: Reported when the compare operation fails.</p> <p>FLS_17_DMU_E_READ_FAILED: Reported when the read operation on DFLASH0 fails.</p> <p>FLS_17_DMU_E_BLANKCHECK_FAILED: Reported when the blank-check operation fails.</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

Fls_17_Dmu driver**11.3.6 Interrupt service routines****11.3.6.1 Fls_17_Dmu_Isr****Table 789 Specification for Fls_17_Dmu_Isr API**

Syntax	<pre>void Fls_17_Dmu_Isr (void)</pre>	
Service ID	0x2D	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interrupt is mapped to the node: SRC_DMU0. This services the Write and Erase Jobs.	
Source	IFX	
Error handling	<p>DET:</p> <p>FLS_17_DMU_E_VERIFY_WRITE_FAILED: Reported when the write verification (compare) fails. This DET is detected and notified only when the Infineon FEE is not used.</p> <p>FLS_17_DMU_E_VERIFY_ERASE_FAILED: Reported when the erase verification(blank check) fails. This DET is detected and notified only when the Infineon FEE is not used.</p> <p>Note: When the development error or safety error detection is ON, the erase verification during interrupt mode is done by means EVER bit in the HF_ERRSR register of DMU, as the EVER bit provides a much more precise result of the erase verification for the erased cells in the hardware.</p> <p>Runtime Errors:</p> <p>FLS_17_DMU_E_ERASE_FAILED: Reported when the erase operation on DFLASH0 fails.</p> <p>FLS_17_DMU_E_WRITE_FAILED: Reported when write operation on DFLASH0 fails.</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>FLS_17_DMU_SE_INVALID_ISR: Error is reported as a safety error when there are spurious(not valid) interrupts.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

Fls_17_Dmu driver
Table 789 Specification for Fls_17_Dmu_Isr API (continued)

Configuration dependencies	FlsUseInterrupts
User hints	-

11.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

11.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Note: The following error IDs are also reported as safety errors.

Table 790 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
Reported when the any FLS driver API service is called while the driver is still busy executing previous operation.	AUTOSAR	FLS_17_DMU_E_BUSY=0x06	Fls_17_Dmu_SetMode, Fls_17_Dmu_Init, Fls_17_Dmu_Compare, Fls_17_Dmu_BlankCheck, Fls_17_Dmu_Read, Fls_17_Dmu_Write, Fls_17_Dmu_Erase
Reported when the FLS driver API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned.	AUTOSAR	FLS_17_DMU_E_PARAM_ADDRESS=0x02	Fls_17_Dmu_BlankCheck, Fls_17_Dmu_Read, Fls_17_Dmu_Write, Fls_17_Dmu_Erase, Fls_17_Dmu_Compare
Reported when the FLS driver API service is called with a wrong parameter.	AUTOSAR	FLS_17_DMU_E_PARAM_CONFIG=0x01	Fls_17_Dmu_Init
Reported when the FLS driver API service is called with the value of source/target address as NULL pointer.	AUTOSAR	FLS_17_DMU_E_PARAM_DATA=0x04	Fls_17_Dmu_Compare, Fls_17_Dmu_Write, Fls_17_Dmu_Read
Reported when the FLS driver API service is called with wrong length.	AUTOSAR	FLS_17_DMU_E_PARAM_LENGTH=0x03	Fls_17_Dmu_Compare, Fls_17_Dmu_BlankCheck, Fls_17_Dmu_Read, Fls_17_Dmu_Write, Fls_17_Dmu_Erase
Reported when the FLS driver's	AUTOSAR	FLS_17_DMU_E_PARAM_POINTER=0x0a	Fls_17_Dmu_GetVersionInfo

Fls_17_Dmu driver
Table 790 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
Fls_17_Dmu_GetVersionInfo() API service is called with a NULL pointer as argument.			
Reported when the timeout limit is exceeded during the execution of an FLS driver's API.	AUTOSAR	FLS_17_DMU_E_TIMEOUT=0x09	Fls_17_Dmu_MainFunction
Reported when any of the FLS driver's API service is called without properly initializing the driver.	AUTOSAR	FLS_17_DMU_E_UNINIT=0x05	Fls_17_Dmu_SetMode, Fls_17_Dmu_MainFunction, Fls_17_Dmu_GetJobResult, Fls_17_Dmu_Erase, Fls_17_Dmu_Compare, Fls_17_Dmu_BlankCheck, Fls_17_Dmu_Cancel, Fls_17_Dmu_Read, Fls_17_Dmu_Write
Reported when the erase verification(blank check) fails. This DET is detected and notified only when the Infineon FEE is not used. Note: When the development error or safety error detection is ON, the erase verification during interrupt mode is done by means EVER bit in the HF_ERRSR register of DMU, as the EVER bit provides a much more precise result of the erase verification for the erased cells in the hardware.	AUTOSAR	FLS_17_DMU_E_VERIFY_ERASE_FAILED=0x07	Fls_17_Dmu_Write, Fls_17_Dmu_Isr, Fls_17_Dmu_MainFunction
Reported when the write verification (compare) fails. This DET is detected and notified only when the Infineon FEE is not used.	AUTOSAR	FLS_17_DMU_E_VERIFY_WRITE_FAILED=0x08	Fls_17_Dmu_Isr, Fls_17_Dmu_MainFunction

11.3.7.2 Production errors

The driver does not report any production errors.

Fls_17_Dmu driver

11.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 791 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
This safety error is raised when the API service is called while the FLS driver is still busy.	IFX	FLS_17_DMU_SE_BUSY=0x06	Fls_17_Dmu_ResumeErase, Fls_17_Dmu_IsHardeningRequired, Fls_17_Dmu_VerifySectorErase, Fls_17_Dmu_VerifyErase, Fls_17_Dmu_CompareWordsSync, Fls_17_Dmu_ReadWordssync
This is reported if the DFLASH0 flash bank is still busy with the operation.	IFX	FLS_17_DMU_SE_HW_BUSY=0x6E	Fls_17_Dmu_ResumeErase, Fls_17_Dmu_Write, Fls_17_Dmu_Erase, Fls_17_Dmu_Read, Fls_17_Dmu_Compare, Fls_17_Dmu_BlankCheck, Fls_17_Dmu_IsHardeningRequired, Fls_17_Dmu_Init, Fls_17_Dmu_VerifySectorErase, Fls_17_Dmu_VerifyErase, Fls_17_Dmu_CompareWordsSync, Fls_17_Dmu_ReadWordssync
This safety error is raised when the wait time for the execution of the suspend/resume operation expires.	IFX	FLS_17_DMU_SE_HW_TIMEOUT=0x73	Fls_17_Dmu_IsHardeningRequired, Fls_17_Dmu_SuspendErase, Fls_17_Dmu_ResumeErase
This safety error is raised when the erase operation is suspended and a new erase operation is initiated.	IFX	FLS_17_DMU_SE_ILLGL_OPERTN=0x64	Fls_17_Dmu_Erase
This safety error is reported when the FLS erase operation is suspended and Fls_17_Dmu_Init() is invoked or the DFLASH0	IFX	FLS_17_DMU_SE_INIT_FAILED=0x5F	Fls_17_Dmu_Init

Fls_17_Dmu driver**Table 791 Description of safety errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
emulation mode is not set to single ended sensing mode.			
Error is reported as a safety error when there are spurious(not valid) interrupts.	IFX	FLS_17_DMU_SE_INVALID_ISR=0x78	Fls_17_Dmu_Isr
Reported when the API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned.	IFX	FLS_17_DMU_SE_PARAM_ADDRESS=0x02	Fls_17_Dmu_VerifySectorErase, Fls_17_Dmu_VerifyErase, Fls_17_Dmu_CompareWordsSync, Fls_17_Dmu_ReadWordssync, Fls_17_Dmu_IsHardeningRequired
Reported when the API service is called, with the source/target address as NULL pointer.	IFX	FLS_17_DMU_SE_PARAM_DATA=0x04	Fls_17_Dmu_VerifySectorErase, Fls_17_Dmu_VerifyErase, Fls_17_Dmu_CompareWordsSync, Fls_17_Dmu_ReadWordssync
This safety error is reported when the parameter passed as argument of the function is not valid.	IFX	FLS_17_DMU_SE_PARAM_INVLD=0x5A	Fls_17_Dmu_VerifySectorErase, Fls_17_Dmu_ControlTimeOutDet, Fls_17_Dmu_IsHardeningRequired, Fls_17_Dmu_SetMode
Reported when the API service is called with wrong length.	IFX	FLS_17_DMU_SE_PARAM_LENGTH=0x03	Fls_17_Dmu_CompareWordsSync, Fls_17_Dmu_ReadWordssync
This safety error is raised when the suspend error(ERR) in the suspend register(HF_SUSPEND) is set.	IFX	FLS_17_DMU_SE_SUSPNDERASE_FAIL=0x50	Fls_17_Dmu_SuspendErase

11.3.7.4 Runtime errors

The following table lists all the runtime errors reported by the driver.

Fls_17_Dmu driver
Table 792 Description of runtime errors reported

Description	Source	Error code and value	Applicable APIs
Reported when the blank-check operation fails.	IFX	FLS_17_DMU_E_BLANKCHECK_FAILED=0x1E	Fls_17_Dmu_MainFunction
Reported when the compare operation fails.	AUTOSAR	FLS_17_DMU_E_COMPARE_FAILED=0x04	Fls_17_Dmu_CompareWordsSync, Fls_17_Dmu_MainFunction
Reported when the erase operation on DFLASH0 fails.	AUTOSAR	FLS_17_DMU_E_ERASE_FAILED=0x01	Fls_17_Dmu_Isr, Fls_17_Dmu_MainFunction, Fls_17_Dmu_Erase
This is reported when the hardening check fails due to the hardware error.	IFX	FLS_17_DMU_E_HARDENCHK_FAILED=0x37	Fls_17_Dmu_IsHardeningRequired
This runtime error is reported if OPER error is detected during initialization.	IFX	FLS_17_DMU_E_INIT_FAILED=0x39	Fls_17_Dmu_Init
Reported when the read operation on DFLASH0 fails.	AUTOSAR	FLS_17_DMU_E_READ_FAILED=0x03	Fls_17_Dmu_MainFunction, Fls_17_Dmu_ReadWordsSync
This is reported when the resume of the erase operation fails due to the hardware error.	IFX	FLS_17_DMU_E_RESUME_FAILED=0x38	Fls_17_Dmu_ResumeErase
Reported when write operation on DFLASH0 fails.	AUTOSAR	FLS_17_DMU_E_WRITE_FAILED=0x02	Fls_17_Dmu_Isr, Fls_17_Dmu_MainFunction, Fls_17_Dmu_Write

11.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

11.3.8.1 Deviations

The section describes the deviations from software specification.

Table 793 Known deviations

Reference	Deviation
Protection settings is not used	Protection settings is not used in the FLS driver as it is more relevant for the FlsLoader driver and therefore the parameter FlsProtection is NOT supported.

Fls_17_Dmu driver**Table 793 Known deviations (continued)**

Reference	Deviation
FlsMaxWriteFastMode and FlsMaxWriteNormalMode is not supported	FlsMaxWriteFastMode / FlsMaxWriteNormalMode is NOT supported as write is performed for 8/32 bytes depending on the data size and page start address.
Runtime error detection	The detection and reporting of runtime errors is performed only if the configuration parameter FLS_RUNTIME_ERROR_DETECT is set.
External flash driver	External flash driver is NOT supported.
Unexpected flash ID error	FLS_E_UNEXPECTED_FLASH_ID error is NOT supported as external flash driver is not configured.
FlsAcLoadOnJobStart	FlsAcLoadOnJobStart is NOT supported because write and erase flash access code is executed from flash.
FlsAcLocationWrite	FlsAcLocationWrite parameter is not supported because the write access code is executed from flash.
FlsAcLocationErase	FlsAcLocationErase parameter is not supported because the erase access code is executed from flash.
FLS_E_VERIFY_ERASE_FAILED in Interrupt mode	When the development error detection or safety error reporting is ON, the erase verification during interrupt mode is done by means of EVER bit in the DMU register HF_ERRSR and not by read back and compare of data from flash, as EVER provides a much more precise result of the erase verification for the erased cells done in the HW.
Behavior of timeouts for erase and write jobs	<p>The timeout monitoring for the user job is done based on the number of <code>Fls_17_Dmu_MainFunction()</code> cycles needed to complete the user job, rather than the timeout monitoring of individual DFLASH0 command cycles. For the purpose of calculation of timeout duration, it is assumed that the <code>Fls_17_Dmu_MainFunction()</code> is called at the rate specified by the <code>FlsCallCycle</code> configuration parameter.</p> <p>For example,</p> <p>When the write address is burst aligned and 512 bytes of data is to be written, the write will be done in bursts and the total number of <code>Fls_17_Dmu_MainFunction()</code> cycles needed will be 16 cycles((512 bytes)/(32 bytes)), where burst write = 32 bytes.</p> <p>When the write address is not burst aligned and 512 bytes of data is to be written, the write will be done page-wise and the total number of <code>Fls_17_Dmu_MainFunction()</code> cycles needed will be 64 cycles ((512 bytes)/(8 bytes)), where page write = 8 bytes.</p> <p>For erase operation, a maximum of 256K can be erased in one command sequence execution. Therefore, if <code>FlsCallCyle</code> = 100us, the length to be erased = 512K, the max time taken for 256K erase = 1500000us. The total number of <code>Fls_17_Dmu_MainFunction()</code> cycles needed will be 30000 and additional 2 cycles is given as a margin, so the number of cycles will be 30002.</p>

Fls_17_Dmu driver

11.3.8.2 Limitations

The section describes the limitations from software specification.

Table 794 Known limitations

Reference	Limitation
FlsMaxWriteNormalMode, FlsMaxWriteFastMode	These parameters are fixed to 32 bytes.
Fls_17_Dmu_Cancel	Although the API is synchronous, hardware may be still busy after returning from Fls_17_Dmu_Cancel API due to already issued flash erase or write command sequence. In such scenarios, any new job issued may get rejected with return value as E_NOT_OK and safety error as FLS_17_DMU_E_HW_BUSY. The user may choose to retry or re-issue the same job again.
FLS_17_DMU_E_TIMEOUT DET	If the DET reporting is ON, the FLS_17_DMU_E_TIMEOUT DET is reported by the Fls_17_Dmu_MainFunction() API. This may occur due to some hardware issue. In such a scenario, the Fls_17_Dmu_MainFunction() will keep reporting the FLS_17_DMU_E_TIMEOUT DET on subsequent calls as well. The FLS driver status will remain MEMIF_BUSY and job result will remain MEMIF_JOB_PENDING.
Fls_17_Dmu_Write API	In case of Fls_17_Dmu_Write the SourceAddressPtr containing the content to be written should be word aligned.
FlsCallCycle	The minimum call cycle(FlsCallCycle) value for invoking Fls_17_Dmu_MainFunction() API is 100us. However, the burst write operation takes 140us. Therefore, it has to be ensured that the minimum value for the call cycle(FlsCallCycle) is 200us. This value is suggested to ensure adequate buffer to avoid unintended timeout.
Timeout of flash operations	All timeout values used by the FLS module are calculated assuming the FSI operation at 100MHz.

11.3.9 Unsupported hardware features

The following hardware features of DMU DFLASH0 are not supported

- The operations on PFlash are not supported, as it is assumed that the FLS driver is only for EEPROM flash emulation.
- Flash protection: DMU_HF_PROTECT is not supported by the FLS driver as it is handled by the FlsLoader.
- The following registers are not used or handled by the FLS driver. These remain at their default values after reset or use the values configured by the user during boot:
 - UCB_DFLASH_ORIG: Recommended value is UNREAD.

Fls_17_Dmu driver

- UCB_DFLASH_COPY: If the value of ‘UCB_DFLASH_ORIG’ is as UNREAD above, then the value of this is Don’t Care.
- DMU_HF_PROCONUSR: Recommended value is 0x00000000. For any other value, user is advised to refer to the Target Spec.
- DMU_HF_PROCOND.RPRO: Recommended value is 0 for this bit. For any other value, user is advised to refer to the Target Spec.
- HF_CONTROL.DDFD: Recommended value is 0 for this bit. For any other value, the user is advised to refer to the Target Spec.

Note: *If the Protection is enabled, the user has to disable Protection, in which case the value of the above register has to be set appropriately. Please refer to the Target Spec.*

GPT driver**12 GPT driver****12.1 User information****12.1.1 Description**

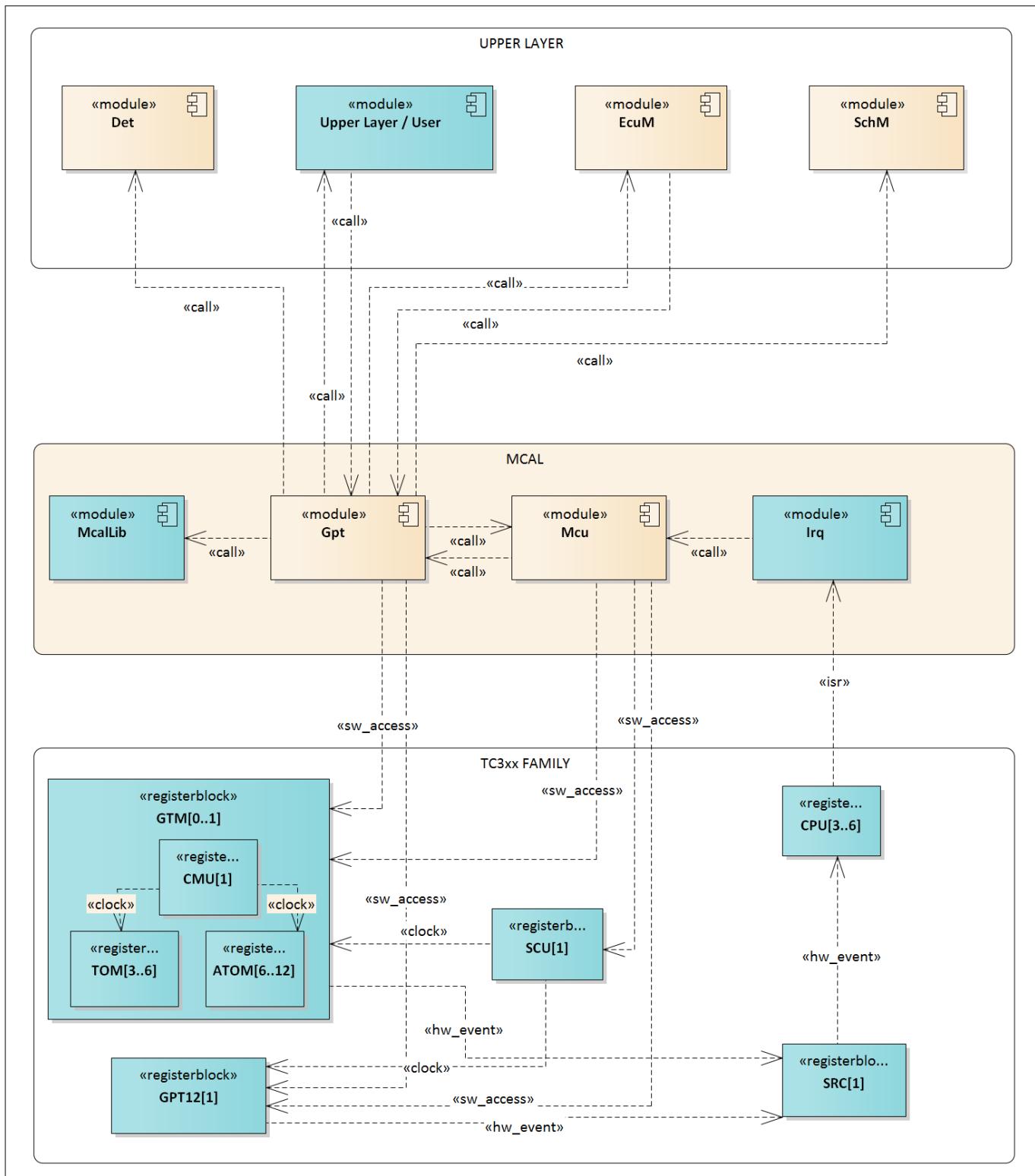
The GPT driver is responsible for providing APIs for standard timer functions specified by AUTOSAR. The underlying timer engines are GTM timer channel [TOM/ATOM slice] and GPT12 timers that are available in the second generation of AURIX™ microcontrollers. The user can configure the driver with multiple channels and the following operations can be performed on each channel:

- Start/stop the timer
- Enable/disable wakeup functionality
- Enable/disable notification
- One-Shot or Continuous mode of operation
- GPT Predef Timer feature implemented using the TOM slices

Apart from the configurable parameters specified by AUTOSAR, the GPT driver also provides additional parameters to configure the GTM timer slice and GPT12 timers.

12.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

GPT driver

Figure 125 **Mapping of hardware-software interfaces**
12.1.2.1 GTM: primary hardware peripheral
Hardware functional features

The GPT driver uses the GTM IP for realizing the continuous timer mode, one-shot timer mode and predef timers. Continuous and one-shot mode logical channels need only one timer channel from TOM/ATOM slice.

GPT driver

Predef timers use only the TOM slice of the GTM IP because TOM is 16-bit timer and derivation of 16-, 24- and 32-bits are easier compared to ATOM which is 24 bit. The implementation of a predef timer takes a minimum of one TOM channel and a maximum of three TOM channels depending on the user configuration. If only 16-bit 1 µs timer is configured and the fGTM is configured in such a way that 1 µs can be directly derived, then only one TOM channel is used. Otherwise the TRIGOUT feature of the hardware is used, where the previous channel triggers the current TOM channel on a compare match event leading to 1 µs per tick. Same feature is used for realizing the 24- and 32-bits predefined timers, where the TOM channel is triggered by the immediate previous TOM channel once its maximum count is reached. Similarly for 100 µs Predef timer, channel (x) will clock the channel (x+1) in desired frequency. For example, if the fGTM is configured in such a way that 1 µs and 100 µs cannot be directly derived and both 32-bit 1 µs and 32-bit 100 µs timers are configured, then 6 TOM channels are used to implement the Predef timers.

The key hardware functional features of GTM-TOM/ATOM IP used by the GPT driver are:

- Continuous counting up mode
- One-shot counting up mode
- ATOM signal output mode PWM (SOMP)

The unsupported features of the GTM-TOM/ATOM IP are:

- Duty cycle, period and clock frequency update mechanisms
- Continuous counting up-down mode
- One-shot counting up-down mode
- Pulse count modulation mode
- Trigger generation
- ATOM signal output mode immediate (SOMI)
- ATOM signal output mode compare (SOMC)
- ATOM signal output mode serial (SOMS)
- ATOM signal output mode buffered compare (SOMB)

Users of the hardware

The GTM-TOM/ATOM IP is shared by multiple drivers such as GPT, OCU, PWM, WDG and ADC. The MCU driver provides APIs to program the GTM SFRs. The GPT driver uses these APIs to write the GTM SFRs. Additionally, updates to channel-specific SFRs are performed by the GPT driver. Since these channels are exclusively reserved for the GPT driver, access to the channel-specific SFRs from other drivers or user software is not allowed.

Hardware diagnostic features

Not applicable.

Hardware events

The GPT driver uses the following hardware event from the GTM-TOM/ATOM IP:

- Period match interrupt to provide notification when target time is elapsed

12.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The GPT driver depends upon the SCU IP for the clock. The driver requires the fGTM and fSPB clock signals for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

GPT driver

The SMU alarms configured for the SCU IP are not monitored by the GPT driver.

Hardware events

Hardware events from the SCU are not used by the GPT driver.

12.1.2.3 GPT12: primary hardware peripheral

Hardware functional features

The GPT driver uses the GPT12 IP for realizing the continuous timer mode and one-shot timer mode. Predef timer is not supported by GPT12 IP. Implementation of a continuous mode logical channel using GPT1 block needs two timers, core timer T3 and one of the auxiliary timer(T2 or T4). Implementation of a continuous mode logical channel using GPT2 block needs only T6(using CAPREL register as reload register). Implementation of one-shot mode logical channel needs only one timer from either of GPT1 or GPT2 block.

The key hardware functional feature of GPT12 IP used by the GPT driver is:

- Timer mode

The unsupported features of the GPT12 IP are:

- Gated Timer mode
- Counter mode
- Incremental Interface mode

Users of the hardware

The GPT12 IP is shared between the GPT and the ICU drivers. The MCU driver provides an API to initialize the GPT12 control registers. The GPT driver uses this API to initialize the GPT12 control registers. Updates to channel-specific SFRs are performed by the GPT driver. Since the GPT12 timer/s are exclusively reserved for the GPT channel, access to the channel-specific SFRs from other drivers or user software is not allowed.

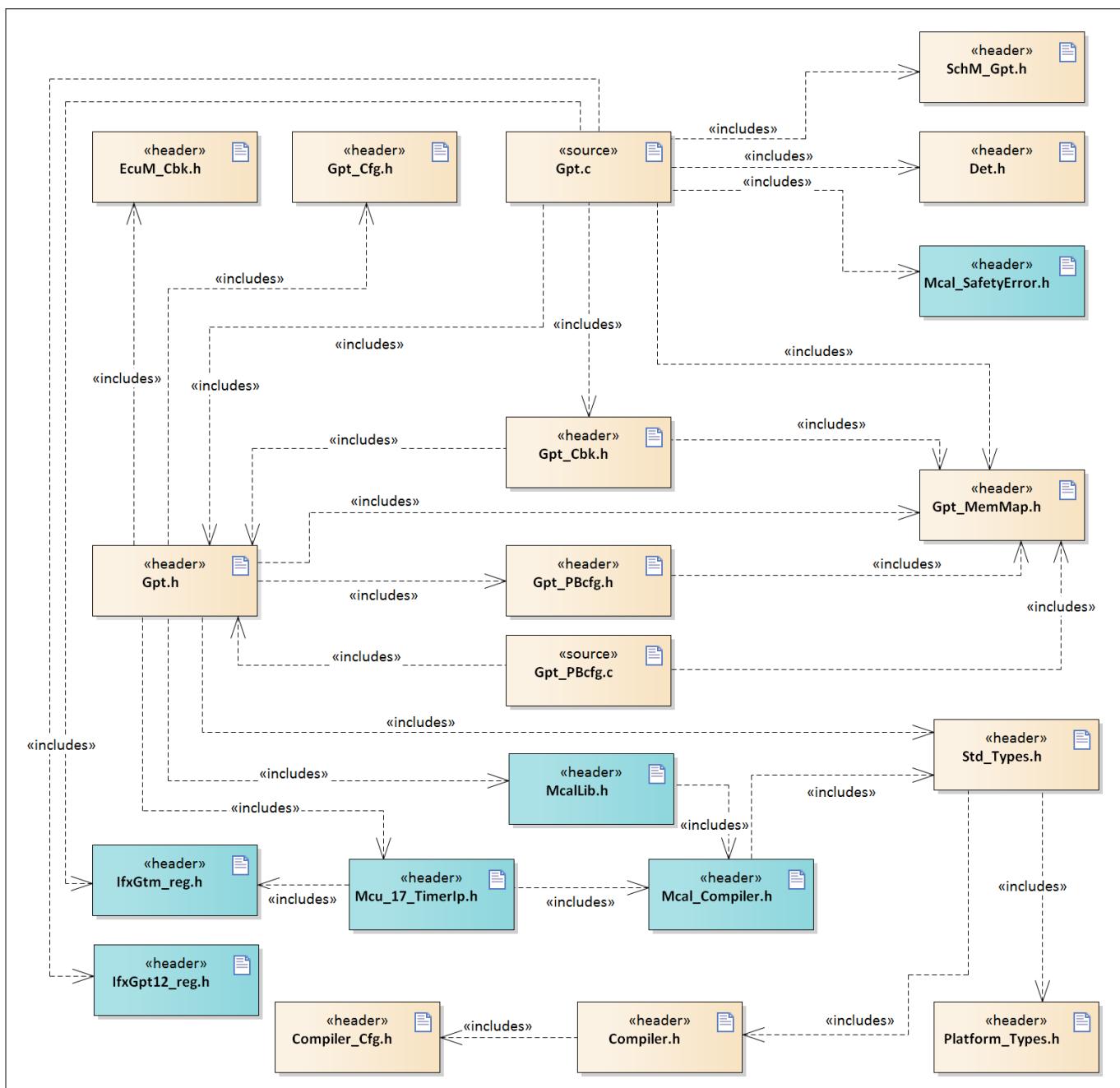
Hardware diagnostic features

Not applicable.

Hardware events

The GPT driver uses the following hardware event from the GPT12 IP:

- Timer counter underflow event to provide notification when target time is elapsed

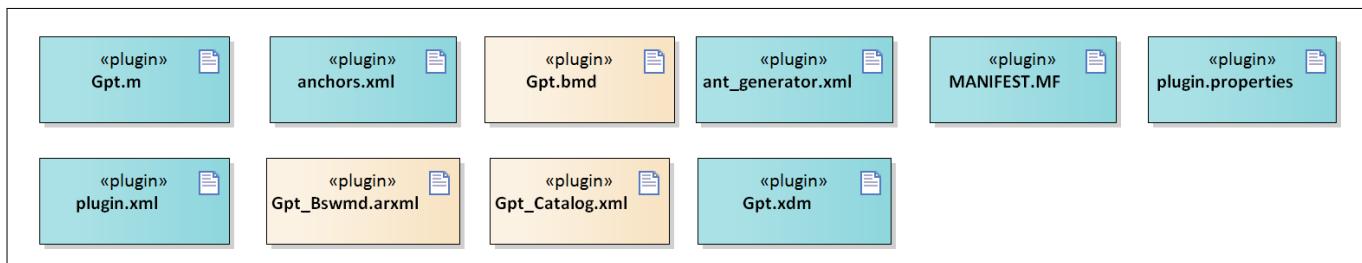
GPT driver**12.1.3 File structure****12.1.3.1 C file structure****Figure 126 C file structure****Table 795 C file structure**

File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Compiler_Cfg.h	Configuration header file for compiler abstraction
Det.h	Provides the exported interfaces of Development Error Tracer

GPT driver**Table 795 C file structure (continued)**

File name	Description
EcuM_Cbk.h	Header file containing declarations of the EcuM callbacks
Gpt.c	C file providing implementation of APIs
Gpt.h	Header file providing prototypes of APIs and data types
Gpt_Cbk.h	Header file providing prototypes of callback APIs
Gpt_Cfg.h	Generated header file containing definitions for all pre-compile time configuration parameters defined as pre-processor directive (#define) for the GPT driver
Gpt_MemMap.h	File containing the memory section definitions used by the GPT driver
Gpt_PBcfg.c	Generated header file containing configuration data of the user
Gpt_PBcfg.h	File (Generated) containing declaration of the post-build configuration data structures
IfxGpt12_reg.h	SFR header file for GPT12
IfxGtm_reg.h	SFR header file for GTM
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
Mcal_Compiler.h	Provides abstraction for TriCore™-intrinsic instruction
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Gpt.h	File containing the critical sections declarations
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

12.1.3.2 Code generator plugin files

**Figure 127 Code generator plugin files****Table 796 Code generator plugin files**

File name	Description
Gpt.bmd	Code template macro file for the GPT driver

GPT driver
Table 796 Code generator plugin files (continued)

File name	Description
Gpt.m	Code template macro file for the GPT driver
Gpt.xdm	Tresos format XML data model schema file
Gpt_Bswmd.arxml	AUTOSAR format module description file
Gpt_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the metadata for the GPT driver
anchors.xml	AUTOSAR format module description file
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point
plugin.properties	Tresos plugin support file for the GPT driver
plugin.xml	Tresos plugin support file for the GPT driver

12.1.4 Integration hints

This section lists the key points that an integrator or user of the GPT driver must consider.

12.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the GPT driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM is also responsible for the wakeup operation of configured wakeup capable channels when the driver is in sleep mode therefore the user must configure the wake-up information in EcuM configuration, which will be assigned to every wake-up capable GPT channel. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the Gpt_MemMap.h file.

The Gpt_MemMap.h file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

GPT driver

```

/* Sample implementation of Gpt_MemMap.h */
/* Core specific data [ DSRAM[X] ] */
#if defined GPT_START_SEC_VAR_CLEARED_ASIL_B_CORE[X]_32
/* User pragmas to allocate to DSRAM[X] */
#undef GPT_START_SEC_VAR_CLEARED_ASIL_B_CORE[X]_32
#undef MEMMAP_ERROR
#elif defined GPT_STOP_SEC_VAR_CLEARED_ASIL_B_CORE[X]_32
/* User pragmas to end to allocation to DSRAM[X] */
#undef GPT_STOP_SEC_VAR_CLEARED_ASIL_B_CORE[X]_32
#undef MEMMAP_ERROR

/* Global data [NON-CACHED LMU] */
#elif defined GPT_START_SEC_VAR_INIT_ASIL_B_GLOBAL_32
/* User pragmas to allocate to Non-cached LMU */
#undef GPT_START_SEC_VAR_INIT_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR
#elif defined GPT_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_32
/* User pragmas to end allocation to Non-cached LMU */
#undef GPT_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR

/* Core specific constant data [ PFLASH[X] ] */
#elif defined GPT_START_SEC_CONST_ASIL_B_CORE[X]_8
/* User pragmas to allocate to PFLASH[X] */
#undef GPT_START_SEC_CONST_ASIL_B_CORE[X]_8
#undef MEMMAP_ERROR
#elif defined GPT_STOP_SEC_CONST_ASIL_B_CORE0_8
/* User pragmas to end allocation to PFLASH[X] */
#undef GPT_STOP_SEC_CONST_ASIL_B_CORE[X]_8
#undef MEMMAP_ERROR

/* Global configuration data [ PFLASH[X] ] */
#elif defined GPT_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/* User pragmas to allocate to PFLASH[X] */
#undef GPT_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR
#elif defined GPT_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/* User pragmas to end allocation to PFLASH[X] */
#undef GPT_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

/* Code [ PFLASH[X] ] */
#elif defined GPT_START_SEC_CODE_ASIL_B_GLOBAL
/* User pragmas to allocate to PFLASH[X] */
#undef GPT_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#elif defined GPT_STOP_SEC_CODE_ASIL_B_GLOBAL
/* User pragmas to end allocation to PFLASH[X] */
#undef GPT_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#endif

```

GPT driver

```
#if defined MEMMAP_ERROR
#error "Gpt_MemMap.h, wrong pragma command"
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The GPT driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the GPT driver must process all the errors reported to the DET module through the API `Det_ReportError()`.

The files `Det.h` and `Det.c` are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

DEM module is not required for the integration of the GPT driver.

- **SchM**

The SchM module is a part of the RTE that manages the BSW Scheduler. The GPT driver uses the exclusive areas defined in the `SchM_Gpt.h` file to protect the SFRs and variables from concurrent accesses from different threads. SchM is used for predef timers only. The SchMs identified for the GPT driver are:

- Get100UsPredefTimerValue
- Get1UsPredefTimerValue
- GtmStartTimer
- Gpt12StartTimer

The `SchM_Gpt.h` and `SchM_Gpt.c` files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the GPT driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows:

GPT driver

```
***** Sample implementation of SchM_Adc.c ****/
#include "Os.h"

void SchM_Enter_Gpt_Get1UsPredefTimerValue(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();/* Suspend CPU core interrupt */
}

void SchM_Exit_Gpt_Get1UsPredefTimerValue(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts();/* Resume CPU core interrupt */
}

void SchM_Enter_Gpt_Get100UsPredefTimerValue(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();/* Suspend CPU core interrupt */
}

void SchM_Exit_Gpt_Get100UsPredefTimerValue(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts();/* Resume CPU core interrupt */
}

void SchM_Enter_Gpt_GtmStartTimer(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();/* Suspend CPU core interrupt */
}

void SchM_Exit_Gpt_GtmStartTimer(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts();/* Resume CPU core interrupt */
}

void SchM_Enter_Gpt_Gpt12StartTimer(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();/* Suspend CPU core interrupt */
}

void SchM_Exit_Gpt_Gpt12StartTimer(void)
{
    /* End of Critical Section */
}
```

GPT driver

```
ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

The GPT driver will report all the detected safety errors through the API `Mcal_ReportSafetyError()`.

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The API `Mcal_ReportSafetyError()` is provided in the files `Mcal_SafetyError.c` and `Mcal_SafetyError.h` as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The GPT driver itself does not implement any notifications. However, GPT driver invokes the notification function configured for a GPT channel on timer count match. These notification functions can be configured by the user in Tresos for each GPT Channel(capable of issuing notifications) separately.

GPT does not provide any call-backs.

- **Operating system (OS)**

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. The enabling and disabling of interrupts must also be managed by the OS or the application.

The OS files provided by MCAL package is only an example code and must be updated by the integrator with the actual OS files for the desired function.

12.1.4.2 Multicore and resource manager

The GPT driver supports execution of its APIs simultaneously from all CPU cores. The user should allocate resources of TOM/ATOM to CPU cores at pre-compile time using the Resource manager module. The following are the key points to be considered with respect to multicore in the driver:

- GPT channels can be allocated to CPU cores at pre-compile time.
- Predefined timers are available only after master core initialization.
- GPT channel configured as Predef timer can be allocated to any core but will be controlled (started/ stopped) only by the logical master core. All slave core can access(/read) the Predef timer count.
- It must be ensured that GPT Channel passed as parameter while invoking an API belong to the same core on which the API is invoked.
- DETs will be raised in case APIs are invoked with mismatch of core and GPT Channel.
- GPT channels using GTM-TOM[i]_CH[X] and TOM[i]_CH[x+1] must be allocated to the same core as these two channels share the same interrupt line.
- GPT channels using GTM-ATOM[i]_CH[X] and ATOM[i]_CH[x+1] must be allocated to the same core as these two channels share the same interrupt line.
- Interrupts raised by individual TOM/ATOM/GPT12 channels must be serviced by the CPU core to which the TOM/ATOM/GPT12 channels have been allocated to.
- Locating constants, variables and configuration data to correct memory space should be done by the user. Memory sections are marked GLOBAL (common to all cores) and CORE[x](specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of the GPT driver is placed under single MemMap section. It can be relocated to any PFlash region.

Data section:

GPT driver

The RAM variable memory sections marked as specific to a core should be relocated to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region.

Configuration data and constants:

The configuration data sections marked as specific to a core should be relocated to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

Note: Relocating of code, data or constants to a distant memory region would impact execution timings.

Note: If the driver operates from single (master) core, all the sections may be relocated to the PFlash/DSPR/DLMU of the same CPU core.

12.1.4.3 MCU support

The GPT driver is dependent on the MCU driver for clock configuration and timer IP-related services. The initialization of GPT driver must be started only after completing the MCU initialization. The following must be considered while configuring the MCU driver in the EBtresos:

- GPT driver uses the services of the MCU to configure the GTM-related trigger events during runtime. The GTM-TOM/GTM-ATOM channels used by GPT driver must be reserved in the MCU configuration for exclusive use by the GPT driver.

12.1.4.4 Port support

The GPT driver does not use any services provided by the PORT driver.

12.1.4.5 DMA support

The GPT driver does not use any services provided by the DMA driver.

12.1.4.6 Interrupt connections

If wakeups/notifications are required, user should enable interrupts in the interrupt configuration register.

The interrupt configuration registers of different hardware used by GPT channels are given below:

Table 797 SRC registers

Hardware used	SRC register
GTM-TOM	SRC_GTMTOMwx (w= TOM module, x= 2n and 2n+1 TOM channel)
GTM-ATOM	SRC_GTMATOMwx (w= ATOM module, x= 2n and 2n+1 ATOM channel)
GPT12	SRC_GPT12_GPT120_Tx (x=2 to 6)

All the ISR to GTM-TOM, GTM-ATOM and GPT12 must be routed to the `Mcu_17_Gtm_TomChannelIsr`, `Mcu_17_Gtm_AtomChannelIsr` and `Mcu_17_Gpt12_ChannelIsr` APIs respectively which in turn invokes the `Gpt_Isr` API.

GPT driver

The user must ensure that the MCU interrupt handler is invoked from the ISR for which the sample code is shown as follows:

```
ISR(GTMTOM0SR0_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();

    /* Parameter is TOM Module 1 and Channel 5 */
    Mcu_17_Gtm_TomChannelIsr(1, 5);
}
```

```
ISR(GTMATOM0SR0_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();

    /* Parameter is ATOM Module 2 and Channel 6 */
    Mcu_17_Gtm_AtomChannelIsr(2, 6);
}
```

```
ISR(GPT12_T3_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();

    /* Parameter is GPT12 timer2 */
    Mcu_17_Gpt12_ChannelIsr(0);
}
```

GPT driver

12.1.4.7 Example usage

Pre-condition

For any functionality of GPT, MCU should be initialized, all the clock configurations should be completed and PLL clock should be distributed to all the peripherals.

Example initialization sequence is shown as follows.

```
/*Initialize Mcu */
Mcu_Init(&Mcu_Config);

/* Initialize the PLLs and other MCU specific clock options */
(void)Mcu_InitClock( 0 );

while(Mcu_GetPllStatus() != MCU_PLL_LOCKED)
{
};

/* Select the PLL clock as source for MCU clock tree distribution */
(void)Mcu_DistributePllClock();
```

Initialization

User must include Gpt_PBcfg.h to access the GPT configuration structure needed for initialization.

```
/* Include Gpt.h to access interfaces of GPT driver */
#include "Gpt.h"

/* Module Initialization */
void Gpt_Sample_Init(void)
{
    /* Initialize Gpt*/
    Gpt_Init(&Gpt_Config);
}
```

Start and stop timer

GPT driver

The user must call `Gpt_StartTimer` API with the logical channel ID and the target time for any normal (non-predefined timer) channel. The timer can be stopped by invoking the `Gpt_StopTimer` API.

```
/* GPT module Initialization is necessary to start timer */
Gpt_Init(&Gpt_Config);

/* Start the timer */
Gpt_StartTimer(<LogicalChannelSymbolicName>, <targetCount>);

/* Count the number of ticks currently */
Gpt_ValueType timeElapsed = Gpt_GetTimeElapsed(<LogicalChannelSymbolicName>);

/* Count the number of ticks remaining to reach the target count */
Gpt_ValueType timeRemaining =
Gpt_GetTimeRemaining(<LogicalChannelSymbolicName>);

/* Stop the timer */
Gpt_StopTimer(<LogicalChannelSymbolicName>);
```

General API

`Gpt_EnableNotification` and `Gpt_DisableNotification` should be invoked after calling `Gpt_Init` and only on a channel that has a valid notification function.

`Gpt_EnableWakeup` and `Gpt_DisableWakeup` should be invoked after the call to `Gpt_Init` and only on a wakeup capable channel.

`Gpt_SetMode` should be invoked after the call to `Gpt_Init` to change the state of GPT driver to SLEEP or NORMAL.

GPT driver

`Gpt_DeInit`, should be invoked after the call to `Gpt_Init` to reset the initialization state of the Gpt driver. After the call to `Gpt_DeInit`, `Gpt_Init` should be invoked again to start any functionality of the GPT driver.

```

/* GPT module Initialization is necessary to start timer */
Gpt_Init(&Gpt_Config);

/* Enable wakeup for a wakeup capable channel */
Gpt_EnableWakeup(<WakeupCapableLogicalChannelSymbolicName>);

/* Change mode to sleep */
Gpt_SetMode(GPT_MODE_SLEEP);

/* provide signal on the wakeup capable channel */
/* EcuMCheckWakeups will be invoked from ISR to indicate a wakeup signal */

/* Change mode to normal */
Gpt_SetMode(GPT_MODE_NORMAL);

/* Disable wakeup for a wakeup capable channel */
Gpt_DisableWakeup(<WakeupCapableLogicalChannelSymbolicName>>);

/* Change mode to sleep */
Gpt_SetMode(GPT_MODE_SLEEP);

/* provide signal on the wakeup capable channel */
/* EcuMCheckWakeups will not be invoked */

```

12.1.5 Key architectural considerations

12.1.5.1 Hardware dependency

The hardware provides GTM (TOM and ATOM slice) and GPT12 timers, which are used as the underlying hardware for GPT. As all TOM, ATOM and GPT12 channels are independent, each channel can be mapped to one of the cores. GTM (TOM and ATOM Slice) is not available for the TC35xx devices.

12.1.5.2 User mode support

The GPT driver supports the User-1 and Supervisor modes. Additional configuration parameters are not provided to switch the modes.

[cover parentID GPT={1A65EADD-AFD0-4845-B2D2-8257E086DD67}]

GPT driver

12.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Configuration check**

User shall ensure that the generated configuration is correct against the GUI configurations provided by the user.

[cover parentID GPT={6223236A-4BFD-40eb-812A-2AE1B0A896DD}]

- **Disabling the interrupts in critical section**

User shall protect the read sequence of Predef timer with resolution more than 16 bits by disabling the interrupts in the critical section Get1UsPredefTimerValue and Get100UsPredefTimerValue.

[cover parentID GPT={F38180FC-D9DF-4e32-9DD1-37B3464914AB}]

- **Execution sequence for initialization check**

User shall invoke the Gpt_InitCheck API only after invoking the Gpt_Init API, and before invoking any runtime API other than the Gpt_GetVersionInfo API.

[cover parentID GPT={B75D5D36-FD97-4130-9A97-7A58DADDAB11}]

- **Freedom from interference for MCAL data**

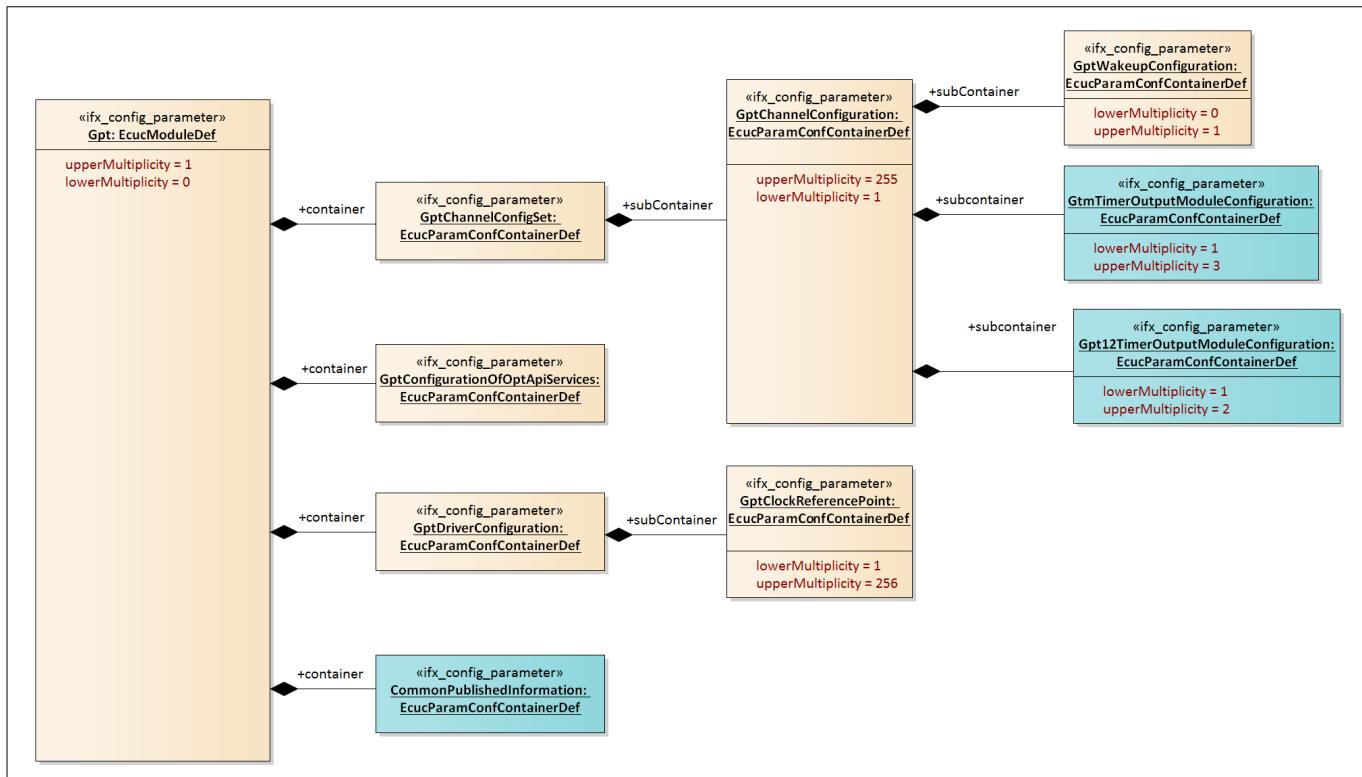
User shall provide protection for the MCAL memory and SFRs from the QM software.

[cover parentID GPT={53C19FC7-7847-481a-B828-3CC536391721}]

- **Handling of underflow situation for one-shot mode with GPT12**

The Gpt_GetTimeElapsed and Gpt_GetTimeRemaining APIs for channels using GPT12 auxillary timers should not be invoked from interrupts, which have higher priority than the particular GPT12 channel interrupt.

[cover parentID GPT={DEE11E9A-DEE3-4e23-93E5-8591E2F7235D}]

GPT driver**12.3 Reference information****12.3.1 Configuration interfaces****Figure 128 Container hierarchy along with their configuration parameters****12.3.1.1 Container: CommonPublishedInformation**

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

12.3.1.1.1 ArMajorVersion**Table 798 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-

GPT driver**Table 798 Specification for ArMajorVersion (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.1.2 ArMinorVersion**Table 799 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.1.3 ArPatchVersion**Table 800 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

GPT driver**Table 800 Specification for ArPatchVersion (continued)**

Dependency	-
-------------------	---

12.3.1.1.4 ModuleId**Table 801 Specification for ModuleId**

Name	ModuleId		
Description	Module ID of this module from Module List		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 65535		
Default value	100		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.1.5 Release**Table 802 Specification for Release**

Name	Release		
Description	Aurix derivative used for the implementation Note: Default value will be selected based on the target device to ensure the relevance.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	_TRICORE_TC3xx (xx will be replaced as per target device. Ex: For TC399 xx will be 99)		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

GPT driver**12.3.1.1.6 SwMajorVersion****Table 803 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.		
Multiplicity	1..1	Type	EcuclIntegerParamDef
Range	0 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.1.7 SwMinorVersion**Table 804 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.		
Multiplicity	1..1	Type	EcuclIntegerParamDef
Range	0 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.1.8 SwPatchVersion**Table 805 Specification for SwPatchVersion**

Name	SwPatchVersion	
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.	

GPT driver**Table 805 Specification for SwPatchVersion (continued)**

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.1.9 VendorId**Table 806 Specification for VendorId**

Name	VendorId		
Description	This parameter provides the Vendor Id		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.2 Container: Gpt12TimerOutputModuleConfiguration

This container contains the configuration elements for configuring GPT channel of GPT12 timer hardware and the input clock divider selection for the respective timer. For One-shot mode, the multiplicity is 1..1 For Continuous mode the multiplicity is 1..2

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

12.3.1.2.1 Gpt12ChannelClockDivider**Table 807 Specification for Gpt12ChannelClockDivider**

Name	Gpt12ChannelClockDivider
-------------	--------------------------

GPT driver**Table 807 Specification for Gpt12ChannelClockDivider (continued)**

Description	This parameter decides the clock divider value for GPT12 timer. Default value is zero because the default value of the corresponding register bits are zero.		
Multiplicity	0..7	Type	EcucReferenceDef
Range	Reference to Node:		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.2.2 Gpt12TimerUsed**Table 808 Specification for Gpt12TimerUsed**

Name	Gpt12TimerUsed		
Description	This parameter is defined by IFX. It is the reference to GPT12 timer used by GPT Driver. Note: Referred timer channel in MCU should have timer usage as GPT_TIMER_USED_BY_GPT_DRIVER. Since the name of the dependent container is user configurable, the default value is kept as NULL		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node:		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.3 Container: GtmTimerOutputModuleConfiguration

This container contains the configuration elements for configuring TOM/ATOM channel of GTM timer hardware and the input clock selection for the respective timer. For Continuous mode and One-shot mode, the multiplicity is 1..1 For Predef timers, the multiplicity is 2..3 This container is not configurable for TC35x.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

GPT driver**12.3.1.3.1 GtmTimerClockSelect****Table 809 Specification for GtmTimerClockSelect**

Name	GtmTimerClockSelect		
Description	<p>This parameter decides the Clock Source for TOM/ATOM timer.</p> <p>Default value: GTM_FIXED_CLOCK_0 selected because TOM channels will run with fixed clock.</p> <p>This parameter is not configurable for TC35x.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>GTM_CONFIGURABLE_CLOCK_X: Configurable Clock X is selected for ATOM module, where X = 0 to 7</p> <p>GTM_FIXED_CLOCK_0: Fixed Clock 0 is selected for TOM module.</p> <p>GTM_FIXED_CLOCK_1: Fixed Clock 1 is selected for TOM module.</p> <p>GTM_FIXED_CLOCK_2: Fixed Clock 2 is selected for TOM module.</p> <p>GTM_FIXED_CLOCK_3: Fixed Clock 3 is selected for TOM module.</p> <p>GTM_FIXED_CLOCK_4: Fixed Clock 4 is selected for TOM module.</p>		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerUsed, GptTimerChannelUsage		

12.3.1.3.2 GtmTimerUsed**Table 810 Specification for GtmTimerUsed**

Name	GtmTimerUsed		
Description	<p>This parameter is defined by IFX. It is the reference to GTM timer channel (TOM/ATOM) used by GPT Driver.</p> <p>Note: Referred timer channel in MCU should have TomChannelUsage/ AtomChannelUsage as GTM_TOM_CHANNEL_USED_BY_GPT / GTM_ATOM_CHANNEL_USED_BY_GPT.</p> <p>Since the name of the dependent container is user configurable, the default value is kept as NULL</p> <p>This parameter is not configurable for TC35x.</p>		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuGtmAtomChannelAllocationConf, McuGtmTomChannelAllocationConf		

GPT driver**Table 810 Specification for GtmTimerUsed (continued)**

Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.4 Container: Gpt

Configuration of the GPT (General Purpose Timer) driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

12.3.1.4.1 Config Variant**Table 811 Specification for Config Variant**

Name	Config Variant		
Description	Selects the config-variant for the GPT driver		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VariantPostBuild: Post Build Support		
Default value	VariantPostBuild		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.5 Container: GptChannelConfigSet

This container is the base of a Configuration Set which contains the configured GPT channels. This way, different configuration sets can be defined for the post-build process.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

GPT driver

12.3.1.6 Container: GptChannelConfiguration

Configuration of an individual GPT channel

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

12.3.1.6.1 GptAssignedHwUnit

Table 812 Specification for GptAssignedHwUnit

Name	GptAssignedHwUnit		
Description	Specifies the hardware used for the GPT channel(GPT12 or GTM). By default this will contain GPT12 since it is present across all TC3xx controller variants.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GPT12: The GPT channel will be realized using GPT12 hardware. GTM: The GPT channel will be realized using GTM(ATOM/TOM) hardware.		
Default value	GPT12		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.6.2 GptChannelClkSrcRef

Table 813 Specification for GptChannelClkSrcRef

Name	GptChannelClkSrcRef		
Description	Reference to the GptClockReferencePoint from which the clock for the channel is derived.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: GptClockReferencePoint		
Default value	NA		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

GPT driver**12.3.1.6.3 GptChannelId****Table 814 Specification for GptChannelId**

Name	GptChannelId		
Description	Channel Id of the GPT channel. This value will be assigned to the symbolic name derived from the GptChannelConfiguration container short name. The value should be unique and continuous starting from 0. Note: Default value is set to minimum value.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - (Total number of channels - 1)		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.6.4 GptChannelMode**Table 815 Specification for GptChannelMode**

Name	GptChannelMode		
Description	Specifies the behavior of the timer channel after the target time is reached. By default this will refer to continues mode to ensure that the timer is running all the times.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GPT_CH_MODE_CONTINUOUS: After reaching the target time, the timer continues running with the value "zero" again. GPT_CH_MODE_ONESHOT: After reaching the target time, the timer stops automatically (timer expired).		
Default value	GPT_CH_MODE_CONTINUOUS		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	GptTimerChannelUsage		

GPT driver**12.3.1.6.5 GptChannelTickFrequency****Table 816 Specification for GptChannelTickFrequency**

Name	GptChannelTickFrequency		
Description	<p>Specifies the tick frequency of the timer channel in Hz. This is not used for implementation and disabled for the configuration GUI.</p> <p>The GPT channel tick value and tick frequency are handled internally with the internal pre-scalars and dividers coming from CMU. Therefore these configurations should be done at the MCU module and they will not be available at GPT channel configurations.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0 - 65535		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.6.6 GptChannelTickCountMax**Table 817 Specification for GptChannelTickCountMax**

Name	GptChannelTickCountMax		
Description	<p>Maximum value in ticks, the timer channel is able to count. With the next tick, the timer rolls over to zero. This is not used for implementation and disabled for the configuration GUI.</p> <p>The GPT channel tick value and tick frequency are handled internally with the internal pre-scalars and dividers coming from CMU. Therefore these configurations should be done at the MCU module and they will not be available at GPT channel configurations.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 18446744073709551615		
Default value	65535		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

GPT driver**12.3.1.6.7 GptEnableWakeup****Table 818 Specification for GptEnableWakeup**

Name	GptEnableWakeup		
Description	Enables wakeup capability of MCU for a channel. Note: Not applicable if the channel is configured for Predef Timer. The optional features are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	GptPredefTimer1usEnablingGrade, GptPredefTimer100us32bitEnable, GptReportWakeupSource		

12.3.1.6.8 GptNotification**Table 819 Specification for GptNotification**

Name	GptNotification		
Description	The GptNotification is used by the GPT driver to invoke the user-defined function for notification purposes(for non-wakeup notification). The parameter can be a name or the address(numeric value) of the notification function. Note1: The GPT driver does not validate the configured function name or address for correctness and the responsibility falls on the user. Note2: Notification is not applicable if the channel is configured for Predef Timer.		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	IoHwAb_GptNotification<#channel>		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL

GPT driver**Table 819 Specification for GptNotification (continued)**

Dependency	GptTimerChannelUsage, GptEnableDisableNotificationApi, GptPredefTimer1usEnablingGrade, GptPredefTimer100us32bitEnable
-------------------	--

12.3.1.6.9 GptTimerChannelUsage**Table 820 Specification for GptTimerChannelUsage**

Name	GptTimerChannelUsage		
Description	<p>Specifies the usage of the timer channel as a predefined timer or normal GPT timer.</p> <p>By default it will be normal channel because a normal channel can be used for all possible configurations (free running / OSM etc).</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>GPT_PREDEF_TIMERCH_100US_32BIT: Channel is selected as Gpt Predef Timer with tick duration 100 us and range 32bit.</p> <p>This literal cannot be used for TC35x devices.</p> <p>GPT_PREDEF_TIMERCH_1US_16BIT: Channel is selected as Gpt Predef Timer with tick duration 1 us and range 16 bit.</p> <p>This literal cannot be used for TC35x devices.</p> <p>GPT_PREDEF_TIMERCH_1US_16_24BIT: Channel is selected as Gpt Predef Timer with tick duration 1 us and range 24bit.</p> <p>This literal cannot be used for TC35x devices.</p> <p>GPT_PREDEF_TIMERCH_1US_16_24_32BIT: Channel selected as Gpt Predef Timer with tick duration 1 us and range 32bit.</p> <p>This literal cannot be used for TC35x devices.</p> <p>GPT_TIMER_CHANNEL_NORMAL: Channel is selected as normal Gpt timer</p>		
Default value	GPT_TIMER_CHANNEL_NORMAL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.7 Container: GptClockReferencePoint

This container contains a parameter, which represents a reference to a container of the type McuClockReferencePoint (defined in module MCU). A container is needed to support multiple clock references (hardware dependent). Since the name of the dependent container is user configurable, the default value is kept as NULL

Post-Build Variant Multiplicity: FALSE

GPT driver

Multiplicity Configuration Class: Post-Build

12.3.1.7.1 GptClockReference

Table 821 Specification for GptClockReference

Name	GptClockReference		
Description	<p>Reference to a container of the type McuClockReferencePoint, to select an input clock.</p> <p>The configuration editor for the GPT driver can support the integrator by only allowing a selection of those clock reference points that can be connected physically to the GPT hardware peripheral.</p> <p>The desired frequency (desired by the GPT driver) has to be the same as the selected and provided frequency of the MCU configuration. This has to be checked automatically.</p> <p>Since the name of the dependent container is user configurable, the default value is kept as NULL</p>		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NA		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.8 Container: GptConfigurationOfOptApiServices

This container contains all configuration switches for configuring optional API services of the GPT driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

12.3.1.8.1 GptDeinitApi

Table 822 Specification for GptDeinitApi

Name	GptDeinitApi		
Description	<p>Adds/removes the service Gpt_Deinit() from the code.</p> <p>Note: The optional APIs are disabled by default to minimize the executable code size</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		

GPT driver**Table 822 Specification for GptDeinitApi (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.8.2 GptEnableDisableNotificationApi**Table 823 Specification for GptEnableDisableNotificationApi**

Name	GptEnableDisableNotificationApi		
Description	Adds/removes the services Gpt_EnableNotification() and Gpt_DisableNotification from the code. Note: The optional APIs are disabled by default to minimize the executable code size		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.8.3 GptInitCheckApi**Table 824 Specification for GptInitCheckApi**

Name	GptInitCheckApi		
Description	Adds/removes the service Gpt_InitCheck() from the code. Note: The detection of safety related errors is enable by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		

GPT driver**Table 824 Specification for GptInitCheckApi (continued)**

Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.8.4 GptTimeElapsedApi**Table 825 Specification for GptTimeElapsedApi**

Name	GptTimeElapsedApi		
Description	Adds/removes the service Gpt_GetTimeElapsed() from the code. Note: The optional APIs are disabled by default to minimize the executable code size		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.8.5 GptTimeRemainingApi**Table 826 Specification for GptTimeRemainingApi**

Name	GptTimeRemainingApi		
Description	Adds/removes the service Gpt_GetTimeRemaining() from the code. Note: The optional APIs are disabled by default to minimize the executable code size		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		

GPT driver**Table 826 Specification for GptTimeRemainingApi (continued)**

Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.8.6 GptVersionInfoApi**Table 827 Specification for GptVersionInfoApi**

Name	GptVersionInfoApi		
Description	Adds/removes the service Gpt_GetVersionInfo() from the code. Note: The optional APIs are disabled by default to minimize the executable code size		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.8.7 GptWakeupFunctionalityApi**Table 828 Specification for GptWakeupFunctionalityApi**

Name	GptWakeupFunctionalityApi		
Description	Adds/removes the services Gpt_SetMode(), Gpt_EnableWakeup(), Gpt_DisableWakeup() and Gpt_CheckWakeup() from the code. Note: The optional APIs are disabled by default to minimize the executable code size		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE		

GPT driver**Table 828 Specification for GptWakeupFunctionalityApi (continued)**

	FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.9 Container: GptDriverConfiguration

This container contains the module-wide configuration (parameters) of the GPT driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

12.3.1.9.1 GptDevErrorDetect**Table 829 Specification for GptDevErrorDetect**

Name	GptDevErrorDetect		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.1.9.2 GptMultiCoreErrorDetect**Table 830 Specification for GptMultiCoreErrorDetect**

Name	GptMultiCoreErrorDetect
-------------	-------------------------

GPT driver**Table 830 Specification for GptMultiCoreErrorDetect (continued)**

Description	This parameter enables or disables the Multi core related default error tracer (Det) detection and reporting. It is applicable only when DETs are enabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GptDevErrorDetect		

12.3.1.9.3 GptPredefTimer100us32bitEnable**Table 831 Specification for GptPredefTimer100us32bitEnable**

Name	GptPredefTimer100us32bitEnable		
Description	Enables/disables the GPT Predef Timer 100us 32bit. The optional APIs are disabled by default to minimize the executable code size. This parameter is not configurable for TC35x.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

12.3.1.9.4 GptPredefTimer1usEnablingGrade**Table 832 Specification for GptPredefTimer1usEnablingGrade**

Name	GptPredefTimer1usEnablingGrade
-------------	--------------------------------

GPT driver**Table 832 Specification for GptPredefTimer1usEnablingGrade (continued)**

Description	Specifies the grade of enabling the GPT Predef Timers with 1 us tick duration. The optional APIs are disabled by default to minimize the executable code size. This parameter is not configurable for TC35x.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GPT_PREDEF_TIMER_1US_16BIT_ENABLED: 16bit timer enabled GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED: 16 and 24bit timers enabled GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED: 16, 24 and 32bit timers enabled GPT_PREDEF_TIMER_1US_DISABLED: Specifies that GPT Predef Timers with 1 us tick duration is disabled.		
Default value	GPT_PREDEF_TIMER_1US_DISABLED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

12.3.1.9.5 GptReportWakeupSource**Table 833 Specification for GptReportWakeupSource**

Name	GptReportWakeupSource		
Description	Enables/Disables wakeup source reporting. The optional features are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

GPT driver**12.3.1.9.6 GptSafetyEnable****Table 834 Specification for GptSafetyEnable**

Name	GptSafetyEnable		
Description	Switch to enable/disable the safety check and reporting. Note: The detection of safety related errors are enabled by default to ensure the safety issues are addressed during the product life cycle.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

12.3.1.10 Container: GptWakeupConfiguration

Function pointer to callback function (for wakeup notification).

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

12.3.1.10.1 GptWakeupSourceRef**Table 835 Specification for GptWakeupSourceRef**

Name	GptWakeupSourceRef		
Description	In case the wakeup-capability is true this value is transmitted to the ECU State Manager. Since the name of the dependent container is user configurable, the default value is kept as NULL		
Multiplicity	1..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: EcuMWakeupSource		
Default value	NA		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

GPT driver

Table 835 Specification for GptWakeupSourceRef (continued)

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

12.3.2 Functions - Type definitions

12.3.2.1 Gpt_ChannelType

Table 836 Specification for Gpt_ChannelType

Syntax	Gpt_ChannelType				
Type	uint8				
File	Gpt.h				
Range	0 - 196	The range is based on the number of TOM channels, ATOM channels and GPT12 timer channels for the device variant. The maximum number of channels will vary depending on the device variant. 197 is for the superset device variant.			
Description	Numeric ID of a GPT channel. Note: Maximum number of channels depends on the device.				
Source	AUTOSAR				

12.3.2.2 Gpt ModeType

Table 837 Specification for Gpt ModeType

Syntax	Gpt_ModeType	
Type	Enumeration	
File	Gpt.h	
Range	1 - GPT_MODE_NORMAL	Normal operation mode of the GPT
	0 - GPT_MODE_SLEEP	Operation for reduced power operation mode. In sleep mode only wakeup capable channels are available.
Source	AUTOSAR	

12.3.2.3 Gpt NotificationPtrType

Table 838 Specification for Gpt NotificationPtrType

Syntax Gpt NotificationPtrType

GPT driver**Table 838 Specification for Gpt_NotificationPtrType (continued)**

Type	Pointer to a function of type void Function_Name (void)
File	Gpt.h
Description	Timer Channel notification function pointer type.
Source	IFX

12.3.2.4 Gpt_PrefdefTimerType**Table 839 Specification for Gpt_PrefdefTimerType**

Syntax	Gpt_PrefdefTimerType	
Type	Enumeration	
File	Gpt.h	
Range	0 - GPT_PREDEF_TIMER_1US_16BIT	GPT Predef Timer with tick duration 1µs and range 16bit
	1 - GPT_PREDEF_TIMER_1US_24BIT	GPT Predef Timer with tick duration 1µs and range 24bit
	2 - GPT_PREDEF_TIMER_1US_32BIT	GPT Predef Timer with tick duration 1µs and range 32bit
	3 - GPT_PREDEF_TIMER_100US_32BIT	GPT Predef Timer with tick duration 100µs and range 32bit
Description	Type for GPT Predef Timers	
Source	AUTOSAR	

12.3.2.5 Gpt_ConfigType**Table 840 Specification for Gpt_ConfigType**

Syntax	Gpt_ConfigType	
Type	Structure	
File	Gpt.h	
Range	-[]	The elements of the data structure are specific to the micro-controller.
Description	This is the type of the data structure including the configuration set required for initializing the GPT timer unit.	
Source	AUTOSAR	

12.3.2.6 Gpt_ValueType**Table 841 Specification for Gpt_ValueType**

Syntax	Gpt_ValueType
---------------	---------------

GPT driver**Table 841 Specification for Gpt_ValueType (continued)**

Type	uint32	
File	Gpt.h	
Range	0 - 0xFFFFFFF	Maximum supported timer value/setting periodic timer values (in number of ticks).
Description	Type for reading and setting the timer values (in number of ticks).	
Source	AUTOSAR	

12.3.3 Functions - APIs**12.3.3.1 Gpt_GetVersionInfo****Table 842 Specification for Gpt_GetVersionInfo API**

Syntax	<pre>void Gpt_GetVersionInfo (Std_VersionInfoType * const VersionInfoPtr)</pre>	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer where the version information of this driver is stored.
Parameters (in - out)	-	-
Return	void	-
Description	The function provides the version information of the GPT driver.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>GPT_E_PARAM_POINTER: Error reported when the parameter passed is a null pointer.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	GptVersionInfoApi	

GPT driver**Table 842 Specification for Gpt_GetVersionInfo API (continued)**

User hints	None
-------------------	------

12.3.3.2 Gpt_Init**Table 843 Specification for Gpt_Init API**

Syntax	<pre>void Gpt_Init (const Gpt_ConfigType * const ConfigPtr)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to a selected configuration structure
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function initializes the GPT driver in the context of the core from where the API is invoked.</p> <p>Note: Initialization should be performed in the following sequence:</p> <ol style="list-style-type: none"> 1. Initialize Gpt driver from the master core. 2. Initialize Gpt driver from all applicable slave cores. 	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>GPT_E_ALREADY_INITIALIZED: Error reported when the driver is already initialized for the current core.</p> <p>GPT_E_INIT_FAILED: Error reported when Gpt_Init is called with null pointer as argument.</p> <p>GPT_E_MASTER_UNINIT: Error reported when slave core initializing is called without initializing Master core.</p> <p>GPT_E_NOT_CONFIGURED: Error reported when GPT driver is not configured for the core from where the API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	

GPT driver**Table 843 Specification for Gpt_Init API (continued)**

User hints	None
-------------------	------

12.3.3.3 Gpt_DeInit**Table 844 Specification for Gpt_DeInit API**

Syntax	void Gpt_DeInit (void)	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function de-initializes the GPT driver in the context of core from where this API is invoked.</p> <p>Note: All slave cores must be de-initialized before the master core.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked.</p> <p>GPT_E_BUSY: Error reported when timer channel is still busy (running).</p> <p>GPT_E_SLAVE_INIT: Error reported when master de-Initialization is called without de-Initializing slave core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	GptDeinitApi	
User hints	None	

GPT driver**12.3.3.4 Gpt_StartTimer****Table 845 Specification for Gpt_StartTimer API**

Syntax	<pre>void Gpt_StartTimer (const Gpt_ChannelType Channel, const Gpt_ValueType Value)</pre>	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Channel Value	Numeric identifier of the GPT channel. Target time in number of ticks.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function starts the selected timer 'Channel' with a defined target time, 'Value', only if the channel is available in the core from where this API is invoked.</p> <p>Note: The maximum range for the target time, 'Value', depends on the type of the timer channel.</p> <p>If the 'Channel' is:</p> <ol style="list-style-type: none"> 1. TOM channel, the MaxValue is 0xFFFF (16 Bit). 2. ATOM channel, the MaxValue is 0xFFFFFFFF (24 Bit). 	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked.</p> <p>GPT_E_PARAM_VALUE: Error reported when the value is not within the allowed range.</p> <p>GPT_E_PARAM_CHANNEL: Error reported when the parameter 'Channel' is invalid.</p> <p>GPT_E_BUSY: Error reported when timer channel is still busy (running).</p> <p>GPT_E_CORE_CHANNEL_MISMATCH: The parameter 'Channel' is not configured for the core from where the API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>GPT_E_MODE: Error reported when the driver is in sleep mode</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

GPT driver**Table 845 Specification for Gpt_StartTimer API (continued)**

Configuration dependencies	-
User hints	None

12.3.3.5 Gpt_StopTimer**Table 846 Specification for Gpt_StopTimer API**

Syntax	<pre>void Gpt_StopTimer (const Gpt_ChannelType Channel)</pre>	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The function stops the selected timer 'Channel', only if the channel is available in the core from where this API is invoked.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked.</p> <p>GPT_E_PARAM_CHANNEL: Error reported when the parameter 'Channel' is invalid.</p> <p>GPT_E_CORE_CHANNEL_MISMATCH: The parameter 'Channel' is not configured for the core from where the API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

GPT driver**12.3.3.6 Gpt_GetTimeElapsed****Table 847 Specification for Gpt_GetTimeElapsed API**

Syntax	<pre>Gpt_ValueType Gpt_GetTimeElapsed (const Gpt_ChannelType Channel)</pre>	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Gpt_ValueType	Remaining timer value (in number of ticks).
Description	The function provides the time elapsed in ticks since the previous timer overflow/start, only if the channel is available in the core from where this API is invoked.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked.</p> <p>GPT_E_PARAM_CHANNEL: Error reported when the parameter 'Channel' is invalid.</p> <p>GPT_E_CORE_CHANNEL_MISMATCH: The parameter 'Channel' is not configured for the core from where the API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	GptTimeElapsedApi	
User hints	None	

12.3.3.7 Gpt_GetTimeRemaining**Table 848 Specification for Gpt_GetTimeRemaining API**

Syntax	<pre>Gpt_ValueType Gpt_GetTimeRemaining (const Gpt_ChannelType Channel)</pre>	
---------------	---	--

GPT driver**Table 848 Specification for Gpt_GetTimeRemaining API (continued)**

Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Gpt_ValueType	Remaining timer value (in number of ticks).
Description	The function returns the time remaining until the target time is reached, only if the channel is available in the core from where this API is invoked..	
Source	AUTOSAR	
Error handling	DET: GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked. GPT_E_PARAM_CHANNEL: Error reported when the parameter 'Channel' is invalid. GPT_E_CORE_CHANNEL_MISMATCH: The parameter 'Channel' is not configured for the core from where the API is invoked. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	GptTimeRemainingApi	
User hints	None	

12.3.3.8 Gpt_GetPredefTimerValue**Table 849 Specification for Gpt_GetPredefTimerValue API**

Syntax	<pre>Std_ReturnType Gpt_GetPredefTimerValue (const Gpt_PredefTimerType PredefTimer, uint32 * const TimeValuePtr)</pre>
Service ID	0x0D
Sync/Async	Synchronous
ASIL Level	B

GPT driver**Table 849 Specification for Gpt_GetPredefTimerValue API (continued)**

Re-entrancy	Reentrant	
Parameters (in)	PredefTimer	GPT Predef Timer
Parameters (out)	TimeValuePtr	Pointer to time value destination data in RAM
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: No error has been detected E_NOT_OK: Aborted due to errors
Description	The function delivers the current value of the desired GPT Predef Timer. This function can be invoked from any core.	
Source	AUTOSAR	
Error handling	DET: GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked. GPT_E_PARAM_PREDEF_TIMER: Error reported when the parameter Predef Timer is invalid. GPT_E_PARAM_POINTER: Error reported when the parameter passed is a null pointer. Runtime Errors: None DEM: None Safety Errors: GPT_E_MODE: Error reported when the driver in sleep mode <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

12.3.3.9 Gpt_EnableNotification**Table 850 Specification for Gpt_EnableNotification API**

Syntax	<pre>void Gpt_EnableNotification (const Gpt_ChannelType Channel)</pre>	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Channel	Numeric identifier of the GPT channel

GPT driver**Table 850 Specification for Gpt_EnableNotification API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The function enables reporting of Notification for the GPT channel.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked.</p> <p>GPT_E_PARAM_CHANNEL: Error reported when the parameter 'Channel' is invalid.</p> <p>GPT_E_CORE_CHANNEL_MISMATCH: The parameter 'Channel' is not configured for the core from where the API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	GptEnableDisableNotificationApi	
User hints	None	

12.3.3.10 Gpt_DisableNotification**Table 851 Specification for Gpt_DisableNotification API**

Syntax	<pre>void Gpt_DisableNotification (const Gpt_ChannelType Channel)</pre>	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

GPT driver**Table 851 Specification for Gpt_DisableNotification API (continued)**

Description	The function disables the interrupt notification for a channel (relevant in normal mode).
Source	AUTOSAR
Error handling	<p>DET:</p> <p>GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked.</p> <p>GPT_E_PARAM_CHANNEL: Error reported when the parameter 'Channel' is invalid.</p> <p>GPT_E_CORE_CHANNEL_MISMATCH: The parameter 'Channel' is not configured for the core from where the API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	GptEnableDisableNotificationApi
User hints	None

12.3.3.11 Gpt_EnableWakeup**Table 852 Specification for Gpt_EnableWakeup API**

Syntax	<pre>void Gpt_EnableWakeup (const Gpt_ChannelType Channel)</pre>	
Service ID	0x0B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The function enables wakeup capability for the GPT Channel.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked.</p>	

GPT driver**Table 852 Specification for Gpt_EnableWakeup API (continued)**

	<p>GPT_E_PARAM_CHANNEL: Error reported when the parameter 'Channel' is invalid.</p> <p>GPT_E_CORE_CHANNEL_MISMATCH: The parameter 'Channel' is not configured for the core from where the API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	GptWakeupFunctionalityApi,GptReportWakeupSource
User hints	None

12.3.3.12 Gpt_DisableWakeup**Table 853 Specification for Gpt_DisableWakeup API**

Syntax	<pre>void Gpt_DisableWakeup (const Gpt_ChannelType Channel)</pre>	
Service ID	0x0A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The function disables the wakeup interrupt of the channel (relevant in sleep mode only).	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked.</p> <p>GPT_E_PARAM_CHANNEL: Error reported when the parameter 'Channel' is invalid.</p> <p>GPT_E_CORE_CHANNEL_MISMATCH: The parameter 'Channel' is not configured for the core from where the API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p>	

GPT driver**Table 853 Specification for Gpt_DisableWakeup API (continued)**

	Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	GptReportWakeupSource
User hints	This API has no impact if invoked in Normal Mode.

12.3.3.13 Gpt_CheckWakeup**Table 854 Specification for Gpt_CheckWakeup API**

Syntax	void Gpt_CheckWakeup (const EcuM_WakeupSourceType WakeupSource)	
Service ID	0x0C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	WakeupSource	Information on wakeup source to be checked. The associated GPT channel can be determined from configuration data.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The function checks if a wakeup capable GPT channel is the source for a wakeup event and calls the ECU state manager service, EcuM_SetWakeupEvent, in case of a valid GPT channel wakeup event.	
Source	AUTOSAR	
Error handling	DET: GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	GptWakeupFunctionalityApi,GptReportWakeupSource	
User hints	None	

GPT driver**12.3.3.14 Gpt_SetMode****Table 855 Specification for Gpt_SetMode API**

Syntax	<pre>void Gpt_SetMode (const Gpt_ModeType Mode)</pre>	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Mode	GPT_MODE_NORMAL: Normal operation mode of the GPT driver GPT_MODE_SLEEP: Sleep mode of the GPT driver (wakeup capable).
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The function sets the mode of the GPT Driver to 'SLEEP' mode or 'NORMAL' mode.	
Source	AUTOSAR	
Error handling	<p>DET: GPT_E_PARAM_MODE: Error reported when Gpt_SetMode is called with invalid parameter (Mode). GPT_E_UNINIT: Error reported when the driver is not initialized in context to the core from where the API is invoked. Runtime Errors: None DEM: None Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	GptWakeupFunctionalityApi,GptReportWakeupSource	
User hints	For Predef timers only the master can start or stop the timers.	

12.3.3.15 Gpt_InitCheck**Table 856 Specification for Gpt_InitCheck API**

Syntax	<pre>Std_ReturnType Gpt_InitCheck (const Gpt_ConfigType ConfigPtr)</pre>
---------------	--

GPT driver**Table 856 Specification for Gpt_InitCheck API (continued)**

Service ID	0x0E	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	ConfigPtr	Pointer to a selected configuration structure
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK - Initialization comparison is success E_NOT_OK - Initialization comparison failed
Description	<p>The function verifies the initialization the GPT driver in context to the core from where the API is invoked.</p> <p>Note: Init check should be performed in the following sequence:</p> <ol style="list-style-type: none"> 1. Call Gpt_Init from a core (master/slave core). 2. Call Gpt_InitCheck from the same core. 	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	GptInitCheckApi	
User hints	None	

12.3.4 Notifications and Callbacks**12.3.4.1 Gpt_Isr****Table 857 Specification for Gpt_Isr API**

Syntax	<pre>void Gpt_Isr (const Gpt_ChannelType LogicalChId, const uint32 StatusFlags)</pre>
Service ID	0x0F
Sync/Async	Synchronous

GPT driver
Table 857 Specification for Gpt_Isr API (continued)

ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	LogicalChild StatusFlags	GPT channel number This flag is not used.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The ISR function is used to call the user-defined notification function as well as the Ecum_CheckWakeup function for wakeup notification.	
Source	IFX	
Error handling	DET: GPT_E_NOT_CONFIGURED: Error reported when GPT driver is not configured for the core from where the API is invoked. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	GptEnableDisableNotificationApi,GptReportWakeupSource,GptWakeupFunctionalityApi	
User hints	None	

12.3.5 Scheduled functions

Not applicable for the GPT driver.

12.3.6 Interrupt service routines

Not applicable for the GPT driver.

12.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

12.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Note: *The following error IDs are also reported as safety errors.*

GPT driver**Table 858 Description of development errors reported**

Description	Source	Error code and value	Applicable APIs
Error reported when the driver is already initialized for the current core.	AUTOSAR	GPT_E_ALREADY_INITIALIZED=0x0D	Gpt_Init
Error reported when timer channel is still busy (running).	AUTOSAR	GPT_E_BUSY=0xB	Gpt_StartTimer, Gpt_DelInit
The parameter 'Channel' is not configured for the core from where the API is invoked.	IFX	GPT_E_CORE_CHANNEL_MISMATCH=0x65	Gpt_DisableWakeups, Gpt_EnableWakeups, Gpt_StopTimer, Gpt_StartTimer, Gpt_GetTimeRemaining , Gpt_GetTimeElapsed, Gpt_DisableNotification, Gpt_EnableNotification
Error reported when Gpt_Init is called with null pointer as argument.	AUTOSAR	GPT_E_INIT_FAILED=0x0E	Gpt_Init
Error reported when slave core initializing is called without initializing Master core.	IFX	GPT_E_MASTER_UNINIT=0x66	Gpt_Init
Error reported when GPT driver is not configured for the core from where the API is invoked.	IFX	GPT_E_NOT_CONFIGURED=0x64	Gpt_Isr, Gpt_Init
Error reported when the parameter 'Channel' is invalid.	AUTOSAR	GPT_E_PARAM_CHANNEL=0x14	Gpt_EnableWakeups, Gpt_DisableWakeups, Gpt_DisableNotification, Gpt_EnableNotification, Gpt_StopTimer, Gpt_StartTimer, Gpt_GetTimeRemaining , Gpt_GetTimeElapsed
Error reported when Gpt_SetMode is called with invalid parameter (Mode).	AUTOSAR	GPT_E_PARAM_MODE=0x1F	Gpt_SetMode
Error reported when the parameter passed is a null pointer.	AUTOSAR	GPT_E_PARAM_POINTER=0x16	Gpt_GetVersionInfo, Gpt_GetPredefTimerValue
Error reported when the parameter Predef Timer is invalid.	AUTOSAR	GPT_E_PARAM_PREDEF_TIMER=0x17	Gpt_GetPredefTimerValue

GPT driver

Table 858 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
Error reported when the value is not within the allowed range.	AUTOSAR	GPT_E_PARAM_VALUE=0x15	Gpt_StartTimer
Error reported when master de-Initialization is called without de-Initializing slave core.	IFX	GPT_E_SLAVE_INIT=0x67	Gpt_DelInit
Error reported when the driver is not initialized in context to the core from where the API is invoked.	AUTOSAR	GPT_E_UNINIT=0xA	Gpt_GetPredefTimerValue, Gpt_CheckWakeup, Gpt_EnableWakeup, Gpt_DisableWakeup, Gpt_SetMode, Gpt_DisableNotification, Gpt_EnableNotification, Gpt_StopTimer, Gpt_StartTimer, Gpt_GetTimeRemaining, Gpt_GetTimeElapsed, Gpt_DelInit

12.3.7.2 Production errors

The driver does not report any production errors.

12.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 859 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
Error reported when the driver is in sleep mode.	AUTOSAR	GPT_E_MODE=0x0C	Gpt_StartTimer, Gpt_GetPredefTimerValue

12.3.7.4 Runtime errors

The driver does not report any runtime errors.

12.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

12.3.8.1 Deviations

The section describes the deviations from software specification.

GPT driver
Table 860 Known deviations

Reference	Deviation
Predefined timer	The maximum value for 16-bit predefined timer must be 65535, but due to hardware limitation the maximum count for 16-bit predefined timer is 65534.
ECUC_Gpt_00330 : GptClockReference	According to AUTOSAR, the reference node for GptClockReferecne should be the container McuClockReferencePoint. In order to calculate the exact clock frequency of the selected GTM channel it is necessary that the referecne be made to McuClockReferencePointConfig (Infineon specific node). By accessing parameters inside the container the McuClockReferencePointConfig the clock frequency for GTM is calculated.

12.3.8.2 Limitations

The section describes the limitations from software specification.

Table 861 Known limitations

Reference	Limitation
Gpt_SetMode API	The Gpt_SetMode API when called with SLEEP shall stop all non-wakeup capable timers. However, in multicore environment, the Gpt_SetMode API will stop the predefined timer only when invoked from the logical master core.

12.3.9 Unsupported hardware features

Not applicable for the GPT driver.

Icu_17_TimerIp driver

13 Icu_17_TimerIp driver

13.1 User information

13.1.1 Description

The ICU driver is responsible for providing standard signal measurement services specified by AUTOSAR. The underlying capture engine of an ICU channel can be a TIM channel of the GTM unit, a CC6 comparator of the CCU6 module, an ERU channel or a GPT12 timer.

13.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

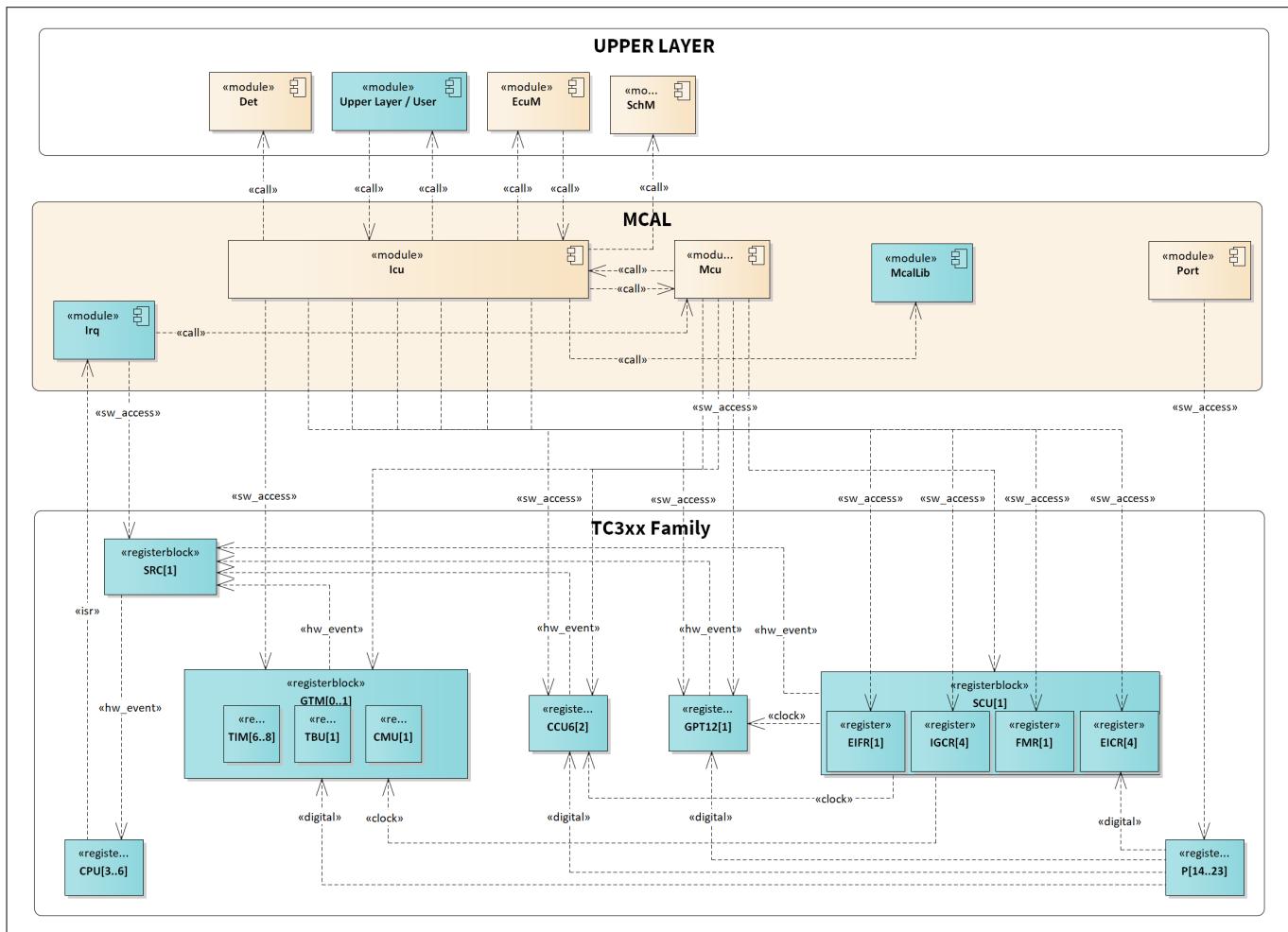


Figure 129 **Mapping of hardware-software interfaces**

13.1.2.1 GTM-TIM: primary hardware peripheral

The TIM slice of the GTM peripheral can be used to realize all the ICU modes.

Hardware functional features

The ICU driver uses the GTM-TIM for edge detection, edge counting, signal measurement and time stamping of the input signal.

Icu_17_TimerIp driver

The key hardware functional features used by the driver are:

- Input selection
- Filter configuration
- Channel clock source configuration
- TIM channel in TPWM, TIEM and TIPM modes

The unsupported features of the GTM-TIM are:

- Timeout Detection Unit (TDU) of TIM is not supported in the ICU driver as it is not relevant to the ICU requirements
- The driver does not support the usage of ARU in TIM
- Generating filter input using the lookup table functionality is not used
- The GTM interrupt mode, GTM_INTERRUPT_PULSE_MODE, cannot be used to achieve the ICU functionalities. In this mode, the interrupt bit in IRQ_NOTIFY register is always cleared if IRQ_EN is enabled. Hence, the MCU driver whose responsibility is to route the ISR to the ICU driver cannot check/determine/validate the interrupt source (from IRQ_NOTIFY flags). Therefore, GTM_INTERRUPT_PULSE_MODE selection is not available.

Users of the hardware

A TIM channel of the GTM is exclusively used by the ICU driver. The TIM channel is not shared with any other driver. The MCU driver provides APIs to program the GTM SFRs. The ICU driver uses these APIs to write the GTM SFRs. Additionally, updates to channel-specific SFRs are performed by the ICU driver. Since these channels are exclusively reserved for the ICU driver, access to the channel-specific SFRs from other drivers or user software is not allowed.

Hardware diagnostic features

Not applicable.

Hardware events

The ICU driver uses the following hardware events from the GTM-TIM IP:

- New measurement value(NEWVAL): for edge detection, edge counting, signal measurement and time stamping
- CNT counter overflow(CNTOFL): to identify counter overflows

13.1.2.2 CCU6: primary hardware peripheral

Hardware functional features

The ICU driver uses the CC6 slices of the T12 timer instance for realizing the signal measurement, time stamping and signal edge detection functionalities.

The key hardware functional features used by the driver are:

- Clock divider and pre-scalar configuration
- Selection of input signal of CC6 slice
- CC6 comparator in the Capture mode

The unsupported features of the CCU6 are:

- CCU6 kernel T13 is not used by the ICU driver
- Compare modes and multi-input capture modes are not used by the ICU driver

Users of the hardware

A CC6x comparator belonging to T12 of the CCU6 kernel is used by an ICU channel. A CCU6 kernel can be reserved to PWM or ICU. However all the comparators of a kernel reserved for the ICU are exclusive for the ICU. The ADC can also use the CCU6 trigger events that do not impact any functionality of the ICU. The MCU driver provides APIs to program the CCU6 SFRs. The ICU driver uses these APIs to write the CCU6 SFRs. Additionally,

Icu_17_TimerIp driver

updates to channel-specific SFRs are performed by the ICU driver. Since these channels are exclusively reserved for the ICU driver, access to the channel-specific SFRs from other drivers or user software is not allowed.

Hardware diagnostic features

Not applicable.

Hardware events

The ICU driver uses the following hardware event from the CCU6 IP:

- Edge detected interrupt generated by the CC6 slices

13.1.2.3 SCU

ERU: primary hardware peripheral

The ERU module can be used for signal edge detection and notification purpose.

Hardware functional features

Channel edge detection feature of the ERU functional block is configured and accessed by the ICU driver.

The unsupported features of the ERU are:

- Hardware event on a pattern detection at the input is not used by the ICU driver as the feature is not relevant for the ICU functionality
- ERU filter external input filter register, filter clock pre-divider and glitch filter depth configuration parameters will not be part of the ICU driver as these parameters are applicable to the complete ERU unit and not per channel.

Users of the hardware

An ERU input channel for input selection and ERU output channel for interrupt trigger are used by the ICU driver. As the two ERU output channels share the same interrupt line, both channels must be allocated to the same module. An ERU input channel can be used by ICU, ADC or DSADC driver.

Hardware diagnostic features

Not applicable.

Hardware events

The ICU driver handles the edge detect interrupts generated by the ETL block.

SCU: dependent hardware peripheral

Hardware functional features

The ICU driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB and fGTM clock signals for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver, and is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the ICU driver.

Hardware events

Hardware events from the SCU are not used by the ICU driver.

13.1.2.4 GPT12: primary hardware peripheral

Hardware functional features

The ICU driver uses the GPT timer of the GPT12 peripheral to realize edge detection, edge counting and incremental interface modes.

Icu_17_TimerIp driver

The key hardware functional features used by the driver are:

- Clock pre-scalar configuration
- Input selection
- Timer in the Counter and Incremental interface modes

The unsupported features of the GPT12 are:

- Timer and Gated timer modes

Users of the hardware

A GPT timer instance of the GPT12 is used by the ICU driver. The GPT12 timer block can be shared with GPT driver. The MCU driver provides APIs to program the GPT12 SFRs. The ICU driver uses these APIs to write the GPT12 SFRs. Additionally, updates to channel-specific SFRs are performed by the ICU driver. Since these channels are exclusively reserved for the ICU driver, access to the channel-specific SFRs from other drivers or user software is not allowed.

Hardware diagnostic features

Not applicable.

Hardware events

The ICU driver uses the following hardware event from the GPT12 IP:

- Timer counter overflow event

13.1.2.5 Port: dependent hardware peripheral

Hardware functional features

The complex input signals are routed to the capture engines of GTM-TIM, GPT12, CCU6, ERU through the port pads. These are configured and enabled through the PORT driver.

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

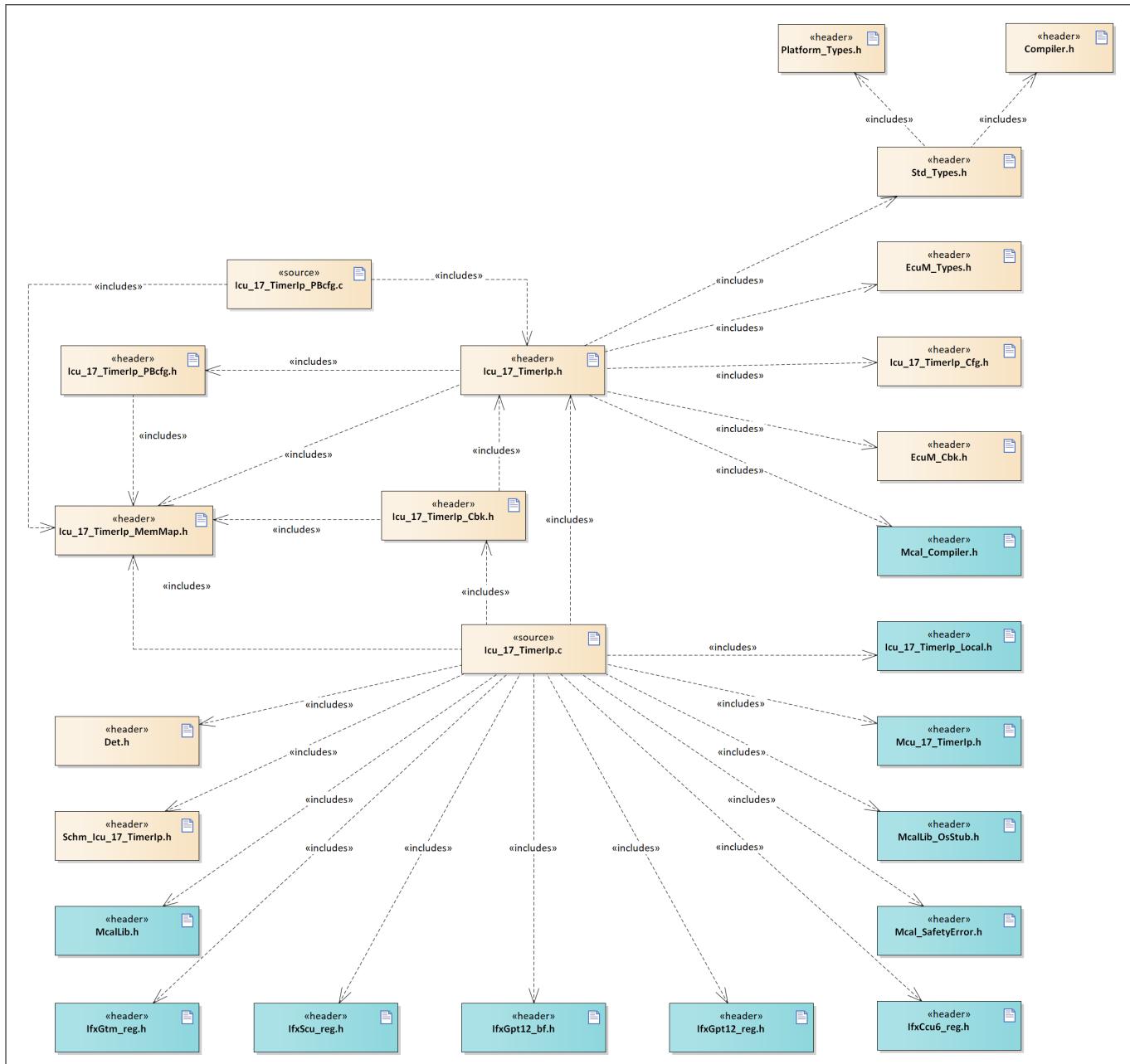
Hardware events

Hardware events from port pads are not used by the ICU driver.

13.1.3 File structure

13.1.3.1 C file structure

The section provides details of the C files of the ICU driver.

Icu_17_TimerIp driver

Figure 130 C file structure
Table 862 C file structure

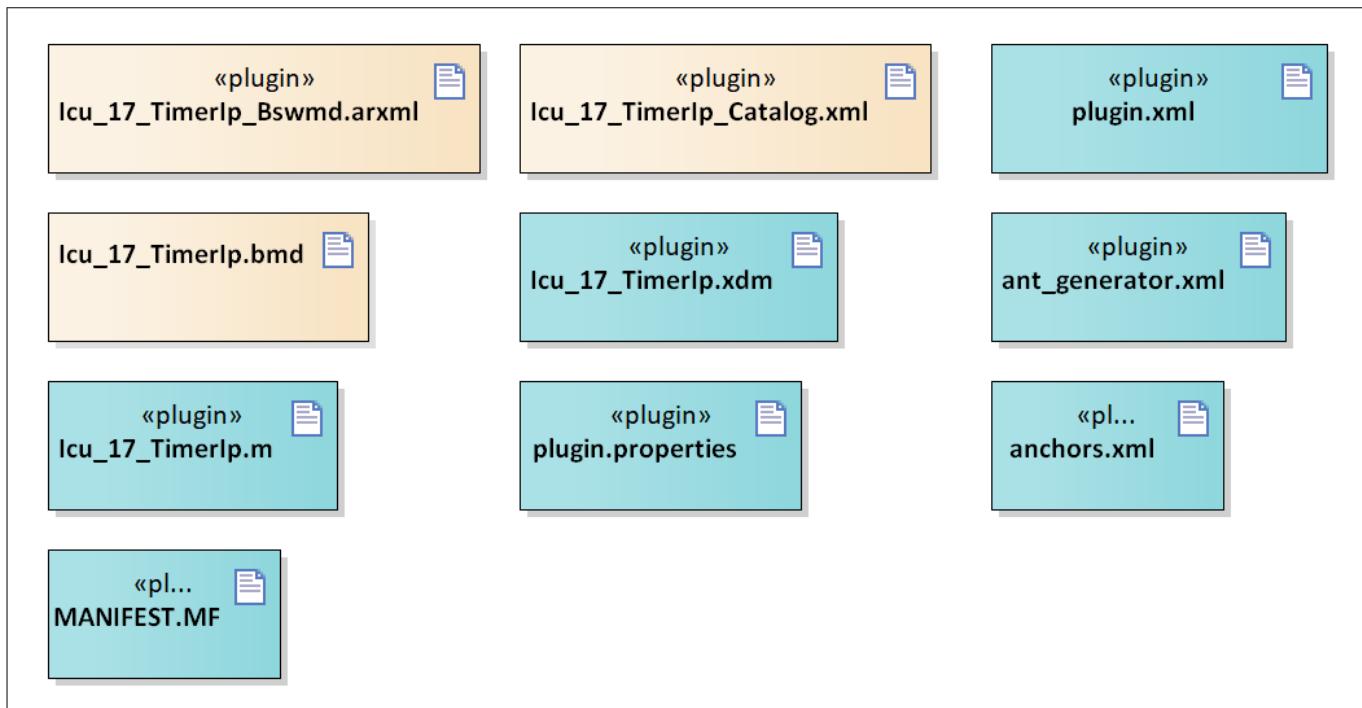
File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Det.h	Provides the exported interfaces of Development Error Tracer
EcuM_Cbk.h	Header file containing declarations of the EcuM callbacks
EcuM_Types.h	Header file containing type declarations of the EcuM
Icu_17_TimerIp.c	File (static) containing implementation of APIs
Icu_17_TimerIp.h	Header file (static) defining prototypes of configuration data structures and APIs
Icu_17_TimerIp_Cbk.h	Header file to declare the callback APIs

Icu_17_TimerIp driver
Table 862 C file structure (continued)

File name	Description
Icu_17_TimerIp_Cfg.h	Header file (generated) containing constants and pre-processor macros
Icu_17_TimerIp_Local.h	Header file defining type definition of global data and inline APIs, which can be used across source files
Icu_17_TimerIp_MemMap.h	File (static) containing the memory section definitions used by the ICU driver
Icu_17_TimerIp_PBcfg.c	File (generated) containing objects to data structures
Icu_17_TimerIp_PBcfg.h	File (generated) containing declaration of the post-build configuration data structures
IfxCcu6_reg.h	SFR header file for CCU6
IfxGpt12_bf.h	SFR header file for GPT12
IfxGpt12_reg.h	SFR header file for GPT12
IfxGtm_reg.h	SFR header file for GTM
IfxScu_reg.h	SFR header file for SCU
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore™. This shall be included by other drivers to call OS APIs.
Mcal_Compiler.h	Provides abstraction for TriCore™-intrinsic instruction
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
Schm_Icu_17_TimerIp.h	File containing the critical sections declarations
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

13.1.3.2 Code generator plugin files

The section provides details of the code generator plugin files of the ICU driver.

Icu_17_TimerIp driver

Figure 131 **Code generator plugin files**
Table 863 **Code generator plugin files**

File name	Description
Icu_17_TimerIp.m	Code template macro file for ICU driver
Icu_17_TimerIp.xdm	Tresos format XML data model schema file
Icu_17_TimerIp_Bswmd.arxml	AUTOSAR format module description file
Icu_17_TimerIp_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the metadata for the ICU driver
anchors.xml	Tresos anchors support file for the ICU driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point
plugin.properties	Tresos plugin support file for the ICU driver
plugin.xml	Tresos plugin support file for the ICU driver

13.1.4 Integration hints

This section lists the key points that an integrator or user of the ICU driver must consider. The ICU measurement modes and hardware configurations as follows:

Table 864 **ICU measurement modes and hardware configurations**

Measurement mode	Supported hardware	Supported features
Edge detection	GTM(TIM), CCU6, GPT12 and ERU	Notifications, wake-up capable

Icu_17_TimerIp driver
Table 864 ICU measurement modes and hardware configurations (continued)

Measurement mode	Supported hardware	Supported features
Multi Edge detection	GTM(TIM) and GPT12	Notifications
Edge count	GTM(TIM) and GPT12	Edge counting up to 32-bit
Signal measurement	GTM(TIM) and CCU6	High time, low time, period and duty cycle
Time stamp	GTM(TIM) and CCU6	Linear and circular buffer, notifications
Incremental interface mode	GPT12(Only T2, T3 and T4)	Detect direction and position from incremental encoder. No interrupts needed.

13.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the ICU driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of the MCAL, the EcuM is used for initialization and de-initialization of the software drivers. User shall configure the wake-up information in EcuM configuration, which will be assigned to every wake-up capable ICU channel. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user specific memory regions. To achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Icu_17_TimerIp_MemMap.h` file.

The `Icu_17_TimerIp_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the

Icu_17_TimerIp driver

elements are re-located to the correct memory region. A sample implementation listing the memory-section macros shown as follows.

```
***** GLOBAL RAM DATA -- NON-CACHED LMU *****/
#if defined ICU_17_TIMERIP_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*User pragmas here for Non-cached LMU****/
#define ICU_17_TIMERIP_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR
#elif defined ICU_17_TIMERIP_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*User pragmas here for Non-cached LMU****/
#define ICU_17_TIMERIP_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR

***** CORE[x] RAM DATA -- DSPr *****/ /*[x]=0..5*/
#elif defined ICU_17_TIMERIP_START_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
/*User pragmas here for CORE[x] DSPr****/
#define ICU_17_TIMERIP_START_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
#define MEMMAP_ERROR
#elif defined ICU_17_TIMERIP_STOP_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
/*User pragmas here for CORE[x] DSPr****/
#define ICU_17_TIMERIP_STOP_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
#define MEMMAP_ERROR

***** CORE[x] RAM DATA INIT -- DSPr *****/ /*[x]=0..5*/
#elif defined ICU_17_TIMERIP_START_SEC_VAR_INIT_ASIL_B_CORE[x]_32
/*User pragmas here for CORE[x] DSPr****/
#define ICU_17_TIMERIP_START_SEC_VAR_INIT_ASIL_B_CORE[x]_32
#define MEMMAP_ERROR
#elif defined ICU_17_TIMERIP_STOP_SEC_VAR_INIT_ASIL_B_CORE[x]_32
/*User pragmas here for CORE[x] DSPr****/
#define ICU_17_TIMERIP_STOP_SEC_VAR_INIT_ASIL_B_CORE[x]_32
#define MEMMAP_ERROR

***** GLOBAL CONST DATA -- PF[x] *****/
#elif defined ICU_17_TIMERIP_START_SEC_CONST_ASIL_B_GLOBAL_32
/*User pragmas here for PF[x]****/
#define ICU_17_TIMERIP_START_SEC_CONST_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR
#elif defined ICU_17_TIMERIP_STOP_SEC_CONST_ASIL_B_GLOBAL_32
/*User pragmas here for PF[x]****/
#define ICU_17_TIMERIP_STOP_SEC_CONST_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR

***** GLOBAL CONFIG DATA -- PF[x] *****/
#elif defined ICU_17_TIMERIP_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/*User pragmas here for PF[x]****/
#define ICU_17_TIMERIP_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined ICU_17_TIMERIP_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/*User pragmas here for PF[x]****/
#define ICU_17_TIMERIP_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR
```

Icu_17_TimerIp driver

```

***** CORE[x] CONFIG DATA -- PF[x] ***** / *[x]=0..5*/
#elif defined ICU_17_TIMERIP_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
    /******User pragmas here for PF[x]*****/
#define ICU_17_TIMERIP_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined ICU_17_TIMERIP_STOP_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
    /******User pragmas here for PF[x]*****/
#define ICU_17_TIMERIP_STOP_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
#define MEMMAP_ERROR

***** CODE -- PF[x] *****/
#elif defined ICU_17_TIMERIP_START_SEC_CODE_ASIL_B_GLOBAL
    /******User pragmas here for PF[x]*****/
#define ICU_17_TIMERIP_START_SEC_CODE_ASIL_B_GLOBAL
#define MEMMAP_ERROR
#elif defined ICU_17_TIMERIP_STOP_SEC_CODE_ASIL_B_GLOBAL
    /******User pragmas here for PF[x]*****/
#define ICU_17_TIMERIP_STOP_SEC_CODE_ASIL_B_GLOBAL
#define MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Icu_17_TimerIp_MemMap.h, wrong pragma command"
#endif

```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The Icu driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the ICU driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

DEM module is not required for integrating the ICU driver.

- **SchM**

The SchM module is a part of the RTE that manages the BSW Scheduler. The Icu driver uses the exclusive areas defined in the `SchM_Icu_17_TimerIp.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the ICU driver are:

- ResetEdgeCount
- SetActivationCondition
- GtmEnableEdgeCount
- GtmGetDutyCycle

The `SchM_Icu_17_TimerIp.h` and `SchM_Icu_17_TimerIp.c` files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions

Icu_17_TimerIp driver

defined by the Icu driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows:

```
***** Sample implementation of SchM_Icu_17_TimerIp.c ****/
#include "Os.h"

/* Disable the interrupts for entering critical section */
void SchM_Enter_Icu_17_TimerIp_ResetEdgeCount(void)
{
    SuspendAllInterrupts();
}

/* Re-enable the interrupt for exiting the critical section */
void SchM_Exit_Icu_17_TimerIp_ResetEdgeCount(void)
{
    ResumeAllInterrupts();
}

/* Disable the interrupts for entering critical section */
void SchM_Enter_Icu_17_TimerIp_SetActivationCondition(void)
{
    SuspendAllInterrupts();
}

/* Re-enable the interrupt for exiting the critical section */
void SchM_Exit_Icu_17_TimerIp_SetActivationCondition(void)
{
    ResumeAllInterrupts();
}

/* Disable the interrupts for entering critical section */
void SchM_Enter_Icu_17_TimerIp_GtmEnableEdgeCount(void)
{
    SuspendAllInterrupts();
}

/* Re-enable the interrupt for exiting the critical section */
void SchM_Exit_Icu_17_TimerIp_GtmEnableEdgeCount(void)
{
    ResumeAllInterrupts();
}

/* Disable the interrupts for entering critical section */
void SchM_Enter_Icu_17_TimerIp_GtmGetDutyCycle(void)
{
    SuspendAllInterrupts();
}

/* Re-enable the interrupt for exiting the critical section */
void SchM_Exit_Icu_17_TimerIp_GtmGetDutyCycle(void)
{
    ResumeAllInterrupts();
}
```

- **Safety error**

The ICU driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API.

Icu_17_TimerIp driver

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The ICU driver does not implement any notifications. However, ICU driver reports the detection of edges and desired timestamps are captured through the notification functions. These notification functions can be configured by the user in Tresos for each channel (in edge detect and time stamping mode).

ICU does not expect any callbacks from application. But the ICU needs the call-back ISR from the MCU.

- **Operating system (OS)**

OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. The enabling and disabling of interrupts must also be managed by the OS or application.

The OS files provided by MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

13.1.4.2 Multicore and Resource Manager

The Icu driver supports execution of its APIs simultaneously from all CPU cores. The user has to allocate each channel of the ICU to CPU cores at pre-compile time using the Resource Manager module. The following are the key points to be considered with respect to multicore in the ICU driver:

- Each ICU channel can be allocated to any core using the Resource Manager.
- For ICU channel dependent on ERU, channels using `OGU[x]` and `OGU[x+4]` where ($x=0-3$), must be allocated to same core as these two channels share same interrupt line. For example, ICU channels using OGU0 and OGU4 should be allocated to same core.
- The locating of constants, variables and configuration data to correct memory space should be done by the user. Memory sections are marked GLOBAL(common to all cores) and CORE[x](specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of the ICU driver is placed under single MemMap section. It can be relocated to any PFlash/DFlash region.

Data section:

The RAM variable memory sections marked as specific to a core should be re-located to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region.

Configuration data and constants:

The configuration data sections marked as specific to a core should be re-located to the PFlash/DFlash of the same core. The sections marked as global should be relocated to the PFlash/DFlash of the master core.

Note: Relocating of code, data or constants to a distant memory region would impact execution timings.

Note: If the driver operates from single (master) core, all the sections may be relocated to the PFlash/DSPR/DLMU of the same CPU core.

13.1.4.3 MCU support

The Icu driver is dependent on MCU driver for clock configuration and timer-IP related services. The initialization of the ICU driver must be initialized only after successful completion of the MCU initialization. The following must be considered while configuring the MCU driver in the Eb tresos:

Icu_17_TimerIp driver

- GTM-TIM Icu channel: The GTM-TIM channels used by Icu driver must be reserved in the MCU configuration for exclusive use by Icu. The reserved TIM channel can be used for one ICU channel.
- CCU6 Icu channel: The CCU6 kernel used by Icu driver must be reserved in the MCU configuration for exclusive use by Icu. Hence, each of the three comparators(CC60, CC61 and CC62) of the reserved CCU6 kernel can be used for three different ICU channels.
- GPT12 Icu channel: The GPT12 timer used by Icu driver must be reserved in the MCU configuration for exclusive use by Icu. The reserved timer can be used for one ICU channel.
- ERU Icu channel: The ERS channel and corresponding OGU channel used by Icu driver must be reserved in the MCU configuration for exclusive use by Icu. An ERS channel can be paired with any OGU channel.

13.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the ICU driver through the Port configuration and initialize the port pins prior to invoking the ICU driver initialization.

13.1.4.5 DMA support

The ICU driver does not use any services provided by the DMA driver.

13.1.4.6 Interrupt connections

The interrupt configuration registers of different hardware used by ICU channels are as follows:

Table 865 SRC registers

Hardware used	SRC register
GTM-TIM	SRC_GTTMTIMwx (w= TIM module; x= TIM channel)
CCU6	SRC_CCU6xSRy (x= CCU6 kernel; y=0-3)
ERU	SRC_SCUERUx (x=0-3)
GPT12	SRC_GPT120Tx (x=2-6, GPT12 timer)

All the ISR to GTM-TIM must be routed to the `Mcu_17_Gtm_TimChannelIsr` API, which further invokes `Icu_17_TimerIp_Timer_Isr`. The example ISR handling is shown as follows:

```

ISR(GTTMTIM0SR0_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    /* Parameter is TIM module number and TIM channel number */
    Mcu_17_Gtm_TimChannelIsr(0, 2); /* For TIM 0 CH2*/
}

```

Icu_17_TimerIp driver

All the ISR to CCU6 comparator must be routed to the `Mcu_17_Ccu6_ChannelIsr` API, which further invokes `Icu_17_TimerIp_Timer_Isr`. The example ISR handling is shown as follows:

```
ISR(CCU60SR0_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    /* Parameter are CCU6 kernel and comparator */
    Mcu_17_Ccu6_ChannelIsr(CCU6_KERNEL_0,CCU6_CHANNEL_0);
}
```

All the ISR to GPT12 timer must be routed to the `Mcu_17_Gpt12_ChannelIsr` API, which further invokes `Icu_17_TimerIp_Timer_Isr`. The example ISR handling is shown as follows:

```
ISR(GPT12_T2_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    /* Parameter is GPT12 timer number 0 for T2, 1 for T3 and so on.*/
    Mcu_17_Gpt12_ChannelIsr(0); /* For T2 timer */
}
```

All the ISR to ERU channel must be routed to the `Mcu_17_Eru_GatingIsr` API, which further invokes `Icu_17_TimerIp_Timer_Isr`. The example ISR handling is shown as follows:

```
ISR(SCUERUSR0_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    /* Call Mcu Interrupt function, parameter is SRC index. */
    Mcu_17_Eru_GatingIsr(0); /* For ERS 0*/
}
```

Icu_17_TimerIp driver

13.1.4.7 Example usage

Initialization

User must include `Icu_17_TimerIp.h` file to access the ICU configuration structure needed for initialization.

```

/* Include Icu.h to access configuration structures */
#include "Icu_17_TimerIp.h"
/* Module Initialization */
void Icu_Sample_Init(void)
{
    /* MCU initializations */
    Mcu_Init(&Mcu_Config);
    (void)Mcu_InitClock( 0 );
    while(Mcu_GetPllStatus() != MCU_PLL_LOCKED)
    {
    };
    (void)Mcu_DistributePllClock();

    /* Initialize ICU */
    Icu_17_TimerIp_Init(&Icu_17_TimerIp_Config);

    /* Initialize Check for ICU */
    Error = Icu_17_TimerIp_InitCheck(&Icu_17_TimerIp_Config);

    if (Error == E_OK)
    {
        /*ICU InitCheck should pass, then Call other APIs related to ICU */
    }
}

```

Edge count mode

Edge counting activity on an edge count configured ICU channel starts by the call to `Icu_17_TimerIp_EnableEdgeCount`. The `Icu_17_TimerIp_GetEdgeNumbers` API returns the number of edges counted after the call to `Icu_17_TimerIp_Init` or `Icu_17_TimerIp_ResetEdgeCount`. Edge counting activity is stopped by the call to `Icu_17_TimerIp_DisableEdgeCount`. The `Icu_17_TimerIp_ResetEdgeCount` API resets the number of counted edges even if the edge counting activity is stopped.

Icu_17_TimerIp driver

```

/* ICU module Initialization is necessary to start the edge counting feature */
*Icu_17_TimerIp_Init(&Icu_17_TimerIp_Config);
Icu_17_TimerIp_EnableEdgeCount(<logical channel symbolic name>);

/* Provide edges on the port pin of the logical channel */

/* Read the number of edges detected. Returns the number of provided edges */
CountedEdges = Icu_17_TimerIp_GetEdgeNumbers(<logical channel symbolic name>);

/* Reset the counted edges */
Icu_17_TimerIp_ResetEdgeCount(<logical channel symbolic name>);

/* Read edge count after reset edge count. "0" will be returned */
CountedEdges = Icu_17_TimerIp_GetEdgeNumbers(<logical channel symbolic name>);

/* Provide edges on the port pin of the logical channel */

/* Read the number of edges detected. Returns the number of provided edges */
CountedEdges = Icu_17_TimerIp_GetEdgeNumbers(<logical channel symbolic name>);

/* Disable edge counting */
Icu_17_TimerIp_DisableEdgeCount(<logical channel symbolic name>);

/* Reset the counted edges */
Icu_17_TimerIp_ResetEdgeCount(<logical channel symbolic name>);

/* Read the number of edges detected. Returns "0" as edge counting is
reset.*/
CountedEdges = Icu_17_TimerIp_GetEdgeNumbers(<logical channel symbolic name>);

/* Provide edges on the port pin of the logical channel */

/* Read the number of edges detected. Returns "0" as edge counting is
disabled.*/
CountedEdges = Icu_17_TimerIp_GetEdgeNumbers(<logical channel symbolic name>);

```

Time stamp mode

Capturing of time stamps on the configured active edge is started by the call to

`Icu_17_TimerIp_StartTimestamp.` `Icu_17_TimerIp_EnableNotification` should be invoked to receive notifications after receiving a specific number(configured at the start of activity) of timestamps (applicable only if the notification function is configured in the module configuration).

`Icu_17_TimerIp_DisableNotification` stops issuing the notifications.

Icu_17_TimerIp driver

Icu_17_TimerIp_StopTimestamp stops capturing time stamps. Icu_17_TimerIp_GetTimestampIndex returns the buffer position which is to be filled next.

```

/* ICU module Initialization is necessary to start the time stamping feature */
Icu_17_TimerIp_Init(&Icu_17_TimerIp_Config);

/* Buffer to fill timestamps. BUFFER_SIZE indicates the size of the buffer */
Icu_17_TimerIp_ValueType Buffer[BUFFER_SIZE];

/* Start the time stamping activity. NOTIFY_INTERVAL is the number of
timestamps to be received to issue notification*/
Icu_17_TimerIp_StartTimestamp(<logical channel symbolic name>, Buffer,
BUFFER_SIZE, NOTIFY_INTERVAL);

/* Enable notifications to receive notifications */
Icu_17_TimerIp_EnableNotification(<logical channel symbolic name>);

/* Provide edges on the port pin of the logical channel */

/* Read the buffer index next to be filled */
NextIndex = Icu_17_TimerIp_GetTimestampIndex(<logical channel symbolic name>);

/* Notification function would have been invoked if sufficient number of
edges are provided */

/* Disable Notifications */
Icu_17_TimerIp_DisableNotification(<logical channel symbolic name>);

/* Provide edges on the port pin of the logical channel */

/* Read the buffer index next to be filled. The index read here will be
different from the previous read as the time stamping activity is still active */
NextIndex = Icu_17_TimerIp_GetTimestampIndex(<logical channel symbolic name>);

/* Notification function would not have been invoked even if sufficient
number of edges are provided */

/* Disable time stamping */
Icu_17_TimerIp_StopTimestamp(<logical channel symbolic name>);

/* Provide edges on the port pin of the logical channel */

/* Notification function would not have been invoked even if sufficient
number of edges are provided */

/* Read the buffer index next to be filled. The index read here will be same
from the previous read as the time stamping activity is disabled */

NextIndex = Icu_17_TimerIp_GetTimestampIndex(<logical channel symbolic name>);

```

Icu_17_TimerIp driver**Signal measurement mode**

Measurement of high time, low time, period or duty cycle (as per the configuration) starts after the call to `Icu_17_TimerIp_StartSignalMeasurement`. The availability of a new measured value is identified by the call to `Icu_17_TimerIp_GetInputState`. `Icu_17_TimerIp_GetTimeElapsed` reads the measured high time, low time or period. `Icu_17_TimerIp_GetDutyCycleValues` reads the measured coherent period and active time. The signal measurement activity is stopped by the call to `Icu_17_TimerIp_StopSignalMeasurement` and restarted by the call to `Icu_17_TimerIp_StartSignalMeasurement`.

Icu_17_TimerIp driver

```

/* ICU module Initialization is necessary to start the signal measurement
feature */
Icu_17_TimerIp_Init(&Icu_17_TimerIp_Config);

/* Start signal measurement activity */
Icu_17_TimerIp_StartSignalMeasurement(<non duty cycle logical channel symbolic
name>);
Icu_17_TimerIp_StartSignalMeasurement(<duty cycle logical channel symbolic
name>);

/* Provide edges on the port pin of the logical channel */

/* Check channel status before reading the corresponding values.*/
if(ICU_ACTIVE == Icu_17_TimerIp_GetInputState(<non duty cycle logical channel
symbolic name>))
{
    /* Channel status is active read the measured time */
    SignalMeasureValue = Icu_17_TimerIp_GetTimeElapsed(<non duty cycle logical
channel symbolic name>);

    /* Read the measured time again, this will return 0 */
    SignalMeasureValue = Icu_17_TimerIp_GetTimeElapsed(<non duty cycle logical
channel symbolic name>);
}

if(ICU_ACTIVE == Icu_17_TimerIp_GetInputState(<duty cycle logical channel
symbolic name>))
{
    Icu_17_TimerIp_DutyCycleType DutyCycle;

    /* Channel status is active, read the duty cycle values */
    Icu_17_TimerIp_GetDutyCycleValues(<duty cycle logical channel symbolic name>,
&DutyCycle);

    /* read the duty cycle values again. this will return 0 */
    Icu_17_TimerIp_GetDutyCycleValues(<duty cycle logical channel symbolic name>,
&DutyCycle);
}

/* Stop signal measurement activity */
Icu_17_TimerIp_StopSignalMeasurement(<non duty cycle logical channel symbolic
name>);
Icu_17_TimerIp_StopSignalMeasurement(<duty cycle logical channel symbolic
name>);

/* Provide edges on the port pin of the logical channel */

/* Check channel status, will return IDLE as signal measurement activity is
stopped */
ChannelState = Icu_17_TimerIp_GetInputState(<non duty cycle logical channel
symbolic name>);
ChannelState = Icu_17_TimerIp_GetInputState(<duty cycle logical channel
symbolic name>);

```

Icu_17_TimerIp driver

```
/* Start signal measurement activity */
Icu_17_TimerIp_StartSignalMeasurement(<non duty cycle logical channel symbolic
name>);
Icu_17_TimerIp_StartSignalMeasurement(<duty cycle logical channel symbolic
name>);

/* Provide edges on the port pin of the logical channel */

/* Check channel status, will return ACTIVE as signal measurement activity is
started */
ChannelState = Icu_17_TimerIp_GetInputState(<non duty cycle logical channel
symbolic name>);
ChannelState = Icu_17_TimerIp_GetInputState(<duty cycle logical channel
symbolic name>);
```

Edge detection mode

Detection of edges (rising, falling or any as per configuration) on an edge detect configured ICU channel starts after the call to `Icu_17_TimerIp_Init`. The ICU channel status is identified by the call to `Icu_17_TimerIp_GetInputState`. `Icu_17_TimerIp_EnableNotification` should be invoked to receive notifications on the configured edges (applicable only if the notification function is configured in the module configuration). `Icu_17_TimerIp_DisableNotification` stops issuing the notifications. The edge detection activity is stopped by the call to `Icu_17_TimerIp_DisableEdgeDetection` and re-enabled by the call to

Icu_17_TimerIp driver

Icu_17_TimerIp_EnableEdgeDetection. Icu_17_TimerIp_EnableMultiEdgeDetection detects multiple edges and issue notifications after multiple edges are detected.

```

/* ICU module Initialization is necessary to start the edge detection activity */
*Icu_17_TimerIp_Init(&Icu_17_TimerIp_Config);

/* Provide edges on the port pin of the logical channel */

/* Check channel status, will return ACTIVE and no notifications are detected */
*/
ChannelState = Icu_17_TimerIp_GetInputState(<logical channel symbolic name>);

/* Read channel state again, will return IDLE */
ChannelState = Icu_17_TimerIp_GetInputState(<logical channel symbolic name>);

/* Enable Notifications */
Icu_17_TimerIp_EnableNotification(<logical channel symbolic name>);

/* Provide edges on the port pin of the logical channel */

/* Channel respective notification function would have been issued.*/

/* Disable Notifications */
Icu_17_TimerIp_DisableNotification(<logical channel symbolic name>);

/* Provide edges on the port pin of the logical channel */

/* Channel respective notification function would not have been issued */

/* Check channel status, will return ACTIVE */
ChannelState = Icu_17_TimerIp_GetInputState(<logical channel symbolic name>);

/* Disable Edge detection */
Icu_17_TimerIp_DisableEdgeDetection(<logical channel symbolic name>);

/* Provide edges on the port pin of the logical channel */

/* Channel respective notification function would not have been issued */

/* Check channel status, will return IDLE and no notifications are detected */
ChannelState = Icu_17_TimerIp_GetInputState(<logical channel symbolic name>);

/* Enable multiple edge detection. EDGE_COUNT number of edge will be detected */
*/
Icu_17_TimerIp_EnableMultiEdgeDetection(<logical channel symbolic name>, EDGE_COUNT);

/* Enable notifications */
Icu_17_TimerIp_EnableNotification(<logical channel symbolic name>);

/* Provide multiple edges on the port pin of the logical channel */

```

Icu_17_TimerIp driver

```
/* Notification would have been issued after detecting EDGE_COUNT edges.*/
```

General API

`Icu_17_TimerIp_SetActivationCondition` should be invoked after the call to `Icu_17_TimerIp_Init` and only for channels which are in edge detection, time stamping and edge counting modes. `Icu_17_TimerIp_EnableWakeup` and `Icu_17_TimerIp_DisableWakeup` should be invoked after the call to `Icu_17_TimerIp_Init` and only on a wakeup capable channel. `Icu_17_TimerIp_SetMode` should be invoked after the call to `Icu_17_TimerIp_Init` to change the state of the ICU driver to **SLEEP** or **NORMAL**. `Icu_17_TimerIp_DeInit`, should be invoked after the call to `Icu_17_TimerIp_Init` to reset the initialization state of ICU module. After the call to `Icu_17_TimerIp_DeInit`, `Icu_17_TimerIp_Init` should be invoked again to start any functionality of the ICU driver.

```
/* ICU module Initialization is necessary to start the ICU channel activities */
Icu_17_TimerIp_Init(&Icu_17_TimerIp_Config);

/* Default edge for detection, edge counting and time stamping is taken from
configuration */

/* Change the default active edge to RISING_EDGE */
Icu_17_TimerIp_SetActivationCondition(<logical channel symbolic
name>, ICU_RISING_EDGE);

/* Edge detection, edge counting and time stamping will be done on rising edge
*/

/* Enable wakeup for a wakeup capable channel */
Icu_17_TimerIp_EnableWakeup(<wakeup capable logical channel symbolic name>);

/* Change mode to sleep */
Icu_17_TimerIp_SetMode(ICU_MODE_SLEEP);

/* provide signal on the wakeup capable channel */

/* EcuMCheckWakeups will be invoked from ISR to indicate a wakeup signal */

/* Change mode to normal */
Icu_17_TimerIp_SetMode(ICU_MODE_NORMAL);

/* Disable wakeup for a wakeup capable channel */
Icu_17_TimerIp_DisableWakeup(<wakeup capable logical channel symbolic name>);

/* Change mode to sleep */
Icu_17_TimerIp_SetMode(ICU_MODE_SLEEP);

/* provide signal on the wakeup capable channel */

/* EcuMCheckWakeups will not be invoked */
```

Icu_17_TimerIp driver

Incremental interface mode

The encoder count is set to 0 after initialization. Detection of encoder edges on an incremental interface configured ICU channel starts after the call to `Icu_17_TimerIp_StartIncInterface`. The encoder count is identified by the call to `Icu_17_TimerIp_ReadEncCount` and direction by the call to `Icu_17_TimerIp_ReadEncCountDir`. `Icu_17_TimerIp_CalibratePos` should be invoked to calibrate the initial encoder position. The incremental interface activity is stopped by the call to `Icu_17_TimerIp_StopIncInterface`.

```

/* ICU module Initialization is necessary to start the incremental interface
activity */
Icu_17_TimerIp_Init(&Icu_17_TimerIp_Config);

/* Check encoder count and direction, will return 0 and UP direction(HW
default) */
EncCount = Icu_17_TimerIp_ReadEncCount(<logical channel symbolic name>);
EncDir = Icu_17_TimerIp_ReadEncCountDir(<logical channel symbolic name>);

/* Enable incremental interface channel */
Icu_17_TimerIp_StartIncInterface(<logical channel symbolic name>);

/* Provide edges on the port pin(both counter and direction signal) of the
logical channel */

/* Check encoder count and direction, will return non zero and current counter
direction(HW default) as per the given input signals */
EncCount = Icu_17_TimerIp_ReadEncCount(<logical channel symbolic name>);
EncDir = Icu_17_TimerIp_ReadEncCountDir(<logical channel symbolic name>);

/* Update the encoder position */
Icu_17_TimerIp_CalibratePos(<logical channel symbolic name>, <new counter
position to be set>);

/* Provide edges on the port pin(both counter and direction signal) of the
logical channel */

/* Check encoder count and direction, will return non zero and current counter
direction(HW default) as per the given input signals starting from the encoder
position set */
EncCount = Icu_17_TimerIp_ReadEncCount(<logical channel symbolic name>);
EncDir = Icu_17_TimerIp_ReadEncCountDir(<logical channel symbolic name>);

/* Disable incremental interface channel */
Icu_17_TimerIp_StopIncInterface(<logical channel symbolic name>);

/* Provide edges on the port pin(both counter and direction signal) of the
logical channel */

/* Check encoder count and direction, will return same as the previous call as
the incremental mode is disabled. */
EncCount = Icu_17_TimerIp_ReadEncCount(<logical channel symbolic name>);
EncDir = Icu_17_TimerIp_ReadEncCountDir(<logical channel symbolic name>);

```

Icu_17_TimerIp driver**13.1.5 Key architectural considerations****13.1.5.1 Overflow handling for signal measurement**

For ICU channel in signal measurement mode using the GTM-TIM hardware, the measured value overflow will be detected and measured value will be returned as 0 until next valid measurement.

13.1.5.2 Accessing shared SFR

ICU channel using ERU will access `MODULE_SCU.EICR`, `MODULE_SCU.IGCR` and `MODULE_SCU.FMR` to configure the corresponding ERS and OGU channels. The ICU driver ensures the access is done atomically and with proper masks to not update the unintended part of register. Hence any application module accessing these register must also perform atomic access and with proper mask to ensure no interference with the ICU functionalities.

Icu_17_TimerIp driver

13.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Edge counter overflow**

Edge counter value returned by the API restarts the count from 0 once it reaches 0xFFFFFFFF. User should consider this behavior while using the Icu_17_TimerIp_GetEdgeNumbers API.

[cover parentID ICU={3A84EBDA-FA2B-49dd-9CBB-67ADDAD9CFE6}]

- **Execution sequence for initialization check**

If configured, the Icu_17_TimerIp_InitCheck() API shall be called after the ICU driver initialization and before starting any functionality of the ICU driver.

[cover parentID ICU={578BB26C-7DA4-4f00-8D8D-B8F6D5418CD2}]

- **Generated configuration structure - AoU**

User shall ensure the generated configuration structures are correct against the intended GUI configurations.

[cover parentID ICU={64E84977-40E5-4ebb-9F06-FE12BE63B8E3}]

- **ICU common ERU ISR**

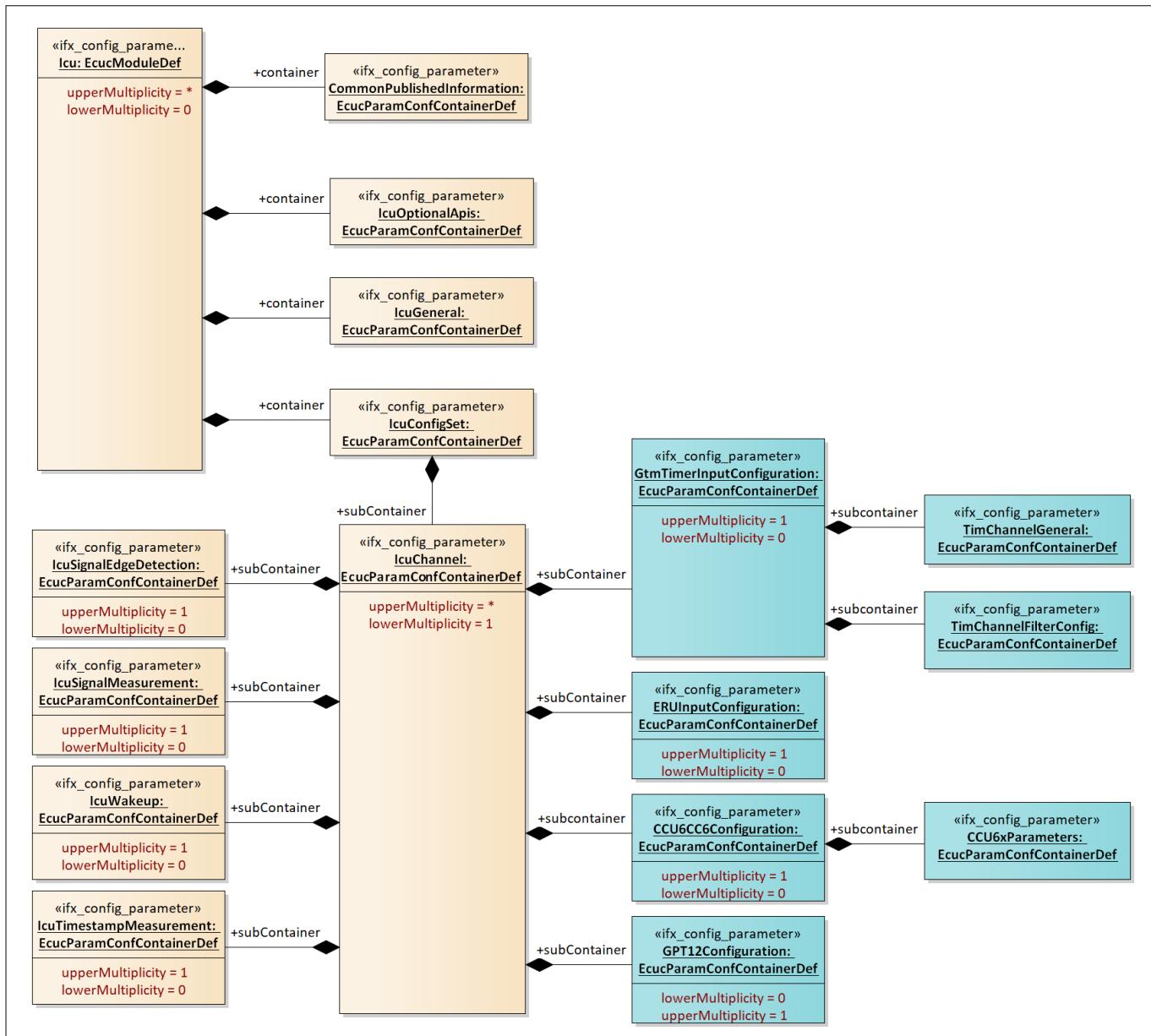
ICU channels using OGU[x] and OGU[x+4] (x=0-3), in case of ERU hardware, shall be allocated to the same core as these two channels share the same interrupt line. For example, ICU channels using OGU0 and OGU4 should be allocated to the same core.

[cover parentID ICU={B6252F81-63DA-442d-B8EE-1CA887EEA003}]

- **ICU signal measurement for CCU6 hardware**

For an ICU channel configured in the signal measurement mode using CCU6 hardware, the user must ensure the measured value must be in the 16-bit range. The overflow cannot be identified due to hardware limitation.

[cover parentID ICU={5FE037D2-D596-4418-9ADA-0E39896716F8}]

Icu_17_TimerIp driver**13.3 Reference information****13.3.1 Configuration interfaces****Figure 132 Container hierarchy along with their configuration parameters****13.3.1.1 Container: CCU6CC6Configuration**

The container contains the configuration for CCU6x comparator if the hardware unit selected is CCU6.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

Icu_17_TimerIp driver

13.3.1.1.1 CCChannelInputSelection

Table 866 Specification for CCChannelInputSelection

Name	CCChannelInputSelection		
Description	Input selection for Cc6xchannel. NOTE: The parameter reads from property file and depends upon channel selection. Default value is set to first encountered value from Resource property file.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CCINx_[inputline]: x can be A,B,C and so on depending on HW. 'inputline' can be PORT PIN or internal signal from other peripheral.		
Default value	CCINx_[inputline]		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CCU6KernelUsed, Cc6xChannel		

13.3.1.1.2 CCU6KernelUsed

Table 867 Specification for CCU6KernelUsed

Name	CCU6KernelUsed		
Description	The parameter is a list of CCU6 kernels available to the ICU Driver. The CCU6 kernel chosen must necessarily have been reserved for usage by ICU. NOTE: Default value is set to blank as user has to select the appropriate reference value from MCU.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuCcu6ModuleAllocationConf		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Icu_17_TimerIp driver**13.3.1.1.3 Cc6xChannel****Table 868 Specification for Cc6xChannel**

Name	Cc6xChannel		
Description	Selection of a CC6x channel. NOTE: Default value is chosen as Cc60 as it is the CCU6 lowest comparator.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	Cc60: Selection of CC60 Capture Cc61: Selection of CC61 Capture Cc62: Selection of CC62 Capture		
Default value	Cc60		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.2 Container: CCU6xParameters

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

13.3.1.2.1 CCU6InterruptNode**Table 869 Specification for CCU6InterruptNode**

Name	CCU6InterruptNode		
Description	Interrupt node to be used for the kernel. NOTE: Default value is chosen as 0 as Hardware Default value is 0.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NODE_SR0: Service request output SR0 is selected NODE_SR1: Service request output SR1 is selected NODE_SR2: Service request output SR2 is selected NODE_SR3: Service request output SR3 is selected		
Default value	NODE_SR0		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Icu_17_TimerIp driver**Table 869 Specification for CCU6InterruptNode (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.2.2 T12ClkSelection**Table 870 Specification for T12ClkSelection**

Name	T12ClkSelection		
Description	T12 clock divider configuration for the kernel used. NOTE1: Effective clock divider is $2^{\text{value configured}}$. NOTE2: Default value is chosen as 0 as Hardware Default value is 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CCU6KernelUsed		

13.3.1.2.3 T12PrescalerEnabled**Table 871 Specification for T12PrescalerEnabled**

Name	T12PrescalerEnabled		
Description	The parameter to determine if the additional (1/256) pre-scalar should be added to the clock path. If T12PrescalerEnabled is false no additional pre-scalar is added else pre-scalar is added. Note: Default value is chosen as FALSE as Hardware Default value is FALSE.		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

Icu_17_TimerIp driver**Table 871 Specification for T12PrescalerEnabled (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CCU6KernelUsed		

13.3.1.3 Container: CommonPublishedInformation

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

13.3.1.3.1 ArMajorVersion**Table 872 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	AUTOSAR major version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.3.2 ArMinorVersion**Table 873 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	AUTOSAR minor version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Icu_17_TimerIp driver**Table 873 Specification for ArMinorVersion (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.3.3 ArPatchVersion**Table 874 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	AUTOSAR patch version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.3.4 ModuleId**Table 875 Specification for ModuleId**

Name	ModuleId		
Description	Parameter to provide the module identifier.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	122		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Icu_17_TimerIp driver**13.3.1.3.5 Release****Table 876 Specification for Release**

Name	Release		
Description	Aurix derivative used for the implementation.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per the configuration.		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.3.6 SwMajorVersion**Table 877 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Module major version.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.3.7 SwMinorVersion**Table 878 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Module minor version.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		

Icu_17_Timerlp driver**Table 878 Specification for SwMinorVersion (continued)**

Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.3.8 SwPatchVersion**Table 879 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Module patch version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.3.9 VendorApiInfix**Table 880 Specification for VendorApiInfix**

Name	VendorApiInfix		
Description	The parameter is used to specify the vendor specific name. Note: "Timerlp" is chosen as VendorApiInfix as all functionalities of ICU are achieved using the timer modules GTM-TIM, CCU6 and GPT12.		
Multiplicity	1..1	Type	EcuStringParamDef
Range	String		
Default value	Timerlp		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Icu_17_TimerIp driver**Table 880 Specification for VendorApiInfix (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.3.10 Vendorld**Table 881 Specification for Vendorld**

Name	VendorId		
Description	Infineon vendor ID in HIS software specification.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.4 Container: ERUInputConfiguration

The container contains the configuration for ERS and OGU channel if the hardware unit selected is ERU.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

13.3.1.4.1 EruErsReference**Table 882 Specification for EruErsReference**

Name	EruErsReference		
Description	<p>The parameter is a reference to the ERS container in the MCU. The ERS channel chosen must necessarily have been reserved for usage by ICU.</p> <p>NOTE: Default value is set to blank as user has to select the appropriate reference value from MCU.</p>		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuEruChannelInputLineConf		
Default value	NULL		

Icu_17_TimerIp driver**Table 882 Specification for EruErsReference (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.4.2 EruInputPin**Table 883 Specification for EruInputPin**

Name	EruInputPin		
Description	The input pin selection for the ERU unit. NOTE: The parameter reads from property file and depends upon channel selection. Default value is set to first encountered value from Resource property file.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ERU_INPUTNx_[inputline]: n is the ERU input channel selected. x can be A, B, C, D, E or F depending on hardware. 'inputline' can be PORT PIN or internal signal from other peripheral.		
Default value	ERU_INPUTx_PORTy_PINz		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	EruErsReference		

13.3.1.4.3 EruOguReference**Table 884 Specification for EruOguReference**

Name	EruOguReference
Description	The parameter is a reference to the ERU container in the MCU. The parameter lists down all of ERU-OGU slices (ERU output processors) available on the device. If the referred EruOguReference is not marked as ERU_CHANNEL_OUT_USED_BY_ICU_DRIVER for the parameter McuEruOutputLineConfChannel in the MCU driver, an error message is raised. NOTE: Default value is set to blank as the user has to select the appropriate reference value from MCU.

Icu_17_TimerIp driver**Table 884 Specification for EruOguReference (continued)**

Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuEruChannelOutputUnitConf		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.5 Container: GPT12Configuration

The container contains the configuration for GPTx timer if the hardware unit selected is GPT12.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

13.3.1.5.1 GPT12BlockReference**Table 885 Specification for GPT12BlockReference**

Name	GPT12BlockReference		
Description	GPT12 timer selection. Allocation is done in the MCU driver. The timer cell chosen here must be reserved by the user during the MCU driver configuration for the ICU driver. If the channel is incremental interface mode, only T2, T3 and T4 are possible. NOTE: Default value is set to blank as the user has to select the appropriate reference value from MCU.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuGpt12ModuleAllocationConf		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	IcuMeasurementMode		

Icu_17_TimerIp driver**13.3.1.5.2 GPT12CounterType****Table 886 Specification for GPT12CounterType**

Name	GPT12CounterType		
Description	Counting mechanism for Incremental interface mode. Note: Default value is set to the minimum of the count inputs.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ICU_1_COUNT_INPUT: Only Input is used for counting ICU_2_COUNT_INPUT: Both Input and Direction is used for counting		
Default value	ICU_1_COUNT_INPUT		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	IcuMeasurementMode		

13.3.1.5.3 GPT12DirPortSelection**Table 887 Specification for GPT12DirPortSelection**

Name	GPT12DirPortSelection		
Description	Direction Input selection for the GPT12 unit. If the ICU channel is configured for Incremental interface mode, the parameter should be configured with a valid port pin. Else the parameter should be configured as NONE.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GPT12_TnEUDx_PORTy_PINz: n is the GPT12 Timer selected x can be A or B depending on hardware. y is port number corresponding to GPT12 timer. z is port pin number corresponding to GPT12 timer. NONE: No pin selected.		
Default value	NONE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GPT12BlockReference, IcuMeasurementMode		

Icu_17_TimerIp driver**13.3.1.5.4 GPT12InputPortSelection****Table 888 Specification for GPT12InputPortSelection**

Name	GPT12InputPortSelection		
Description	Port pin selection for input. NOTE: The parameter reads from property file and depends upon channel selection. Default value is set to first encountered value from Resource property file.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GPT12_TnINx_[inputline]: n is the GPT12 Timer selected x can be A, B, C and so on depending on hardware. y is port number corresponding to GPT12 timer. 'inputline' can be PORT PIN or internal signal from other peripheral.		
Default value	GPT12_TnINx_[inputline]		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GPT12BlockReference		

13.3.1.6 Container: GtmTimerInputConfiguration

The container contains the configuration for TIM channel if the hardware unit selected is GTM.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

13.3.1.6.1 GtmTimerUsed**Table 889 Specification for GtmTimerUsed**

Name	GtmTimerUsed		
Description	The parameter is essentially a list of GTM TIM timer cells available for usage by an ICU logical channel. Referred timer channel in MCU should have McuGtmTimChannelAllocationConf as GTM_TIM_CHANNEL_USED_BY_ICU. NOTE: Default value is set to blank as user has to select the appropriate reference value from MCU.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuGtmTimChannelAllocationConf		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Icu_17_Timerlp driver**Table 889 Specification for GtmTimerUsed (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.7 Container: IcuTimestampMeasurement

The container contains the configuration parameters in case the measurement mode is time stamp.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

13.3.1.7.1 IcuTimestampMeasurementProperty**Table 890 Specification for IcuTimestampMeasurementProperty**

Name	IcuTimestampMeasurementProperty		
Description	Configures the handling of the buffer in case the mode is timestamp. Implementation type of this parameter is Icu_17_Timerlp_TimestampBufferType. Note: Default value is chosen as Linear buffer which is represented by a numerical value of 0.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ICU_CIRCULAR_BUFFER: After reaching the end of the buffer, the driver restarts at the beginning of the buffer ICU_LINEAR_BUFFER: The buffer will be filled once		
Default value	ICU_LINEAR_BUFFER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuMeasurementMode		

13.3.1.7.2 IcuTimestampNotification**Table 891 Specification for IcuTimestampNotification**

Name	IcuTimestampNotification
Description	The parameter is used by the ICU driver to invoke the user-defined function if the requested number of time stamps are acquired. The parameter can be configured as a name or an address(numeric value) of the notification function.

Icu_17_TimerIp driver**Table 891 Specification for IcuTimestampNotification (continued)**

	<p>Note1: By default, the notification parameter will be NULL , to remove dependency from user defined functions.</p> <p>Note2: The ICU driver does not validate the configured function name or address for correctness and hence the responsibility falls on the user.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuTimestampApi, IcuMeasurementMode		

13.3.1.8 Container: TimChannelFilterConfig

The container contains the filter configuration for TIM channel.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

13.3.1.8.1 TimChFilterCounterFreqSelect**Table 892 Specification for TimChFilterCounterFreqSelect**

Name	TimChFilterCounterFreqSelect		
Description	<p>The parameter decides the filter counter frequency for the TIM channel. The parameter modifies the FLT_CNT_FRQ of the TIM channel.</p> <p>Note: Default value is set to Hardware default value.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_CONFIGURABLE_CLOCK_0: Configurable clock 0 clocks the filter counter GTM_CONFIGURABLE_CLOCK_1: Configurable clock 1 clocks the filter counter GTM_CONFIGURABLE_CLOCK_6: Configurable clock 6 clocks the filter counter GTM_CONFIGURABLE_CLOCK_7: Configurable clock 7 clocks the filter counter		
Default value	GTM_CONFIGURABLE_CLOCK_0	Post-build variant multiplicity	-
Post-build variant value	TRUE	Multiplicity configuration class	-
Value configuration class	Post-Build		

Icu_17_TimerIp driver**Table 892 Specification for TimChFilterCounterFreqSelect (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.8.2 TimChFilterModeForFallingEdge**Table 893 Specification for TimChFilterModeForFallingEdge**

Name	TimChFilterModeForFallingEdge		
Description	The parameter decides the filter mode for falling edge of the TIM channel input. Note: Default value is set to Hardware default value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DEGLITCH_WITH_HOLD_COUNTER: Each edge of an input signal will be filtered with an individual de-glitch threshold filter value. Filter counter holds after inactive edge until active edge. DEGLITCH_WITH_UPDOWN_COUNTER: Each edge of an input signal will be filtered with an individual de-glitch threshold filter value. Filter counter counts down after inactive edge until active edge. IMMEDIATE_EDGE_PROPAGATION_MODE: After detection of an edge the new signal level is propagated and the new signal level remains unchanged until the configured acceptance time expires.		
Default value	IMMEDIATE_EDGE_PROPAGATION_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.8.3 TimChFilterModeForRisingEdge**Table 894 Specification for TimChFilterModeForRisingEdge**

Name	TimChFilterModeForRisingEdge		
Description	The parameter decides the filter mode for rising edge of the TIM channel input. Note: Default value is set to Hardware default value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DEGLITCH_WITH_HOLD_COUNTER: Each edge of an input signal will be filtered with an individual de-glitch threshold filter value. Filter counter holds after inactive edge until active edge.		

Icu_17_TimerIp driver**Table 894 Specification for TimChFilterModeForRisingEdge (continued)**

	DEGLITCH_WITH_UPDOWN_COUNTER: Each edge of an input signal will be filtered with an individual de-glitch threshold filter value. Filter counter counts down after inactive edge until active edge. IMMEDIATE_EDGE_PROPAGATION_MODE: After detection of an edge the new signal level is propagated and the new signal level remains unchanged until the configured acceptance time expires.		
Default value	IMMEDIATE_EDGE_PROPAGATION_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.8.4 TimChFilterTimeForFallingEdge**Table 895 Specification for TimChFilterTimeForFallingEdge**

Name	TimChFilterTimeForFallingEdge		
Description	The parameter specifies the filter time for falling edge of the TIM channel input. Note: Default value is set to Hardware default value.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.8.5 TimChFilterTimeForRisingEdge**Table 896 Specification for TimChFilterTimeForRisingEdge**

Name	TimChFilterTimeForRisingEdge		
Description	The parameter specifies the filter time for rising edge of the TIM channel input. Note: Default value is set to Hardware default value.		
Multiplicity	1..1	Type	EcclIntegerParamDef

Icu_17_TimerIp driver**Table 896 Specification for TimChFilterTimeForRisingEdge (continued)**

Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.8.6 TimChannelFilterEnable**Table 897 Specification for TimChannelFilterEnable**

Name	TimChannelFilterEnable		
Description	The parameter enables filter for the channel. Sets FLT_EN for the TIM channel. Note: Default value is set to Hardware default value.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.9 Container: TimChannelGeneral

The container contains the TIM channel specific configuration.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

13.3.1.9.1 OverflowISRThreshold**Table 898 Specification for OverflowISRThreshold**

Name	OverflowISRThreshold
-------------	----------------------

Icu_17_TimerIp driver**Table 898 Specification for OverflowISRThreshold (continued)**

Description	The threshold denotes the maximum latency between the actual TIM counter overflow interrupt and execution flow entering ICU ISR. User must configure the threshold to contain the maximum latency and must be lesser than the actual measured time. The threshold will be in the ticks of the CMU_CLK selected for that TIM channel. The threshold should consider interrupt latency, other high priority interrupts and the latency to reach the ICU ISR. The threshold is applicable only for Signal measurement, HIGH TIME and LOW TIME. Note: Default value is set to minimum value(0).		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.9.2 TimChannelClockSelect**Table 899 Specification for TimChannelClockSelect**

Name	TimChannelClockSelect		
Description	The parameter decides the clock source for TIM channel. Note: Default value is set to Hardware default value.		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	GTM_CONFIGURABLE_CLOCK_0: Configurable clock 0 will be supplied to the TIM channel GTM_CONFIGURABLE_CLOCK_1: Configurable clock 1 will be supplied to the TIM channel GTM_CONFIGURABLE_CLOCK_2: Configurable clock 2 will be supplied to the TIM channel GTM_CONFIGURABLE_CLOCK_3: Configurable clock 3 will be supplied to the TIM channel GTM_CONFIGURABLE_CLOCK_4: Configurable clock 4 will be supplied to the TIM channel GTM_CONFIGURABLE_CLOCK_5: Configurable clock 5 will be supplied to the TIM channel GTM_CONFIGURABLE_CLOCK_6: Configurable clock 6 will be supplied to the TIM channel GTM_CONFIGURABLE_CLOCK_7: Configurable clock 7 will be supplied to the TIM channel		
Default value	GTM_CONFIGURABLE_CLOCK_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Icu_17_TimerIp driver**Table 899 Specification for TimChannelClockSelect (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.9.3 TimChannelGpr0InputSelect**Table 900 Specification for TimChannelGpr0InputSelect**

Name	TimChannelGpr0InputSelect		
Description	<p>The parameter decides the reference timer for GPR0 register of TIM channel. The timer selected as reference should be enabled in MCU configurations. The GPR0 input can be selected only if the channel mode is time stamp.</p> <p>An error will be issued if the selected TBU channel is not enabled in MCU configuration.</p> <p>Note: Default value is set to Hardware default value.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	TIMEBASE_TBU_TS0: TBU_TS0 will be captured in GPRO TIMEBASE_TBU_TS1: TBU_TS1 will be captured in GPRO TIMEBASE_TBU_TS2: TBU_TS2 will be captured in GPRO		
Default value	TIMEBASE_TBU_TS0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTbuChannelEnable		

13.3.1.9.4 TimChannelInputSelect**Table 901 Specification for TimChannelInputSelect**

Name	TimChannelInputSelect		
Description	<p>The parameter decides the Input for the TIM channel.</p> <p>Note: Default value is set to Hardware default value.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	INPUT_OF_CURRENT_TIM_CHANNEL: Input to the current TIM channel will be the input assigned to current channel.		

Icu_17_TimerIp driver**Table 901 Specification for TimChannelInputSelect (continued)**

	INPUT_OF_PREVIOUS_TIM_CHANNEL: Input to the current TIM channel will be the input assigned to the previous channel.		
Default value	INPUT_OF_CURRENT_TIM_CHANNEL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.9.5 TimChannelPortPinSelect**Table 902 Specification for TimChannelPortPinSelect**

Name	TimChannelPortPinSelect		
Description	<p>The parameter decides the input port and pin for the GTM timer.</p> <p>Port pin is selectable only if input selected is of current TIM channel.</p> <p>Available port pins depends on target device and selected timer unit.</p> <p>Refer to target specification for more information.</p> <p>If TimChannelInputSelect parameter is selected as "INPUT_OF_PREVIOUS_TIM_CHANNEL", the parameter should be set as "SEL0_NONE". Else a valid port pin should be selected.</p> <p>NOTE: The parameter reads from property file and depends upon channel selection. Default value is set to first en-countered value from Resource property file.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>SEL0_NONE: No pin selected.</p> <p>SELx_PORTy_PINz: x can be 0-15 depending on hardware.</p> <p>y is port number corresponding to TIM channel.</p> <p>z is port pin number corresponding to TIM channel.</p>		
Default value	SEL0_NONE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	TimChannelInputSelect, GtmTimerUsed, TimChannelInputSelect		

Icu_17_TimerIp driver**13.3.1.9.6 TimInterruptMode****Table 903 Specification for TimInterruptMode**

Name	TimInterruptMode		
Description	The parameter decides the interrupt mode to be used. Note: Default value is set to Hardware default value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_INTERRUPT_LEVEL_MODE: Selects level mode for interrupt. GTM_INTERRUPT_PULSE_NOTIFY_MODE: Selects pulse notify mode for interrupt. GTM_INTERRUPT_SINGLE_PULSE_MODE: Selects single pulse mode for interrupt.		
Default value	GTM_INTERRUPT_LEVEL_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.10 Container: Icu

Configuration of ICU (Input Capture Unit) module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

13.3.1.10.1 Config Variant**Table 904 Specification for Config Variant**

Name	Config Variant		
Description	Selects the config-variant for the ICU driver. Note: Default value is set to post build as ICU support to post build value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VariantPostBuild: Post-build variant supported.		
Default value	VariantPostBuild		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

Icu_17_TimerIp driver
Table 904 Specification for Config Variant (continued)

Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.11 Container: IcuChannel

Configuration of an individual ICU channel.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

13.3.1.11.1 IcuAssignedHwUnit

Table 905 Specification for IcuAssignedHwUnit

Name	IcuAssignedHwUnit		
Description	The parameter chooses the capture engine required by an ICU channel. Note: Default value is chosen as Gtm which is represented by a numerical value of 0.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CCU6: Selects CCU6 as hardware unit to realize ICU channel ERU: Selects ERU as hardware unit to realize ICU channel GPT12: Selects GPT12 as hardware unit to realize ICU channel GTM: Selects GTM-TIM as hardware unit to realize ICU channel		
Default value	GTM		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	IcuMeasurementMode		

13.3.1.11.2 IcuChannelId

Table 906 Specification for IcuChannelId

Name	IcuChannelId		
Description	Logical channel identifier of the ICU channel. The parameters value will be assigned to the symbolic name derived from the IcuChannel container short name. The value of IcuChannelId should be unique in a configuration set. Note: Default value is set to minimum value.		
Multiplicity	1..1	Type	EcucIntegerParamDef

Icu_17_TimerIp driver**Table 906 Specification for IcuChannelId (continued)**

Range	0 - (Total number of channels - 1)		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

13.3.1.11.3 IcuDefaultStartEdge**Table 907 Specification for IcuDefaultStartEdge**

Name	IcuDefaultStartEdge		
Description	<p>Configures the default activation edge which will be used for the ICU channel.</p> <p>For Signal measurement, following conventions are to be adhered:</p> <p>PERIOD - Denotes the start of period.</p> <p>DUTY - Denotes the start of Period and Active time.</p> <p>HIGH TIME and LOW TIME, the parameter is irrelevant and will be un-editable.</p> <p>If BOTH EDGES is configured for DUTY CYCLE measurement, first edge seen after first call of Icu_17_TimerIp_StartSignalMeasurement is considered as default start edge for the entire Init - Delnit cycle.</p> <p>The parameter is unused and hence will be non-editable for incremental interface mode channel.</p> <p>Note: Default value is chosen as Rising Edge which is represented by a numerical value of 0(minimum).</p>		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	ICU_BOTH_EDGES: Both edges are used ICU_FALLING_EDGE: Falling edge is the used ICU_RISING_EDGE: Rising edge is the used		
Default value	ICU_RISING_EDGE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuMeasurementMode, IcuSignalMeasurementProperty		

Icu_17_TimerIp driver
13.3.1.11.4 IcuMeasurementMode
Table 908 Specification for IcuMeasurementMode

Name	IcuMeasurementMode
Description	<p>Configures the measurement mode of the ICU channel.</p> <p>ICU_MODE_SIGNAL_EDGE_DETECT: The channel is used for detecting the edges which are configured by the call of the service Icu_17_TimerIp_SetActivationCondition().</p> <p>The following API services support this mode:</p> <ul style="list-style-type: none"> - Icu_17_TimerIp_EnableEdgeDetection() - Icu_17_TimerIp_DisableEdgeDetection() - Icu_17_TimerIp_EnableMultiEdgeDetection() - Icu_17_TimerIp_EnableNotification() - Icu_17_TimerIp_DisableNotification() - Icu_17_TimerIp_GetInputState() <p>Edge detection mode can be configured if IcuEdgeDetectApi is switched on.</p> <p>Note: Default value is chosen as Edge detection which is represented by a numerical value of 0.</p> <p>ICU_MODE_SIGNAL_MEASUREMENT: The channel is used to measure different times between various configurable edges. The configuration of the period-start edges is done by configuration and cannot be changed during runtime.</p> <p>The following API services support this mode:</p> <ul style="list-style-type: none"> - Icu_17_TimerIp_StartSignalMeasurement() - Icu_17_TimerIp_StopSignalMeasurement() - Icu_17_TimerIp_GetTimeElapsed() - Icu_17_TimerIp_GetDutyCycleValues() - Icu_17_TimerIp_GetInputState() <p>Signal measurement mode can be configured if at least one of the following switches are set to TRUE:</p> <ul style="list-style-type: none"> - IcuGetDutyCycleValuesApi - IcuGetTimeElapsedApi <p>ICU_MODE_TIMESTAMP: The channel is used to capture timer values on the edges which are configured by the call of the service Icu_SetActivationCondition().</p> <p>The following API services support this mode:</p> <ul style="list-style-type: none"> - Icu_17_TimerIp_StartTimestamp() - Icu_17_TimerIp_StopTimestamp() - Icu_17_TimerIp_GetTimestampIndex() <p>Time stamping mode can be configured if IcuTimeStampApi is switched on.</p> <p>ICU_MODE_EDGE_COUNTER: The channel is used to count the edges which are configured by the call of the service Icu_SetActivationCondition().</p> <p>The following API services support this mode:</p> <ul style="list-style-type: none"> - Icu_17_TimerIp_EnableEdgeCount() - Icu_17_TimerIp_DisableEdgeCount() - Icu_17_TimerIp_GetEdgeNumbers()

Icu_17_TimerIp driver**Table 908 Specification for IcuMeasurementMode (continued)**

	<p>- Icu_17_TimerIp_ResetEdgeCount() Edge counting mode can be configured if IcuEdgeVountApi is switched on. ICU_MODE_INCREMENTAL_INTERFACE: The channel is configured to count the encoder edges(using the incremental interface mode of GPT12 peripheral). The following API services support this mode: Icu_17_TimerIp_StartIncInterface() Icu_17_TimerIp_StopIncInterface() Icu_17_TimerIp_CalibratePos() Icu_17_TimerIp_ReadEncCount() Icu_17_TimerIp_EncCountDirType() Incremental interface mode can be configured if IcuIncrementalInterfaceApi is switched on.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ICU_MODE_EDGE_COUNTER: The channel is configured to count the edges which are set by the call of service Icu_17_TimerIp_SetActivationCondition(). ICU_MODE_INCREMENTAL_INTERFACE: The channel is configured to count the encoder edges(using the incremental interface mode of GPT12 peripheral). ICU_MODE_SIGNAL_EDGE_DETECT: The channel is configured for detecting the edges which are set by the call of service Icu_17_TimerIp_SetActivationCondition(). ICU_MODE_SIGNAL_MEASUREMENT: The channel is configured to measure signal properties. The configuration of the period start edges is done during configuration and cannot be changed during runtime. ICU_MODE_TIMESTAMP: The channel is configured to capture timer values on the edges which are set by the call of service Icu_17_TimerIp_SetActivationCondition().		
Default value	ICU_MODE_SIGNAL_EDGE_DETECT		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.11.5 IcuWakeupCapability**Table 909 Specification for IcuWakeupCapability**

Name	IcuWakeupCapability
Description	Information about the wakeup-capability of the ICU channel. TRUE: Channel is wakeup capable. FALSE: Channel is not wakeup capable. Wakeup capability value can be TRUE only if the channel is an edge detect channel.

Icu_17_TimerIp driver
Table 909 Specification for IcuWakeupCapability (continued)

	Note: By default value is set to False, to remove the dependency from ECUM.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuMeasurementMode		

13.3.1.12 Container: IcuConfigSet

The container contains the configuration parameters and sub containers of the ICU driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

13.3.1.12.1 IcuMaxChannel

Table 910 Specification for IcuMaxChannel

Name	IcuMaxChannel		
Description	The parameter contains the number of channels configured. The parameters value will be gathered by tools during the configuration stage. calculationFormula = Number of configured ICU channels Implementation Type: Icu_17_TimerIp_ChannelType Note: The parameter is non-editable as it is not used in any code generation. The value shall not be used for any references.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuChannel		

Icu_17_TimerIp driver**13.3.1.13 Container: IcuGeneral**

Configuration of general ICU parameters. Note: By default all the error reporting (Development, Safety and Multi-core) are enable, to ensure proper driver functionality.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

13.3.1.13.1 IcuDevErrorDetect**Table 911 Specification for IcuDevErrorDetect**

Name	IcuDevErrorDetect		
Description	Enables or disables the Default Error Tracer (DET) detection and reporting. true: enabled (ON). false: disabled (OFF).		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.13.2 IcuIndex**Table 912 Specification for IcuIndex**

Name	IcuIndex		
Description	Specifies the instance Id of the ICU driver. If only one instance is present, the value of the parameter should be 0. Note: Default value is set to the minimum value.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

Icu_17_TimerIp driver**Table 912 Specification for IcuIndex (continued)**

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.13.3 IcuInitDeInitApiMode**Table 913 Specification for IcuInitDeInitApiMode**

Name	IcuInitDeInitApiMode		
Description	<p>Pre-processor switch to enable or disable protected register access in Icu_17_TimerIp_Init and Icu_17_TimerIp_DeInit APIs. If IcuRuntimeApiMode is set to ICU_MCAL_SUPERVISOR, IcuInitDeInitApiMode has to be set to ICU_MCAL_SUPERVISOR.</p> <p>Note: By default access level of all the runtime APIs set to Supervisor so that there is no dependency on the OS functions to write into the access protected SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ICU_MCAL_SUPERVISOR: ICU init APIs will run in supervisor mode. ICU_MCAL_USER1: ICU init APIs will run in user1 mode.		
Default value	ICU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	IcuRuntimeApiMode		

13.3.1.13.4 IcuMultiCoreErrorDetect**Table 914 Specification for IcuMultiCoreErrorDetect**

Name	IcuMultiCoreErrorDetect		
Description	The parameter enables or disables the multi core related default error tracer (DET) detection and reporting. It is applicable only when DETs are enabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Icu_17_TimerIp driver**Table 914 Specification for IcuMultiCoreErrorDetect (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	IcuDevErrorDetect		

13.3.1.13.5 IcuReportWakeupSource**Table 915 Specification for IcuReportWakeupSource**

Name	IcuReportWakeupSource		
Description	Switch for enabling wakeup source reporting. true: Report wakeup source. false: Do not report wakeup source.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.13.6 IcuRuntimeApiMode**Table 916 Specification for IcuRuntimeApiMode**

Name	IcuRuntimeApiMode		
Description	Pre-processor switch to enable or disable protected register access in runtime APIs. Note: By default access level of all the runtime APIs set to Supervisor so that there is no dependency on the OS functions to write into the access protected SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ICU_MCAL_SUPERVISOR: ICU runtime APIs will run in supervisor mode. ICU_MCAL_USER1: ICU runtime APIs will run in user1 mode.		
Default value	ICU_MCAL_SUPERVISOR		

Icu_17_TimerIp driver**Table 916 Specification for IcuRuntimeApiMode (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.13.7 IcuSafetyEnable**Table 917 Specification for IcuSafetyEnable**

Name	IcuSafetyEnable		
Description	Pre-processor switch for enabling the safety features of ICU driver.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.14 Container: IcuOptionalApis

The container contains all configuration switches for configuring optional API services of the ICU driver. Note: All optional APIs set to False except initCheck , to minimize executable code size.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

13.3.1.14.1 IcuDeInitApi**Table 918 Specification for IcuDeInitApi**

Name	IcuDeInitApi
Description	Adds / removes the service Icu_17_TimerIp_DeInit() from the code. TRUE: Icu_17_TimerIp_DeInit() can be used. FALSE: Icu_17_TimerIp_DeInit() cannot be used.

Icu_17_TimerIp driver**Table 918 Specification for IcuDeInitApi (continued)**

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.14.2 IcuDisableWakeupApi**Table 919 Specification for IcuDisableWakeupApi**

Name	IcuDisableWakeupApi		
Description	Adds / removes the service Icu_17_TimerIp_DisableWakeup() from the code. IcuDisableWakeupApi can be set to true only if IcuEnableWakeupApi is true. TRUE: Icu_17_TimerIp_DisableWakeup() can be used. FALSE: Icu_17_TimerIp_DisableWakeup() cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.14.3 IcuEdgeCountApi**Table 920 Specification for IcuEdgeCountApi**

Name	IcuEdgeCountApi
-------------	-----------------

Icu_17_TimerIp driver**Table 920 Specification for IcuEdgeCountApi (continued)**

Description	Adds / removes all services related to the edge counting functionality, as listed below, from the code: Icu_17_TimerIp_ResetEdgeCount(), Icu_17_TimerIp_EnableEdgeCount(), Icu_17_TimerIp_DisableEdgeCount(), Icu_17_TimerIp_GetEdgeNumbers(). IcuEdgeCountApi must be set to true if there is at least one channel in edge count mode. IcuEdgeCountApi must be set to false if there are no GPT12 and GTM channels. TRUE: The services listed above can be used. FALSE: The services listed above cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuAssignedHwUnit, IcuMeasurementMode		

13.3.1.14.4 IcuEdgeDetectApi**Table 921 Specification for IcuEdgeDetectApi**

Name	IcuEdgeDetectApi		
Description	Adds / removes the services related to the edge detection functionality from the code: Icu_17_TimerIp_EnableEdgeDetection(), Icu_17_TimerIp_DisableEdgeDetection() and Icu_17_TimerIp_EnableMultiEdgeDetection IcuEdgeDetectApi must be set to true if there is at least one channel in edge detect mode. TRUE: These services can be used. FALSE: These services cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

Icu_17_TimerIp driver**Table 921 Specification for IcuEdgeDetectApi (continued)**

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuMeasurementMode		

13.3.1.14.5 IcuEnableWakeupApi**Table 922 Specification for IcuEnableWakeupApi**

Name	IcuEnableWakeupApi		
Description	Adds / removes the service Icu_17_TimerIp_EnableWakeup() from the code. IcuEnableWakeupApi must be set to true if there is at least one channel's wake-up capability is true. TRUE: Icu_17_TimerIp_EnableWakeup() can be used. FALSE: Icu_17_TimerIp_EnableWakeup() cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.14.6 IcuGetDutyCycleValuesApi**Table 923 Specification for IcuGetDutyCycleValuesApi**

Name	IcuGetDutyCycleValuesApi		
Description	Adds / removes the service Icu_17_TimerIp_GetDutyCycleValues() from the code. IcuGetDutyCycleValuesApi must be set to true if there is at least one channel measuring duty cycle. IcuGetDutyCycleValuesApi can be set to true only if IcuSignalMeasurementApi is true. TRUE: Icu_17_TimerIp_GetDutyCycleValues() can be used. FALSE: Icu_17_TimerIp_GetDutyCycleValues() cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		

Icu_17_TimerIp driver**Table 923 Specification for IcuGetDutyCycleValuesApi (continued)**

Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuSignalMeasurementApi		

13.3.1.14.7 IcuGetInputStateApi**Table 924 Specification for IcuGetInputStateApi**

Name	IcuGetInputStateApi		
Description	Adds / removes the service Icu_17_TimerIp_GetInputState() from the code. TRUE: Icu_17_TimerIp_GetInputState() can be used. FALSE: Icu_17_TimerIp_GetInputState() cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.14.8 IcuGetTimeElapsedApi**Table 925 Specification for IcuGetTimeElapsedApi**

Name	IcuGetTimeElapsedApi
Description	Adds / removes the service Icu_17_TimerIp_GetTimeElapsed() from the code. IcuGetTimeElapsedApi must be set to true if there is at least one channel in signal measurement mode measuring a non duty cycle value. IcuGetTimeElapsedApi can be set to true only if IcuSignalMeasurementApi is true. TRUE: Icu_17_TimerIp_GetTimeElapsed() can be used. FALSE: Icu_17_TimerIp_GetTimeElapsed() cannot be used.

Icu_17_TimerIp driver**Table 925 Specification for IcuGetTimeElapsedApi (continued)**

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuSignalMeasurementApi		

13.3.1.14.9 IcuGetVersionInfoApi**Table 926 Specification for IcuGetVersionInfoApi**

Name	IcuGetVersionInfoApi		
Description	Adds / removes the service Icu_17_TimerIp_GetVersionInfo() from the code. TRUE: Icu_17_TimerIp_GetVersionInfo() can be used. FALSE: Icu_17_TimerIp_GetVersionInfo() cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.14.10 IcuIncrementalInterfaceApi**Table 927 Specification for IcuIncrementalInterfaceApi**

Name	IcuIncrementalInterfaceApi
-------------	----------------------------

Icu_17_TimerIp driver**Table 927 Specification for IcuIncrementalInterfaceApi (continued)**

Description	<p>Adds / removes all services related to the incremental interface functionality, as listed below, from the code: Icu_17_TimerIp_StartInclInterface(), Icu_17_TimerIp_StopInclInterface(), Icu_17_TimerIp_CalibratePos(), Icu_17_TimerIp_ReadEncCount(), Icu_17_TimerIp_ReadEncCountDir().</p> <p>IcuIncrementalInterfaceApi must be set to true if there is at least one channel in incremental interface mode.</p> <p>IcuIncrementalInterfaceApi must be set to false if there are no GPT12 channels.</p> <p>TRUE: The services listed above can be used.</p> <p>FALSE: The services listed above cannot be used.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	IcuMeasurementMode, IcuAssignedHwUnit		

13.3.1.14.11 IcuInitCheckApi**Table 928 Specification for IcuInitCheckApi**

Name	IcuInitCheckApi		
Description	<p>Pre-processor switch for enabling/disabling the safety feature Icu_17_TimerIp_InitCheck() which verifies the initialization done by ICU driver.</p> <p>If the parameter is set to TRUE, the Icu_17_TimerIp_InitCheck() API can be used to verify the initialization done by ICU driver .</p> <p>If this parameter is set to FALSE, Icu_17_TimerIp_InitCheck() API cannot be used.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Icu_17_TimerIp driver**Table 928 Specification for IcuInitCheckApi (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

13.3.1.14.12 IcuSetModeApi**Table 929 Specification for IcuSetModeApi**

Name	IcuSetModeApi		
Description	Adds / removes the service Icu_17_TimerIp_SetMode() from the code. TRUE: Icu_17_TimerIp_SetMode() can be used. FALSE: Icu_17_TimerIp_SetMode() cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.14.13 IcuSignalMeasurementApi**Table 930 Specification for IcuSignalMeasurementApi**

Name	IcuSignalMeasurementApi
Description	Adds / removes the services Icu_17_TimerIp_StartSignalMeasurement() and Icu_17_TimerIp_StopSignalMeasurement() from the code. IcuSignalMeasurementApi should be set to true if there is at least one channel in signal measurement mode. IcuSignalMeasurementApi should be set to false if there are no GTM and CCU6 channels. TRUE: Icu_17_TimerIp_StartSignalMeasurement() and Icu_17_TimerIp_StopSignalMeasurement() can be used. FALSE: Icu_17_TimerIp_StartSignalMeasurement() and Icu_17_TimerIp_StopSignalMeasurement() cannot be used.

Icu_17_TimerIp driver**Table 930 Specification for IcuSignalMeasurementApi (continued)**

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuAssignedHwUnit, IcuMeasurementMode		

13.3.1.14.14 IcuTimestampApi**Table 931 Specification for IcuTimestampApi**

Name	IcuTimestampApi		
Description	Adds / removes all services related to the time stamping functionality, as listed below, from the code: Icu_17_TimerIp_StartTimestamp(), Icu_17_TimerIp_StopTimestamp(), Icu_17_TimerIp_GetTimestampIndex(). IcuTimestampApi should be set to true if there is at least one channel in time stamping mode. IcuTimestampApi should be set to false if there are no GTM channels and no CCU6 channels. TRUE: The services listed above can be used. FALSE: The services listed above cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuAssignedHwUnit, IcuMeasurementMode		

Icu_17_TimerIp driver**13.3.1.14.15 IcuWakeupFunctionalityApi****Table 932 Specification for IcuWakeupFunctionalityApi**

Name	IcuWakeupFunctionalityApi		
Description	Adds / removes the service Icu_17_TimerIp_CheckWakeups() from the code. IcuWakeupFunctionalityApi should be set to true if at least one channel's wake-up capability is set to true. IcuWakeupFunctionalityApi should be set to false if there are no channels with wake-up capability set to true. TRUE: Icu_17_TimerIp_CheckWakeups() can be used. FALSE: Icu_17_TimerIp_CheckWakeups() cannot be used.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.1.15 Container: IcuSignalEdgeDetection

The container contains the configuration (parameters) in case the measurement mode is edge detection.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

13.3.1.15.1 IcuSignalNotification**Table 933 Specification for IcuSignalNotification**

Name	IcuSignalNotification		
Description	The parameter is used by the ICU driver to invoke the user-defined function if the configured edge is detected. The parameter can be configured as a name or an address(numeric value) of the notification function. Note1: By default, the notification parameter will be NULL , to remove dependency from user defined functions. Note2: The ICU driver does not validate the configured function name or address for correctness and hence the responsibility falls on the user.		
Multiplicity	0..1	Type	EcucFunctionNameDef

Icu_17_TimerIp driver
Table 933 Specification for IcuSignalNotification (continued)

Range	String		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuMeasurementMode		

13.3.1.16 Container: IcuSignalMeasurement

The container contains the configuration (parameters) in case the measurement mode is signal measurement.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

13.3.1.16.1 IcuSignalMeasurementProperty
Table 934 Specification for IcuSignalMeasurementProperty

Name	IcuSignalMeasurementProperty		
Description	Configures the property that could be measured in case the mode is signal measurement. The signal measurement property can not be changed during runtime. Duty cycle can only be selected if IcuGetDutyCyclesApi is available. High time/low time/period can be selected only if IcuGetTimeElapsedApi is available. For period measurement, IcuDefaultStartEdge should not be Both edges. Implementation type: Icu_17_TimerIp_SignalMeasurementPropertyType. Note: Default value is chosen as Low time which is represented by a numerical value of 0.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ICU_DUTY_CYCLE: The channel is configured to read values which are needed for calculating the duty cycle (coherent active and period time). ICU_HIGH_TIME: The channel is configured for reading the elapsed signal high time. ICU_LOW_TIME: The channel is configured for reading the elapsed signal low time. ICU_PERIOD_TIME: The channel is configured for reading the elapsed signal period time.		
Default value	ICU_LOW_TIME		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

Icu_17_TimerIp driver
Table 934 Specification for IcuSignalMeasurementProperty (continued)

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	IcuGetTimeElapsedApi, IcuGetDutyCycleValuesApi, IcuMeasurementMode		

13.3.1.17 Container: IcuWakeup

The container contains the configuration (parameters) needed to configure a wake-up capable channel.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

13.3.1.17.1 IcuChannelWakeuInfo
Table 935 Specification for IcuChannelWakeuInfo

Name	IcuChannelWakeuInfo		
Description	<p>If the wakeup-capability is true the wakeup source referenced is transmitted to the ECU State Manager (EcUM).</p> <p>IcuChannelWakeuInfo is editable only if channel wakeup capability is true and IcuReportWakeuSource is true.</p> <p>Implementation type: reference to EcUM_WakeuSourceType.</p> <p>Note: By default wake-up info is set to NULL , to remove dependency from EcUM wake-up configurations.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: EcUMWakeuSource		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

13.3.2 Functions - Type definitions
13.3.2.1 Icu_17_TimerIp_NotifiPtrType
Table 936 Specification for Icu_17_TimerIp_NotifiPtrType

Syntax	Icu_17_TimerIp_NotifiPtrType
Type	Pointer to a function of type void Function_Name (void)

Icu_17_TimerIp driver
Table 936 Specification for Icu_17_TimerIp_NotifiPtrType (continued)

File	Icu_17_TimerIp.h
Description	Channel notification function pointer (notification function applicable in case of channel configured for edge detect or time stamp mode).
Source	IFX

Icu_17_TimerIp_ModeType
Table 937 Specification for Icu_17_TimerIp_ModeType

Syntax	Icu_17_TimerIp_ModeType	
Type	Enumeration	
File	Icu_17_TimerIp.h	
Range	0 - ICU_17_TIMERIP_MODE_NORMAL	Normal operation, all used interrupts are enabled according to the notification requests.
	1 - ICU_17_TIMERIP_MODE_SLEEP	Reduced power operation. In sleep mode only those notifications are available which are configured as wakeup capable.
Description	Allow enabling / disabling of all interrupts which are not required for the ECU wakeup.	
Source	AUTOSAR	

Icu_17_TimerIp_ChannelType
Table 938 Specification for Icu_17_TimerIp_ChannelType

Syntax	Icu_17_TimerIp_ChannelType	
Type	uint8	
File	Icu_17_TimerIp.h	
Range	0-82	The range includes the total number of TIM channels, CCU6 comparators, ERU channels and GPT12 timers. The maximum number of channels may vary depending on the device variant. 82 is considering the superset device variant. NOTE: This is the maximum possible valid range. The actual valid range is 0-(Total number of ICU channels configured - 1)
Description	Numeric identifier of an ICU channel.	
Source	AUTOSAR	

Icu_17_TimerIp driver**13.3.2.4 Icu_17_TimerIp_EncCountDirType****Table 939 Specification for Icu_17_TimerIp_EncCountDirType**

Syntax	Icu_17_TimerIp_EncCountDirType	
Type	Enumeration	
File	Icu_17_TimerIp.h	
Range	1 - ICU_17_TIMERIP_ENC_COUNT_DOWN	Encoder counting down
	0 - ICU_17_TIMERIP_ENC_COUNT_UP	Encoder counting up
Description	Encoder counting direction for an incremental interface mode ICU channel.	
Source	IFX	

13.3.2.5 Icu_17_TimerIp_InputStateType**Table 940 Specification for Icu_17_TimerIp_InputStateType**

Syntax	Icu_17_TimerIp_InputStateType	
Type	Enumeration	
File	Icu_17_TimerIp.h	
Range	1 - ICU_17_TIMERIP_ACTIVE	An activation edge has been detected
	0 - ICU_17_TIMERIP_IDLE	No activation edge has been detected since the last call of Icu_GetInputState() or Icu_Init().
Description	Input state of an ICU channel.	
Source	AUTOSAR	

13.3.2.6 Icu_17_TimerIp_ConfigType**Table 941 Specification for Icu_17_TimerIp_ConfigType**

Syntax	Icu_17_TimerIp_ConfigType	
Type	Structure	
File	Icu_17_TimerIp.h	
Range	--	The elements of the data structure are specific to the micro-controller.
Description	The data type contains initialization data.	
Source	IFX	

13.3.2.7 Icu_17_TimerIp_ActivationType**Table 942 Specification for Icu_17_TimerIp_ActivationType**

Syntax	Icu_17_TimerIp_ActivationType
---------------	-------------------------------

Icu_17_TimerIp driver**Table 942 Specification for Icu_17_TimerIp_ActivationType (continued)**

Type	Enumeration	
File	Icu_17_TimerIp.h	
Range	0 - ICU_17_TIMERIP_RISING_EDGE	An appropriate action will be executed when a rising edge occurs on the ICU input signal.
	1 - ICU_17_TIMERIP_FALLING_EDGE	An appropriate action will be executed when a falling edge occurs on the ICU input signal.
	2 - ICU_17_TIMERIP_BOTH_EDGES	An appropriate action will be executed when either a rising or falling edge occurs on the ICU input signal.
	3 - ICU_17_TIMERIP_NO_EDGE	No edge is selected.
Description	Definition of the type of activation of an ICU channel.	
Source	AUTOSAR	

Icu_17_TimerIp_ValueType**Table 943 Specification for Icu_17_TimerIp_ValueType**

Syntax	Icu_17_TimerIp_ValueType	
Type	uint32	
File	Icu_17_TimerIp.h	
Range	0-16777215	
Description	Width of the buffer for timestamp ticks and measured elapsed timeticks. 24-bit range for GTM(TIM) Channel. 16-bit range for GPT12 and CCU6 Channel.	
Source	AUTOSAR	

Icu_17_TimerIp_DutyCycleType**Table 944 Specification for Icu_17_TimerIp_DutyCycleType**

Syntax	Icu_17_TimerIp_DutyCycleType	
Type	Structure	
File	Icu_17_TimerIp.h	
Range	Icu_17_TimerIp_ValueType ActiveTime	Coherent active time measured on a channel.
	Icu_17_TimerIp_ValueType PeriodTime	Coherent period time measured on a channel.
Description	Contains the values needed for calculating duty cycles.	
Source	AUTOSAR	

Icu_17_TimerIp driver**13.3.2.10 Icu_17_TimerIp_IndexType****Table 945 Specification for Icu_17_TimerIp_IndexType**

Syntax	Icu_17_TimerIp_IndexType
Type	uint16
File	Icu_17_TimerIp.h
Range	0-65535
Description	Type, to abstract the return value of the service Icu_17_TimerIp_GetTimestampIndex(). Since circular buffer handling is supported and Icu_17_TimerIp_GetTimestampIndex can return zero as a legally true value.
Source	AUTOSAR

13.3.2.11 Icu_17_TimerIp_EdgeNumberType**Table 946 Specification for Icu_17_TimerIp_EdgeNumberType**

Syntax	Icu_17_TimerIp_EdgeNumberType
Type	uint32
File	Icu_17_TimerIp.h
Range	0-4294967295
Description	Type to abstract the return value of the service Icu_17_TimerIp_GetEdgeNumbers().
Source	AUTOSAR

13.3.2.12 Icu_17_TimerIp_MeasurementModeType**Table 947 Specification for Icu_17_TimerIp_MeasurementModeType**

Syntax	Icu_17_TimerIp_MeasurementModeType	
Type	Enumeration	
File	Icu_17_TimerIp.h	
Range	0 - ICU_17_TIMERIP_MODE_SIGNAL_EDGE_DETECT	Edge Detection Mode
	1 - ICU_17_TIMERIP_MODE_SIGNAL_MEASUREMENT	Mode for measuring different times between various configurable edges
	2 - ICU_17_TIMERIP_MODE_TIMESTAMP	Mode for capturing timer values on configurable edges
	3 - ICU_17_TIMERIP_MODE_EDGE_COUNTER	Mode for counting edges on configurable edges

Icu_17_TimerIp driver**Table 947 Specification for Icu_17_TimerIp_MeasurementModeType (continued)**

	4 - ICU_17_TIMERIP_MODE_INCREMENTEL_INTERFACE	Incremental Interface mode
Description	Definition of ICU measurement mode. Member of a config structure.	
Source	AUTOSAR	

13.3.2.13 Icu_17_TimerIp_SignalMeasurementPropertyType**Table 948 Specification for Icu_17_TimerIp_SignalMeasurementPropertyType**

Syntax	Icu_17_TimerIp_SignalMeasurementPropertyType	
Type	Enumeration	
File	Icu_17_TimerIp.h	
Range	0 - ICU_17_TIMERIP_LOW_TIME	The channel is configured for reading the elapsed signal low time
	1 - ICU_17_TIMERIP_HIGH_TIME	The channel is configured for reading the elapsed signal high time
	2 - ICU_17_TIMERIP_PERIOD_TIME	The channel is configured for reading the elapsed signal period time
	3 - ICU_17_TIMERIP_DUTY_CYCLE	The channel is configured to read values which are needed for calculating the duty cycle (coherent active and period time)
Description	Definition of the measurement property type.	
Source	AUTOSAR	

13.3.2.14 Icu_17_TimerIp_TimestampBufferType**Table 949 Specification for Icu_17_TimerIp_TimestampBufferType**

Syntax	Icu_17_TimerIp_TimestampBufferType	
Type	Enumeration	
File	Icu_17_TimerIp.h	
Range	0 - ICU_17_TIMERIP_LINEAR_BUFFER	Buffer will be filled once
	1 - ICU_17_TIMERIP_CIRCULAR_BUFFER	After reaching the end of the buffer, the driver restarts at the beginning of the buffer
Description	Definition of the timestamp measurement property type.	
Source	AUTOSAR	

Icu_17_TimerIp driver**13.3.3 Functions - APIs****13.3.3.1 Icu_17_TimerIp_CalibratePos****Table 950 Specification for Icu_17_TimerIp_CalibratePos API**

Syntax	<pre>void Icu_17_TimerIp_CalibratePos (const Icu_17_TimerIp_ChannelType Channel, const uint16 Position)</pre>	
Service ID	0x23	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel Position	Numeric identifier of the ICU channel Start point to be set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function calibrates the start point for incremental interface mode functionality.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuIncrementalInterfaceApi	
User hints	-	

Icu_17_TimerIp driver**13.3.3.2 Icu_17_TimerIp_CheckWakeups****Table 951 Specification for Icu_17_TimerIp_CheckWakeups API**

Syntax	<pre>void Icu_17_TimerIp_CheckWakeups (const EcuM_WakeupSourceType WakeupSource)</pre>	
Service ID	0x15	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	WakeupSource	Information on wakeup source to be checked. The associated ICU channel can be determined from configuration data.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Checks if a wakeup capable ICU channel is the source for a wakeup event and calls the ECU state manager service EcuM_SetWakeupEvent in case of a valid ICU channel wakeup event.</p> <p>For multicore, the ICU channel should be allocated to the core in which this function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET: ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuReportWakeupSource,IcuWakeupFunctionalityApi	
User hints	-	

Icu_17_TimerIp_DeInit**Table 952 Specification for Icu_17_TimerIp_DeInit API**

Syntax	<pre>void Icu_17_TimerIp_DeInit (void)</pre>
---------------	--

Icu_17_TimerIp driver**Table 952 Specification for Icu_17_TimerIp_DeInit API (continued)**

Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function de-initializes the ICU driver.</p> <p>For multicore, the function will de-initialize those channels allocated to the core in which the function is invoked. Additionally if called from master core, de-initialize the resources shared among the cores.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_SLAVE_CORE_INIT: Error reported when master de-initialization is called without de-initializing slave core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuDelInitApi	
User hints	User should not call Icu_17_TimerIp_DeInit during a running operation (e. g. timestamp measurement or edge counting)	

Icu_17_TimerIp_DisableEdgeCount**Table 953 Specification for Icu_17_TimerIp_DisableEdgeCount API**

Syntax	void Icu_17_TimerIp_DisableEdgeCount (const Icu_17_TimerIp_ChannelType Channel)
Service ID	0x0e
Sync/Async	Synchronous
ASIL Level	B

Icu_17_TimerIp driver**Table 953 Specification for Icu_17_TimerIp_DisableEdgeCount API (continued)**

Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function disables the counting of edges of the given channel.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuEdgeCountApi	
User hints	-	

Icu_17_TimerIp_DisableNotification**Table 954 Specification for Icu_17_TimerIp_DisableNotification API**

Syntax	void Icu_17_TimerIp_DisableNotification (const Icu_17_TimerIp_ChannelType Channel)
Service ID	0x06
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for different channel

Icu_17_TimerIp driver
Table 954 Specification for Icu_17_TimerIp_DisableNotification API (continued)

Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The function disables the notification of a channel. For multicore, the ICU channel should be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	DET: ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization. ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode. ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core. Runtime Errors: None DEM: None Safety Errors: ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

Icu_17_TimerIp_DisableWakeup
Table 955 Specification for Icu_17_TimerIp_DisableWakeup API

Syntax	void Icu_17_TimerIp_DisableWakeup (const Icu_17_TimerIp_ChannelType Channel)
Service ID	0x03
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for different channel

Icu_17_TimerIp driver**Table 955 Specification for Icu_17_TimerIp_DisableWakeup API (continued)**

Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function disables the wakeup capability of a single ICU channel.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuDisableWakeupApi	
User hints	-	

Icu_17_TimerIp_EnableEdgeCount**Table 956 Specification for Icu_17_TimerIp_EnableEdgeCount API**

Syntax	void Icu_17_TimerIp_EnableEdgeCount (const Icu_17_TimerIp_ChannelType Channel)
Service ID	0x0d
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for different channel

Icu_17_TimerIp driver**Table 956 Specification for Icu_17_TimerIp_EnableEdgeCount API (continued)**

Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function enables the counting of edges of the given channel.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuEdgeCountApi	
User hints	-	

Icu_17_TimerIp_EnableMultiEdgeDetection**Table 957 Specification for Icu_17_TimerIp_EnableMultiEdgeDetection API**

Syntax	void Icu_17_TimerIp_EnableMultiEdgeDetection (const Icu_17_TimerIp_ChannelType Channel, const uint32 EdgeCount)
Service ID	0x19
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for different channel

Icu_17_TimerIp driver**Table 957 Specification for Icu_17_TimerIp_EnableMultiEdgeDetection API (continued)**

Parameters (in)	Channel EdgeCount	Numeric identifier of the ICU channel Number of edges before interrupt occurs
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The multi edge detection feature is provided to generate interrupts after specified number of edges. It is possible to enable this feature at runtime for desired number of edges.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	IFX	
Error handling	<p>DET:</p> <ul style="list-style-type: none"> ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization. ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode. ICU_17_TIMERIP_E_PARAM_EDGE_NUMBER: Edge count parameter is set as zero or not in the range supported by channel. Valid only for multi edge detection API. ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core. Runtime Errors: None DEM: None Safety Errors: ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables. <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuEdgeDetectApi	
User hints	-	

Icu_17_TimerIp_EnableNotification**Table 958 Specification for Icu_17_TimerIp_EnableNotification API**

Syntax	void Icu_17_TimerIp_EnableNotification (const Icu_17_TimerIp_ChannelType Channel)
Service ID	0x07
Sync/Async	Synchronous

Icu_17_TimerIp driver**Table 958 Specification for Icu_17_TimerIp_EnableNotification API (continued)**

ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function enables the notification on the given channel.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p>ICU_17_TIMERIP_E_INVALID_NOTIF: Notification invoked on a non- notification function configured channel.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

Icu_17_TimerIp_EnableWakeup**Table 959 Specification for Icu_17_TimerIp_EnableWakeup API**

Syntax	void Icu_17_TimerIp_EnableWakeup (const Icu_17_TimerIp_ChannelType Channel)
Service ID	0x04

Icu_17_TimerIp driver**Table 959 Specification for Icu_17_TimerIp_EnableWakeup API (continued)**

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function (re-)enables the wakeup capability of the given ICU channel.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuEnableWakeupApi	
User hints	-	

Icu_17_TimerIp_GetDutyCycleValues**Table 960 Specification for Icu_17_TimerIp_GetDutyCycleValues API**

Syntax	void Icu_17_TimerIp_GetDutyCycleValues (const Icu_17_TimerIp_ChannelType Channel, Icu_17_TimerIp_DutyCycleType * const DutyCycleValues)
Service ID	0x11

Icu_17_TimerIp driver**Table 960 Specification for Icu_17_TimerIp_GetDutyCycleValues API (continued)**

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	DutyCycleValues	Pointer to a buffer where the results (active time and period time) will be placed
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function reads the coherent active time and period time for the given ICU Channel.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_PARAM_POINTER: API called with invalid pointer.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuGetDutyCycleValuesApi	
User hints	<p>For a GTM channel the overflow is identified and ZERO will be returned.</p> <p>For a CCU6 channel there is no unique way to identify overflow and hence the input signal must be within the 16-bit range.</p>	

Icu_17_TimerIp_GetEdgeNumbers**Table 961 Specification for Icu_17_TimerIp_GetEdgeNumbers API**

Syntax	Icu_17_TimerIp_EdgeNumberType Icu_17_TimerIp_GetEdgeNumbers (
---------------	--

Icu_17_TimerIp driver**Table 961 Specification for Icu_17_TimerIp_GetEdgeNumbers API (continued)**

	const Icu_17_TimerIp_ChannelType Channel)	
Service ID	0x0f	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Icu_17_TimerIp_EdgeNumberType	Edge Count for an ICU channel.
Description	<p>The function reads the number of counted edges.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuEdgeCountApi	
User hints	This API can be invoked even if edge counting activity is not active.	

Icu_17_TimerIp_GetInputState**Table 962 Specification for Icu_17_TimerIp_GetInputState API**

Syntax	Icu_17_TimerIp_InputStateType Icu_17_TimerIp_GetInputState (
---------------	---

Icu_17_TimerIp driver**Table 962 Specification for Icu_17_TimerIp_GetInputState API (continued)**

	const Icu_17_TimerIp_ChannelType Channel)	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Icu_17_TimerIp_InputStat eType	ICU_17_TIMERIP_ACTIVE/ICU_17_TIMERIP_IDLE
Description	<p>The function returns the status of the ICU input.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuGetInputStateApi	
User hints	If Icu_GetInputState API is invoked for a channel in multi-edge detection mode, the channel status is set to ACTIVE only after the required number of edges are detected and not on the next detected single active edge.	

Icu_17_TimerIp driver**13.3.3.14 Icu_17_TimerIp_GetTimeElapsed****Table 963 Specification for Icu_17_TimerIp_GetTimeElapsed API**

Syntax	<pre>Icu_17_TimerIp_ValueType Icu_17_TimerIp_GetTimeElapsed (const Icu_17_TimerIp_ChannelType Channel)</pre>	
Service ID	0x10	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Icu_17_TimerIp_ValueType	Signal Low time, High timer or period value for the channel.
Description	<p>This function reads the elapsed Signal Measurement Time for the given channel</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuGetTimeElapsedApi	
User hints	<p>For a GTM channel the overflow is identified and ZERO will be returned.</p> <p>For a CCU6 channel there is no unique way to identify overflow and hence the input signal must be within the 16-bit range.</p>	

Icu_17_TimerIp driver**13.3.3.15 Icu_17_TimerIp_GetTimestampIndex****Table 964 Specification for Icu_17_TimerIp_GetTimestampIndex API**

Syntax	<pre>Icu_17_TimerIp_IndexType Icu_17_TimerIp_GetTimestampIndex (const Icu_17_TimerIp_ChannelType Channel)</pre>	
Service ID	0x0b	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Icu_17_TimerIp_IndexType	Timestamp index next to be written.
Description	<p>The function reads the timestamp index of the given channel. For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET: ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization. ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode. ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuTimestampApi	
User hints	This API will return the size of the buffer if the buffer is full and buffer configuration is linear. This API can be invoked even if there is no active time stamping activity.	

Icu_17_TimerIp driver**13.3.3.16 Icu_17_TimerIp_GetVersionInfo****Table 965 Specification for Icu_17_TimerIp_GetVersionInfo API**

Syntax	<pre>void Icu_17_TimerIp_GetVersionInfo (Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x12	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where to store the version information.
Parameters (in - out)	-	-
Return	void	-
Description	The function returns the version information of the ICU driver.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_PARAM_VINFO: Icu_17_TimerIp_GetVersionInfo API called with a NULL_PTR.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuGetVersionInfoApi	
User hints	The API can be called before ICU initialization.	

13.3.3.17 Icu_17_TimerIp_InitCheck**Table 966 Specification for Icu_17_TimerIp_InitCheck API**

Syntax	<pre>Std_ReturnType Icu_17_TimerIp_InitCheck (const Icu_17_TimerIp_ConfigType * const ConfigPtr)</pre>
Service ID	0x30
Sync/Async	Synchronous
ASIL Level	B

Icu_17_TimerIp driver**Table 966 Specification for Icu_17_TimerIp_InitCheck API (continued)**

Re-entrancy	Reentrant	
Parameters (in)	ConfigPtr	Pointer to a selected configuration structure
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK - if initialization comparison is success.E_NOT_OK - if initialization comparison fails.
Description	<p>Will check against all SFRs or variables initialized by Init API including initialization status flag.</p> <p>It does not modify any SFR/variable, only a read operation is done.</p> <p>If any failure in comparison, it reports an error.</p> <p>For multicore, the function will check the initialization of those channels allocated to the core in which this function is invoked. Additionally for master core, the function will check the initialization of the resources which are shared among cores.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuInitCheckApi	
User hints	-	

Icu_17_TimerIp_ReadEncCount**Table 967 Specification for Icu_17_TimerIp_ReadEncCount API**

Syntax	<pre>uint16 Icu_17_TimerIp_ReadEncCount (const Icu_17_TimerIp_ChannelType Channel)</pre>	
Service ID	0x24	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel

Icu_17_TimerIp driver**Table 967 Specification for Icu_17_TimerIp_ReadEncCount API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint16	Encoder counter value
Description	<p>The function reads the current encoder count value. The encoder count and direction are not impacted by the call of the API.</p> <p>If a DET/Safety error is identified, 0 is returned.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuIncrementalInterfaceApi	
User hints	-	

Icu_17_TimerIp_ReadEncCountDir**Table 968 Specification for Icu_17_TimerIp_ReadEncCountDir API**

Syntax	Icu_17_TimerIp_EncCountDirType Icu_17_TimerIp_ReadEncCountDir (const Icu_17_TimerIp_ChannelType Channel)
Service ID	0x25
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for different channel

Icu_17_TimerIp driver**Table 968 Specification for Icu_17_TimerIp_ReadEncCountDir API (continued)**

Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Icu_17_TimerIp_EncCount DirType	Counting direction
Description	<p>The function to read the direction of rotation. The encoder count and direction are not impacted by the call of the API.</p> <p>If a DET/Safety error is identified, ICU_17_TIMERIP_ENC_COUNT_UP is returned.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuIncrementalInterfaceApi	
User hints	-	

Icu_17_TimerIp_ResetEdgeCount**Table 969 Specification for Icu_17_TimerIp_ResetEdgeCount API**

Syntax	void Icu_17_TimerIp_ResetEdgeCount (const Icu_17_TimerIp_ChannelType Channel)
Service ID	0x0c
Sync/Async	Synchronous
ASIL Level	B

Icu_17_TimerIp driver**Table 969 Specification for Icu_17_TimerIp_ResetEdgeCount API (continued)**

Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function resets the value of the counted edges to zero.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuEdgeCountApi	
User hints	No active edge should be seen on the input pin during the execution of the API.	

Icu_17_TimerIp_SetActivationCondition**Table 970 Specification for Icu_17_TimerIp_SetActivationCondition API**

Syntax	void Icu_17_TimerIp_SetActivationCondition (const Icu_17_TimerIp_ChannelType Channel, const Icu_17_TimerIp_ActivationType Activation)
Service ID	0x05
Sync/Async	Synchronous
ASIL Level	B

Icu_17_TimerIp driver**Table 970 Specification for Icu_17_TimerIp_SetActivationCondition API (continued)**

Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel Activation	Numeric identifier of the ICU channel Type of activation edge to be configured <ul style="list-style-type: none">- ICU_17_TIMERIP_RISING_EDGE- ICU_17_TIMERIP_FALLING_EDGE- ICU_17_TIMERIP_BOTH_EDGES
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function sets the activation-edge for the given channel. For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET: ICU_17_TIMERIP_E_PARAM_ACTIVATION: Invalid activation parameter in API. ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization. ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode. ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p>ICU_17_TIMERIP_E_BUSY_CHANNEL: Activation edge of a time stamp channel modified during an active time stamping operation.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	<p>The channel on which Icu_SetActivationCondition is invoked, must not have any on-going operations to ensure proper functionality.</p> <p>The API must be invoked only on channels configured in edge detection, edge counting and time stamping mode.</p> <p>The API will issue a ICU_E_PARAM_CHANNEL DET/Safety error if the channel parameter corresponds to a signal measurement channel.</p>	

Icu_17_TimerIp driver**13.3.3.22 Icu_17_TimerIp_StartIncInterface****Table 971 Specification for Icu_17_TimerIp_StartIncInterface API**

Syntax	<pre>void Icu_17_TimerIp_StartIncInterface (const Icu_17_TimerIp_ChannelType Channel)</pre>	
Service ID	0x21	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function starts the incremental interface mode activity of the ICU channel. The encoder count and direction are not impacted by the call of the API.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuIncrementalInterfaceApi	
User hints	-	

Icu_17_TimerIp driver**13.3.3.23 Icu_17_TimerIp_StartSignalMeasurement****Table 972 Specification for Icu_17_TimerIp_StartSignalMeasurement API**

Syntax	<pre>void Icu_17_TimerIp_StartSignalMeasurement (const Icu_17_TimerIp_ChannelType Channel)</pre>	
Service ID	0x13	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function starts the measurement of signals.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	IcuSignalMeasurementApi	
User hints	-	

Icu_17_TimerIp driver**13.3.3.24 Icu_17_TimerIp_StartTimestamp****Table 973 Specification for Icu_17_TimerIp_StartTimestamp API**

Syntax	<pre>void Icu_17_TimerIp_StartTimestamp (const Icu_17_TimerIp_ChannelType Channel, Icu_17_TimerIp_ValueType * const BufferPtr, const uint16 BufferSize, const uint16 NotifyInterval)</pre>	
Service ID	0x09	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel BufferPtr BufferSize NotifyInterval	Numeric identifier of the ICU channel Pointer to the buffer-array where the timestamp values will be placed. Size of the external buffer (number of entries) Notification interval (number of events).
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function starts the capturing of timer values on the edges.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_POINTER: API called with invalid pointer.</p> <p>ICU_17_TIMERIP_E_PARAM_NOTIFY_INTERVAL: Icu_17_TimerIp_StartTimeStamp API called with invalid NotifyInterval parameter. Zero is considered as invalid value.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_PARAM_BUFFER_SIZE: Invalid buffer size used in API.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p>	

Icu_17_TimerIp driver**Table 973 Specification for Icu_17_TimerIp_StartTimestamp API (continued)**

	<p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p>ICU_17_TIMERIP_E_PARAM_IMPLAUSIBLE_NOTIFY_INTERVAL: Notify interval is greater than buffer size in case of a Linear buffer.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	IcuTimestampApi
User hints	-

13.3.3.25 Icu_17_TimerIp_StopIncInterface**Table 974 Specification for Icu_17_TimerIp_StopIncInterface API**

Syntax	<pre>void Icu_17_TimerIp_StopIncInterface (const Icu_17_TimerIp_ChannelType Channel)</pre>	
Service ID	0x22	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function stops the incremental interface mode activity of the ICU channel. The encoder count and direction are not impacted by the call of the API.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>Runtime Errors: None</p> <p>DEM: None</p>	

Icu_17_TimerIp driver**Table 974 Specification for Icu_17_TimerIp_StopIncInterface API (continued)**

	<p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	IcuIncrementalInterfaceApi
User hints	-

13.3.3.26 Icu_17_TimerIp_StopSignalMeasurement**Table 975 Specification for Icu_17_TimerIp_StopSignalMeasurement API**

Syntax	<pre>void Icu_17_TimerIp_StopSignalMeasurement (const Icu_17_TimerIp_ChannelType Channel)</pre>	
Service ID	0x14	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function stops the measurement of signals of the given channel.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p>	

Icu_17_TimerIp driver**Table 975 Specification for Icu_17_TimerIp_StopSignalMeasurement API (continued)**

	ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables. <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	IcuSignalMeasurementApi
User hints	-

13.3.3.27 Icu_17_TimerIp_StopTimestamp**Table 976 Specification for Icu_17_TimerIp_StopTimestamp API**

Syntax	void Icu_17_TimerIp_StopTimestamp (const Icu_17_TimerIp_ChannelType Channel)	
Service ID	0x0a	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The function stops the timestamp measurement of the given channel. For multicore, the ICU channel should be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	DET: ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode. ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization. ICU_17_TIMERIP_E_NOT_STARTED: An ICU API which stops a particular feature is called before the start of that feature. ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core. Runtime Errors: None DEM: None	

Icu_17_TimerIp driver**Table 976 Specification for Icu_17_TimerIp_StopTimestamp API (continued)**

	<p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	IcuTimestampApi
User hints	None.

Icu_17_TimerIp_EnableEdgeDetection**Table 977 Specification for Icu_17_TimerIp_EnableEdgeDetection API**

Syntax	<pre>void Icu_17_TimerIp_EnableEdgeDetection (const Icu_17_TimerIp_ChannelType Channel)</pre>	
Service ID	0x16	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function enables / re-enables the detection of edges of the given channel.</p> <p>For multicore, the ICU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization.</p> <p>ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode.</p> <p>ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p>	

Icu_17_TimerIp driver**Table 977 Specification for Icu_17_TimerIp_EnableEdgeDetection API (continued)**

	ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables. <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	IcuEdgeDetectApi
User hints	-

13.3.3.29 Icu_17_TimerIp_DisableEdgeDetection**Table 978 Specification for Icu_17_TimerIp_DisableEdgeDetection API**

Syntax	void Icu_17_TimerIp_DisableEdgeDetection (const Icu_17_TimerIp_ChannelType Channel)	
Service ID	0x17	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The function disables the detection of edges of the given channel. For multicore, the ICU channel should be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	DET: ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization. ICU_17_TIMERIP_E_PARAM_CHANNEL: Invalid channel number or the channel is not configured for the required measurement mode. ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH: An API is called with the channel not allocated to executing core. Runtime Errors: None DEM: None Safety Errors:	

Icu_17_TimerIp driver
Table 978 Specification for Icu_17_TimerIp_DisableEdgeDetection API (continued)

	ICU_17_TIMERIP_E_INVALID_MODE: API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables. <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	IcuEdgeDetectApi
User hints	-

13.3.3.30 Icu_17_TimerIp_Init
Table 979 Specification for Icu_17_TimerIp_Init API

Syntax	void Icu_17_TimerIp_Init (const Icu_17_TimerIp_ConfigType * const ConfigPtr)	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to a selected configuration structure
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function initializes the driver.</p> <p>For multicore, the function will initialize those channels allocated to the core in which this function is invoked. Additionally for master core, the function will initialize the resources which are shared among cores.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>ICU_17_TIMERIP_E_INIT_FAILED: Configuration pointer is NULL_PTR.</p> <p>ICU_17_TIMERIP_E_ALREADY_INITIALIZED: Icu_17_TimerIp_Init API service called when the ICU driver and the hardware are already initialized.</p> <p>ICU_17_TIMERIP_E_MASTER_CORE_UNINIT: Error reported when slave core init is called without initializing master core.</p> <p>ICU_17_TIMERIP_E_CORE_NOT_CONFIGURED: Error reported when the ICU driver is not configured for the core in which an API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p>	

Icu_17_TimerIp driver**Table 979 Specification for Icu_17_TimerIp_Init API (continued)**

	Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	Signal measurement will not be started after Init. A call to Icu_17_TimerIp_StartSignalMeasurement is required to start the signal measurement activity.

Icu_17_TimerIp_SetMode**Table 980 Specification for Icu_17_TimerIp_SetMode API**

Syntax	void Icu_17_TimerIp_SetMode (const Icu_17_TimerIp_ModeType Mode)	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Mode	ICU_17_TIMERIP_MODE_NORMAL: Normal operation, all used interrupts are enabled according to the notification requests. ICU_17_TIMERIP_MODE_SLEEP: Reduced power mode. In sleep mode only those notifications are available which are configured as wakeup capable.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The function sets the ICU mode.</p> <p>The DET, ICU_17_TIMERIP_E_BUSY_OPERATION, is issued if SLEEP mode is requested during a running operation of edge count channel, incremental interface channel, time stamp channel or signal measurement channel.</p> <p>For multicore, the function sets the mode of the core with which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	DET: ICU_17_TIMERIP_E_BUSY_OPERATION: Icu_17_TimerIp_SetMode is called when a channel is in running condition. ICU_17_TIMERIP_E_UNINIT: API service used without the driver initialization. ICU_17_TIMERIP_E_PARAM_MODE: Invalid mode is passed for the API.	

Icu_17_TimerIp driver**Table 980 Specification for Icu_17_TimerIp_SetMode API (continued)**

	<p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	IcuSetModeApi
User hints	None.

13.3.4 Notifications and Callbacks**13.3.4.1 Icu_17_TimerIp_Timer_Isr****Table 981 Specification for Icu_17_TimerIp_Timer_Isr API**

Syntax	<pre>void Icu_17_TimerIp_Timer_Isr (const Icu_17_TimerIp_ChannelType Channel, const uint32 Flags)</pre>	
Service ID	0x20	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Channel	Logical channel identifier.
	Flags	Interrupt flags responsible for ISR
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Callback function from MCU to service timer (ERU, GTM-TIM, CCU6 and GPT12) interrupts for all modes of ICU. The ISR is reentrant because access to any non-channel based timer resource is protected by protection mechanisms.	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>ICU_17_TIMERIP_E_INVALID_ISR: ISR invoked on a spurious interrupt.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

Icu_17_TimerIp driver

Table 981 Specification for Icu_17_TimerIp_Timer_Isr API (continued)

Configuration dependencies	-
User hints	-

13.3.5 Scheduled functions

There are no scheduled functions for the ICU driver.

13.3.6 Interrupt service routines

ICU does not require any interrupt handler.

13.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

13.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Table 982 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
Icu_17_TimerIp_Init API service called when the ICU driver and the hardware are already initialized.	AUTOSAR	ICU_17_TIMERIP_E_ALREADY_INITIALIZED=0x17	Icu_17_TimerIp_Init
Icu_17_TimerIp_SetMode is called when a channel is in running condition.	AUTOSAR	ICU_17_TIMERIP_E_BUSY_OPERATION=0x16	Icu_17_TimerIp_SetMode
An API is called with the channel not allocated to executing core.	IFX	ICU_17_TIMERIP_E_CORE_CHANNEL_MISMATCH=0x65	Icu_17_TimerIp_ReadEncoderCountDir, Icu_17_TimerIp_ReadEncoderCount, Icu_17_TimerIp_CalibratePos, Icu_17_TimerIp_StopInclineInterface, Icu_17_TimerIp_StartInclineInterface, Icu_17_TimerIp_EnableEdgeDetection, Icu_17_TimerIp_EnableNotification, Icu_17_TimerIp_DisableNotification, Icu_17_TimerIp_DisableEncoder

Icu_17_TimerIp driver**Table 982 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
			dgeDetection, Icu_17_TimerIp_DisableEdgeCount, Icu_17_TimerIp_DisableWakeup, Icu_17_TimerIp_EnableEdgeCount, Icu_17_TimerIp_EnableWakeup, Icu_17_TimerIp_GetDutyCycleValues, Icu_17_TimerIp_GetEdgeNumbers, Icu_17_TimerIp_GetInputState, Icu_17_TimerIp_GetTimeElapsed, Icu_17_TimerIp_GetTimestampIndex, Icu_17_TimerIp_ResetEdgeCount, Icu_17_TimerIp_SetActivationCondition, Icu_17_TimerIp_StartSignMeasurement, Icu_17_TimerIp_StartTimeStamp, Icu_17_TimerIp_StopSignMeasurement, Icu_17_TimerIp_StopTimeStamp, Icu_17_TimerIp_EnableMultiEdgeDetection
Error reported when the ICU driver is not configured for the core in which an API is invoked.	IFX	ICU_17_TIMERIP_E_CORE_NOT_CONFIGURED=0x64	Icu_17_TimerIp_Init
Configuration pointer is NULL_PTR.	AUTOSAR	ICU_17_TIMERIP_E_INIT_FAILED=0xd	Icu_17_TimerIp_Init
Error reported when slave core init is called without initializing master core.	IFX	ICU_17_TIMERIP_E_MASTER_CORE_UNINIT=0x66	Icu_17_TimerIp_Init
An ICU API which stops a particular feature is called before the start of that feature.	AUTOSAR	ICU_17_TIMERIP_E_NOT_STARTED=0x15	Icu_17_TimerIp_StopTimeStamp

Icu_17_TimerIp driver**Table 982 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
Invalid activation parameter in API.	AUTOSAR	ICU_17_TIMERIP_E_PARAM_ACTIVATION=0x0c	Icu_17_TimerIp_SetActivationCondition
Invalid buffer size used in API.	AUTOSAR	ICU_17_TIMERIP_E_PARAM_BUFFER_SIZE=0x0e	Icu_17_TimerIp_StartTimestamp
Invalid channel number or the channel is not configured for the required measurement mode.	AUTOSAR	ICU_17_TIMERIP_E_PARAM_CHANNEL=0x0b	Icu_17_TimerIp_ReadEncoderCountDir, Icu_17_TimerIp_ReadEncoderCount, Icu_17_TimerIp_CalibratePos, Icu_17_TimerIp_StopInCounterface, Icu_17_TimerIp_StartInCounterface, Icu_17_TimerIp_SetActivationCondition, Icu_17_TimerIp_EnableNotification, Icu_17_TimerIp_DisableNotification, Icu_17_TimerIp_EnableMultiEdgeDetection, Icu_17_TimerIp_GetInputState, Icu_17_TimerIp_StartSignalMeasurement, Icu_17_TimerIp_StopSignalMeasurement, Icu_17_TimerIp_GetTimeElapsed, Icu_17_TimerIp_GetDutyCycleValues, Icu_17_TimerIp_ResetEdgeCount, Icu_17_TimerIp_GetEdgeNumbers, Icu_17_TimerIp_DisableEdgeCount, Icu_17_TimerIp_EnableEdgeCount, Icu_17_TimerIp_EnableEdgeDetection, Icu_17_TimerIp_DisableEdgeDetection, Icu_17_TimerIp_GetTimestampIndex, Icu_17_TimerIp_StopTime

Icu_17_TimerIp driver**Table 982 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
			stamp, Icu_17_TimerIp_StartTim estamp, Icu_17_TimerIp_EnableW akeup, Icu_17_TimerIp_DisableW akeup
Edge count parameter is set as zero or not in the range supported by channel. Valid only for multi edge detection API.	IFX	ICU_17_TIMERIP_E_PARAM_EDGE_N UMBER=0x21	Icu_17_TimerIp_EnableM ultiEdgeDetection
Invalid mode is passed for the API.	AUTOSAR	ICU_17_TIMERIP_E_PARAM_MODE=0 x0f	Icu_17_TimerIp_SetMode
Icu_17_TimerIp_StartTime Stamp API called with invalid NotifyInterval parameter. Zero is considered as invalid value.	AUTOSAR	ICU_17_TIMERIP_E_PARAM_NOTIFY_I NTerval=0x18	Icu_17_TimerIp_StartTim estamp
API called with invalid pointer.	AUTOSAR	ICU_17_TIMERIP_E_PARAM_POINTER =0x0a	Icu_17_TimerIp_GetDutyC ycleValues, Icu_17_TimerIp_StartTim estamp
Icu_17_TimerIp_GetVersio nInfo API called with a NULL_PTR.	AUTOSAR	ICU_17_TIMERIP_E_PARAM_VINFO=0 x19	Icu_17_TimerIp_GetVersio nInfo
Error reported when master de-initialization is called without de-initializing slave core.	IFX	ICU_17_TIMERIP_E_SLAVE_CORE_INI T=0x67	Icu_17_TimerIp_DelInit
API service used without the driver initialization.	AUTOSAR	ICU_17_TIMERIP_E_UNINIT=0x14	Icu_17_TimerIp_ReadEnc CountDir, Icu_17_TimerIp_ReadEnc Count, Icu_17_TimerIp_Calibrate Pos, Icu_17_TimerIp_StopIncl nterface, Icu_17_TimerIp_StartIncl nterface, Icu_17_TimerIp_SetActivat ionCondition, Icu_17_TimerIp_EnableN otification,

Icu_17_TimerIp driver

Table 982 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
			Icu_17_TimerIp_DisableNotification, Icu_17_TimerIp_EnableMultiEdgeDetection, Icu_17_TimerIp_GetInputState, Icu_17_TimerIp_StartSignalMeasurement, Icu_17_TimerIp_StopSignalMeasurement, Icu_17_TimerIp_GetTimeElapsed, Icu_17_TimerIp_GetDutyCycleValues, Icu_17_TimerIp_ResetEdgeCount, Icu_17_TimerIp_GetEdgeNumbers, Icu_17_TimerIp_DisableEdgeCount, Icu_17_TimerIp_EnableEdgeCount, Icu_17_TimerIp_EnableEdgeDetection, Icu_17_TimerIp_DisableEdgeDetection, Icu_17_TimerIp_GetTimestampIndex, Icu_17_TimerIp_StopTimeStamp, Icu_17_TimerIp_StartTimeStamp, Icu_17_TimerIp_CheckWakeUp, Icu_17_TimerIp_EnableWakeUp, Icu_17_TimerIp_DisableWakeUp, Icu_17_TimerIp_SetMode, Icu_17_TimerIp_DelInit

13.3.7.2 Production errors

The driver does not report any production errors.

13.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Icu_17_TimerIp driver**Table 983 Description of safety errors reported**

Description	Source	Error code and value	Applicable APIs
Activation edge of a time stamp channel modified during an active time stamping operation.	IFX	ICU_17_TIMERIP_E_BUSY_CHANNEL =0xCC	Icu_17_TimerIp_SetActivationCondition
ISR invoked on a spurious interrupt.	IFX	ICU_17_TIMERIP_E_INVALID_ISR=0xC9	Icu_17_TimerIp_Timer_Isr
API is invoked in SLEEP mode, with an intention to modify channel hardware registers or global variables.	IFX	ICU_17_TIMERIP_E_INVALID_MODE=0xCA	Icu_17_TimerIp_GetInputState, Icu_17_TimerIp_ReadEncCountDir, Icu_17_TimerIp_ReadEncCount, Icu_17_TimerIp_GetEdgeNumbers, Icu_17_TimerIp_CalibratePos, Icu_17_TimerIp_StopInclineInterface, Icu_17_TimerIp_StartInclineInterface, Icu_17_TimerIp_StopTimeStamp, Icu_17_TimerIp_StopSignMeasurement, Icu_17_TimerIp_SetActivationCondition, Icu_17_TimerIp_ResetEdgeCount, Icu_17_TimerIp_GetTimeElapsed, Icu_17_TimerIp_GetDutyCycleValues, Icu_17_TimerIp_StartTimestamp, Icu_17_TimerIp_StartSignMeasurement, Icu_17_TimerIp_EnableWakeups, Icu_17_TimerIp_EnableNotification, Icu_17_TimerIp_EnableMultiEdgeDetection, Icu_17_TimerIp_EnableEdgeDetection, Icu_17_TimerIp_EnableEdgeCount, Icu_17_TimerIp_DisableWakeups

Icu_17_TimerIp driver

Table 983 Description of safety errors reported (continued)

Description	Source	Error code and value	Applicable APIs
			akeup, Icu_17_TimerIp_DisableNotification, Icu_17_TimerIp_DisableEdgeDetection, Icu_17_TimerIp_DisableEdgeCount
Notification invoked on a non-notification function configured channel.	IFX	ICU_17_TIMERIP_E_INVALID_NOTIF=0xCB	Icu_17_TimerIp_EnableNotification
Notify interval is greater than buffer size in case of a Linear buffer.	IFX	ICU_17_TIMERIP_E_PARAM_IMPLAUSIBLE_NOTIFY_INTERVAL=0xCD	Icu_17_TimerIp_StartTimestamp

13.3.7.4 Runtime errors

The driver does not report any runtime errors.

13.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

13.3.8.1 Deviations

There are no deviations reported for the ICU driver.

13.3.8.2 Limitations

The section describes the limitations from software specification.

Table 984 Known limitations

Reference	Limitation
Icu_17_TimerIp_GetInputState, Icu_17_TimerIp_GetTimeElapsed, Icu_17_TimerIp_GetDutyCycleValues	For an ICU channel configured in the signal measurement mode using the CCU6 hardware, the measured value will be in the 16-bit range. The overflow cannot be identified due to hardware limitation.
Icu_17_TimerIp_GetInputState, Icu_17_TimerIp_GetTimeElapsed, Icu_17_TimerIp_GetDutyCycleValues	For an ICU channel configured in the signal measurement mode using the GTM-TIM hardware, the measured value will be in the 24-bit range. If the input signal is such that the measured value is more than 24-bit, zero shall be returned.

13.3.9 Unsupported hardware features

The Timeout Detection Unit hardware feature of the GTM-TIM is not supported by the ICU driver:

IRQ driver

14 IRQ driver

14.1 User information

14.1.1 Description

The IRQ driver provides the necessary configuration parameters and APIs for interrupt configuration, initialization and handling.

The driver is responsible for:

- Configuration of priority number for service requests
- Runtime APIs for initialization of service request nodes with configured priority and service provider (CPU_x, DMA where x varies from 0 to number of available cores)
- Runtime APIs for initialization of general service request nodes with configured priority and service provider which can be used for software trigger service requests
- Runtime APIs for clearing of service request flags for SRNs
- If CAT2 is selected, operating system should take care of interrupt handling

The IRQ driver is implemented as Pre-Compile variant.

14.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

IRQ driver

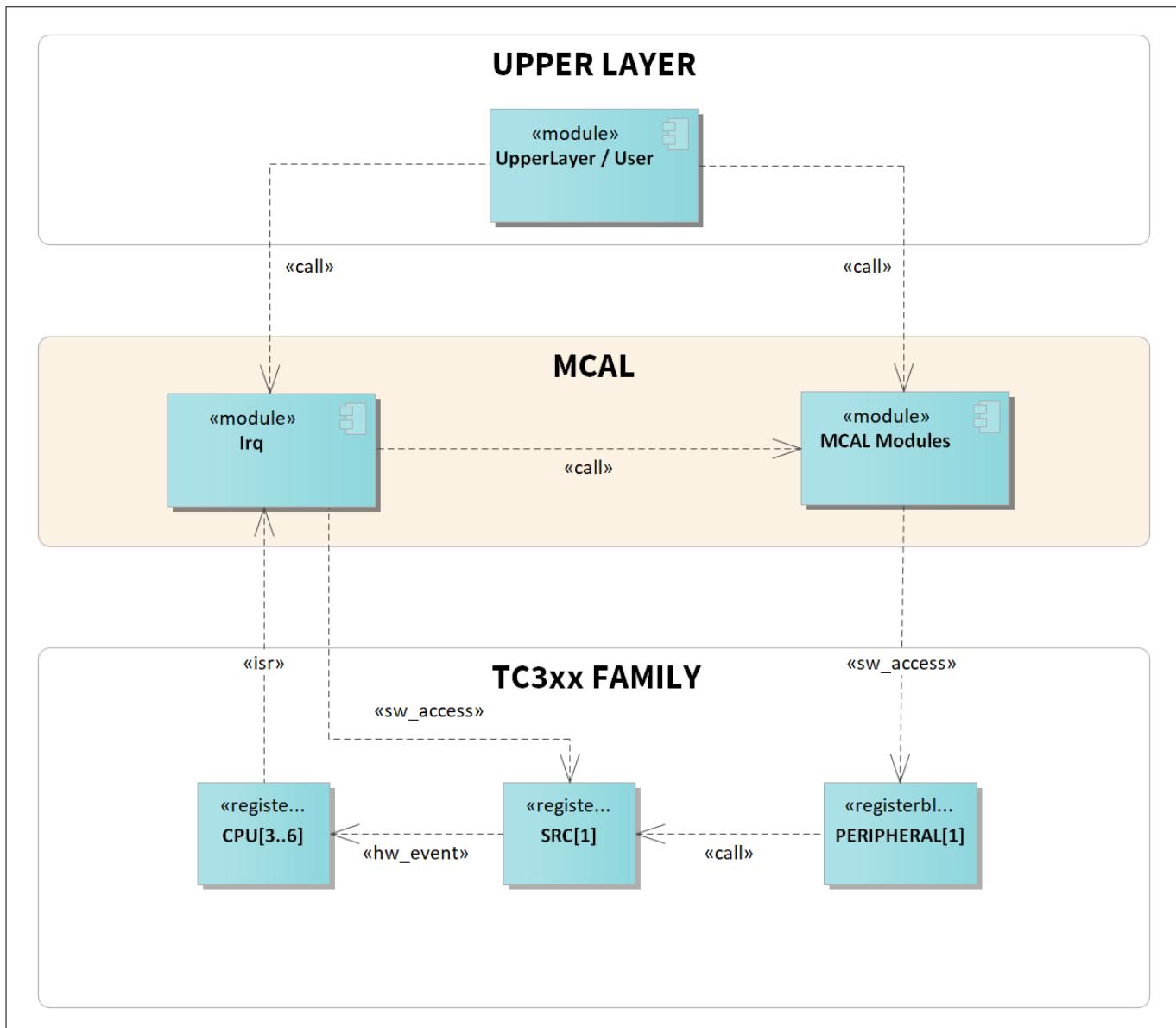


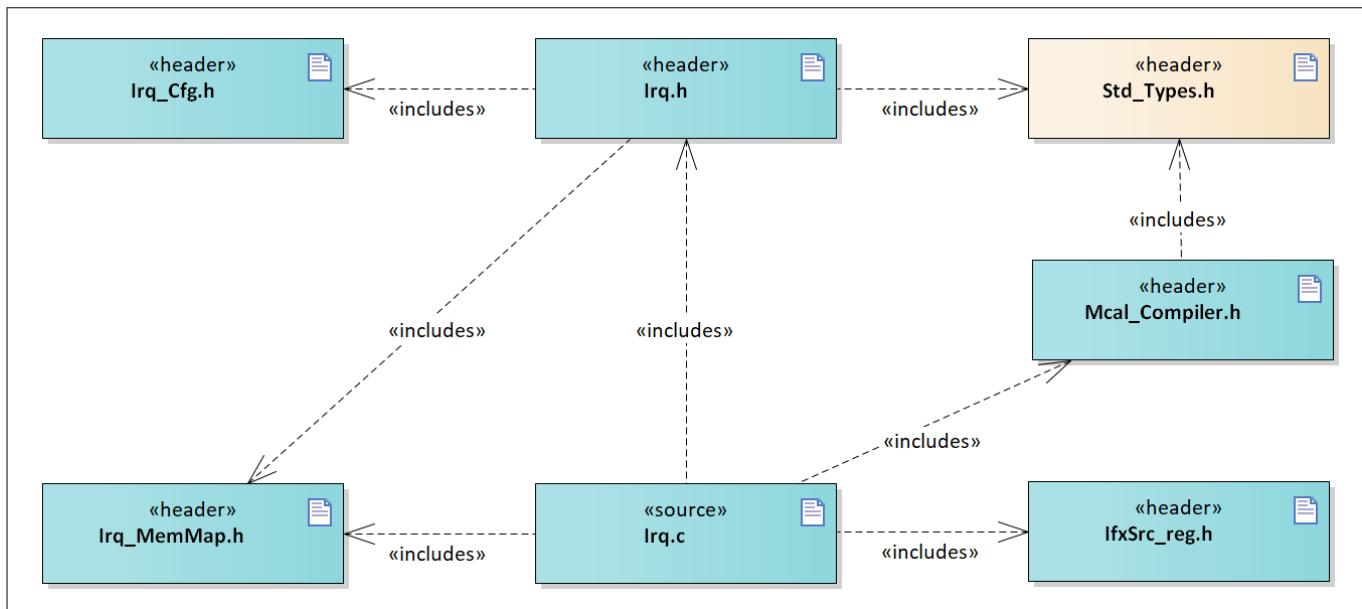
Figure 133 Mapping of hardware-software interfaces

The IRQ driver accesses the registers of interrupt controller. The driver initializes the peripheral-specific SRNs priority. On receiving an interrupt, the interrupt system jumps to the appropriate interrupt handler frame within the IRQ driver and the corresponding interrupt handler function is called. In the MCAL drivers, the interrupt handler functions are located within the software driver.

14.1.3 File structure

14.1.3.1 C file structure

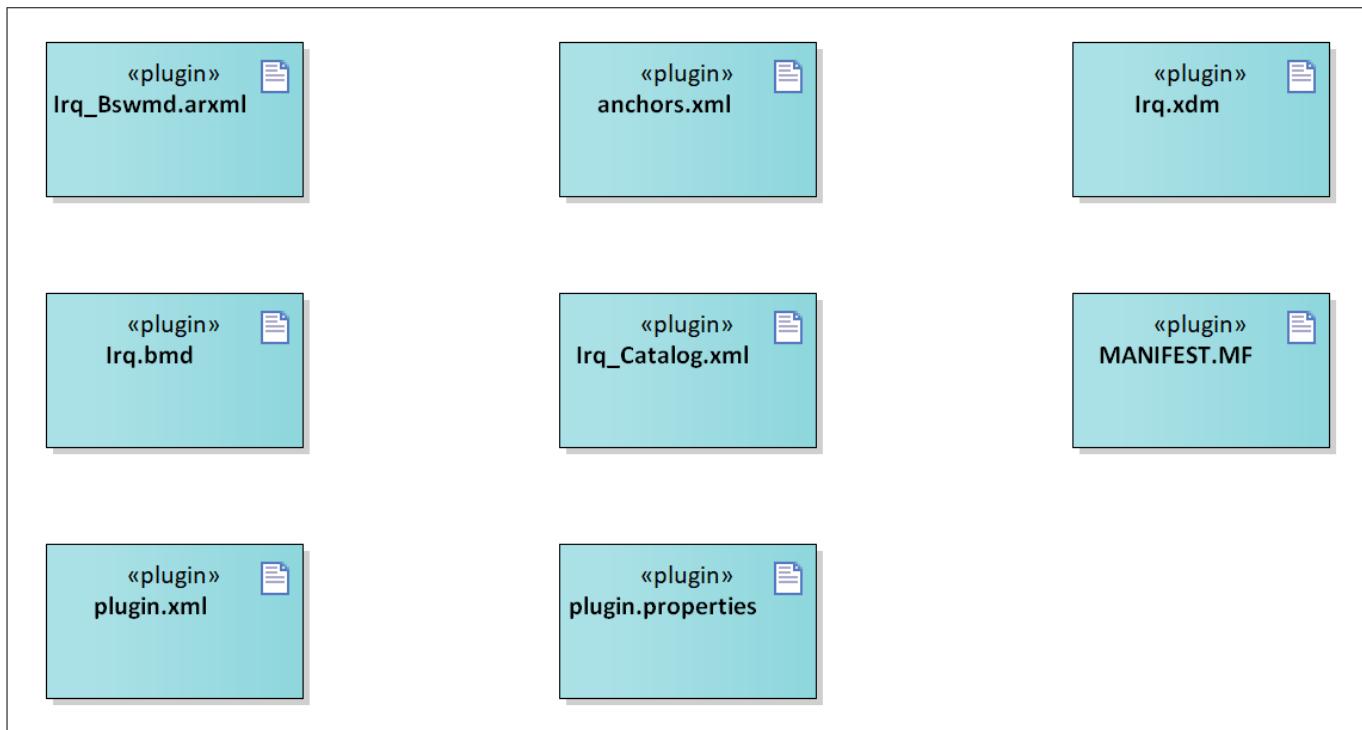
The following diagram explains the file structure of the IRQ driver.

IRQ driver

Figure 134 C file structure
Table 985 C file structure

File name	Description
Irq.h	Header file (static) defining prototypes of data structures and APIs
Irq.c	File (static) containing implementation of APIs
Irq_Cfg.h	Generated header file containing macros and configuration data of interrupt priority and interrupt service providers
Irq_MemMap.h	File (static) containing the memory section definitions used by the IRQ driver
IfxSrc_Reg.h	SRC register definition file
Std_Types.h	Standard data types to be used are declared here
Mcal_Compiler.h	Compiler abstraction of TriCore™ instructions

14.1.3.2 Code generator plugin files

This section provides details on the code generator plugin files of the IRQ driver.

IRQ driver

Figure 135 Code generator plugin files
Table 986 Code generator plugin files

File name	Description
anchors.xml	Tresos anchors support file for the IRQ driver
Irq.xdm	Tresos format XML data model schema file
Irq.bmd	AUTOSAR format XML data model schema file (for each device)
MANIFEST.MF	Tresos plugin support file containing the metadata for the IRQ driver
plugin.properties	Tresos plugin support file for the IRQ driver
plugin.xml	Tresos plugin support file for the IRQ driver
Irq_Bswmd.arxml	AUTOSAR format module description file
Irq_Catalog.xml	AUTOSAR format catalog file

14.1.4 Integration hints

This section lists the key points that an integrator or user of the IRQ driver must consider.

14.1.4.1 Integration with AUTOSAR stack

This section lists the modules that are not part of the MCAL but are required to integrate the IRQ driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM

IRQ driver

module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **DET**

DET module is not required for integrating the IRQ driver.

- **DEM**

DEM module is not required for integrating the IRQ driver.

- **SchM**

SchM is not required for integrating the IRQ driver.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows re-location of text, variables, constants and configuration data to user specific memory regions. In order to achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Irq_MemMap.h` file.

The `Irq_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are relocated to the correct memory region. A sample implementation listing the memorysection macros is depicted below.

```
/* Code Sections */
#if defined IRQ_START_SEC_CODE_QM_GLOBAL
    /* User Pragma here */
    #undef IRQ_START_SEC_CODE_QM_GLOBAL
    #undef MEMMAP_ERROR
#elif defined IRQ_STOP_SEC_CODE_QM_GLOBAL
    /* User Pragma here */
    #undef IRQ_STOP_SEC_CODE_QM_GLOBAL
    #undef MEMMAP_ERROR
#endif
```

Safety error

The IRQ driver does not report any safety error.

Notifications and callbacks

The IRQ driver does not provide any notification or callback functions.

Operating system

The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application.

Operating system files provided by MCAL package is only an example code and must be updated by the integrator with the actual OS files for the desired function.

14.1.4.2 Multicore and Resource Manager

The IRQ driver does not support execution on multiple cores in parallel.

14.1.4.3 MCU support

The IRQ driver is dependent on the MCU driver for the clock service. Therefore, the MCU initialization must be completed prior to IRQ initialization.

IRQ driver

14.1.4.4 PORT support

The IRQ driver does not use any services provided by the PORT driver.

14.1.4.5 DMA support

The IRQ driver does not use any services provided by the DMA driver.

14.1.4.6 Interrupt connections

Defining the interrupt vector table entries and invoking the interrupt handlers must be done by the user. A sample invocation for GTM TOM is depicted as follows:

GTM TOM interrupts definition and invoking:

```
/* SRC_GMTTOM0SR0_ISR */
ISR(GMTTOM0SR0_ISR)
{
    /* Enable global interrupts */
    ENABLE();
    /* Parameter is channel number */
    Mcu_17_Gtm_TomChannelIsr (TOM_MODULE_0 , TOM_CHANNEL_0);
    /* TOM_MODULE_0 = 0, TOM_CHANNEL_0 = 0*/
}
```

14.1.4.7 Example usage

14.1.4.7.1 Configuring the driver

Use the following to configure the module:

- IRQ general
 - Configure IrqOsekEnable to indicate the use of OS. If this parameter is enabled then user is allowed to configure the CAT2 interrupts
 - Configuring IrqOsekEnable is shown in the following figure:



Figure 136 Configuring IrqOsekEnable

- Module interrupt setting
 - Configure Irq<ModuleName>SRN##Cat to indicate the category of the corresponding SRN
 - Configure Irq<ModuleName>SRN##Prio to indicate the priority of the corresponding SRN
 - Configure Irq<ModuleName>SRN##Tos to indicate the category of the corresponding SRN
 - Example for configuring module interrupt is shown in the following figure:

IRQ driver

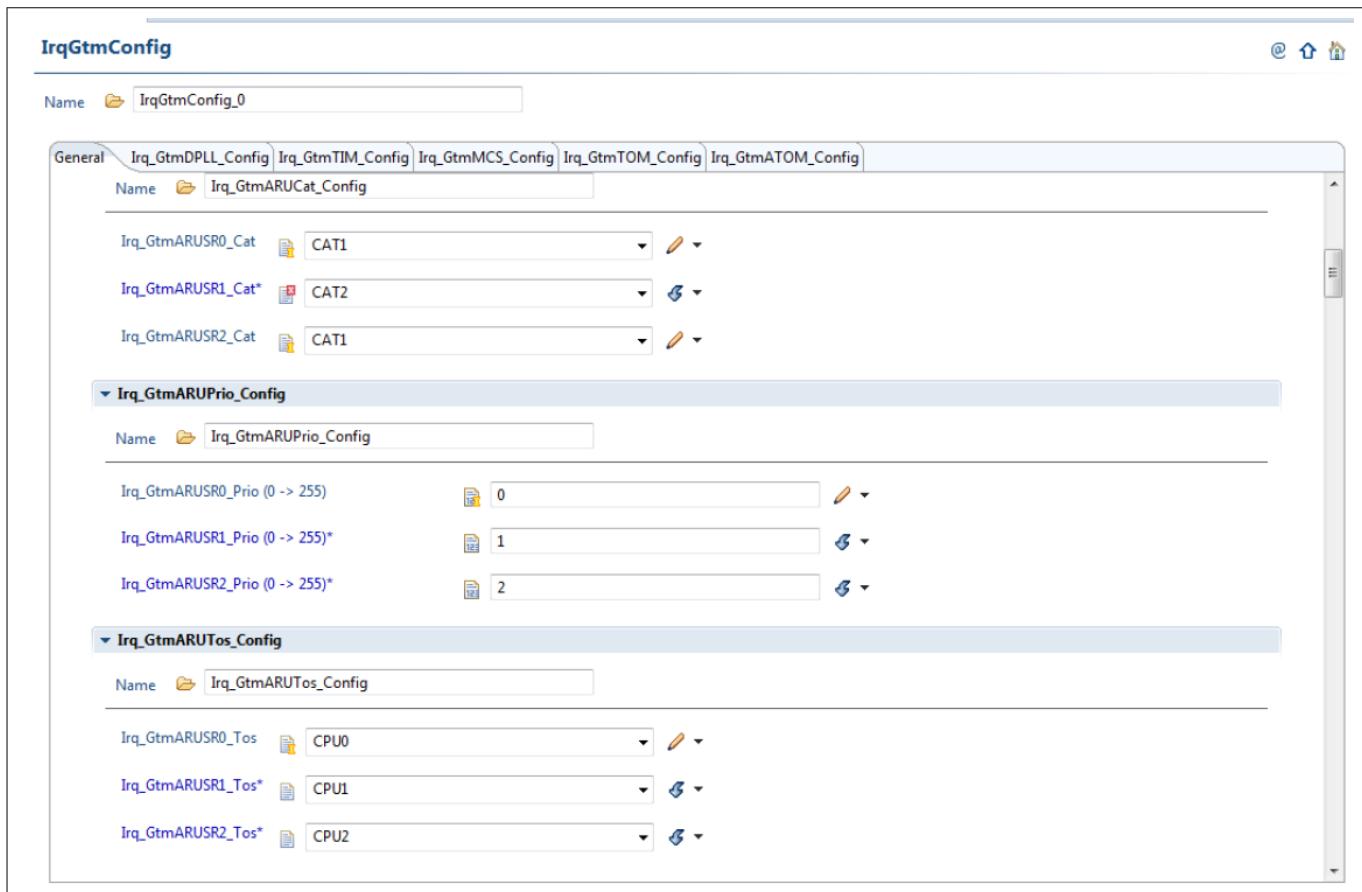


Figure 137 Interrupt configuration

14.1.4.7.2 Initializing interrupts

The module-specific service request control (SRC) registers are initialized with type of service and priority by <ModuleName>_Init functions. Additionally, user should enable module interrupts in interrupt configuration register.

For example, GTM initialization is depicted as follows:

```
/* Initialize Module Interrupt Priority and Service Providers */
IrqGtm_Init();
/* Enable module interrupts using SRE Bit*/
SRC_GMTMTO00.B.SRE = 1U;
```

14.1.4.7.3 Clearing interrupts

The `Irq_ClearAllInterruptFlag()` API clears the SRC registers of all modules.

```
/* Clear Interrupt Flags */
Irq_ClearAllInterruptFlags();
```

IRQ driver

14.1.5 Key architectural considerations

None.

14.2 Assumptions of Use (AoUs)

The operating system should take care of handling the CAT2 interrupts.

14.3 Reference information

14.3.1 Configuration interfaces

The following diagram depicts the hierarchy along with the extensions provided for IRQ module.

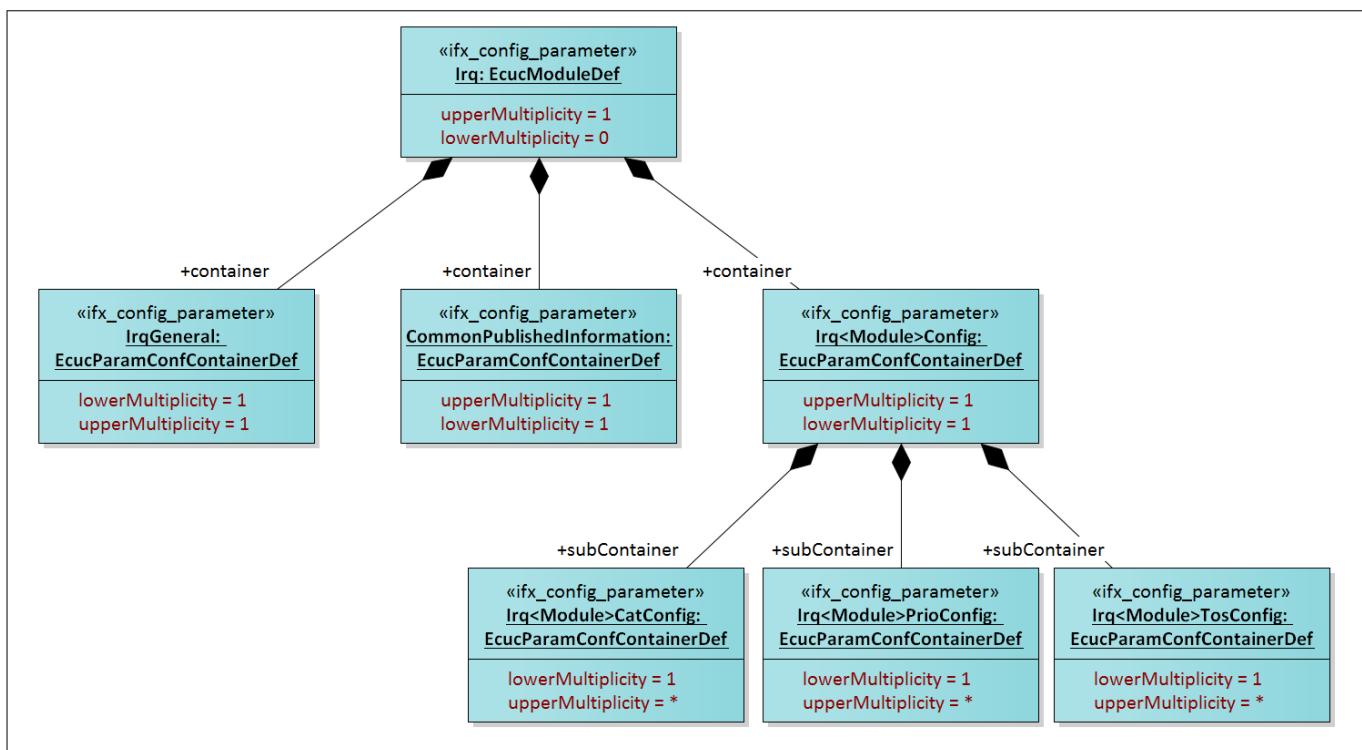


Figure 138 IRQ module configuration

14.3.1.1 Container: IrqGeneral

This container includes general configuration parameters of the IRQ driver.

14.3.1.1.1 IrqOSekEnable

Table 987 Specification for IrqOSekEnable

Name	IrqOSekEnable		
Description	Parameter to identify the use of OS. This controls the available configuration options for interrupt category.		
Multiplicity	1..1	Type	EcucBooleanParamDef

IRQ driver**Table 987 Specification for IrqOSekEnable (continued)**

Range	TRUE: OSEK OS is used. This allows the user to configure the interrupt category as CAT1 or CAT2 FALSE: OSEK OS is not used. All interrupt must be of category CAT1.		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.1.2 Container: Irq<ModuleName>Config

This container includes module-specific interrupt configuration parameters. Where <ModuleName> stand for Ccu6, Gpt, Can, Adc, Gtm, GpsrGroup, Flexray, Spi, Asclin, Ethernet, Dma, Scu, Dmu, Hssl, Sent, Stm, I²C and Dsadc.

The name of the container Irq<ModuleName>Config is affecting the macro generation. Hence, the container name should not be modified.

Note: The list of available modules is provided in the Release Notes.

14.3.1.2.1 Container: Irq<ModuleName>CatConfig

This container includes the parameter to configure the category of all the SRN supported by the module.

The name of the container Irq<ModuleName>CatConfig is affecting the macro generation. Hence, the container name should not be modified.

Irq<ModuleName><SrName>Cat**Table 988 Specification for Irq<ModuleName><SrName>Cat**

Name	Irq<ModuleName><SrName>Cat		
Description	The interrupt category setting of the corresponding SRN of the module. The SRN number indicates the SRN configured. Depending upon the category the interrupt frame is different. <SrName> stands for name of the service request node.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CAT1 : Category 1 interrupt is selected CAT2 : Category 2 interrupt is selected		
Default value	CAT 1		
Post-build variant value	FALSE	Post-build variant multiplicity	-

IRQ driver**Table 988 Specification for **Irq<ModuleName><SrName>Cat** (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	Only the option CAT1 is allowed to select if the IrqOsekEnable parameter is set to false.		

14.3.1.2.2 Container: **Irq<ModuleName>PrioConfig**

This container includes the parameter to configure the priority of all the SRN supported by the module.

The name of the container **Irq<ModuleName>PrioConfig** is affecting the macro generation. Hence, the container name should not be modified.

Irq<ModuleName><SrName>Prio**Table 989 Specification for **Irq<ModuleName><SrName>Prio****

Name	Irq<ModuleName><SrName>Prio		
Description	The interrupt priority setting of the corresponding SRN of the module. Each SRN should have different priority number.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 to 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.1.2.3 Container: **Irq<ModuleName>TosConfig**

This container includes the parameter to configure the type of service of all the SRN supported by the module.

The name of the container **Irq<ModuleName>TosConfig** is affecting the macro generation. Hence, the container name should not be modified.

Irq<ModuleName><SrName>Tos**Table 990 Specification for **Irq<ModuleName><SrName>Tos****

Name	Irq<ModuleName><SrName>Tos		
Description	The type of service setting of the corresponding SRN of the module. This defines the interrupt control unit, which service the ISR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

IRQ driver**Table 990 Specification for **Irq<ModuleName><SrName>Tos** (continued)**

Range	CPUx: where x depends on derivate DMA		
Default value	CPU0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.1.3 Container: CommonPublishedInformation

This container includes published information about vendor and versions.

14.3.1.3.1 ArMajorVersion**Table 991 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	This parameter specifies AUTOSAR major release version.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 to 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.1.3.2 ArMinorVersion**Table 992 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	This parameter specifies AUTOSAR minor release version.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 to 255		
Default value	2		

IRQ driver**Table 992 Specification for ArMinorVersion (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.1.3.3 ArPatchVersion**Table 993 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	This parameter specifies AUTOSAR patch release version.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 to 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.1.3.4 SwMajorVersion**Table 994 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	This parameter specifies software major release version.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 to 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

IRQ driver**14.3.1.3.5 SwMinorVersion****Table 995 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	This parameter specifies software minor release version.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 to 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.1.3.6 SwPatchVersion**Table 996 Specification for SwPatchVersion**

Name	ArPatchVersion		
Description	This parameter specifies software patch release version.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 to 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.1.3.7 ModuleId**Table 997 Specification for ModuleId**

Name	ModuleId		
Description	This parameter specifies module identification number.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 to 65535		
Default value	255		

IRQ driver**Table 997 Specification for ModuleId (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.1.3.8 VendorId**Table 998 Specification for VendorId**

Name	VendorId		
Description	This parameter specifies vendor identification number.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 to 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

14.3.2 Functions – Type definitions

No datatype are defined in IRQ.

14.3.3 Functions - APIs

This section lists the APIs provided along with a short description of the functionality.

14.3.3.1 Interrupt init functions**Table 999 Specification for Irq<ModuleName>_Init API**

Syntax	void Irq<ModuleName>_Init(void)
Service ID	None
Sync/Async	Synchronous
Reentrancy	Non-reentrant
Parameters (in)	None
Parameters (out)	None

IRQ driver
Table 999 Specification for `Irq<ModuleName>_Init API (continued)`

Parameters (in-out)	None
Return	None
Description	Initializes the priority and type of service of SRNs for modules.
Source	IFX
Error handling	-
Configuration dependencies	-

14.3.3.2 `Irq_ClearAllInterruptFlags`

Table 1000 Specification for `Irq_ClearAllInterruptFlags API`

Syntax	<code>void Irq_ClearAllInterruptFlags(void)</code>
Service ID	None
Sync/Async	Synchronous
Reentrancy	Non-reentrant
Parameters (in)	None
Parameters (out)	None
Parameters (in-out)	None
Return	None
Description	The service clears the service request (SRR), interrupt trigger overflow clear bit (IOVCLR) and software sticky clear bit (WSCLR) flags for available modules SRN.
Source	IFX
Error handling	-
Configuration dependencies	-

14.3.4 Notifications and callbacks

No notifications or callback functions are used for IRQ.

14.3.5 Scheduled functions

No scheduled functions are used for IRQ.

14.3.6 Interrupt service routines

No scheduled functions are used for IRQ.

14.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

IRQ driver**14.3.7.1 Production errors**

No production errors are reported for IRQ.

14.3.7.2 Development errors

No development errors are reported for IRQ.

14.3.7.3 Runtime errors

No runtime errors are reported for IRQ.

14.3.8 Deviations and limitations**14.3.8.1 Deviations**

No deviations are reported from the requirement.

14.3.8.2 Limitations

The IRQ driver does not support SRNs with different service providers to have the same priorities.

14.3.9 Unsupported hardware features

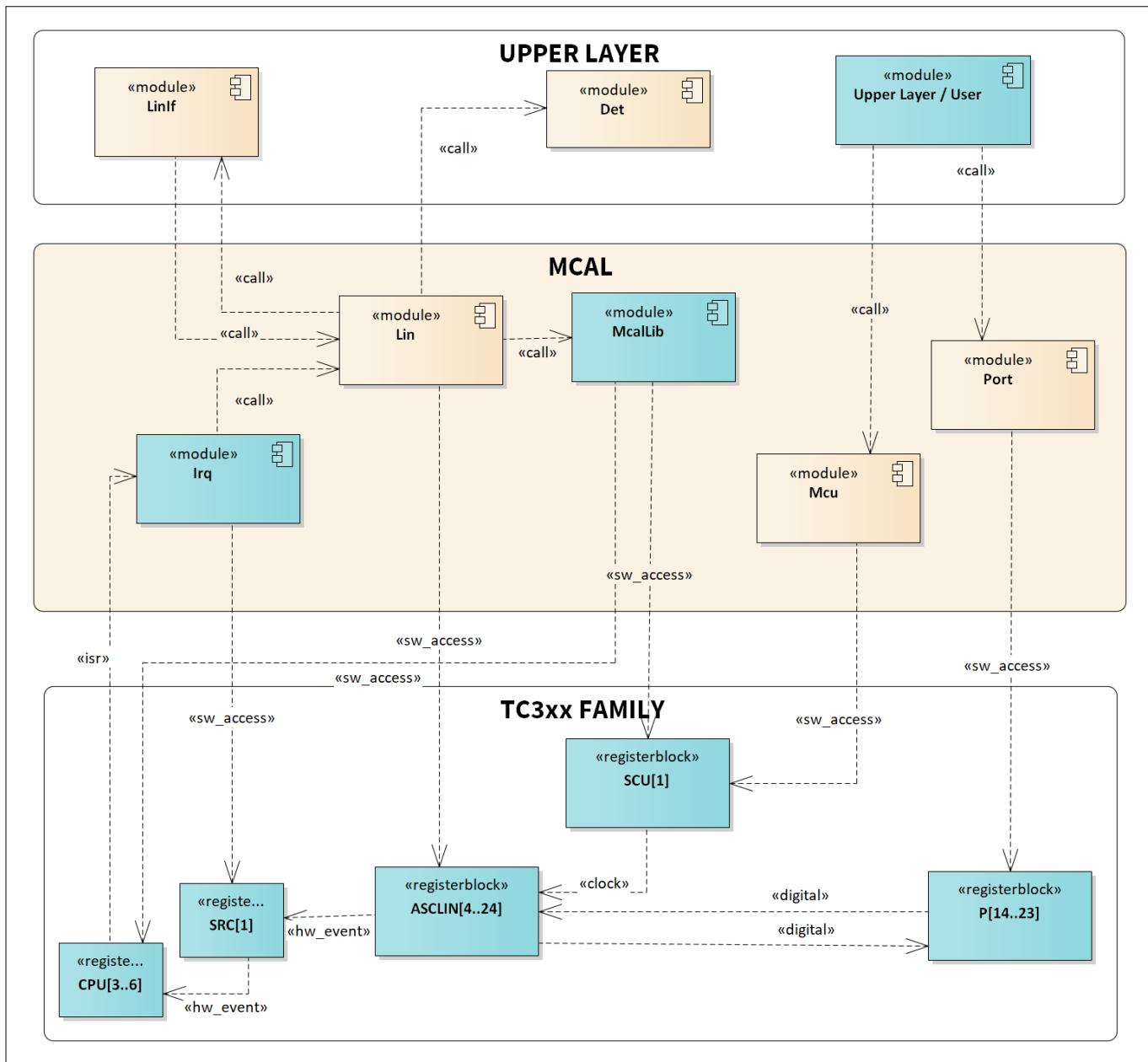
None.

Lin_17_AscLin driver**15 Lin_17_AscLin driver****15.1 User information****15.1.1 Description**

The LIN driver is responsible for providing communication services conforming to LIN 2.1 protocol as specified by AUTOSAR. The ASCLIN module provides hardware support for the LIN protocol. The LIN driver provides UI options to configure the driver parameters described in the AUTOSAR LIN specification (Version AS 4.2.2) and additional parameters to configure the various functional blocks of ASCLIN. The LIN driver supports LIN channels in master mode only. Operating in the slave mode is out of the scope. The LIN driver is implemented as Post-Build variant as specified by AUTOSAR.

15.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

Lin_17_AscLin driver

Figure 139 Mapping of hardware-software interfaces

15.1.2.1 ASCLIN: primary hardware peripheral

Hardware functional features

The LIN driver uses the ASCLIN for LIN communication in the AURIX2G MCAL implementation. The key hardware functional features used by the driver are:

- LIN support features (for Master mode only)
- LSB first
- One stop bit
- 1 K baud to 20 K baud rate signal generation
- Tx and Rx hardware FIFO buffers
- Sync break field generation
- LIN watchdogs (header timeout, response timeout) are used

Lin_17_AscLin driver

- Collision detection feature
- Wake detection and generation
- Interrupts and error handlers flags

The unsupported features of the ASCLIN are:

- LIN slave mode is not used as AUTOSAR requires only the master mode
- Auto-baud detection based on Sync Field measurement
- Break detection
- Struck at zero/one monitoring
- Bus idle time monitoring

Users of the hardware

The LIN and UART drivers utilize the ASCLIN IP. The allocation of ASCLIN channels to LIN/UART driver is done by the MCU driver. Both LIN and UART drivers utilize only the channels allocated to them.

Hardware diagnostic features

The SMU alarms configured for the ASCLIN are not monitored by the LIN driver.

Hardware events

The LIN driver uses the following hardware events from the ASCLIN:

- Tx flag/interrupt upon transmission completed
- Rx flag/interrupt upon reception completed
- Ex flag/interrupt upon occurrences of errors (transmit or receive error)

15.1.2.2 Port: dependent hardware peripheral

Hardware functional features

The ATX and ARX signals are routed to the ASCLIN through the digital port pad. These are configured and enabled through the PORT driver.

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from port pads are not used by the LIN driver.

15.1.2.3 SCU: dependent hardware peripheral

Hardware functional features

The LIN driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB and fASCLIN clock signals for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the LIN driver.

Hardware events

Hardware events from the SCU are not used by the LIN driver.

Lin_17_AscLin driver**15.1.2.4 SRC: dependent hardware peripheral****Hardware functional features**

The LIN driver depends on the interrupt router for raising an interrupt to the CPU based on the transmit/receive/error events, which indicates successful frame transmission and reception respectively.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software.

Hardware diagnostic features

The SMU alarms configured for the interrupt router are not monitored by the LIN driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU. The LIN driver provides interrupt handlers as software interfaces, which must be invoked from the ISR.

15.1.3 File structure**15.1.3.1 C file structure**

This section provides details of the C files of the LIN driver.

Lin_17_AscLin driver

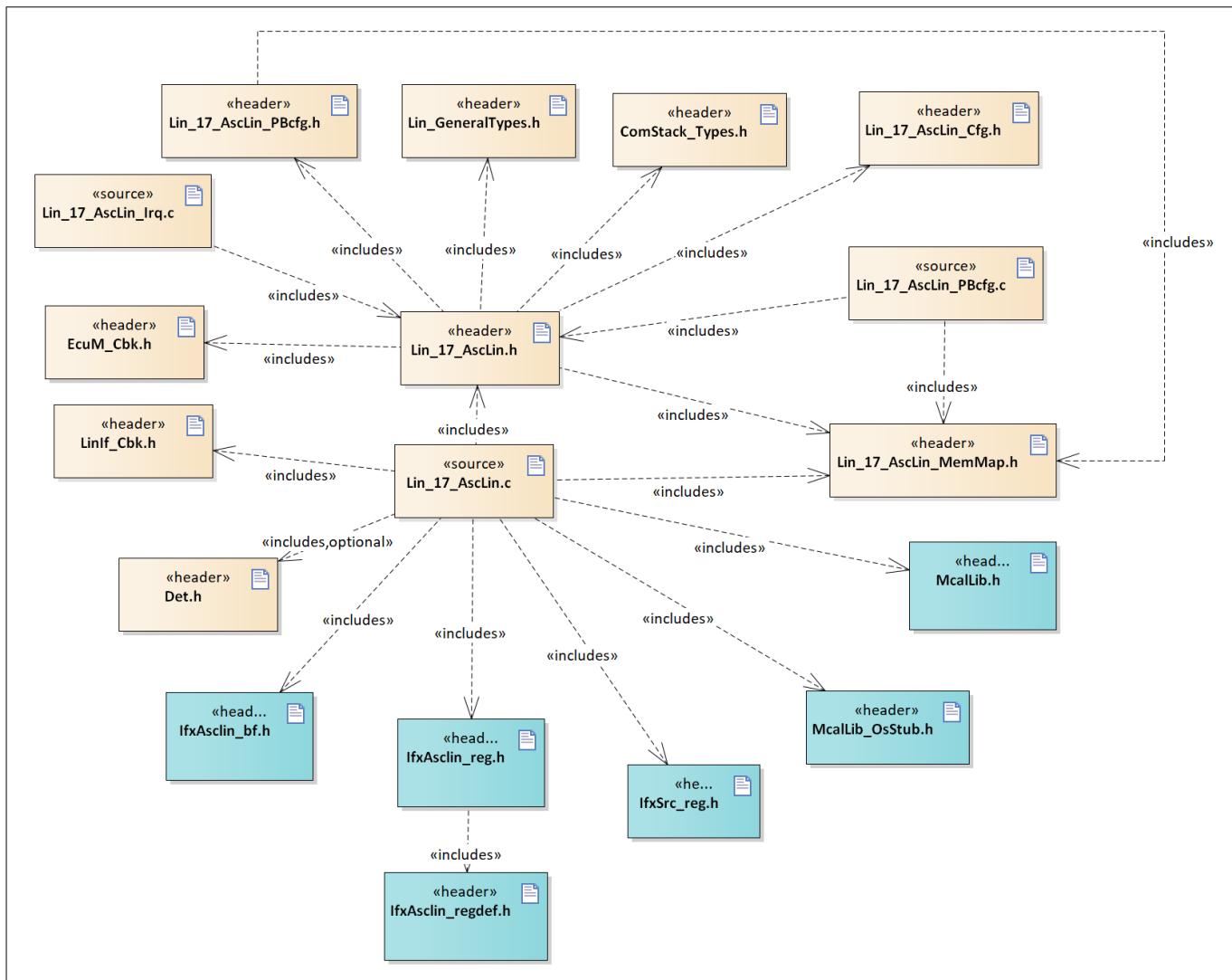


Figure 140 C file structure

Table 1001 C file structure

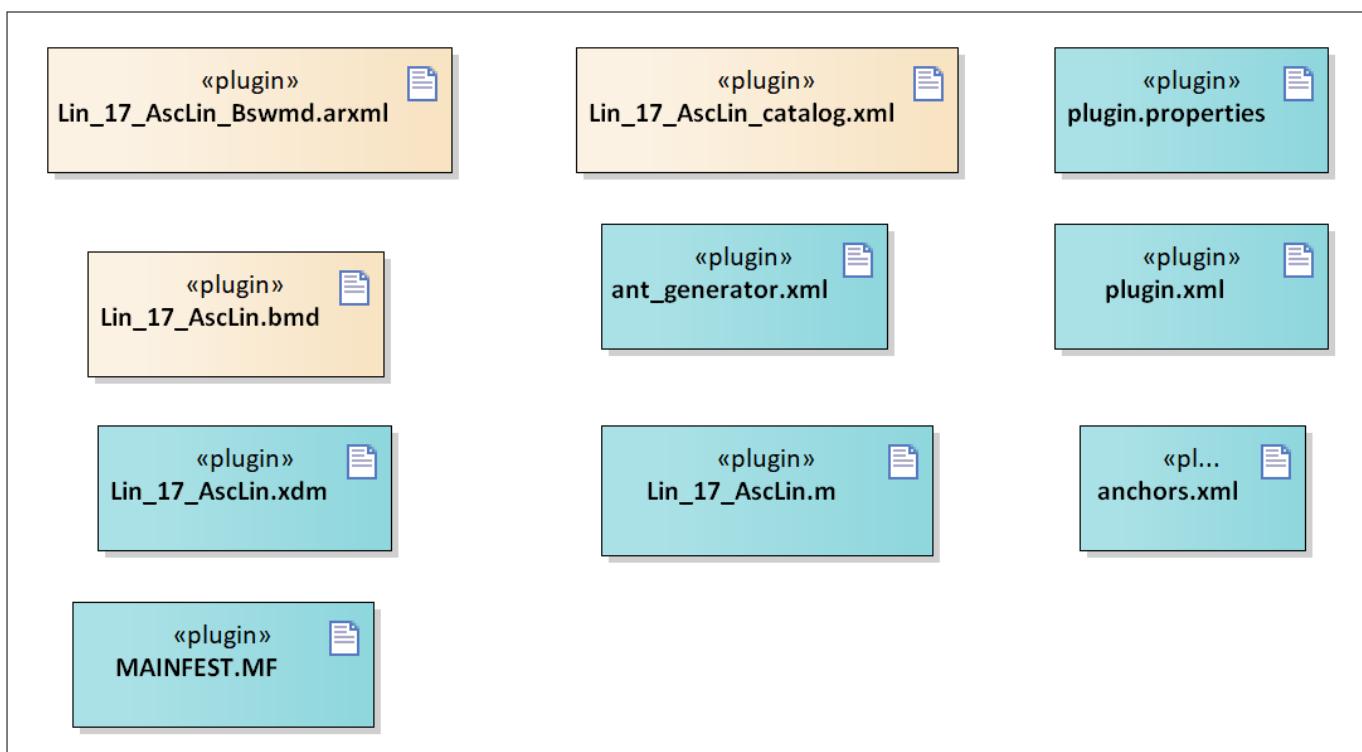
File name	Description
ComStack_Types.h	Type Definition for Com stack
Det.h	Provides the exported interfaces of Development Error Tracer
EcuM_Cbk.h	Header file containing declarations of the EcuM callbacks
IfxAsclin_bf.h	SFR header file for ASCLIN
IfxAsclin_reg.h	SFR header file for ASCLIN
IfxAsclin_Regdef.h	SFR header file for ASCLIN
IfxSrc_Reg.h	SFR header file for Interrupt Controller
LinIf_Cbk.h	The header file contains the function declarations for the callback functions in the LIN interface
Lin_17_AscLin.c	File (Static) containing implementation of the APIs

Lin_17_AscLin driver
Table 1001 C file structure (continued)

File name	Description
Lin_17_AscLin.h	Header file (Static) defining prototypes of data structures, the APIs and Interrupt handlers
Lin_17_AscLin_Cfg.h	Header file (Generated) containing constants and pre-processor macros as #defines
Lin_17_AscLin_Irq.c	Interrupt handler file for LIN
Lin_17_AscLin_MemMap.h	File (Static) containing the memory section definitions used by the LIN driver
Lin_17_AscLin_PBcfg.c	File (Generated) containing definition of the configuration data structures
Lin_17_AscLin_PBcfg.h	File (Generated) containing declaration of the post-build configuration data structures
Lin_GeneralTypes.h	The header file includes general LIN type declarations
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore™. This shall be included by other drivers to call OS APIs.

15.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the LIN driver.


Figure 141 Code generator plugin files

Lin_17_AscLin driver

Table 1002 Code generator plugin files

File name	Description
Lin_17_AscLin.bmd	AUTOSAR format XML data model schema file
Lin_17_AscLin.m	Code template macro file for LIN driver
Lin_17_AscLin.xdm	Tresos format XML data model schema file
Lin_17_AscLin_Bswmd.arxml	AUTOSAR format module description file
Lin_17_AscLin_catalog.xml	AUTOSAR format catalog file
MAINFEST.MF	Tresos plugin support file containing the metadata for the LIN driver
anchors.xml	Tresos anchors support file for the LIN driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configurations when using variation point
plugin.properties	Tresos plugin support file for the LIN driver
plugin.xml	Tresos plugin support file for the LIN driver

15.1.4 Integration hints

This section lists the key points that an integrator or user of the LIN driver must consider.

15.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the LIN driver

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of the MCAL, the EcuM is used for initialization and de-initialization of the software drivers. To start data communication on LIN bus, the EcuM module has to initialize other related AUTOSAR BSW modules such as LIN Interface (LinIf). The LIN module notifies about the wakeup event to the EcuM module. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **LIN interface (LinIf)**

The LIN interface module is a part of the AUTOSAR stack that provides upper layer a hardware independent interface to the LIN communication system. The LIN driver uses the callback function of LinIf to provide wakeup confirmation and indicate successful detection of wakeup signal. The LIN Interface module provided in the MCAL package is a stub code and needs to be replaced with a complete LIN Interface module during the integration phase.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Lin_17_AscLin_MemMap.h` file.

The `Lin_17_AscLin_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the

Lin_17_AscLin driver

elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

```

/*To be used for all global or static variables.*/
#if defined LIN_17_ASCLIN_START_SEC_VAR_CLEARED_QM_LOCAL_32
/* User Pragma here */
#define LIN_17_ASCLIN_START_SEC_VAR_CLEARED_QM_LOCAL_32
#define MEMMAP_ERROR
#elif defined LIN_17_ASCLIN_STOP_SEC_VAR_CLEARED_QM_LOCAL_32
/* User Pragma here */
#define LIN_17_ASCLIN_STOP_SEC_VAR_CLEARED_QM_LOCAL_32
#define MEMMAP_ERROR

#define LIN_17_ASCLIN_START_SEC_VAR_CLEARED_QM_LOCAL_8
/* User Pragma here */
#define LIN_17_ASCLIN_START_SEC_VAR_CLEARED_QM_LOCAL_8
#define MEMMAP_ERROR
#elif defined LIN_17_ASCLIN_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
/* User Pragma here */
#define LIN_17_ASCLIN_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
#define MEMMAP_ERROR

#define LIN_17_ASCLIN_START_SEC_VAR_CLEARED_QM_LOCAL_UNSPECIFIED
/* User Pragma here */
#define LIN_17_ASCLIN_START_SEC_VAR_CLEARED_QM_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined LIN_17_ASCLIN_STOP_SEC_VAR_CLEARED_QM_LOCAL_UNSPECIFIED
/* User Pragma here */
#define LIN_17_ASCLIN_STOP_SEC_VAR_CLEARED_QM_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR

/*To be used for global or static constants.*/
#elif defined LIN_17_ASCLIN_START_SEC_CONST_QM_LOCAL_32
/* User Pragma here */
#define LIN_17_ASCLIN_START_SEC_CONST_QM_LOCAL_32
#define MEMMAP_ERROR
#elif defined LIN_17_ASCLIN_STOP_SEC_CONST_QM_LOCAL_32
/* User Pragma here */
#define LIN_17_ASCLIN_STOP_SEC_CONST_QM_LOCAL_32
#define MEMMAP_ERROR

/* LIN module configuration data */
#elif defined LIN_17_ASCLIN_START_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
/* User Pragma here */
#define LIN_17_ASCLIN_START_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined LIN_17_ASCLIN_STOP_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
/* User Pragma here */
#define LIN_17_ASCLIN_STOP_SEC_CONFIG_DATA_QM_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR

/* Code section */
#elif defined LIN_17_ASCLIN_START_SEC_CODE_QM_LOCAL

```

Lin_17_AscLin driver

```

/* User Pragma here */
#define LIN_17_ASCLIN_START_SEC_CODE_QM_LOCAL
#define MEMMAP_ERROR
#elif defined LIN_17_ASCLIN_STOP_SEC_CODE_QM_LOCAL
/* User Pragma here */
#define LIN_17_ASCLIN_STOP_SEC_CODE_QM_LOCAL
#define MEMMAP_ERROR
#endif

#if defined MEMMAP_ERROR
#error "LIN_17_ASCLIN__MemMap.h, wrong pragma command"
#endif

```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The LIN driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the LIN driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The files `Det.h` and `Det.c` are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is not required for integrating the LIN driver.

- **SchM**

The SchM is not required for the integration of LIN driver.

- **Safety error**

The LIN driver does not report any safety errors.

- **Notifications and callbacks**

The LIN driver does not implement any notifications. However, the LIN driver reports wakeup confirmation through callback functions of the LinIf module.

- **Operating system (OS)**

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application.

The OS files provided by MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

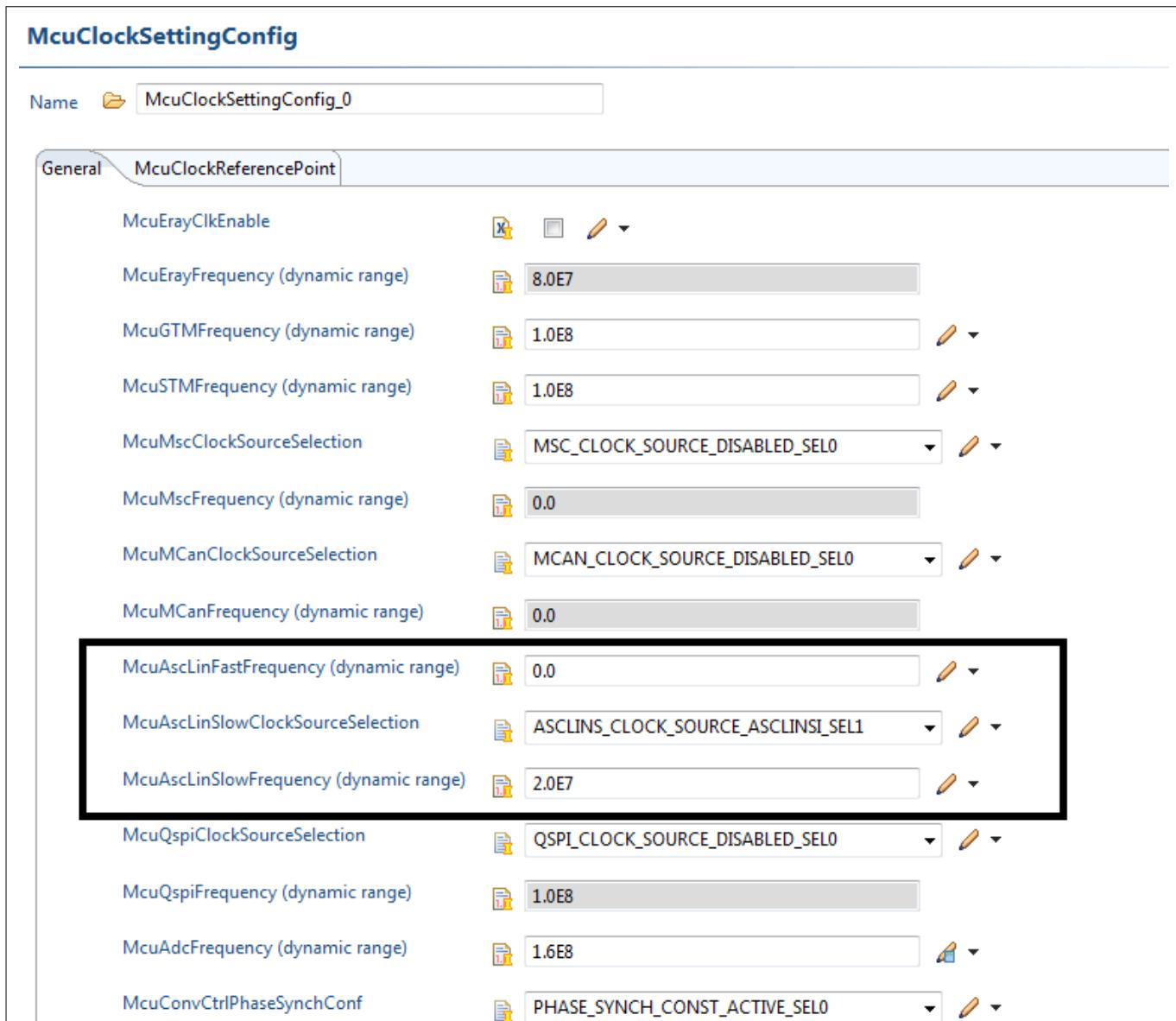
15.1.4.2 Multicore and Resource Manager

The LIN driver does not support execution on multiple cores simultaneously.

15.1.4.3 MCU support

The LIN driver is dependent on the MCU driver for clock configuration and channel allocation services. The initialization of the LIN driver must be started only after completing the MCU initialization. The following must be considered while configuring the MCU driver in the EB tresos tool:

The `fASCLINF` or `fASCLINS` defines the clock frequency for the ASCLIN kernel. To configure clock frequency for `fASCLINF` or `fASCLINS` refer to the `McuAsclinFastFrequency` and `McuAsclinSlowFrequency` parameters from the MCU driver configuration as follows:

Lin_17_AscLin driver

Figure 142 **LIN fast or slow clock frequency configuration**

Note: `LinCsrClksel` configuration parameter in the LIN driver configuration selects the frequency to be used.

The ASCLIN hardware IP is shared between the LIN and the UART drivers. So the LIN driver also depends on MCU driver for allocation of the ASCLIN kernels to the driver. The resource allocation for the ASCLIN kernels configured from the MCU driver using `McuHardwareResourceAllocationConf` container. Following is the example of allocation of ASCLIN1 kernel to the LIN driver. Index ID for the node represents the ASCLIN kernel ID.

Lin_17_AscLin driver

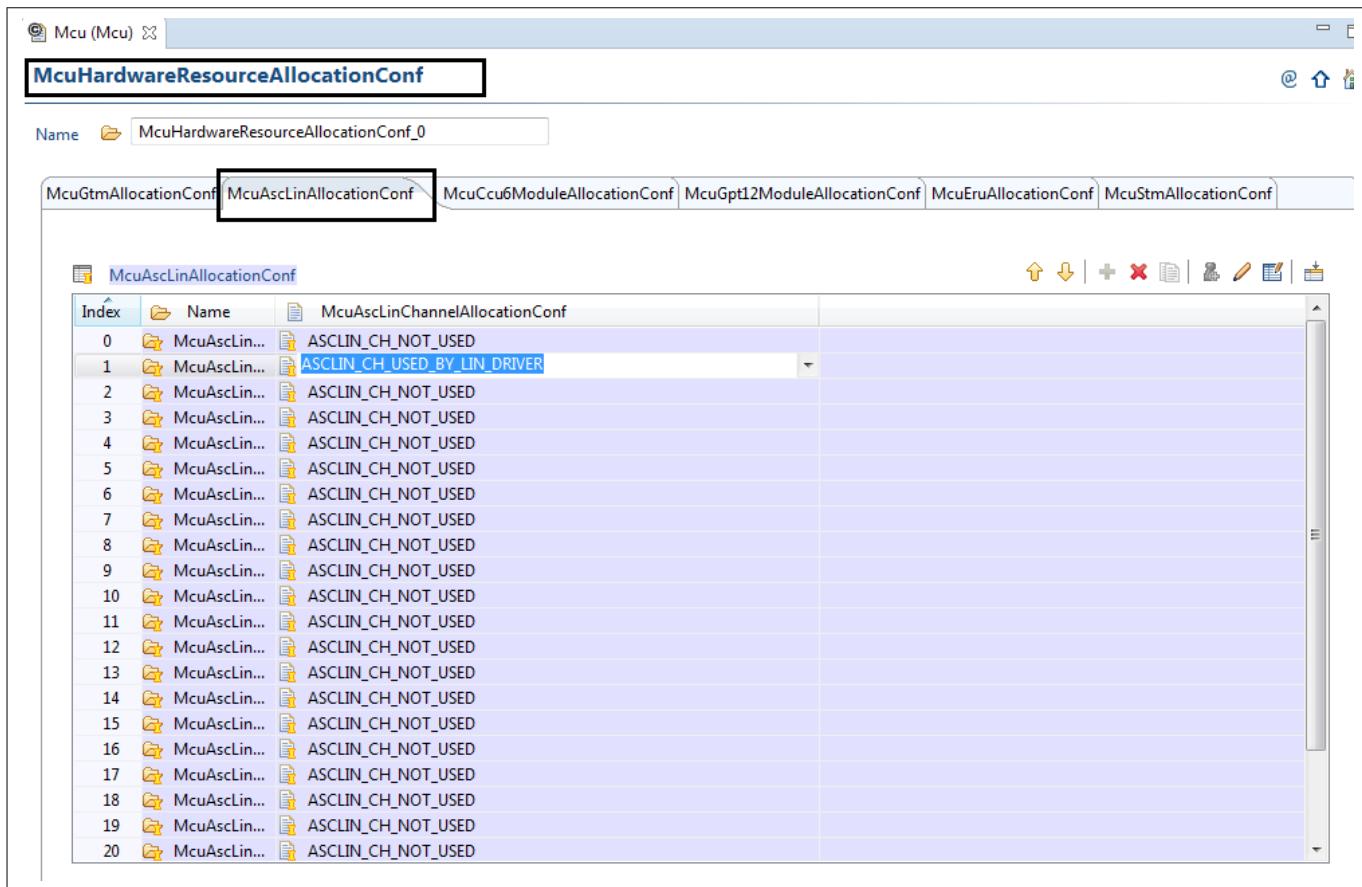


Figure 143 LIN resource allocation using MCU

The MCU initialization must be completed prior to invoking the LIN initialization.

15.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the LIN driver through the PORT configuration and initialize the port pins prior to invoking the LIN initialization. The following must be considered while configuring PORT driver in the EB tresos tool:

- The TxD and RxD pins of the different ASCLIN kernels must be configured with respective direction and configuration in the PORT module. For RxD pin the selection for suitable port and pin shall be made in LIN driver configuration using `LinRxAlternateInputSignal` configuration parameter for a ASCLIN kernel. Refer to the following sample configuration for the PORT driver.

Lin_17_AscLin driver

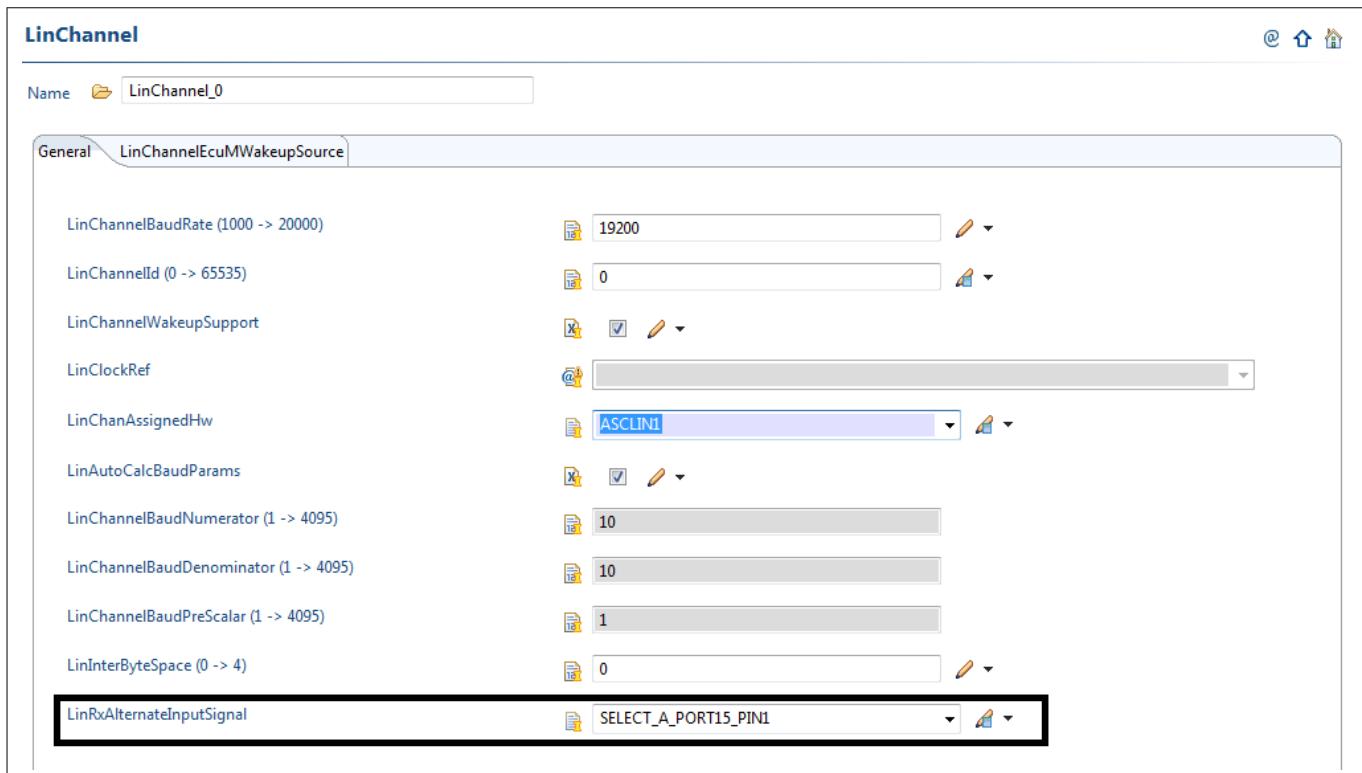


Figure 144 RX input pin selection from LIN configuration for a LIN channel

The screenshot shows the 'PortContainer' configuration window. The 'Name' field is set to 'PortContainer_8'. The 'PortPin' tab is selected, displaying a table of port pins:

Index	Name	PortPinId	PortPinSymbolicName	PortPinDirection	PortPinDi...	PortPinIn...	PortPinL...
0	PortPin_3	243	PORT_15_PIN_3	PORT_PIN_OUT	X	GPIO	PORT_PIN
1	PortPin_0	240	PORT_15_PIN_0	PORT_PIN_OUT	X	ALT2	PORT_PIN
2	PortPin_1	241	PORT_15_PIN_1	PORT_PIN_IN	X	GPIO	PORT_PIN
3	PortPin_10	251	PORT_15_PIN_11	PORT_PIN_IN	X	GPIO	PORT_PIN
4	PortPin_11	252	PORT_15_PIN_12	PORT_PIN_IN	X	GPIO	PORT_PIN
5	PortPin_12	253	PORT_15_PIN_13	PORT_PIN_IN	X	GPIO	PORT_PIN
6	PortPin_13	254	PORT_15_PIN_14	PORT_PIN_IN	X	GPIO	PORT_PIN
7	PortPin_14	255	PORT_15_PIN_15	PORT_PIN_IN	X	GPIO	PORT_PIN
8	PortPin_2	242	PORT_15_PIN_2	PORT_PIN_OUT	X	ALT2	PORT_PIN
9	PortPin_4	244	PORT_15_PIN_4	PORT_PIN_IN	X	GPIO	PORT_PIN
10	PortPin_5	245	PORT_15_PIN_5	PORT_PIN_IN	X	GPIO	PORT_PIN
11	PortPin_6	246	PORT_15_PIN_6	PORT_PIN_IN	X	GPIO	PORT_PIN
12	PortPin_7	247	PORT_15_PIN_7	PORT_PIN_IN	X	GPIO	PORT_PIN
13	PortPin_8	248	PORT_15_PIN_8	PORT_PIN_IN	X	GPIO	PORT_PIN
14	PortPin_9	250	PORT_15_PIN_10	PORT_PIN_IN	X	GPIO	PORT_PIN

Figure 145 Port pin configuration from PORT module

Lin_17_AscLin driver

15.1.4.5 DMA support

The LIN driver does not use any services provided by the DMA driver.

15.1.4.6 Interrupt connections

The following interrupts in LIN are to be configured:

- Tx - Transmit complete interrupt

This interrupt triggers when the header transmission or the master response transmission is completed successfully. A sample invocation for Tx interrupt handler is as follows:

```
#include "Lin_17_AscLin.h"
#include "Irq.h"

*****TX Interrupt for ASCLINO *****/
IFX_INTERRUPT(ASCLINOTX_ISR, 0, IRQ_ASCLINO_TX_PRIO)
ISR(ASCLINOTX_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    /* Call Lin Interrupt function*/
    Lin_17_AscLin_IsrTransmit(0U);
}
```

- Rx - Receive complete interrupt

This interrupt triggers when slave response reception is completed successfully. A sample invocation for Rx interrupt handler is depicted below:

```
#include "Lin_17_AscLin.h"
#include "Irq.h"

*****RX Interrupt for ASCLINO *****/
IFX_INTERRUPT(ASCLINORX_ISR, 0, IRQ_ASCLINO_RX_PRIO)
ISR(ASCLINORX_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    /* Call Lin Interrupt function*/
    Lin_17_AscLin_IsrReceive(0U);
}
```

- Err - Error event interrupt

Lin_17_AscLin driver

This interrupt triggers when erroneous event occurs like collision error, header error. A sample invocation for Err interrupt handler is depicted below:

```
#include "Lin_17_AscLin.h"
#include "Irq.h"

/*****************Err Interrupt for ASCLINO *****/
IFX_INTERRUPT(ASCLINOERR_ISR, 0, IRQ_ASCLINO_ERR_PRIO)
ISR(ASCLINOERR_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    /* Call Lin Interrupt function*/
    Lin_17_AscLin_IsrError(OU);
}
```

Configuration of interrupt category and priority, shall be configured in IRQ module. Following are interrupt configuration example for ASCLINO.

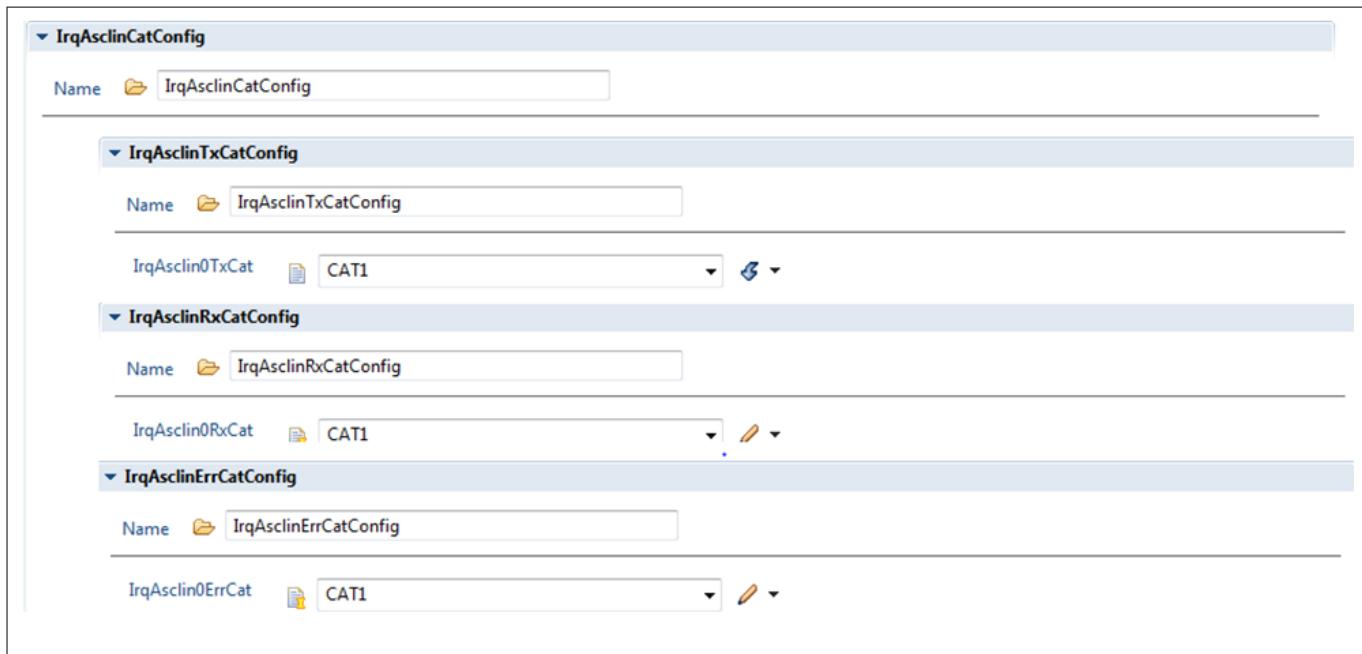
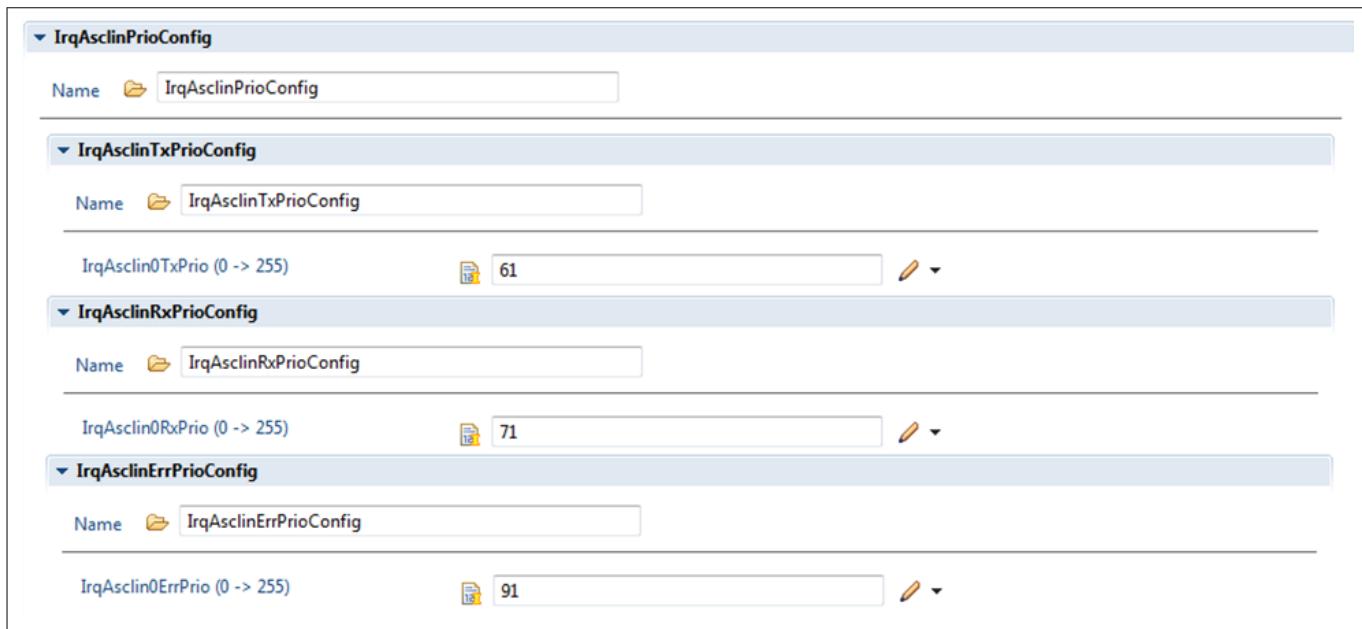


Figure 146 Interrupt category configuration for ASCLINO

Lin_17_AscLin driver**Figure 147 Interrupt priority configuration for ASCLINO**

Lin_17_AscLin driver

15.1.4.7 Example usage

Examples of LIN driver API usage are as follows:

Configuration of the driver

The LIN driver must be configured before usage and configuration files should be generated and made available during the software build process.

To configure the LIN driver, the following guidelines should be followed.

- Configuration of the system clock: Before using the LIN driver, the MCU driver should be configured and initialized so that the system clock is up and running at the required frequency. This configuration is done using the MCU driver.
- Configuration of the port pins: The TXD and RXD (for the relevant Rx pin select) pins of the LIN channels should be configured using the PORT driver.
- Configuration of LIN interrupts: For LIN drivers with interrupt mode enabled, configure the interrupt priority, type of service and interrupt type in the IRQ driver.
- Configuration of the LIN driver: Select the required API configuration and choose channel dependent parameters such as baud-rate, wakeup support, inter byte space and so on.

Initialization of LIN driver

Follow the sequence in the application code:

1. Initialize the MCU driver and clock using the `Mcu_Init()` API.
2. Initialize the PORT driver using the `Port_Init()` API.
3. Initialize the IRQ driver to enable the interrupt generation.
4. Initialize the LIN driver using the `Lin_17_AscLin_Init()` API.

Sample code for LIN driver initialization is as follows:

```
#include "Port.h"
#include "Port_PBcfg.h"
#include "Mcu.h"
#include "Mcu_PBcfg.h"
#include "McalLib.h"
#include "Irq.h"

/* Init MCU */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Init Port */
Port_Init(&Port_Config);

/* Init Irq */
IrqAsclin_Init();

/* Init Driver */
Lin_17_AscLin_Init(&Lin_17_AscLin_Config);
```

Master response communication

Lin_17_AscLin driver

The code sequence sends a header followed by master response data.

```

/* dummy sdu data for master */
uint8 Sdu_Data[][8] =
{
{1,2,3,4,5,6,7,8},
{0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00}
};

/* dummy pdu data */
Lin_PduType Lin_Pdu[] =
{
{0x80, LIN_ENHANCED_CS, LIN_MASTER_RESPONSE, 8, Sdu_Data[0]},
{0xC1, LIN_ENHANCED_CS, LIN_SLAVE_RESPONSE, 8, Sdu_Data[1]},
{0xC2, LIN_ENHANCED_CS, LIN_SLAVE_TO_SLAVE, 8, Sdu_Data[1]}
};

uint8 DataRead[8];
/* dummy Shadow pointer for slave response received data */
volatile uint8 *SlaveSduPtr = DataRead;

void Lin_Master_Response(void)
{
Std_ReturnType Ret1;
Std_ReturnType Ret2;

/* Master response header and frame transmission as referred in the dummy
Lin_Pdu[0]*/
Ret1 = Lin_17_AscLin_SendFrame(LinConf_LinChannel_LinChannel_0, &Lin_Pdu[0]);
/* Ret1 is E_NOT_OK then Error in sending Frame.... */

do
{
Ret2 = Lin_17_AscLin_GetStatus(LinConf_LinChannel_LinChannel_0,
(uint8**) &SlaveSduPtr);

if ((Ret2 != LIN_TX_BUSY) && (Ret2 != LIN_TX_OK) && (Ret2 != OPERATIONAL_STATE))
{
/* Transmit Error.... */
break;
}
}while(Ret2 != LIN_TX_OK );

if (Ret2 == LIN_TX_OK)
{
/* LIN Header (PID 0x80) Transmitted sucessfully.... */
/* Data (0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8) is transmitted. can be verify it at
the slave.... */
}
else
{
/* Lin Master->Slave Response Failed. Response code: Ret2 */
}
}

```

Lin_17_AscLin driver

```

    }
}
```

Slave response communication

The code sequence sends a header followed by slave response received by master.

```

void Lin_Slave_Response(void)
{
    Std_ReturnType Ret1;
    Std_ReturnType Ret2;

    /* Slave response header transmission and reception of the slave response as
    referred in the dummy Lin_Pdu[1]*/
    Ret1 = Lin_17_AscLin_SendFrame(LinConf_LinChannel_LinChannel_0, &Lin_Pdu[1]);
    /* Ret1 is E_NOT_OK then Lin Slave->Master Response Failed.... */

    do
    {
        Ret2 = Lin_17_AscLin_GetStatus(LinConf_LinChannel_LinChannel_0,
        (uint8**)&SlaveSduPtr);

        if ((Ret2 != LIN_TX_BUSY) && (Ret2 != LIN_TX_OK) && (Ret2 != OPERATIONAL_STATE))
        {
            /* Reception Error.... */
            break;
        }
        }while(Ret2 != LIN_RX_OK );

        if (Ret2 == LIN_RX_OK)
        {
            /* LIN Header (PID 0xC1) Transmitted successfully.... */
            /* Data Received by Master.... */
            for (i=0; i < 8 ; i++)
            {
                print_f("%x ",SlaveSduPtr[i]);
            }
        }
        else
        {
            /* Lin Master->Slave Response Failed. Response code: Ret2.... */
        }
    }
```

Global interrupt disabled

The LIN driver provides `LinInterruptDisabled` pre-compile configuration parameter, which allows support for `Lin_17_AscLin_Wakeup()` and `Lin_17_AscLin_GetStatus()` APIs when global interrupts are disabled. If the configuration parameter is true, then status flags will be checked from `Lin_17_AscLin_GetStatus()` API and hence the API need to be polled to keep the driver status updated.

Lin_17_AscLin driver

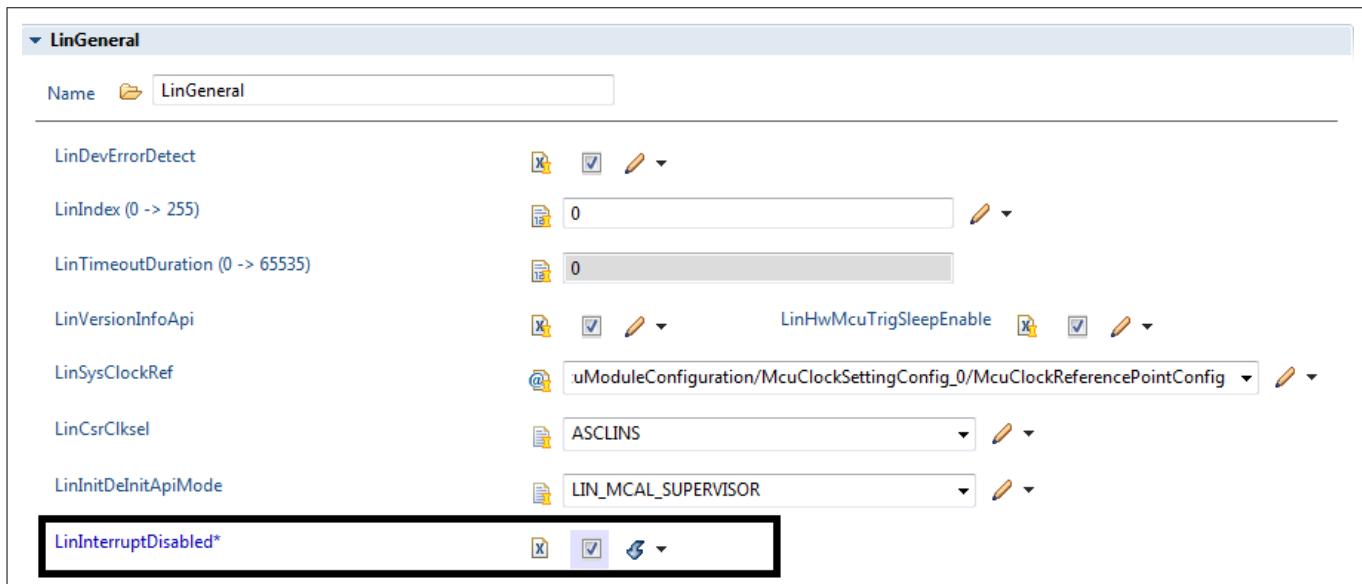


Figure 148 **LinInterruptDisabled configuration parameter**

15.1.5 Key architectural considerations

15.1.5.1 ASCLIN hardware: used for LIN feature

Decision: The LIN driver uses only the LIN features of the ASCLIN hardware.

Rationale: The LIN driver implements LIN protocol 2.1 version as per AUTOSAR. Since ASCLIN hardware supports multiple features like LIN, SPI, ASC, the driver uses only LIN features.

Information: The driver supports only master mode as per AUTOSAR specification

The Master mode has following types of communication:

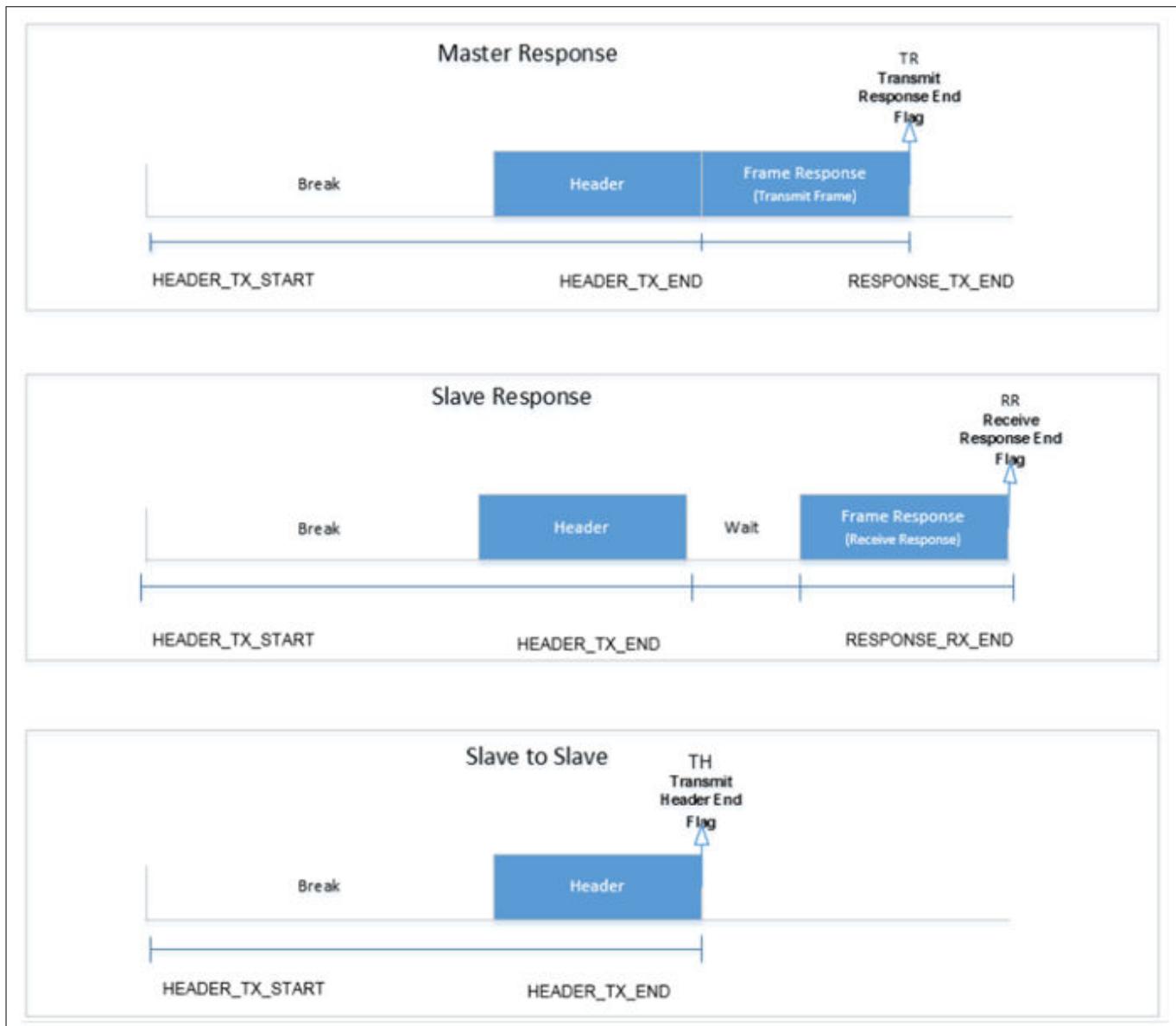
- Master Response - After the header transmission response frame is transmitted, the master node publishes the data frame.
- Slave Response - One of the slave node as defined in the LDF (LIN Description File) publishes the response. Data frame is subscribed by the master node.
- Slave to Slave communication - Master node transmits a header and ignores the response in reply of the header as it is not one of the subscribers.

15.1.5.2 Modes of operation - TxFIFO and RxFIFO modes

Decision: Tx FIFO and Rx FIFO operate in the combined mode.

Rationale: LIN protocol data vary in length (maximum 8 bytes), and data length of the LIN frame to be transferred. In order to raise interrupt at the end of each frame, the mode is used to define required threshold level for each frame transfer.

Information: Three interrupt generation modes are provided by ASCLIN TxFIFO, single move mode, batch move mode and combined move mode with 8 bit data width. For the protocol combined mode is used with frame length . As depicted in the following diagram interrupt can be generated according to LIN frame type.

Lin_17_AscLin driver**Figure 149** LIN interrupt generation for different type of communication**15.1.5.3 Addition of LinInterruptDisabled configuration parameter**

Decision: LinInterruptDisabled configuration parameter is added in the LinGeneral container.

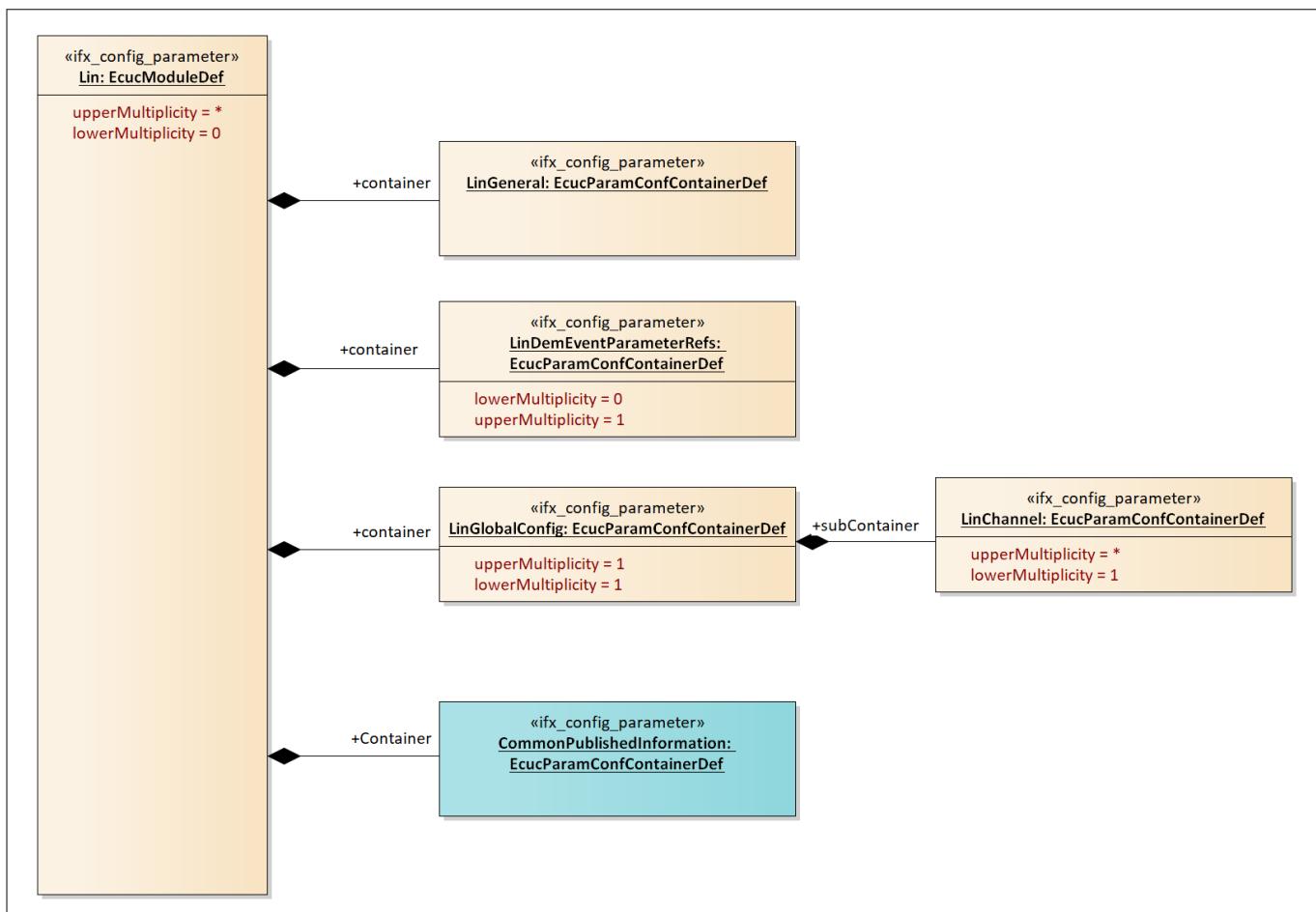
Rationale: As the LIN driver need to support GetStatus and Wakeup APIs when interrupts are disabled (polling mode), the configuration parameter enables polling of status flag in order to update the current LIN status.

Information: When the LIN driver need to operate in the interrupt disabled mode, the parameter shall be true. When the parameter is true then LIN wakeup signal detection cannot be supported by the driver as it depends on hardware interrupt, which triggers in a small interval of time.

Lin_17_AscLin driver

15.2 Assumptions of Use (AoUs)

There are no AoUs for the driver.

Lin_17_AscLin driver**15.3 Reference information****15.3.1 Configuration interfaces****Figure 150 Container hierarchy along with their configuration parameters****15.3.1.1 Container: CommonPublishedInformation**

This section describes the parameters published by the Lin driver module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

15.3.1.1.1 ArMajorVersion**Table 1003 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		

Lin_17_AscLin driver**Table 1003 Specification for ArMajorVersion (continued)**

Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.1.2 ArMinorVersion**Table 1004 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.1.3 ArPatchVersion**Table 1005 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Lin_17_AscLin driver**Table 1005 Specification for ArPatchVersion (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.1.4 ModuleId**Table 1006 Specification for ModuleId**

Name	ModuleId		
Description	Module ID of this module from the Module list.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 65535		
Default value	82		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.1.5 Release**Table 1007 Specification for Release**

Name	Release		
Description	This parameter indicates the TC3xx device derivative used for the implementation.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per hardware derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Lin_17_AscLin driver**15.3.1.1.6 SwMajorVersion****Table 1008 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.		
Multiplicity	1..1	Type	EcuclIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.1.7 SwMinorVersion**Table 1009 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Minor version number of the vendor specific implementation of the module. the numbering is vendor specific.		
Multiplicity	1..1	Type	EcuclIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.1.8 SwPatchVersion**Table 1010 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Patch level version number of the vendor specific implementation of the module. the numbering is vendor specific.		

Lin_17_AscLin driver**Table 1010 Specification for SwPatchVersion (continued)**

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.1.9 VendorApiInfix**Table 1011 Specification for VendorApiInfix**

Name	VendorApiInfix		
Description	This parameter is used to specify the vendor specific name.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	AscLin		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.1.10 VendorId**Table 1012 Specification for VendorId**

Name	VendorId		
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Lin_17_AscLin driver

Table 1012 Specification for VendorId (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.2 Container: Lin

Configuration of the LIN module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

15.3.1.2.1 Config Variant

Table 1013 Specification for Config Variant

Name	Config Variant		
Description	Selects the config-variant for the LIN module. The default value of this parameter is set to VariantPostBuild as per AUTOSAR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VariantPostBuild: Post Build Support.		
Default value	VariantPostBuild		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.3 Container: LinChannel

This container contains the configuration (parameters) of the LIN Controller(s).

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

15.3.1.3.1 LinAutoCalcBaudParams

Table 1014 Specification for LinAutoCalcBaudParams

Name	LinAutoCalcBaudParams
-------------	-----------------------

Lin_17_AscLin driver**Table 1014 Specification for LinAutoCalcBaudParams (continued)**

Description	This parameter enables or disables the automatic Baud rate parameter calculation by the configuration tool. Note: The optional features are disabled by default to minimize the executable code size. Since the calculation of the baud rate is done automatically.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.3.2 LinChanAssignedHw**Table 1015 Specification for LinChanAssignedHw**

Name	LinChanAssignedHw		
Description	This parameter defines which ASCLIN module is selected by LIN channel. Note: Minimum Kernel ID is selected as the default value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ASCLIN0: Signifies ASCLIN hardware kernel 0. ASCLINx: This parameter signifies kernel name, where x is varies from 1 to maximum number of units as per device variant. For example ASCLIN1, ASCLIN2,, ASCLINx, where x depends on device variant.		
Default value	ASCLIN0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Lin_17_AscLin driver**15.3.1.3.3 LinChannelBaudDenominator****Table 1016 Specification for LinChannelBaudDenominator**

Name	LinChannelBaudDenominator		
Description	<p>The parameter value is used to configure the BRG register DENOMINATOR field.</p> <p>The parameter is editable if LinAutoCalcBaudParams parameter is false.</p> <p>Note: Minimum Denominator value is selected as the default value.</p>		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	1 - 4095		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	LinAutoCalcBaudParams		

15.3.1.3.4 LinChannelBaudNumerator**Table 1017 Specification for LinChannelBaudNumerator**

Name	LinChannelBaudNumerator		
Description	<p>The parameter value is used to configure the BRG register NUMERATOR field.</p> <p>The parameter is editable if LinAutoCalcBaudParams parameter is false.</p> <p>Note: Minimum Numerator value is selected as the default value.</p>		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	1 - 4905		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	LinAutoCalcBaudParams		

Lin_17_AscLin driver**15.3.1.3.5 LinChannelBaudPreScalar****Table 1018 Specification for LinChannelBaudPreScalar**

Name	LinChannelBaudPreScalar		
Description	<p>This parameter value is used to configure the BITCON register PRESCALAR field.</p> <p>The parameter is editable if LinAutoCalcBaudParams parameter is false.</p> <p>Note: Minimum Pre-Scalar value is selected as the default value.</p>		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 4095		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	LinAutoCalcBaudParams		

15.3.1.3.6 LinChannelBaudRate**Table 1019 Specification for LinChannelBaudRate**

Name	LinChannelBaudRate		
Description	<p>Specifies the baud rate of the LIN channel.</p> <p>The parameter is editable if LinAutoCalcBaudParams parameter is true.</p> <p>Note: Minimum BaudRate value is selected as the default value.</p>		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	1000 - 20000		
Default value	1000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	LinAutoCalcBaudParams		

Lin_17_AscLin driver

15.3.1.3.7 LinChannelEcuMWakeupSource

Table 1020 Specification for LinChannelEcuMWakeupSource

Name	LinChannelEcuMWakeupSource		
Description	<p>This parameter contains a reference to the Wakeup Source for this controller as defined in the ECU State Manager.</p> <p>Note: Since the name of the dependent container is user configurable, the default value is kept as NULL.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: EcuMWakeupSource		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

15.3.1.3.8 LinChannelId

Table 1021 Specification for LinChannelId

Name	LinChannelId		
Description	<p>Identifies the LIN channel. Replaces LIN_CHANNEL_INDEX_NAME from the LIN SWS.</p> <p>Note: Minimum channel ID is selected as the default value.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Lin_17_AscLin driver**15.3.1.3.9 LinChannelWakeUpSupport****Table 1022 Specification for LinChannelWakeUpSupport**

Name	LinChannelWakeUpSupport		
Description	Specifies if the LIN hardware channel supports wake up functionality. Note: The optional features are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	LinInterruptDisabled		

15.3.1.3.10 LinClockRef**Table 1023 Specification for LinClockRef**

Name	LinClockRef		
Description	Reference to the LIN clock source configuration, which is set in the MCU driver configuration. Note: Not editable because clock configuration is common for the ASCLIN module not configured for each channel. Instead configuration parameter LinSysClockRef serves the purpose.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node:		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Lin_17_AscLin driver**15.3.1.3.11 LinInterByteSpace****Table 1024 Specification for LinInterByteSpace**

Name	LinInterByteSpace		
Description	This parameter is used to configure IDLE delay in bit times. Default- Minimum Inter Byte Space value is selected as the default value.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.3.12 LinRxAlternateInputSignal**Table 1025 Specification for LinRxAlternateInputSignal**

Name	LinRxAlternateInputSignal		
Description	This parameter selects the alternate input for the RX signal for the given LIN channel. Note: The first available data line for configured ASCLIN HW unit is selected as default value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SELECT_A_PORT14_PIN1: Rx data pin option SELECT_x_PORTy_PINz: This parameter varies in availability as per configured LIN kernel, and device variant, where x signifies dataline , y signifies port number and z signifies pin number. For example SELECT_A_PORT14_PIN1.		
Default value	SELECT_A_PORT14_PIN1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Lin_17_AscLin driver

15.3.1.4 Container: LinDemEventParameterRefs

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. Note: LinDemEventParameterRefs configuration container is made non editable as the LIN driver do not raise DEM error.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

15.3.1.4.1 LIN_E_TIMEOUT

Table 1026 Specification for LIN_E_TIMEOUT

Name	LIN_E_TIMEOUT		
Description	<p>Reference to the DemEventParameter which shall be issued when the error "Timeout caused by hardware error" has occurred. If the reference is not configured the error shall be reported as DET error.</p> <p>Since the name of the dependent container is user configurable, the default value is kept as NULL.</p> <p>Note: LIN_E_TIMEOUT configuration parameter is made non editable as the LIN driver do not raise DEM error, only for AUTOSAR compliance.</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

15.3.1.5 Container: LinGeneral

This container contains the parameters related to each LIN Driver Unit.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

15.3.1.5.1 LinCsrClksel

Table 1027 Specification for LinCsrClksel

Name	LinCsrClksel
Description	This parameter selects BaudRate logic clock for the LIN driver.

Lin_17_AscLin driver**Table 1027 Specification for LinCsrClksel (continued)**

	Note: Default value set with fast mode.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ASCLINF: ASCLIN fast clock mode ASCLINS: ASCLIN slow clock mode		
Default value	ASCLINF		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.5.2 LinDevErrorDetect**Table 1028 Specification for LinDevErrorDetect**

Name	LinDevErrorDetect		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. TRUE: enabled (ON). FALSE: disabled (OFF).		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

15.3.1.5.3 LinHwMcuTrigSleepEnable**Table 1029 Specification for LinHwMcuTrigSleepEnable**

Name	LinHwMcuTrigSleepEnable
-------------	-------------------------

Lin_17_AscLin driver**Table 1029 Specification for LinHwMcuTrigSleepEnable (continued)**

Description	Enable or disable the ASCLIN module sleep upon request by setting corresponding CLC register EDIS bit, for all configured channels. The parameter is common for all ASCLIN channels. Note: The optional features are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.5.4 LinIndex**Table 1030 Specification for LinIndex**

Name	LinIndex		
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0. Note: Minimum channel index is selected as the default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

15.3.1.5.5 LinInitDeInitApiMode**Table 1031 Specification for LinInitDeInitApiMode**

Name	LinInitDeInitApiMode
-------------	----------------------

Lin_17_AscLin driver**Table 1031 Specification for LinInitDeInitApiMode (continued)**

Description	This parameter selects the operating mode for MCU initialization, de-initialization of APIs. Since LIN driver accesses the SFRs, it is more efficient to operate the LIN driver in supervisor mode. Hence, the default mode of operation is supervisor.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	LIN_MCAL_SUPERVISOR: Operating mode used is SUPERVISOR. LIN_MCAL_USER1: Operating mode used is USER1.		
Default value	LIN_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.5.6 LinInterruptDisabled**Table 1032 Specification for LinInterruptDisabled**

Name	LinInterruptDisabled		
Description	Specify LIN operation in interrupt disabled or enabled. Note: The optional features are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Lin_17_AscLin driver**15.3.1.5.7 LinSysClockRef****Table 1033 Specification for LinSysClockRef**

Name	LinSysClockRef		
Description	<p>This parameter refers to the system clock configured by MCU driver. This reference is used for BaudRate computation.</p> <p>Note: Since the name of the dependent container is user configurable, the default value is kept as NULL.</p>		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

15.3.1.5.8 LinTimeoutDuration**Table 1034 Specification for LinTimeoutDuration**

Name	LinTimeoutDuration		
Description	<p>Specifies the maximum number of loops for blocking function until a timeout is raised in short term wait loops.</p> <p>Default- Minimum Timeout duration value is selected as the default value.</p> <p>Note: LinTimeoutDuration configuration parameter is made non editable as the requirement for the parameter is not present. Although configuration parameter shall be present in the schema in order to maintain AUTOSAR schema.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Lin_17_AscLin driver**15.3.1.5.9 LinVersionInfoApi****Table 1035 Specification for LinVersionInfoApi**

Name	LinVersionInfoApi		
Description	Switches the Lin_GetVersionInfo function ON or OFF. Note: The optional features are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

15.3.1.6 Container: LinGlobalConfig

This container contains the global configuration parameter of the Lin driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

15.3.2 Functions - Type definitions**15.3.2.1 Lin_17_AscLin_ConfigType****Table 1036 Specification for Lin_17_AscLin_ConfigType**

Syntax	Lin_17_AscLin_ConfigType	
Type	Structure	
File	Lin_17_AscLin.h	
Range	- []	The elements of the data structure are specific to the micro-controller.
Description	This is the type of the external data structure containing the overall initialization data for the LIN driver and the SFR settings affecting the LIN channels. A pointer to such a structure is provided to the LIN driver initialization routine for configuration of the driver, LIN hardware unit and LIN hardware channels.	
Source	AUTOSAR	

Lin_17_AscLin driver**15.3.2.2 Lin_FramePidType****Table 1037 Specification for Lin_FramePidType**

Syntax	Lin_FramePidType	
Type	uint8	
File	Lin_GeneralTypes.h	
Range	0..0xFE	The LIN identifier (0...0x3F) together with its two parity bits.
Description	Represents all valid protected identifier used by Lin_SendFrame().	
Source	AUTOSAR	

15.3.2.3 Lin_FrameCsModelType**Table 1038 Specification for Lin_FrameCsModelType**

Syntax	Lin_FrameCsModelType	
Type	Enumeration	
File	Lin_GeneralTypes.h	
Range	0 - LIN_ENHANCED_CS	Enhanced checksum model
	1 - LIN_CLASSIC_CS	Classic checksum model
Description	This type is used to specify the Checksum model to be used for the LIN Frame.	
Source	AUTOSAR	

15.3.2.4 Lin_FrameResponseType**Table 1039 Specification for Lin_FrameResponseType**

Syntax	Lin_FrameResponseType	
Type	Enumeration	
File	Lin_GeneralTypes.h	
Range	0 - LIN_MASTER_RESPONSE	Response is generated from this (master) node
	1 - LIN_SLAVE_RESPONSE	Response is generated from a remote slave node
	2 - LIN_SLAVE_TO_SLAVE	Response is generated from one slave to another slave, for the master the response will be anonymous, it does not have to receive the response.
Description	This type is used to specify whether the frame processor is required to transmit the response part of the LIN frame.	
Source	AUTOSAR	

Lin_17_AscLin driver**15.3.2.5 Lin_FrameDlType****Table 1040 Specification for Lin_FrameDlType**

Syntax	Lin_FrameDlType	
Type	uint8	
File	Lin_GeneralTypes.h	
Range	1...8	Data length of a LIN Frame
Description	This type is used to specify the number of SDU data bytes to copy.	
Source	AUTOSAR	

15.3.2.6 Lin_PduType**Table 1041 Specification for Lin_PduType**

Syntax	Lin_PduType	
Type	Structure	
File	Lin_GeneralTypes.h	
Range	Lin_FramePidType Pid	Describes Pid number for the current LIN frame for which header is sent.
	Lin_FrameCsModelType Cs	Describes checksum used for the LIN frame
	Lin_FrameResponseType Drc	Describes type of communication required for the LIN frame
	Lin_FrameDlType Dl	Describes data length for the current LIN frame
	uint8 * SduPtr	Pointer to get the data from upper layer
Description	This Type is used to provide PID, checksum model, data length and SDU pointer from the LIN Interface to the LIN driver.	
Source	AUTOSAR	

15.3.2.7 Lin_StatusType**Table 1042 Specification for Lin_StatusType**

Syntax	Lin_StatusType	
Type	Enumeration	
File	Lin_GeneralTypes.h	
Range	0 - LIN_NOT_OK	LIN frame operation return value. Development or production error occurred
	1 - LIN_TX_OK	LIN frame operation return value. Successful transmission.

Lin_17_AscLin driver**Table 1042 Specification for Lin_StatusType (continued)**

	2 - LIN_TX_BUSY	LIN frame operation return value. Ongoing transmission (Header or Response).
	3 - LIN_TX_HEADER_ERROR	LIN frame operation return value. Erroneous header transmission such as: - Mismatch between sent and read back data - Identifier parity error or - Physical bus error
	4 - LIN_TX_ERROR	LIN frame operation return value. Erroneous response transmission such as: - Mismatch between sent and read back data - Physical bus error
	5 - LIN_RX_OK	LIN frame operation return value. Reception of correct response.
	6 - LIN_RX_BUSY	LIN frame operation return value. Ongoing reception: at least one response byte has been received, but the checksum byte has not been received.
	7 - LIN_RX_ERROR	LIN frame operation return value. Erroneous response reception such as: - Framing error - Overrun error - Checksum error or - Short response
	8 - LIN_RX_NO_RESPONSE	LIN frame operation return value. No response byte has been received so far.
	9 - LIN_OPERATIONAL	LIN channel state return value. Normal operation; the related LIN channel is ready to transmit next header. No data from previous frame available (e.g. after initialization)
	10 - LIN_CH_SLEEP	LIN channel state return value. Sleep state operation; in this state wake-up detection from slave nodes is enabled.
Description	LIN operation states for a LIN channel or frame, as returned by the API service Lin_GetStatus().	
Source	AUTOSAR	

Lin_17_AscLin driver**15.3.3 Functions - APIs****15.3.3.1 Lin_17_AscLin_CheckWakeup****Table 1043 Specification for Lin_17_AscLin_CheckWakeup API**

Syntax	<pre>Std_ReturnType Lin_17_AscLin_CheckWakeup (const uint8 Channel)</pre>	
Service ID	0x0a	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Channel	LIN channel to be addressed
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: No error has occurred during execution of the API. E_NOT_OK: An error has occurred during execution of the API.
Description	This function checks if a wakeup has occurred on the addressed LIN channel. The API calls EcuM_SetWakeupEvent and LinIf_WakeupConfirmation, if wakeup (LOW) signal is detected.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>LIN_17_ASCLIN_E_UNINIT: API service used without module initialization.</p> <p>LIN_17_ASCLIN_E_INVALID_CHANNEL: API service used with an invalid or inactive channel parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

Lin_17_AscLin driver**15.3.3.2 Lin_17_AscLin_GetStatus****Table 1044 Specification for Lin_17_AscLin_GetStatus API**

Syntax	<pre>Lin_StatusType Lin_17_AscLin_GetStatus (const uint8 Channel, uint8 ** const Lin_SduPtr)</pre>	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Channel	LIN channel to be checked
Parameters (out)	Lin_SduPtr	Pointer to pointer to a shadow buffer or memory mapped LIN Hardware receive buffer where the current SDU is stored.
Parameters (in - out)	-	-
Return	Lin_StatusType	<p>LIN_NOT_OK: Development or production error occurred.</p> <p>LIN_TX_OK: Successful transmission.</p> <p>LIN_TX_BUSY: Ongoing transmission (Header or Response).</p> <p>LIN_TX_HEADER_ERROR: Erroneous header transmission such as: - collision error, TX FIFO over flow error, or parity error.</p> <p>LIN_TX_ERROR: Erroneous response transmission such as: - collision error, TX FIFO over flow error.</p> <p>LIN_RX_OK: Reception of correct response.</p> <p>LIN_RX_BUSY: Ongoing reception: at least one response byte has been received, but the checksum byte has not been received.</p> <p>LIN_RX_ERROR: Erroneous response reception such as: - LIN frame error, frame checksum error, RX FIFO overflow or under flow error.</p> <p>LIN_RX_NO_RESPONSE: No response byte has been received so far.</p> <p>LIN_OPERATIONAL: Normal operation; the related LIN channel is just initialized or waked up from the LIN_CH_SLEEP and no data has been sent.</p> <p>LIN_CH_SLEEP: Sleep state operation; in this state wake-up detection from slave nodes is enabled.</p>
Description	Gets the status of the LIN driver. Further the API is also used to get the received data from the driver by upper layer using SDU shadow pointer.	
Source	AUTOSAR	
Error handling	DET:	

Lin_17_AscLin driver**Table 1044 Specification for Lin_17_AscLin_GetStatus API (continued)**

	<p>LIN_17_ASCLIN_E_UNINIT: API service used without module initialization.</p> <p>LIN_17_ASCLIN_E_INVALID_CHANNEL: API service used with an invalid or inactive channel parameter.</p> <p>LIN_17_ASCLIN_E_PARAM_POINTER: API service called with a NULL pointer.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

15.3.3.3 Lin_17_AscLin_GoToSleep**Table 1045 Specification for Lin_17_AscLin_GoToSleep API**

Syntax	<pre>Std_ReturnType Lin_17_AscLin_GoToSleep (const uint8 Channel)</pre>	
Service ID	0x06	
Sync/Async	Asynchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Channel	LIN channel to be addressed
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Sleep command has been accepted. E_NOT_OK: Sleep command has not been accepted, development or production error occurred.
Description	The service instructs the driver to transmit a go-to-sleep-command on the addressed LIN channel.	
Source	AUTOSAR	
Error handling	DET: LIN_17_ASCLIN_E_UNINIT: API service used without module initialization. LIN_17_ASCLIN_E_INVALID_CHANNEL: API service used with an invalid or inactive channel parameter.	

Lin_17_AscLin driver**Table 1045 Specification for Lin_17_AscLin_GoToSleep API (continued)**

	<p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

15.3.3.4 Lin_17_AscLin_GoToSleepInternal**Table 1046 Specification for Lin_17_AscLin_GoToSleepInternal API**

Syntax	Std_ReturnType Lin_17_AscLin_GoToSleepInternal (const uint8 Channel)	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Channel	LIN channel to be addressed
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Command has been accepted. E_NOT_OK: Command has not been accepted, development or production error occurred.
Description	Sets the channel state to LIN_CH_SLEEP, enables the wake-up detection and sets the LIN hardware unit to reduced power operation mode by the hardware. Enabling of the configuration parameter LinHwMcuTrigSleepEnable enables hardware sleep mode when the API is called. If the parameter is false only software state changes.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>LIN_17_ASCLIN_E_INVALID_CHANNEL: API service used with an invalid or inactive channel parameter.</p> <p>LIN_17_ASCLIN_E_UNINIT: API service used without module initialization.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p>	

Lin_17_AscLin driver**Table 1046 Specification for Lin_17_AscLin_GoToSleepInternal API (continued)**

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

15.3.3.5 Lin_17_AscLin_Init**Table 1047 Specification for Lin_17_AscLin_Init API**

Syntax	<pre>void Lin_17_AscLin_Init (const Lin_17_AscLin_ConfigType * const Config)</pre>	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Config	Pointer to LIN driver configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Initializes the LIN module.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>LIN_17_ASCLIN_E_STATE_TRANSITION: Invalid state transition for the current state.</p> <p>LIN_17_ASCLIN_E_INVALID_POINTER: API service called with invalid configuration pointer.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

Lin_17_AscLin driver**15.3.3.6 Lin_17_AscLin_SendFrame****Table 1048 Specification for Lin_17_AscLin_SendFrame API**

Syntax	<pre>Std_ReturnType Lin_17_AscLin_SendFrame (const uint8 Channel, Lin_PduType * const PduInfoPtr)</pre>	
Service ID	0x04	
Sync/Async	Asynchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Channel PduInfoPtr	LIN channel to be addressed Pointer to PDU containing the PID, checksum model, response type, DL and SDU data pointer
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Send command has been accepted. E_NOT_OK: Send command has not been accepted, development or production error occurred.
Description	Sends a LIN header and a LIN response, if necessary. The direction of the frame response (master response, slave response, slave-to-slave communication) is provided by the PduInfoPtr.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <ul style="list-style-type: none"> LIN_17_ASCLIN_E_INVALID_CHANNEL: API service used with an invalid or inactive channel parameter. LIN_17_ASCLIN_E_STATE_TRANSITION: Invalid state transition for the current state. LIN_17_ASCLIN_E_UNINIT: API service used without module initialization. LIN_17_ASCLIN_E_PARAM_POINTER: API service called with a NULL pointer. <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

Lin_17_AscLin driver**15.3.3.7 Lin_17_AscLin_Wakeup****Table 1049 Specification for Lin_17_AscLin_Wakeup API**

Syntax	Std_ReturnType Lin_17_AscLin_Wakeup (const uint8 Channel)	
Service ID	0x07	
Sync/Async	Asynchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Channel	LIN channel to be addressed
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Wake-up request has been accepted. E_NOT_OK: Wake-up request has not been accepted, development or production error occurred.
Description	Generates a wake up pulse and sets the channel state to LIN_CH_OPERATIONAL.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>LIN_17_ASCLIN_E_STATE_TRANSITION: Invalid state transition for the current state.</p> <p>LIN_17_ASCLIN_E_UNINIT: API service used without module initialization.</p> <p>LIN_17_ASCLIN_E_INVALID_CHANNEL: API service used with an invalid or inactive channel parameter.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

15.3.3.8 Lin_17_AscLin_WakeupInternal**Table 1050 Specification for Lin_17_AscLin_WakeupInternal API**

Syntax	Std_ReturnType Lin_17_AscLin_WakeupInternal (
---------------	--

Lin_17_AscLin driver**Table 1050 Specification for Lin_17_AscLin_WakeupInternal API (continued)**

	const uint8 Channel)	
Service ID	0x0b	
Sync/Async	Asynchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Channel	LIN channel to be addressed
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Wake-up request has been accepted. E_NOT_OK: Wake-up request has not been accepted, development or production error occurred.
Description	Sets the channel state to LIN_CH_OPERATIONAL without generating a wake up pulse.	
Source	AUTOSAR	
Error handling	DET: LIN_17_ASCLIN_E_UNINIT: API service used without module initialization. LIN_17_ASCLIN_E_STATE_TRANSITION: Invalid state transition for the current state. LIN_17_ASCLIN_E_INVALID_CHANNEL: API service used with an invalid or inactive channel parameter. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

15.3.3.9 Lin_17_AscLin_GetVersionInfo**Table 1051 Specification for Lin_17_AscLin_GetVersionInfo API**

Syntax	void Lin_17_AscLin_GetVersionInfo (Std_VersionInfoType * const versioninfo)
Service ID	0x01
Sync/Async	Synchronous

Lin_17_AscLin driver

Table 1051 Specification for Lin_17_AscLin_GetVersionInfo API (continued)

ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where is stored the version information of this module.
Parameters (in - out)	-	-
Return	void	-
Description	Returns the version information of this module.	
Source	AUTOSAR	
Error handling	DET: LIN_17_ASCLIN_E_PARAM_POINTER: API service called with a NULL pointer. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	LinVersionInfoApi	
User hints	-	

15.3.4 Notifications and Callbacks

The driver does not support any notification and callbacks.

15.3.5 Scheduled functions

The driver does not support any scheduled functions.

15.3.6 Interrupt service routines

This section lists all the interrupt handlers of the driver.

15.3.6.1 Lin_17_AscLin_IsrError

Table 1052 Specification for Lin_17_AscLin_IsrError API

Syntax	void Lin_17_AscLin_IsrError (const uint8 HwUnit)
---------------	---

Lin_17_AscLin driver**Table 1052 Specification for Lin_17_AscLin_IsrError API (continued)**

Service ID	-	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant (For different channels)	
Parameters (in)	HwUnit	Represents HW module number.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This ISR will be called whenever there is a data transmission or reception error or a wakeup signal is detected in ASCLIN. The ISR will call EcuM_CheckWakeup, if channel wakeup support is enabled and wakeup signal is detected.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	LinInterruptDisabled	
User hints	-	

15.3.6.2 Lin_17_AscLin_IsrReceive**Table 1053 Specification for Lin_17_AscLin_IsrReceive API**

Syntax	<pre>void Lin_17_AscLin_IsrReceive (const uint8 HwUnit)</pre>	
Service ID	-	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant (For different channels)	
Parameters (in)	HwUnit	Represents HW module number.

Lin_17_AscLin driver**Table 1053 Specification for Lin_17_AscLin_IsrReceive API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This ISR will be called whenever the Slave response data is completely received by the ASCLIN without any errors.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	LinInterruptDisabled	
User hints	-	

15.3.6.3 Lin_17_AscLin_IsrTransmit**Table 1054 Specification for Lin_17_AscLin_IsrTransmit API**

Syntax	<pre>void Lin_17_AscLin_IsrTransmit (const uint8 HwUnit)</pre>	
Service ID	-	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant (For different channels)	
Parameters (in)	HwUnit	Represents HW module number.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This ISR will be called whenever the response data is successfully transmitted by the ASCLIN without any errors.	
Source	IFX	

Lin_17_AscLin driver
Table 1054 Specification for Lin_17_AscLin_IsrTransmit API (continued)

Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	LinInterruptDisabled
User hints	-

15.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

15.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Table 1055 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
API service used with an invalid or inactive channel parameter.	AUTOSAR	LIN_17_ASCLIN_E_INVALID_CHANNEL=0x02	Lin_17_AscLin_WakeupInternal, Lin_17_AscLin_Wakeup, Lin_17_AscLin_SendFrame, Lin_17_AscLin_GoToSleepInternal, Lin_17_AscLin_GoToSleep, Lin_17_AscLin_GetStatus, Lin_17_AscLin_CheckWakeup
API service called with an invalid configuration pointer.	AUTOSAR	LIN_17_ASCLIN_E_INVALID_POINTER=0x03	Lin_17_AscLin_Init
API service called with a NULL pointer.	AUTOSAR	LIN_17_ASCLIN_E_PARAM_POINTER=0x05	Lin_17_AscLin_GetVersionInfo, Lin_17_AscLin_SendFrame, Lin_17_AscLin_GetStatus
Invalid state transition for the current state.	AUTOSAR	LIN_17_ASCLIN_E_STATE_TRANSITION=0x04	Lin_17_AscLin_WakeupInternal, Lin_17_AscLin_Wakeup, Lin_17_AscLin_SendFrame, Lin_17_AscLin_Init

Lin_17_AscLin driver

Table 1055 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
API service used without module initialization.	AUTOSAR	LIN_17_ASCLIN_E_UNINIT=0x00	Lin_17_AscLin_WakeupInternal, Lin_17_AscLin_Wakeup, Lin_17_AscLin_SendFrame, Lin_17_AscLin_GoToSleepInternal, Lin_17_AscLin_GoToSleep, Lin_17_AscLin_GetStatus, Lin_17_AscLin_CheckWakeup

15.3.7.2 Production errors

The driver does not report any production errors.

15.3.7.3 Safety errors

The driver does not report any safety errors.

15.3.7.4 Runtime errors

The driver does not report any runtime errors.

15.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

15.3.8.1 Deviations

The section describes the deviations from software specification.

Table 1056 Known deviations

Reference	Deviation
AUTOSAR header file inclusion requirement for LIN module [SWS_Lin_00075]	As per AUTOSAR requirement, the <code>EcuM_Cbk.h</code> file should be included in the <code>Lin_17_AscLin.c</code> file. However, the LIN driver configuration structure defined in <code>Lin_17_AscLin.h</code> file refer to the data type <code>EcuM_WakeupSourceType</code> from the EcuM module. Hence to avoid compilation error, <code>EcuM_Cbk.h</code> is included in the <code>Lin_17_AscLin.h</code> file.
AUTOSAR LinClockRef configuration parameter [ECUC_Lin_00094]	The configuration parameter is not used and is not editable, instead clock reference for all LIN channels

Lin_17_AscLin driver**Table 1056 Known deviations (continued)**

	are provided commonly by the <code>LinSysClockRef</code> parameter.
--	---

15.3.8.2 Limitations

There are no limitations for the LIN driver.

15.3.9 Unsupported hardware features

The following ASCLIN hardware features are not supported by the LIN driver:

- LIN slave mode is not used as AUTOSAR requires only the master mode
- Auto-baud detection based on Sync Field measurement
- Break detection
- Stuck at zero/one monitoring
- Bus idle time monitoring

MCALLIB driver

16 MCALLIB driver

16.1 User information

16.1.1 Description

The MCAL Library (MCALLIB) provides a set of utility routines for use by the MCAL drivers. The services provided are ENDINIT management, global-local memory address translation, timer based delay, retrieval of CPU identifier, abstraction of TriCore™-intrinsic instruction and spinlock.

16.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

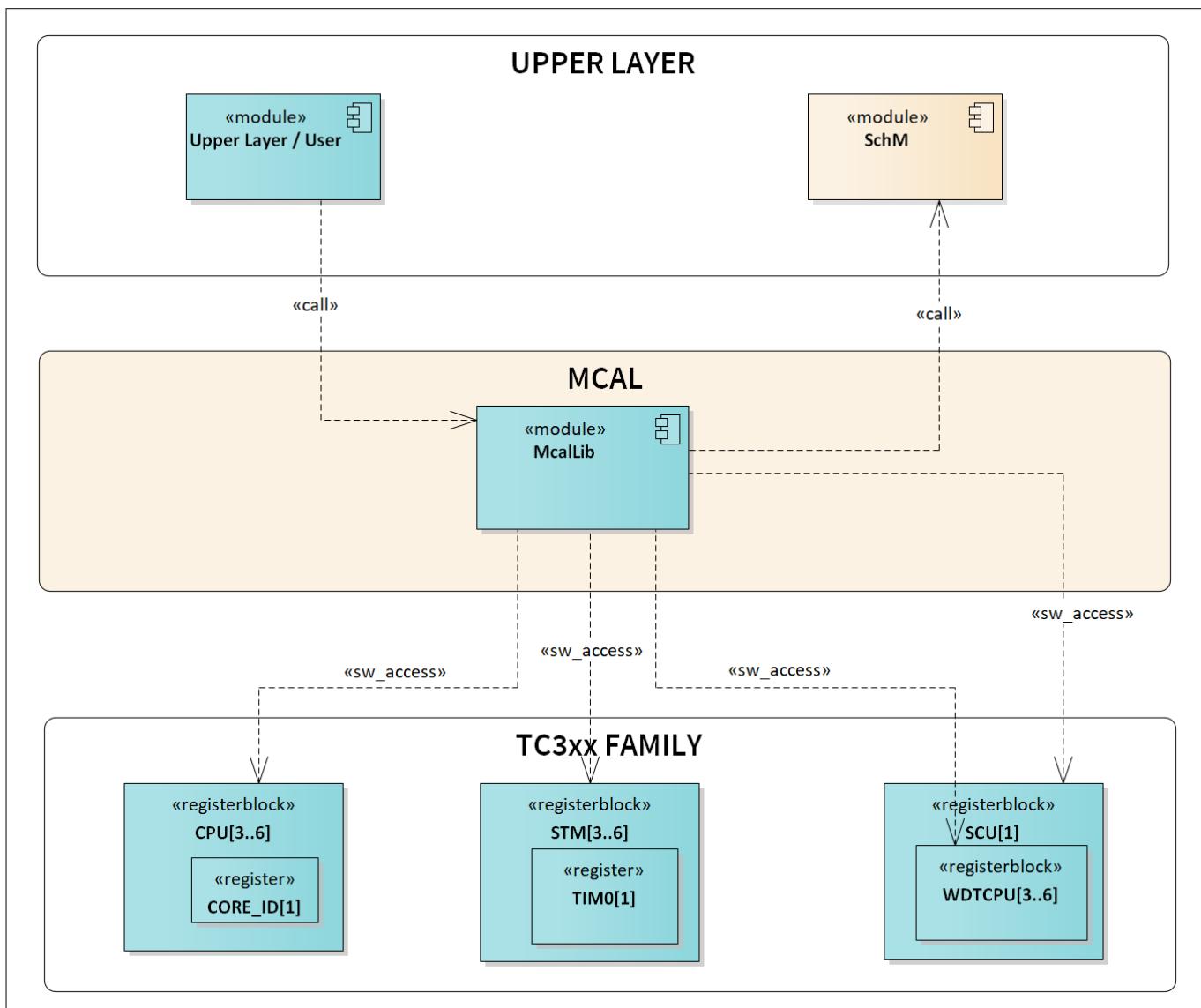


Figure 151 Mapping of hardware-software interfaces

MCALLIB driver

16.1.2.1 STM: primary hardware peripheral

Hardware functional features

The MCALLIB driver only reads the System Timer Bits [31:0]. The key hardware functional feature used by the driver is:

- Free-running system timer

The unsupported feature of the STM is:

- Compare match operation

Users of the hardware

The MCALLIB driver provides API to read current STM tick count.

Hardware diagnostic features

Not applicable.

Hardware events

The MCALLIB driver does not use the hardware events from the STM IP.

16.1.2.2 CORE_ID-CPU: primary hardware peripheral

Hardware functional features

The MCALLIB driver uses the CPU to retrieve the CPU ID on which the code is executing. The key hardware functional feature used by the driver is:

- CPU registers

The unsupported feature of the CPU is:

- Debug support

Users of the hardware

The MCALLIB driver exclusively utilizes the CPU module.

Hardware diagnostic features

The SMU alarms configured for the CPU are not monitored by the MCALLIB driver.

Hardware events

Hardware events from CPU are not used by MCALLIB.

16.1.2.3 SCU: dependent hardware peripheral

Hardware functional features

The MCALLIB driver depends on the SCU IP for the clock functionality. The driver requires the fSPB and fSTM clock signals for functioning.

ENDINIT feature of the SCU is implemented by the MCALLIB driver.

Users of the hardware

The SCU IP supplies clock for all the peripherals. The MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

MCALLIB and WDG driver both update the WDT peripheral related registers. In order to avoid register corruption due to concurrent writes, all the writes to the WDT registers is performed under the same critical section by MCALLIB and WDG driver both.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the MCALLIB driver.

Hardware events

MCALLIB driver

Hardware events from the SCU are not used by the MCALLIB driver.

16.1.3 File structure

16.1.3.1 C file structure

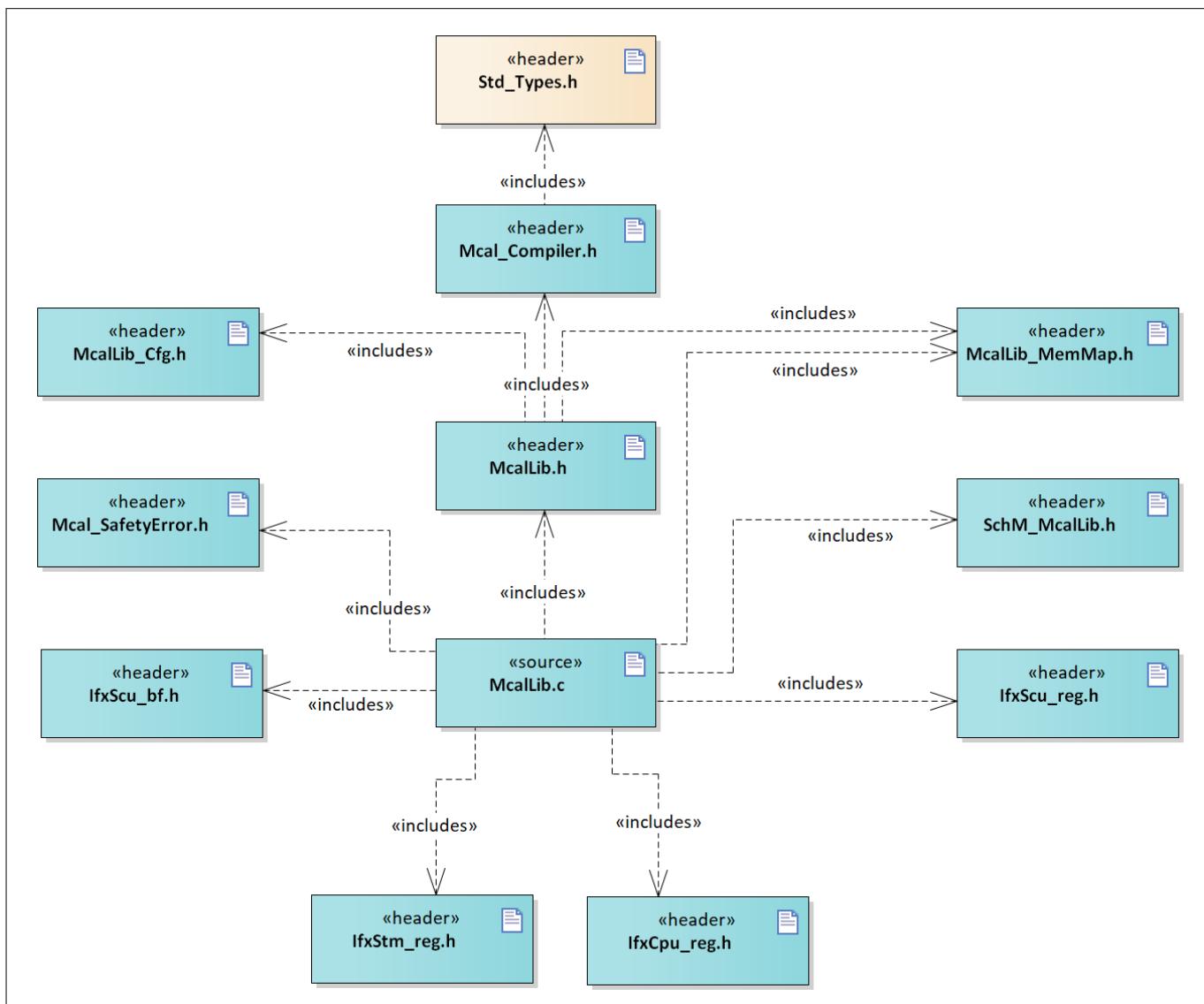


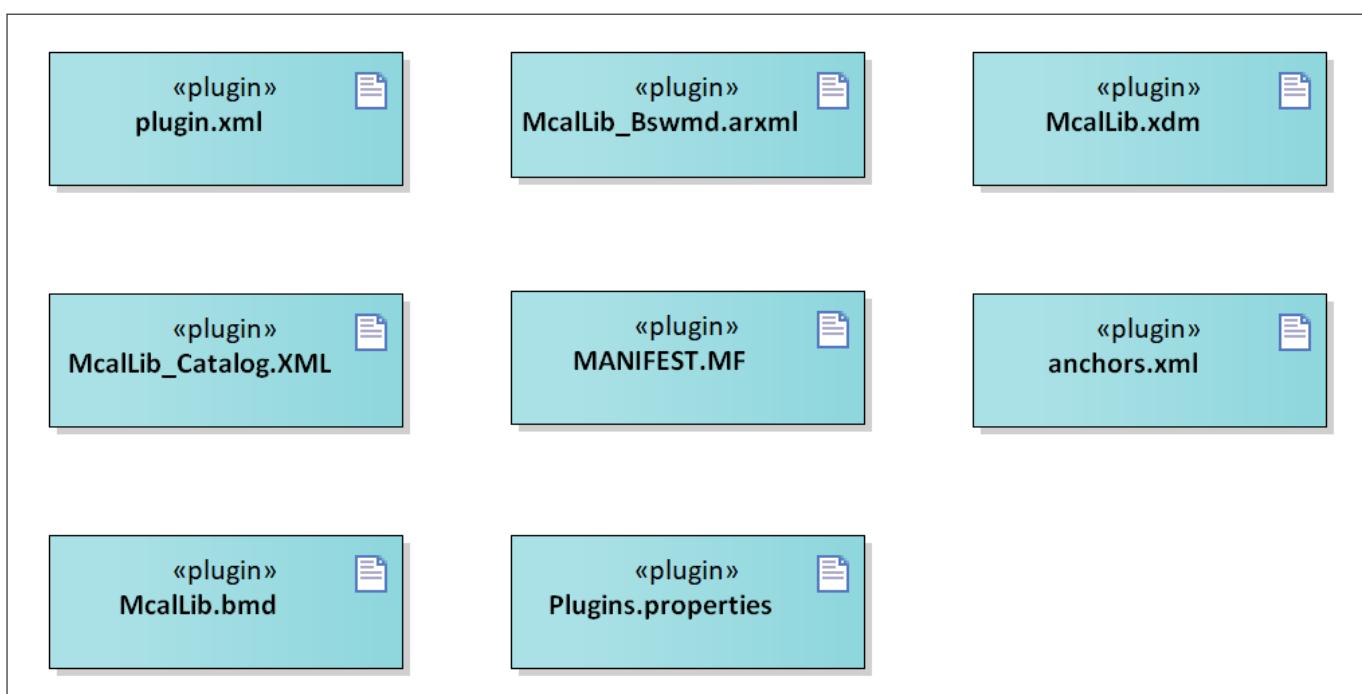
Figure 152 C file structure

Table 1057 C file structure

File name	Description
IfxCpu_Reg.h	SFR header file for CPU
IfxScu_bf.h	SFR header file for SCU
IfxScu_Reg.h	SFR header file for SCU
IfxStm_Reg.h	SFR header file for STM
McalLib.c	Static source code for the MCALLIB

MCALLIB driver
Table 1057 C file structure (continued)

File name	Description
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_Cfg.h	Generated header file providing information on number of cores, DSPR, PSPR (start and end addresses) and system and backup clock information
McalLib_MemMap.h	File containing the memory section definitions used by the MCALLIB
Mcal_Compiler.h	Provides abstraction for TriCore™-intrinsic instruction
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
SchM_McalLib.h	Provides prototype of SchM interfaces needed by the MCALLIB
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform..

16.1.3.2 Code generator plugin files

Figure 153 Code generator plugin files
Table 1058 Code generator plugin files

File name	Description
MANIFEST.MF	Tresos plugin support file containing the metadata for the MCALLIB driver
McalLib.bmd	AUTOSAR format XML data model schema file (for each device)
McalLib.xdm	Tresos format XML data model schema file
McalLib_Bswmd.arxml	AUTOSAR format module description file

MCALLIB driver
Table 1058 Code generator plugin files (continued)

File name	Description
McalLib_Catalog.XML	AUTOSAR format catalog file as per catalog_V3_0_0.ml.xsd
Plugins.properties	Tresos plugin support file for the MCALLIB driver
anchors.xml	Tresos anchors support file for the MCALLIB driver
plugin.xml	Tresos plugin support file for the MCALLIB driver

16.1.4 Integration hints

This section lists the key points that an integrator or user of the MCALLIB must consider. In general, the APIs of MCALLIB driver may be invoked from several CPU cores in parallel with some restrictions, which are also described in this section.

16.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the MCALLIB driver.

- **EcuM**

EcuM module is not required for integrating the MCALLIB driver.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the McallLib_MemMap.h file.

The McallLib_MemMap.h file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows:

MCALLIB driver

```

***** GLOBAL RAM DATA -- NON-CACHED LMU *****
#if defined MCALLIB_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*User pragmas here for Non-cached LMU*****/
#undef MCALLIB_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR
#elif defined MCALLIB_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*User pragmas here for Non-cached LMU*****/
#undef MCALLIB_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR

***** Static Global Constants Sections *****
***** Static Global Constants -- PF[x] *****
#if defined MCALLIB_START_SEC_CONST_ASIL_B_GLOBAL_8
/*User pragmas here for PF[x]*****/
#undef MCALLIB_START_SEC_CONST_ASIL_B_GLOBAL_8
#undef MEMMAP_ERROR
#elif defined MCALLIB_STOP_SEC_CONST_ASIL_B_GLOBAL_8
/*User pragmas here for PF[x]*****/
#undef MCALLIB_STOP_SEC_CONST_ASIL_B_GLOBAL_8
#undef MEMMAP_ERROR
#elif defined MCALLIB_START_SEC_CONST_ASIL_B_GLOBAL_32
/*User pragmas here for PF[x]*****/
#undef MCALLIB_START_SEC_CONST_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR
#elif defined MCALLIB_STOP_SEC_CONST_ASIL_B_GLOBAL_32
/*User pragmas here for PF[x]*****/
#undef MCALLIB_STOP_SEC_CONST_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR

***** CODE -- PF[x] *****
#if defined MCALLIB_START_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here for PF[x]*****/
#undef MCALLIB_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#elif defined MCALLIB_STOP_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here for PF[x]*****/
#undef MCALLIB_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#endif

#if defined MEMMAP_ERROR
#error "MCALLIB_MemMap.h, wrong pragma command"
#endif

```

- **DET**

The DET module is not required for integrating the MCALLIB driver.

- **DEM**

The DEM module is not required for the integration of MCALLIB driver.

- **SchM**

MCALLIB driver

The SchM module is a part of the RTE that manages the BSW. The MCALLIB driver uses the exclusive areas defined in the `SchM_McalLib.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the MCALLIB driver are:

- PeripheralEndInit
- SafetyEndInit
- CpuEndInit
- StmTimerResolution

The files `SchM_McalLib.h` and `SchM_McalLib.c` are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the

MCALLIB driver

MCALLIB driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as below:

```
***** Sample implementation of SchM_McalLib.c ****/
#include "SchM_McalLib.h"

void SchM_Enter_McalLib_PeripheralEndInit(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();
}

void SchM_Exit_McalLib_PeripheralEndInit(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts();
}

void SchM_Enter_McalLib_SafetyEndInit(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();
}

void SchM_Exit_McalLib_SafetyEndInit(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts();
}

void SchM_Enter_McalLib_CpuEndInit(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();
}

void SchM_Exit_McalLib_CpuEndInit(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts();
}

void SchM_Enter_McalLib_StmTimerResolution(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();
}

void SchM_Exit_McalLib_StmTimerResolution(void)
{
    /* End of Critical Section */
}
```

MCALLIB driver

```
    ResumeAllInterrupts () ;  
}
```

- **Safety error**

The MCALLIB driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API.

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the files `Mcal_SafetyError.c` and `Mcal_SafetyError.h` as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The MCALLIB driver does not provide any callbacks or notifications

- **Operating System (OS)**

The integrator shall implement the APIs routed from the MCALLIB via `McalLib_OsStub.h` file when the User-1 mode is used by any driver.

16.1.4.2 Multicore and Resource Manager

The MCALLIB driver supports execution of its APIs from all CPU cores. The following are the key points to be considered with respect to multicore in the driver:

- MCALLIB services accessing global hardware resources (like safety and peripheral endinit protection) would create a critical section and a spinlock around these accesses, which will serialize the shared hardware resource access across cores.

Code section:

The executable code of <Mod> driver is placed under single MemMap section. It can be relocated to any PFlash region.

Data section:

The sections marked as global should be relocated to the non-cached LMU region.

Constants:

The marked as global should be relocated to the non-cached LMU region.

Note: Relocating of code, data or constants to a distant memory region would impact execution timings.

16.1.4.3 MCU support

MCALLIB driver does not use any services provided by the MCU driver.

16.1.4.4 Port support

The MCALLIB driver does not use any services provided by the PORT driver.

16.1.4.5 DMA support

The MCALLIB driver does not use any services provided by the DMA driver.

MCALLIB driver

16.1.4.6 Interrupt connections

The MCALLIB driver does not use any interrupt source.

MCALLIB driver**16.1.4.7 Example usage**

The MCALLIB is a library. All the APIs provided are independent of each other, therefore, there is no example usage for this driver.

16.1.5 Key architectural considerations**16.1.5.1 User mode**

The integrator shall implement the APIs routed from the MCALLIB via `McalLib_OsStub.h` file when the User-1 mode is used by any driver.

16.1.5.2 Spinlock

Timeout value that is passed as an input parameter to the `Mcal_GetSpinlock` API must be in the range of 1 µs to 1048575 µs (timeout when passed as 1 indicate as 1 µs to this API).

MCALLIB driver

16.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **ASIL level of calling module**

User shall ensure that the ASIL level of the calling module is same as that of MCALLIB.

[cover parentID MCALLIB={3982DA82-28CB-453e-8D1C-4B80B83BE3CF}]

- **Common critical section**

User shall ensure that core specific interrupts are disabled in the critical sections SchM_Enter_Wdg_CpuEndInit and SchM_Enter_McalLib_CpuEndInit.

[cover parentID MCALLIB={70616172-E23B-4d86-9C20-7C5DF26143D7}]

- **ENDINIT Protected Register Access**

User shall ensure that all the ENDINIT protected registers are modified using only the write ENDINIT APIs (Mcal_WriteCpuEndInitProtReg, Mcal_WriteSafetyEndInitProtReg, Mcal_WriteSafetyEndInitProtRegMask, Mcal_WritePeripEndInitProtReg).

[cover parentID MCALLIB={845BAE75-B05D-49dc-822F-7480A13C4A84}]

- **Parameter range check for Mcal_SetBitAtomic and Mcal_GetBitAtomic**

The MCALLIB user shall ensure the following while using the APIs Mcal_SetBitAtomic and Mcal_GetBitAtomic: - Sum of the input parameter BitPos and BitLen should not be greater than 32 bits - BitLen should always be constant and non-zero value

[cover parentID MCALLIB={E28707C1-2DDB-451b-8DA6-3625A9EB2244}]

- **Password check**

User can verify the password set by calling the GetPassword APIs (Mcal_GetCpuWdgPassword, Mcal_GetSafetyEndInitPassword, Mcal_GetPeripheralEndInitPassword).

[cover parentID MCALLIB={D3EA116F-A029-4b83-A6E8-BB03A72E7C9B}]

- **STM timer resolution**

User shall call the Mcal_DelayResetTickCalibration API after any change in the clock tree to update the STM timer resolution.

[cover parentID MCALLIB={EF8478C5-1EDD-459e-B5DF-E729EE956664}]

- **Test, Test and set spinlock mechanism**

User shall ensure that: -The lock address passed to the Mcal_Getspinlock API must be at a non-cached memory address. -This API shall not be called within an ISR.

[cover parentID MCALLIB={8EADA6CF-0B73-430a-9545-B24315AAF137}]

- **Valid address (base + offset) are passed as register address for McalLib API**

Valid address (base + offset) shall be passed as the register address to the API

Mcal_WriteSafetyEndInitProtReg16

[cover parentID MCALLIB={81931B95-E9B4-4caa-BF12-7B2E84F1BC58}]

- **Valid CSFR address (only offset) are passed as register address for McalLib APIs**

Valid CSFR address (only offset) shall be passed as the register address for Core Specific SFRs to the APIs Mcal_WriteSafetyEndInitProtReg

[cover parentID MCALLIB={817DF82C-39C1-4767-B78B-9ECE9F585305}]

- **Valid pointer to be passed to APIs**

User shall ensure the correctness of the pointers that is passed as an input parameter before invoking the MCALLIB APIs.

[cover parentID MCALLIB={35C4D569-ECE0-4ff4-A361-4E2E5A06D535}]

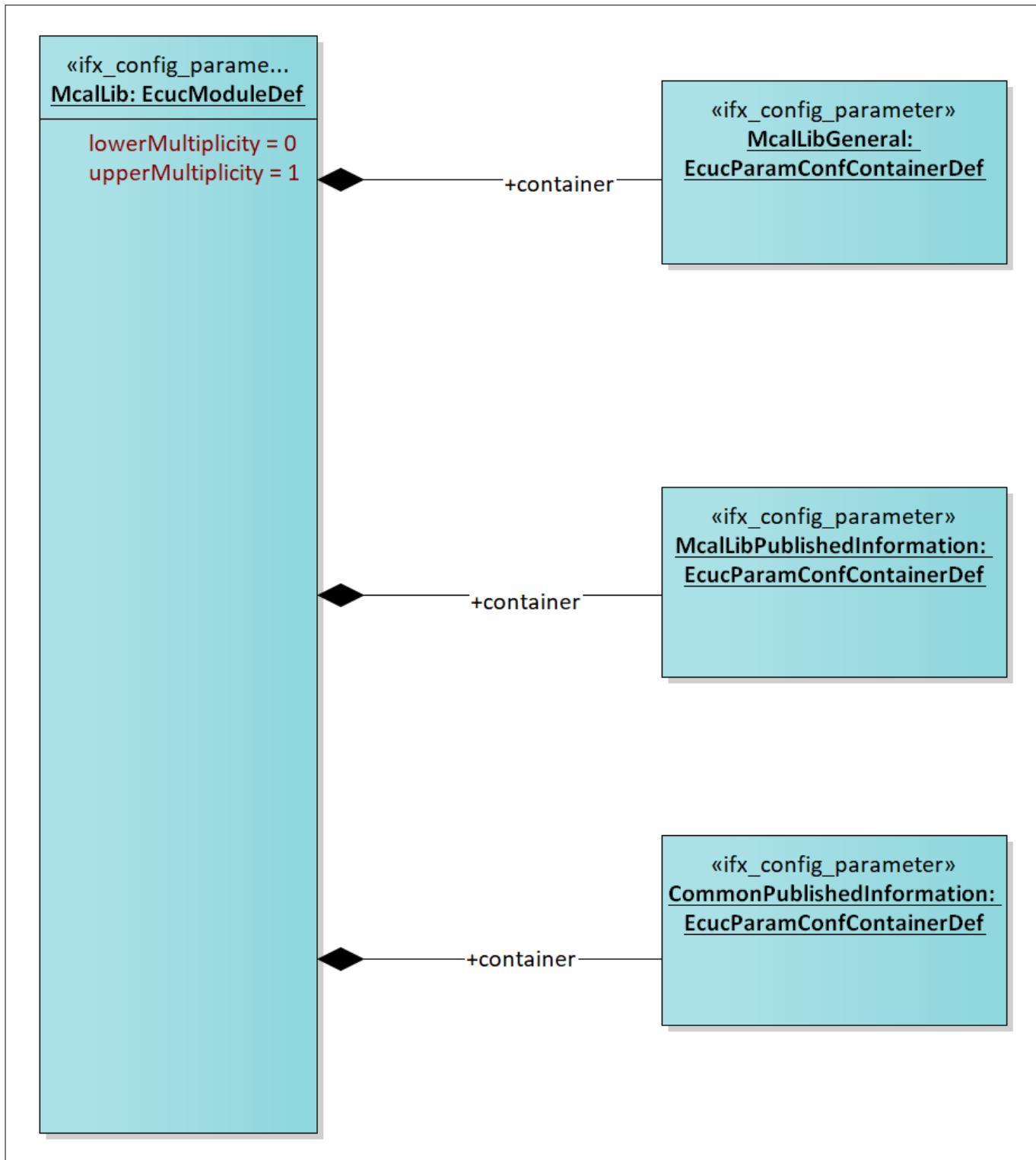
- **Valid value are passed as parameter for MCALLIB APIs**

MCALLIB driver

User shall ensure the correctness of the data value that is passed as an input parameter before invoking the MCALLIB APIs.

[cover parentID MCALLIB={42C179FC-3D4A-4648-B422-6BB895B43B4F}]

MCALLIB driver

16.3 Reference information**16.3.1 Configuration interfaces****Figure 154 Container hierarchy along with their configuration parameters**

MCALLIB driver

16.3.1.1 Container: McalLibGeneral

Container for all the general configuration parameters for the MCALLIB

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

16.3.1.1.1 McalLibSafetyEnable

Table 1059 Specification for McalLibSafetyEnable

Name	McalLibSafetyEnable		
Description	Switch to enable reporting of safety error. True : Safety error reporting is enabled. False: Safety error reporting is disabled. The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.2 Container: CommonPublishedInformation

This container holds all the published information of the Mcal Library.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

16.3.1.2.1 ArMajorVersion

Table 1060 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	Major version number of the AUTOSAR specification on which the implementation is based on		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		

MCALLIB driver**Table 1060 Specification for ArMajorVersion (continued)**

Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.2.2 ArMinorVersion**Table 1061 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	Minor version number of the AUTOSAR specification on which the implementation is based on		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.2.3 ArPatchVersion**Table 1062 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	Patch version number of the AUTOSAR specification on which the implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-

MCALLIB driver**Table 1062 Specification for ArPatchVersion (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.2.4 ModuleId**Table 1063 Specification for ModuleId**

Name	ModuleId		
Description	Module ID of MCALLIB.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	255		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.2.5 Release**Table 1064 Specification for Release**

Name	Release		
Description	Specifies the derivative for which the configuration project is created.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per hardware derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCALLIB driver**16.3.1.2.6 SwMajorVersion****Table 1065 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Specifies the major version of the driver software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.2.7 SwMinorVersion**Table 1066 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Specifies the minor version of the driver software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.2.8 SwPatchVersion**Table 1067 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Specifies the patch version of the driver software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		

MCALLIB driver**Table 1067 Specification for SwPatchVersion (continued)**

Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.2.9 VendorId**Table 1068 Specification for VendorId**

Name	VendorId		
Description	Vendor ID for Infineon.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.3 Container: McalLib

This is the parent container for all configuration parameters of MCALLIB.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

16.3.1.4 Container: McalLibPublishedInformation

Container for all the published information of MCALLIB.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

MCALLIB driver**16.3.1.4.1 McalLibBackUpClockFrequency****Table 1069 Specification for McalLibBackUpClockFrequency**

Name	McalLibBackUpClockFrequency		
Description	Specifies the frequency of the back-up clock.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	100 - 100		
Default value	100		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.2 McalLibDsprCore0EndAddr**Table 1070 Specification for McalLibDsprCore0EndAddr**

Name	McalLibDsprCore0EndAddr		
Description	Specifies the end address of DSPR for Core 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x700FFFFF - 0x700FFFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.3 McalLibDsprCore0StartAddr**Table 1071 Specification for McalLibDsprCore0StartAddr**

Name	McalLibDsprCore0StartAddr		
Description	Specifies the start address of DSPR for core 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x70000000 - 0x70000000		

MCALLIB driver**Table 1071 Specification for McalLibDsprCore0StartAddr (continued)**

Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.4 McalLibDsprCore1EndAddr**Table 1072 Specification for McalLibDsprCore1EndAddr**

Name	McalLibDsprCore1EndAddr		
Description	Specifies the end address of DSPR for Core 1. If Core 1 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0x600FFFFF - 0x600FFFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.5 McalLibDsprCore1StartAddr**Table 1073 Specification for McalLibDsprCore1StartAddr**

Name	McalLibDsprCore1StartAddr		
Description	Specifies the start address of DSPR for Core 1. If Core 1 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0x60000000 - 0x60000000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-

MCALLIB driver**Table 1073 Specification for McalLibDsprCore1StartAddr (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.6 McalLibDsprCore2EndAddr**Table 1074 Specification for McalLibDsprCore2EndAddr**

Name	McalLibDsprCore2EndAddr		
Description	Specifies the end address of DSPR for Core 2. If Core 2 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0x500FFFFF - 0x500FFFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.7 McalLibDsprCore2StartAddr**Table 1075 Specification for McalLibDsprCore2StartAddr**

Name	McalLibDsprCore2StartAddr		
Description	Specifies the start address of DSPR for Core 2. If Core 2 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0x50000000 - 0x50000000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCALLIB driver**Table 1075 Specification for McalLibDsprCore2StartAddr (continued)**

Dependency	-
-------------------	---

16.3.1.4.8 McalLibDsprCore3EndAddr**Table 1076 Specification for McalLibDsprCore3EndAddr**

Name	McalLibDsprCore3EndAddr		
Description	Specifies the end address of DSPR for Core 3. If Core 3 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x400FFFFF - 0x400FFFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.9 McalLibDsprCore3StartAddr**Table 1077 Specification for McalLibDsprCore3StartAddr**

Name	McalLibDsprCore3StartAddr		
Description	Specifies the start address of DSPR for Core 3. If Core 3 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x40000000 - 0x40000000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCALLIB driver**16.3.1.4.10 McalLibDsprCore4EndAddr****Table 1078 Specification for McalLibDsprCore4EndAddr**

Name	McalLibDsprCore4EndAddr		
Description	Specifies the end address of DSPR for Core 4. If Core 4 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x300FFFFF - 0x300FFFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.11 McalLibDsprCore4StartAddr**Table 1079 Specification for McalLibDsprCore4StartAddr**

Name	McalLibDsprCore4StartAddr		
Description	Specifies the start address of DSPR for Core 4. If Core 4 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x30000000 - 0x30000000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.12 McalLibDsprCore5EndAddr**Table 1080 Specification for McalLibDsprCore5EndAddr**

Name	McalLibDsprCore5EndAddr
Description	Specifies the end address of DSPR for Core 5.

MCALLIB driver**Table 1080 Specification for McalLibDsprCore5EndAddr (continued)**

	If Core 5 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0x100FFFFF - 0x100FFFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.13 McalLibDsprCore5StartAddr**Table 1081 Specification for McalLibDsprCore5StartAddr**

Name	McalLibDsprCore5StartAddr		
Description	Specifies the start address of DSPR for Core 5. If Core 5 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0x10000000 - 0x10000000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.14 McalLibMcalAvailableCores**Table 1082 Specification for McalLibMcalAvailableCores**

Name	McalLibMcalAvailableCores		
Description	Specifies the number of cores available for the selected device.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	1 - 6		
Default value	Depends on device		

MCALLIB driver**Table 1082 Specification for McalLibMcalAvailableCores (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.15 McalLibPsprCore0EndAddr**Table 1083 Specification for McalLibPsprCore0EndAddr**

Name	McalLibPsprCore0EndAddr		
Description	Specifies the end address of PSPR for Core 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x7010FFFF - 0x7010FFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.16 McalLibPsprCore0StartAddr**Table 1084 Specification for McalLibPsprCore0StartAddr**

Name	McalLibPsprCore0StartAddr		
Description	Specifies the start address of PSPR for core 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x70100000 - 0x70100000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCALLIB driver**Table 1084 Specification for McalLibPsprCore0StartAddr (continued)**

Dependency	-
-------------------	---

16.3.1.4.17 McalLibPsprCore1EndAddr**Table 1085 Specification for McalLibPsprCore1EndAddr**

Name	McalLibPsprCore1EndAddr		
Description	Specifies the end address of PSPR for Core 1. If Core 1 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x6010FFFF - 0x6010FFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.18 McalLibPsprCore1StartAddr**Table 1086 Specification for McalLibPsprCore1StartAddr**

Name	McalLibPsprCore1StartAddr		
Description	Specifies the start address of PSPR for Core 1. If Core 1 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x60100000 - 0x60100000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCALLIB driver**16.3.1.4.19 McalLibPsprCore2EndAddr****Table 1087 Specification for McalLibPsprCore2EndAddr**

Name	McalLibPsprCore2EndAddr		
Description	Specifies the end address of PSPR for Core 2. If Core 2 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0x5010FFFF - 0x5010FFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.20 McalLibPsprCore2StartAddr**Table 1088 Specification for McalLibPsprCore2StartAddr**

Name	McalLibPsprCore2StartAddr		
Description	Specifies the start address of PSPR for Core 2. If Core 2 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0x50100000 - 0x50100000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.21 McalLibPsprCore3EndAddr**Table 1089 Specification for McalLibPsprCore3EndAddr**

Name	McalLibPsprCore3EndAddr		
Description	Specifies the end address of PSPR for Core 3.		

MCALLIB driver**Table 1089 Specification for McalLibPsprCore3EndAddr (continued)**

	If Core 3 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0x4010FFFF - 0x4010FFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.22 McalLibPsprCore3StartAddr**Table 1090 Specification for McalLibPsprCore3StartAddr**

Name	McalLibPsprCore3StartAddr		
Description	Specifies the start address of PSPR for Core 3. If Core 3 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0x40100000 - 0x40100000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.23 McalLibPsprCore4EndAddr**Table 1091 Specification for McalLibPsprCore4EndAddr**

Name	McalLibPsprCore4EndAddr		
Description	Specifies the end address of PSPR for Core 4. If Core 4 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0x3010FFFF - 0x3010FFFF		
Default value	Depends on device		

MCALLIB driver**Table 1091 Specification for McalLibPsprCore4EndAddr (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.24 McalLibPsprCore4StartAddr**Table 1092 Specification for McalLibPsprCore4StartAddr**

Name	McalLibPsprCore4StartAddr		
Description	Specifies the start address of PSPR for Core 4. If Core 4 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x30100000 - 0x30100000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.25 McalLibPsprCore5EndAddr**Table 1093 Specification for McalLibPsprCore5EndAddr**

Name	McalLibPsprCore5EndAddr		
Description	Specifies the end address of PSPR for Core 5. If Core 5 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x1010FFFF - 0x1010FFFF		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-

MCALLIB driver**Table 1093 Specification for McalLibPsprCore5EndAddr (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.1.4.26 McalLibPsprCore5StartAddr**Table 1094 Specification for McalLibPsprCore5StartAddr**

Name	McalLibPsprCore5StartAddr		
Description	Specifies the start address of PSPR for Core 5. If Core 5 does not exist for the selected device, then the parameter holds a value 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0x10100000 - 0x10100000		
Default value	Depends on device		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

16.3.2 Functions - Type definitions**16.3.2.1 unsigned_int****Table 1095 Specification for unsigned_int**

Syntax	unsigned_int
Type	unsigned int
File	Mcal_Compiler.h
Range	32 bit
Description	This data type is used for defining structure members that are bit fields. Rationale: As per AUTOSAR, all primitive data types needs to have compiler abstraction
Source	IFX

MCALLIB driver**16.3.3 Functions - APIs****16.3.3.1 Mcal_WriteSafetyEndInitProtReg16****Table 1096 Specification for Mcal_WriteSafetyEndInitProtReg16 API**

Syntax	<pre>void Mcal_WriteSafetyEndInitProtReg16 (void * const RegAddress, const uint16 DataValue)</pre>	
Service ID	0x81	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	WithValue	Value to be written to the register located at RegAddress.
Parameters (out)	RegAddress	Safety Endinit protected register address having 16 bit access Note: The pointer will be pointer to volatile since the address passed is of a register.
Parameters (in - out)	-	-
Return	void	-
Description	This API unlocks the safety ENDINIT protection, updates the protected register with 16 bit accesses and then locks back the safety ENDINIT protection. The API writes the value specified in 'WithValue' into the safety ENDINIT protected register, whose address is specified in 'RegAddress'.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCALLIB_E_PARAM_POINTER: This error code is reported if the API is invoked with a null pointer as a parameter. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

MCALLIB driver

16.3.3.2 **Mcal_WriteSafetyEndInitProtRegMask**

Table 1097 Specification for Mcal_WriteSafetyEndInitProtRegMask API

Syntax	<pre>void Mcal_WriteSafetyEndInitProtRegMask (void * const RegAddress, const uint32 DataValue, const uint32 Mask)</pre>	
Service ID	0x8F	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	DataValue Mask	Value to be written to the register located at RegAddress. Mask value for updating the registers. Bits set as 1 in the mask, will be updated in 'RegAddress', all the other bits are unchanged.
Parameters (out)	RegAddress	Address for the safety ENDINIT protected register. Note: The pointer will be pointer to volatile since the address passed is of a register.
Parameters (in - out)	-	-
Return	void	-
Description	The function updates the safety ENDINIT protected register, for which the address is specified in 'RegAddress'. The register is updated with the corresponding data value for the bit position where the mask value is 1. The remaining bits retain their original value. If register address is null pointer, then a safety error is reported. This API disable safety ENDINIT protection, updates the protected register and then enable safety ENDINIT protection.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCALLIB_E_PARAM_POINTER: This error code is reported if the API is invoked with a null pointer as a parameter. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

MCALLIB driver**16.3.3.3 McalLib_GetVersionInfo****Table 1098 Specification for McalLib_GetVersionInfo API**

Syntax	<pre>void McalLib_GetVersionInfo (Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x79	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to store the version information of MCALLIB.
Parameters (in - out)	-	-
Return	void	-
Description	The API returns the version information of MCALLIB.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCALLIB_E_PARAM_POINTER: This error code is reported if the API is invoked with a null pointer as a parameter. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

16.3.3.4 Mcal_GetCpuIndex**Table 1099 Specification for Mcal_GetCpuIndex API**

Syntax	<pre>uint32 Mcal_GetCpuIndex (void)</pre>
Service ID	0x89
Sync/Async	Synchronous
ASIL Level	B

MCALLIB driver**Table 1099 Specification for Mcal_GetCpuIndex API (continued)**

Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Index of the core on which this API is called
Description	Retrieves the index of the core on which this API is invoked. Note: For CPU5, although the actual core ID is 6, the API reports the index as 5. This maintains continuity of index from CPU0 to CPU5.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

16.3.3.5 Mcal_GetCpuPhysicalId**Table 1100 Specification for Mcal_GetCpuPhysicalId API**

Syntax	<pre>uint32 Mcal_GetCpuPhysicalId (void)</pre>	
Service ID	0x8B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Identification number of the core.

MCALLIB driver**Table 1100 Specification for Mcal_GetCpuPhysicalId API (continued)**

Description	Retrieves the identification number of the core on which this API is invoked. Note: For CPU0 to CPU4, the identification number of the core is 0 to 4 respectively. For CPU-5, the identification number of the core is 6.
Source	IFX
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

16.3.3.6 Mcal_DelayGetTick**Table 1101 Specification for Mcal_DelayGetTick API**

Syntax	uint32 Mcal_DelayGetTick (void)	
Service ID	0x8A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Lowest 32 bits of STM0.TIM0.
Description	Retrieves the current value of the lowest 32-bits of the register STM0.TIM0.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

MCALLIB driver**Table 1101 Specification for Mcal_DelayGetTick API (continued)**

Configuration dependencies	-
User hints	None

16.3.3.7 Mcal_DelayResetTickCalibration**Table 1102 Specification for Mcal_DelayResetTickCalibration API**

Syntax	uint32 Mcal_DelayResetTickCalibration (void)	
Service ID	0x86	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	System timer (STM) resolution: Value of 1 STM tick in nano seconds.
Description	<p>This API must be invoked by MCU driver to indicate to MCALLIB, that the clock tree has updated. As a result of invocation of this service, MCALLIB:</p> <ul style="list-style-type: none"> - Calculates the STM resolution based on the new clock tree. - Old STM resolution is updated with the newly calculated value within the library. <p>Note: The service is expected to be invoked only by the MCU driver, which is responsible for configuring the clock tree.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>MCALLIB_E_CLKDISABLE: This error code is reported if the STM clock divider is zero and the returned STM resolution is zero.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	

MCALLIB driver**Table 1102 Specification for Mcal_DelayResetTickCalibration API (continued)**

User hints	The MCU clock tree should be initialized prior calling the API. The API is allowed to be called only by the MCAL MCU driver. NOTE: 1. In the flowchart, the value of Ndiv,Pdiv,K2 div are NDIV+1,PDIV+1(from SYSPLLCON0 register) and K2DIV+1(from SYSPLLCON1 register respectively) 2. In the flowchart, the value of TIMER_RESOL_1_NANOSEC is 10^9, which is used to return STM timer resolution in 1ns resolution.
-------------------	--

16.3.3.8 Mcal_DelayTickResolution**Table 1103 Specification for Mcal_DelayTickResolution API**

Syntax	uint32 Mcal_DelayTickResolution (void)	
Service ID	0x8C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	System timer(STM) resolution: Value of 1 STM tick in nanosecond.
Description	Retrieve the resolution of a STM in nanosecond. Note: A return value of 0 indicates, STM is switched off or Mcal_DelayResetTickCalibration API was never invoked.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

MCALLIB driver**16.3.3.9 Mcal_GetBitAtomic****Table 1104 Specification for Mcal_GetBitAtomic API**

Syntax	<pre>uint32 Mcal_GetBitAtomic (const uint32 DataValue, const uint8 BitPos, const uint8 BitLen)</pre>	
Service ID	NA	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	NA	
Parameters (in)	DataValue BitPos BitLen	Value of the variable/register from which bits needs to extracted. Starting bit position of the data to be extracted Bit length of the data to be extracted
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Bits extracted from 'DataValue'
Description	<p>The API extracts bits of data from the 32-bit value. The start position and length of the data to be extracted is specified by BitPos and BitLen respectively.</p> <p>Note: This function is implemented as a macro.</p>	
Source	IFX	
Error handling	<p>DET: None Runtime Errors: None DEM: None Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

16.3.3.10 Mcal_SetBitAtomic**Table 1105 Specification for Mcal_SetBitAtomic API**

Syntax	<pre>void Mcal_SetBitAtomic (uint32 * const DataPtr, const uint8 BitPos, const uint8 BitLen,</pre>
---------------	---

MCALLIB driver**Table 1105 Specification for Mcal_SetBitAtomic API (continued)**

	const uint32 Data)	
Service ID	NA	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	NA	
Parameters (in)	DataPtr BitPos BitLen Data	Variable or register address to be updated. Starting bit position of the data to be modified. Bit length of the data to be modified Value to be updated to address pointed by DataPtr
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API stores 'Data' at the address location pointed by 'DataPtr' atomically. The start position and length of the data to be updated is specified by 'BitPos' and 'BitLen' respectively. Only the bits specified by BitPos and BitLen are updated, all the other bits are unchanged. Note: This function is implemented as a macro.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

16.3.3.11 Mcal_GetGlobalDsprAddress**Table 1106 Specification for Mcal_GetGlobalDsprAddress API**

Syntax	uint32 Mcal_GetGlobalDsprAddress (const uint32 CpuId, const uint32 LocalDsprAddress)
Service ID	0x7B
Sync/Async	Synchronous

MCALLIB driver**Table 1106 Specification for Mcal_GetGlobalDsprAddress API (continued)**

ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Cpuld LocalDsprAddress	Physical CPU Core ID Note: For CPU5 the physical core ID is 6. Local DSPR address for which the global DSPR address is to be returned
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	<ul style="list-style-type: none"> - If passed parameter is a valid core ID and local DSPR address, then routine return global DSPR address. - If passed parameter is valid global DSPR address corresponding to passed Cpuld then routine return passed address as is. - If passed parameter(Cpuld or LocalDsprAddress or both) is invalid then routine return value 0
Description	Returns the global address of a local DSPR address of a specified CPU.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None.	

16.3.3.12 Mcal_GetGlobalPsprAddress**Table 1107 Specification for Mcal_GetGlobalPsprAddress API**

Syntax	<pre>uint32 Mcal_GetGlobalPsprAddress (const uint32 CpuId, const uint32 LocalPsprAddress)</pre>
Service ID	0x7D
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant

MCALLIB driver**Table 1107 Specification for Mcal_GetGlobalPsprAddress API (continued)**

Parameters (in)	Cpuld LocalPsprAddress	Physical Core ID Local PSPR address for which global PSPR address is to be returned
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	<ul style="list-style-type: none"> - If passed parameter is a valid core ID and local PSPR address, then routine return global PSPR address. - If passed parameter is valid global PSPR address corresponding to the passed Cpuld then routine returns the passed address as is. - If passed parameter(Cpuld or LocalPsprAddress or both) is invalid then routine return a value of 0.
Description	Returns the global address of a local PSPR address of a specified CPU	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None.	

16.3.3.13 Mcal_GetLocalDsprAddress**Table 1108 Specification for Mcal_GetLocalDsprAddress API**

Syntax	<pre>uint32 Mcal_GetLocalDsprAddress (const uint32 GlobalDsprAddress)</pre>	
Service ID	0x83	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	GlobalDsprAddress	Global DSPR address
Parameters (out)	-	-

MCALLIB driver**Table 1108 Specification for Mcal_GetLocalDsprAddress API (continued)**

Parameters (in - out)	-	-
Return	uint32	<ul style="list-style-type: none"> - If passed parameter is a valid global DSPR address, then routine return local DSPR address. - If passed parameter is valid local DSPR address corresponding to currently executing CPU then routine returns the passed address as is. - If passed parameter is an invalid address then routine return a value of 0.
Description	Returns the local DSPR address for a global DSPR address.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None.	

16.3.3.14 Mcal_GetLocalPsprAddress**Table 1109 Specification for Mcal_GetLocalPsprAddress API**

Syntax	<pre>uint32 Mcal_GetLocalPsprAddress (const uint32 GlobalPsprAddress)</pre>	
Service ID	0x84	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	GlobalPsprAddress	Global PSPR address
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	<ul style="list-style-type: none"> - If passed parameter is a valid global PSPR address, then routine return local PSPR address

MCALLIB driver**Table 1109 Specification for Mcal_GetLocalPsprAddress API (continued)**

	<ul style="list-style-type: none"> - If passed parameter is valid local PSPR address corresponding to currently executing CPU then routine returns the passed address as is. - If passed parameter is an invalid address then routine return a value of 0.
Description	Returns the local PSPR address for a global PSPR address
Source	IFX
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	None.

16.3.3.15 Mcal_GetPeripheralEndInitPassword**Table 1110 Specification for Mcal_GetPeripheralEndInitPassword API**

Syntax	uint32 Mcal_GetPeripheralEndInitPassword (void)	
Service ID	0x82	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Current peripheral ENDINIT password.
Description	<p>Retrieves the peripheral ENDINIT password installed in the EPW bitfield of EICON0 register.</p> <p><i>Note: The API reads the current password stored in EICON.EPW, and inverts the bits 0 to 5 of the password before reporting.</i></p>	
Source	IFX	
Error handling	DET: None	

MCALLIB driver**Table 1110 Specification for Mcal_GetPeripheralEndInitPassword API (continued)**

	<p>Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

16.3.3.16 Mcal_GetCpuWdgPassword**Table 1111 Specification for Mcal_GetCpuWdgPassword API**

Syntax	uint32 Mcal_GetCpuWdgPassword (void)	
Service ID	0x88	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Currently installed password for the CPU watchdog.
Description	<p>Retrieves the ENDINIT password for the watchdog of the CPU on which the API is invoked.</p> <p><i>Note: The API reads the current password stored in CON0.PW, and inverts the bits 0 to 5 of the password before reporting.</i></p>	
Source	IFX	
Error handling	<p>DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

MCALLIB driver**16.3.3.17 Mcal_GetSafetyEndInitPassword****Table 1112 Specification for Mcal_GetSafetyEndInitPassword API**

Syntax	<pre>uint32 Mcal_GetSafetyEndInitPassword (void)</pre>	
Service ID	0x87	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Currently installed safety ENDINIT password.
Description	<p>Retrieves the safety ENDINIT password installed in the EPW bit field of SEICON0 register.</p> <p>Note: The API reads the current password stored in SEICON0.EPW and inverts the bits 0 to 5 of the password before reporting..</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

16.3.3.18 Mcal_WriteSafetyEndInitProtReg**Table 1113 Specification for Mcal_WriteSafetyEndInitProtReg API**

Syntax	<pre>void Mcal_WriteSafetyEndInitProtReg (void * const RegAddress, const uint32 DataValue)</pre>
Service ID	0x7F
Sync/Async	Synchronous

MCALLIB driver**Table 1113 Specification for Mcal_WriteSafetyEndInitProtReg API (continued)**

ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	DataValue	Value to be written to the register located at RegAddress.
Parameters (out)	RegAddress	Address for the safety ENDINIT protected register. Note: The pointer will be pointer to volatile since the address passed is of a register.
Parameters (in - out)	-	-
Return	void	-
Description	This API unlocks the safety ENDINIT protection, updates the protected register and then locks back the safety ENDINIT protection. The API writes the value specified in 'DataValue' into the safety ENDINIT protected register, whose address is specified in 'RegAddress'.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCALLIB_E_PARAM_POINTER: This error code is reported if the API is invoked with a null pointer as a parameter. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

16.3.3.19 Mcal_SetCpuWdgPassword**Table 1114 Specification for Mcal_SetCpuWdgPassword API**

Syntax	<pre>uint32 Mcal_SetCpuWdgPassword (const uint32 Password)</pre>	
Service ID	0x85	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant on same CPU, Reentrant for other CPUs	
Parameters (in)	Password	New password to be installed for CPU ENDINIT protection

MCALLIB driver**Table 1114 Specification for Mcal_SetCpuWdgPassword API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Previously installed password
Description	Installs a new ENDINIT password or the watchdog of the CPU on which the API is invoked. The interface internally prepares the password (both for static and automatic password sequencing), installs the password and returns the previously installed password. Note: Bits 0 to 5 of the previously installed password is inverted before reporting.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

16.3.3.20 Mcal_SetPeripheralEndInitPassword**Table 1115 Specification for Mcal_SetPeripheralEndInitPassword API**

Syntax	uint32 Mcal_SetPeripheralEndInitPassword (const uint32 Password)	
Service ID	0x7C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Password	New password to be installed for peripheral ENDINIT.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Previously installed password
Description	Installs a new peripheral ENDINIT password. The interface internally prepares the password, installs the password and returns the previously installed password.	

MCALLIB driver**Table 1115 Specification for Mcal_SetPeripheralEndInitPassword API (continued)**

	Note: Bits 0 to 5 of the previously installed password is inverted before reporting.
Source	IFX
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

16.3.3.21 Mcal_SetSafetyEndInitPassword**Table 1116 Specification for Mcal_SetSafetyEndInitPassword API**

Syntax	<pre>uint32 Mcal_SetSafetyEndInitPassword (const uint32 Password)</pre>	
Service ID	0x80	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Password	New password to be installed for safety ENDINIT protection
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Previously installed password
Description	Installs a new safety ENDINIT password. The interface internally prepares the password, installs the password and returns the previously installed password. <i>Note: Bits 0 to 5 of the previously installed password is inverted before reporting.</i>	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

MCALLIB driver**Table 1116 Specification for Mcal_SetSafetyEndInitPassword API (continued)**

Configuration dependencies	-
User hints	None

16.3.3.22 Mcal_GetSpinlock**Table 1117 Specification for Mcal_GetSpinlock API**

Syntax	<pre>void Mcal_GetSpinlock (volatile uint32 * const LockAddress, const uint32 Timeout)</pre>	
Service ID	0x8D	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	LockAddress Timeout	Address of the spinlock to be acquired. Maximum wait time(micro second) to acquire the spinlock.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API acquires the passed spinlock atomically. It is implemented in MCALLIB using Test, Test and Set Spinlock(TTAS)mechanism. A Timeout shall be passed as input parameter to spinlock API so that TTAS does not enter into an indefinite loop. If spinlock is not acquired within the specified timeout, then the control returns to the application after reporting a safety error.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCALLIB_E_TIMEOUT_FAILED: This error code is reported if the spinlock could not be acquired in the specified timeout. MCALLIB_E_PARAM_POINTER: This error code is reported if the API is invoked with a null pointer as a parameter. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	

MCALLIB driver**Table 1117 Specification for Mcal_GetSpinlock API (continued)**

User hints	User shall ensure that when this interface is used the McalLibSafetyEnable parameter shall be enabled to detect timeout.
-------------------	--

16.3.3.23 Mcal_ReleaseSpinlock**Table 1118 Specification for Mcal_ReleaseSpinlock API**

Syntax	<pre>void Mcal_ReleaseSpinlock (volatile uint32 * const LockAddress)</pre>	
Service ID	0x8E	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	LockAddress	Address of the spinlock to be released.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Releases the spinlock pointed by the lock address.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCALLIB_E_PARAM_POINTER: This error code is reported if the API is invoked with a null pointer as a parameter. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

16.3.3.24 Mcal_WriteCpuEndInitProtReg**Table 1119 Specification for Mcal_WriteCpuEndInitProtReg API**

Syntax	<pre>void Mcal_WriteCpuEndInitProtReg (</pre>
---------------	---

MCALLIB driver**Table 1119 Specification for Mcal_WriteCpuEndInitProtReg API (continued)**

	<pre> void * const RegAddress, const uint32 DataValue) </pre>	
Service ID	0x7E	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	WithValue	Value to be written to the register located at RegAddress.
Parameters (out)	RegAddress	Address of the CPU ENDINIT protected register. Note: The pointer will be pointer to volatile since the address passed is of a register.
Parameters (in - out)	-	-
Return	void	-
Description	This API unlocks the CPU ENDINIT protection, updates the protected register and then locks back the CPU ENDINIT protection. The API writes the value specified in 'WithValue' into the CPU ENDINIT protected register, whose address is specified though 'RegAddress'.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCALLIB_E_PARAM_POINTER: This error code is reported if the API is invoked with a null pointer as a parameter. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

16.3.3.25 Mcal_WritePeripEndInitProtReg**Table 1120 Specification for Mcal_WritePeripEndInitProtReg API**

Syntax	<pre> void Mcal_WritePeripEndInitProtReg (void * const RegAddress, const uint32 DataValue) </pre>
Service ID	0x7A
Sync/Async	Synchronous

MCALLIB driver
Table 1120 Specification for `Mcal_WritePeripEndInitProtReg` API (continued)

ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	DataValue	Value to be written to the register located at RegAddress..
Parameters (out)	RegAddress	Address of the peripheral ENDINIT protected register. Note: The pointer will be pointer to volatile since the address passed is of a register.
Parameters (in - out)	-	-
Return	void	-
Description	This API unlocks the peripheral ENDINIT protection, updates the protected register and then locks back the peripheral ENDINIT protection. The API writes the value specified in 'DataValue' into the peripheral ENDINIT protected register, whose address is specified though 'RegAddress'.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCALLIB_E_PARAM_POINTER: This error code is reported if the API is invoked with a null pointer as a parameter. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

16.3.4 Notifications and Callbacks

Not available for the driver.

16.3.5 Scheduled functions

Not available for the driver.

16.3.6 Interrupt service routines

Not available for the driver.

16.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

MCALLIB driver

16.3.7.1 Development errors

The driver does not report any development errors.

16.3.7.2 Production errors

The driver does not report any production errors.

16.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 1121 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
This error code is reported if the STM clock divider is zero and the returned STM resolution is zero.	IFX	MCALLIB_E_CLKDISABLE=0xD0	Mcal_DelayResetTickCalibration
This error code is reported if the API is invoked with a null pointer as a parameter.	IFX	MCALLIB_E_PARAM_POINTER=0xC9U	Mcal_WriteSafetyEndInitProtReg16, Mcal_GetSpinlock, Mcal_WriteSafetyEndInitProtReg, Mcal_WriteCpuEndInitProtReg, Mcal_WritePeripEndInitProtReg, Mcal_WriteSafetyEndInitProtRegMask, Mcal_ReleaseSpinlock, McalLib_GetVersionInfo
This error code is reported if the spinlock could not be acquired in the specified timeout.	IFX	MCALLIB_E_TIMEOUT_FAILED=0xCCU	Mcal_GetSpinlock

16.3.7.4 Runtime errors

The driver does not report any runtime errors.

16.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

16.3.8.1 Deviations and Assumptions

There are no deviations in this driver.

MCALLIB driver

16.3.8.2 Limitations

This section describes the limitation from software specification

Table 1122 Known Limitation

Reference	Limitation
STM timer resolution	When the STM clock divider is zero, the resolution calculated in the <code>Mcal_DelayResetTickCalibration</code> API is zero. User must ensure that the value of CCUCONO.STMDIV is not zero before using this MCALLIB API.

16.3.9 Unsupported hardware features

There are no hardware features that are unsupported by this driver.

MCU driver**17 MCU driver****17.1 User information****17.1.1 Description**

The MCU driver is responsible for configuring the SCU, GTM, CCU6, GPT12 and STM peripherals. The driver provides runtime services specified by AUTOSAR. The MCU driver is responsible for the following:

- Configuration of Clock, Reset and static low power mode functionalities as specified by AUTOSAR
- Configuration of Trap functionality
- Configuration of global features of GTM, CCU6 and GPT12 required by the BASIC drivers
- Provide library support for other drivers for timer IPs - GTM, CCU6, GPT12 and STM
- Configuration of phase synchronizer necessary for analog converters
- Runtime APIs requested by AUTOSAR for clock, reset, low power management and RAM initialization
- Runtime APIs for Trap management

Additionally, the MCU driver provides a centralized hardware resource reservation mechanism to the configurator for conflict-free allocation to the MCAL drivers. The resources capable of being reserved are CCU6 modules, GTM timer slices, ASCLIN slices, ERU slices and STM comparators. The MCU driver is delivered as a Post-Build variant. Post-Build architecture guarantees the ability to generate an independent HEX file for configuration alone.

17.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

MCU driver

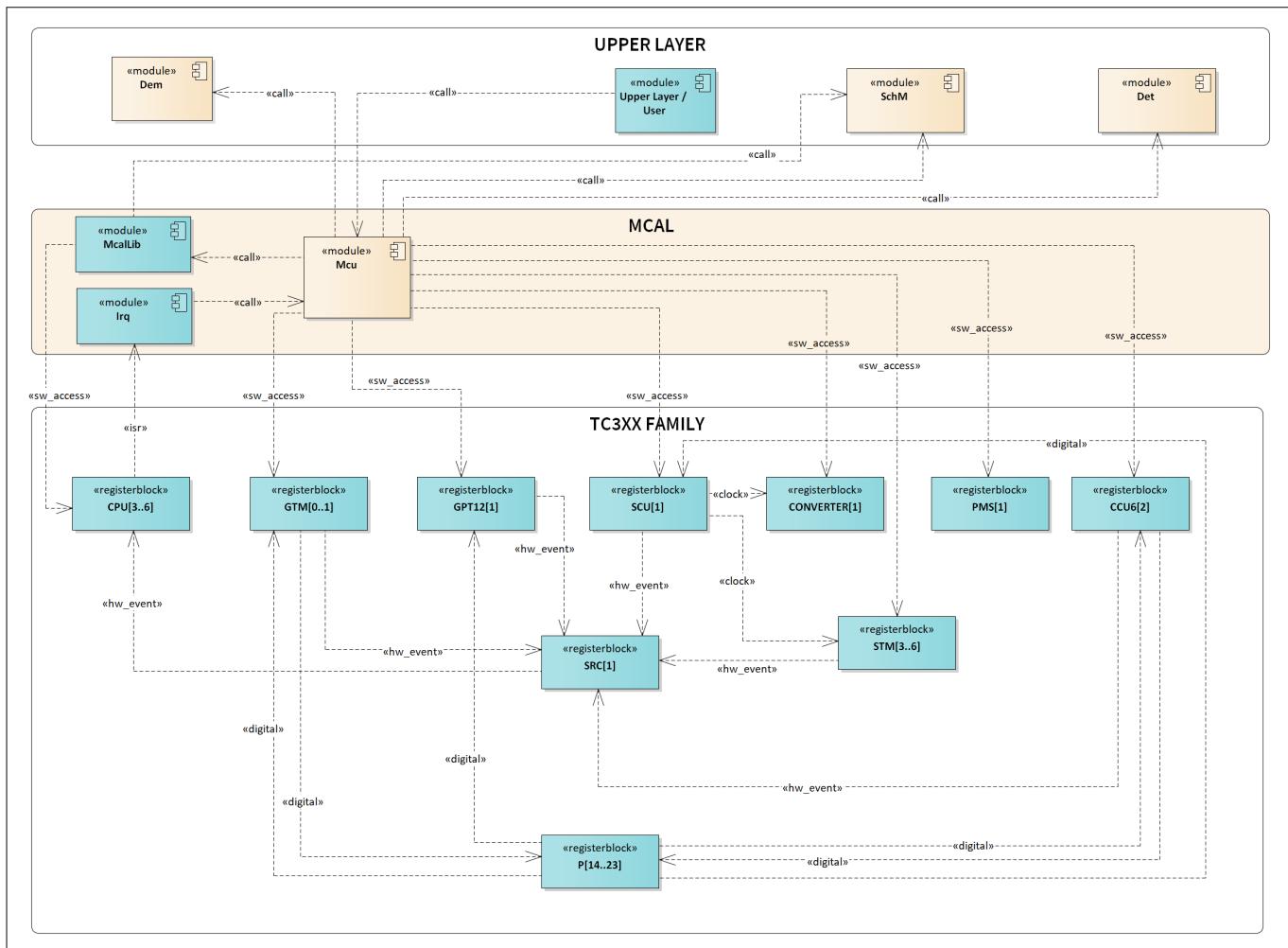


Figure 155 **Mapping of hardware-software interfaces**

17.1.2.1 GTM: primary hardware peripheral

Hardware functional features

The MCU driver only provides configuration interfaces for the GTM IP.

During the initialization the driver is responsible for configuring the global blocks of GTM (CMU, CCM, TBU). The channel specific SFRs are configured at run time (by other drivers).

Users of the hardware

The global functional blocks of GTM are centrally administered by the MCU driver.

The MCU driver provides APIs to program the GTM (TOM, ATOM, TIM) channel SFRs. The PWM, GPT, ADC, DSADC, WDG, OCU and ICU drivers use these APIs to initialize, de-initialize, enable and disable channels.

Additionally, updates to the channel specific SFRs are performed by the MCAL drivers also. Since the channels are exclusively reserved for each driver, access to the channel specific SFRs by the reserving driver is allowed.

Hardware diagnostic features

The SMU alarms configured for the GTM IP are not monitored by the MCU driver.

Hardware events

The hardware event for each channel is enabled based on the user configuration. The MCU driver invokes the callback function provided as interrupt handler by each driver on a hardware event.

MCU driver

17.1.2.2 SCU: primary hardware peripheral

Hardware functional features

The MCU driver uses the SCU IP for the following:

- Configuring the clock tree
- Reset control
- Trap setting
- Power-mode control and transitions
- Configuration of ERU for pattern detection and output gating control

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the MCU driver.

Hardware events

The hardware event for ERU channels is enabled based on the user configuration. The MCU driver invokes the call back function provided as interrupt handler by the ICU and DSADC driver on a hardware event.

17.1.2.3 CONVERTER: primary hardware peripheral

Hardware functional features

The MCU driver configures the convertor control block for providing a clock enable signal to synchronize the clock signals of all analog blocks (EVADC and EDSADC).

Users of the hardware

The phase synchronizer signal is used by the ADC and DSADC drivers, however the configuration for generating the signals is done by the MCU driver.

Hardware diagnostic features

The SMU alarms configured for the convertor control block are not monitored by the MCU driver.

Hardware events

Hardware events from the convertor control block are not used by the MCU driver.

17.1.2.4 CCU6: primary hardware peripheral

Hardware functional features

The MCU driver only provides configuration interfaces for the CCU6 IP. The CCU6 IP is used by other MCAL drivers for various applications.

During the initialization the driver is responsible for enabling the clock for the CCU6 IP. The channel specific SFRs are configured at run time (by other drivers).

Users of the hardware

The MCU driver provides APIs to program the CCU6 SFRs. The PWM and ICU driver use these APIs to initialize, de-initialize, enable and disable channels.

Additionally, updates to the channel specific SFRs are performed by the PWM and ICU drivers. Since the channels are exclusively reserved for each driver, access to the channel specific SFRs by the reserving driver is allowed.

Hardware diagnostic features

Not applicable.

MCU driver
Hardware events

The hardware event for each channel is enabled based on the user configuration. The MCU driver invokes the call back function provided as interrupt handler by each driver on a hardware event.

17.1.2.5 GPT12: primary hardware peripheral

Hardware functional features

The MCU driver only provides configuration interfaces for the GPT12 IP. The GPT12 IP is used by other MCAL drivers for various applications.

During the initialization the driver is responsible for enabling the clock and configuring the block pre-scalers for the GPT12 IP. The channel specific SFRs are configured at run time (by other drivers).

Users of the hardware

The MCU driver provides APIs to program the GPT12 SFRs. The GPT and ICU driver use these APIs to initialize, de-initialize, enable and disable channels.

Additionally, updates to the channel specific SFRs are performed by the GPT and ICU drivers. Since the channels are exclusively reserved for each driver, access to the channel specific SFRs by the reserving driver is allowed.

Hardware diagnostic features

Not applicable.

Hardware events

The hardware event for each channel is enabled based on the user configuration. The MCU driver invokes the call back function provided as interrupt handler by each driver on a hardware event.

17.1.2.6 PMS: primary hardware peripheral

Hardware functional features

The MCU driver uses the PMS IP for changing the active power-mode of the controller. The supported power modes are:

- Normal
- Idle
- Sleep
- Standby

Users of the hardware

The MCU driver exclusively utilizes the PMS IP for power mode management.

Hardware diagnostic features

Not applicable.

Hardware events

The MCU driver configures the wake-up events from the PMS IP.

17.1.2.7 STM: primary hardware peripheral

Hardware functional features

The MCU driver only provides configuration interfaces for the STM IP. The STM IP is used by other MCAL drivers for various applications. The compare match SFRs are configured at run time (by other drivers).

Users of the hardware

The MCU driver provides APIs to program the STM SFRs. The WDG and STM driver use these APIs to utilize the compare match feature of the STM IP.

MCU driver

Additionally, updates to the compare register are performed by the WDG and STM drivers. Since the compare registers are exclusively reserved for each driver, access to the compare registers by the reserving driver is allowed.

Hardware diagnostic features

Not applicable.

Hardware events

The hardware event for each channel is enabled based on the user configuration. The MCU driver invokes the call back function provided as interrupt handler by each driver on a hardware event.

17.1.3 File structure

17.1.3.1 C file structure

This section provides details of the C files of the MCU driver.

MCU driver

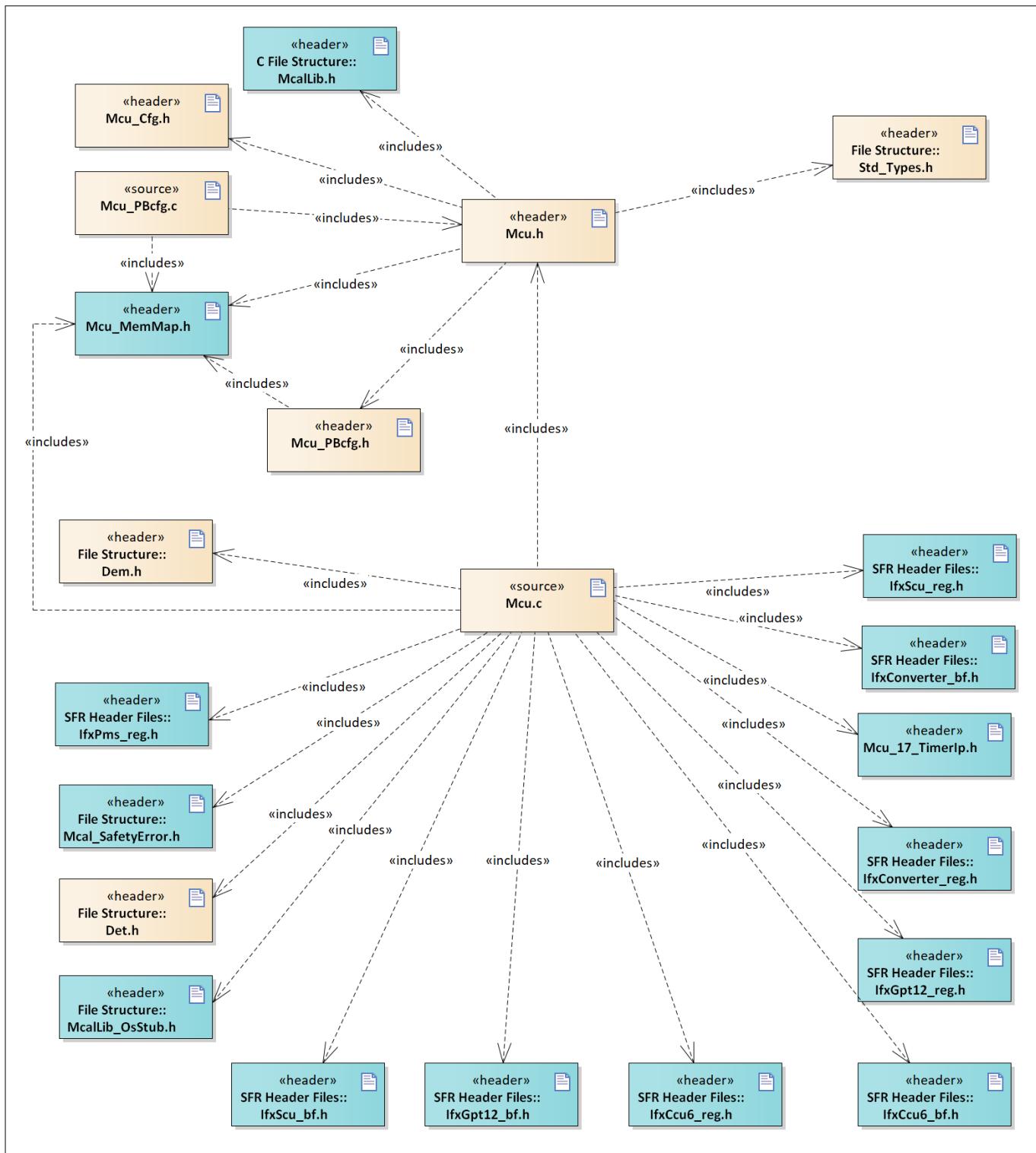
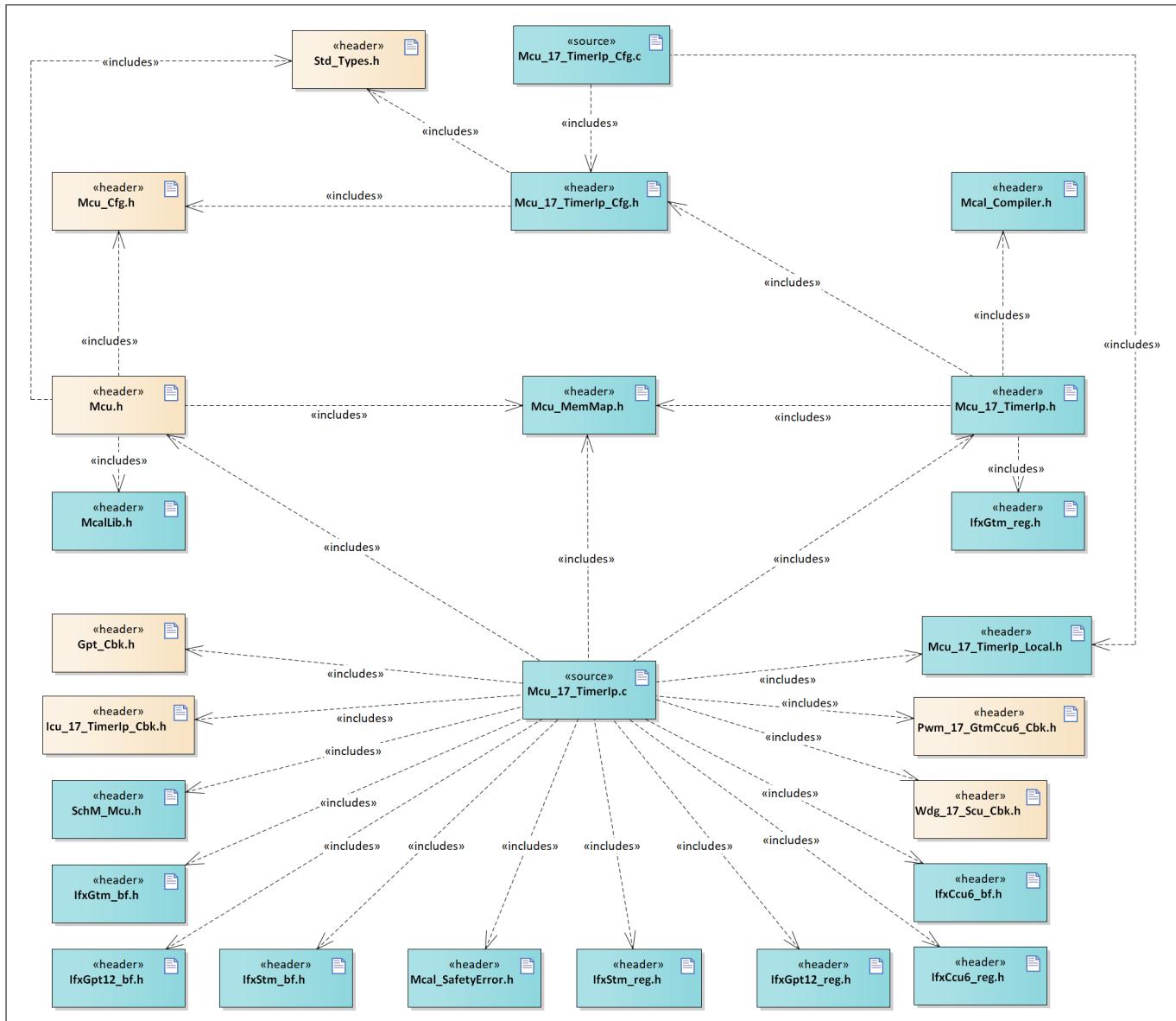


Figure 156

C file structure

MCU driver

Figure 157 C file structure (continued)
Table 1123 C file structure (Mcu.c)

File name	Description
Dem.h	Provides the exported interfaces of Diagnostic Event Manager
Det.h	Provides the exported interfaces of Development Error Tracer
IfxCcu6_bf.h	SFR header file for CCU6
IfxCcu6_reg.h	SFR header file for CCU6
IfxConverter_bf.h	SFR header file for Converter
IfxConverter_reg.h	SFR header file for Converter
IfxGpt12_bf.h	SFR header file for GPT12
IfxGpt12_reg.h	SFR header file for GPT12
IfxPms_reg.h	SFR header file for Pms

MCU driver**Table 1123 C file structure (Mcu.c) (continued)**

File name	Description
IfxScu_bf.h	SFR header file for SCU
IfxScu_reg.h	SFR header file for SCU
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore™. This shall be included by other drivers to call OS APIs.
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcu.c	MCU source file proving implementation of APIs (including AUTOSAR) relating to initialization, clock, power modes, reset, trap, etc
Mcu.h	Header file providing prototypes of APIs and data types. This file exports only necessary interfaces for upper layer
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Mcu_Cfg.h	Generated header file containing macros
Mcu_MemMap.h	File (Static) containing the memory section definitions used by the MCU driver
Mcu_PBcfg.c	Generated header file containing configuration data of the user
Mcu_PBcfg.h	File (Generated) containing declaration of the post-build configuration data structures
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

Table 1124 C file structure (Mcu_17_TimerIp.c)

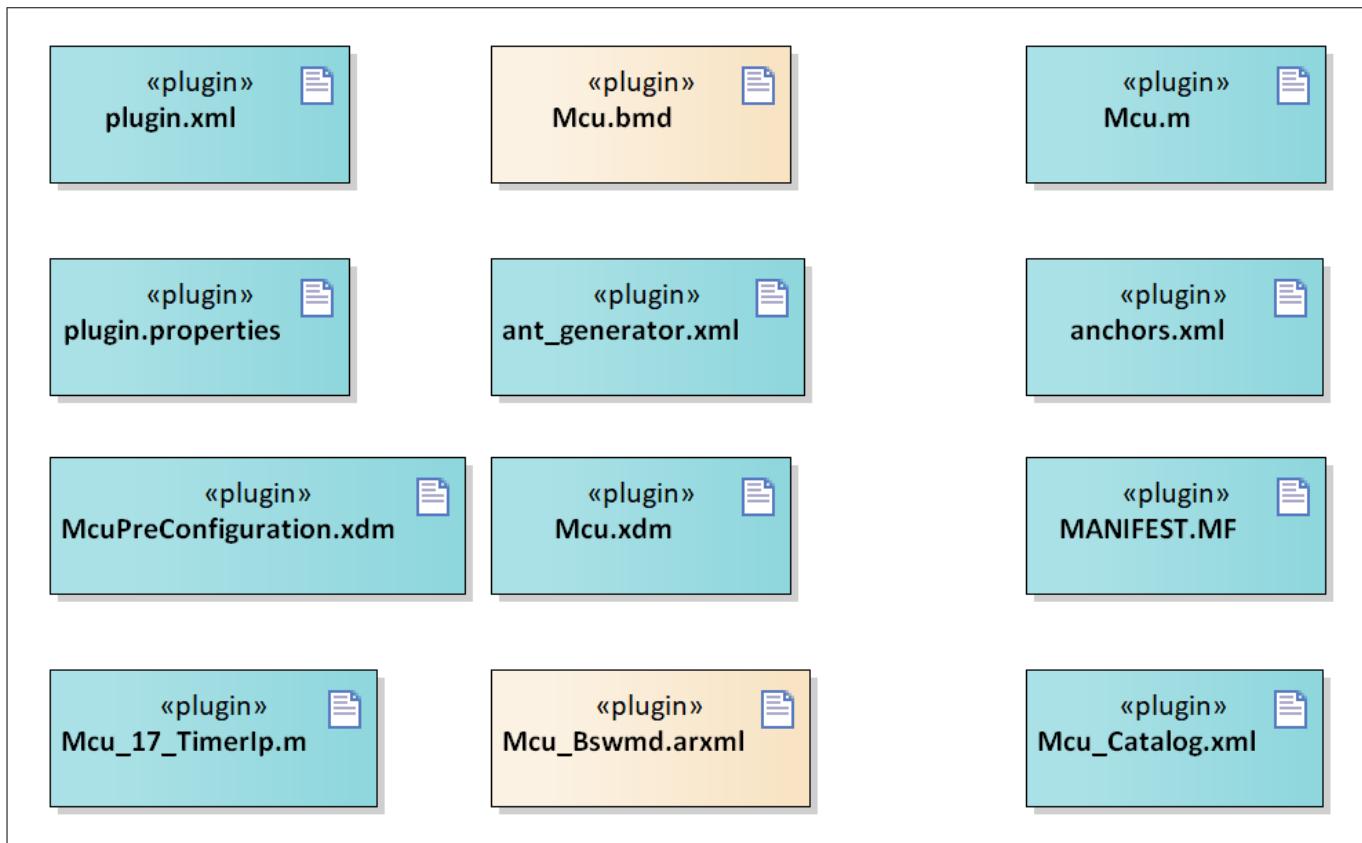
File name	Description
Gpt_Cbk.h	Header file providing prototypes of callback APIs
Icu_17_TimerIp_Cbk.h	Header file to declare the callback APIs
IfxCcu6_bf.h	SFR header file for CCU6
IfxCcu6_reg.h	SFR header file for CCU6
IfxGpt12_bf.h	SFR header file for GPT12
IfxGpt12_reg.h	SFR header file for GPT12
IfxGtm_bf.h	SFR header file for GTM
IfxGtm_reg.h	SFR header file for GTM
IfxStm_bf.h	SFR header file for STM
IfxStm_reg.h	SFR header file for STM
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
Mcal_Compiler.h	Provides abstraction for TriCore™-intrinsic instruction

MCU driver
Table 1124 C file structure (Mcu_17_TimerIp.c) (continued)

File name	Description
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcu.h	Header file providing prototypes of APIs and data types. This file exports only necessary interfaces for upper layer
Mcu_17_TimerIp.c	File (Static) containing implementation of APIs of Timer IPs - GTM, CCU6 and GPT12, initialization, enable, interrupt and other services
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Mcu_17_TimerIp_Cfg.c	Generated source file, which contains the user information for each the Timers - CCU6 , GPT12 and GTM channels
Mcu_17_TimerIp_Cfg.h	Generated header file for Timer IPs APIs
Mcu_17_TimerIp_Local.h	Header file contains declaration of callback data for ERU, CCU6, GPT12, GTM (TIM, TOM, ATOM) and STM
Mcu_Cfg.h	Generated header file containing macros
Mcu_MemMap.h	File (Static) containing the memory section definitions used by the MCU driver
Pwm_17_GtmCcu6_Cbk.h	Includes callback header definition
SchM_Mcu.h	Non-productized file. Contains prototype of SchM_Enter/Exit interfaces needed by Timer APIs
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.
Wdg_17_Scu_Cbk.h	Header file contains call back function of WDG driver

17.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the MCU driver.

MCU driver**Figure 158** **Code generator plugin files****Table 1125** **Code generator plugin files**

File name	Description
MANIFEST.MF	Tresos plugin support file containing metadata for the MCU driver
Mcu.bmd	AUTOSAR format XML data model schema file (for each device)
Mcu.m	Code template macro file for the MCU driver
Mcu.xdm	Tresos format XML data model schema file
McuPreConfiguration.xdm	Tresos format XML data model schema file
Mcu_17_TimerIp.m	Code template macro file for Timer APIs in the MCU driver
Mcu_Bswmd.arxml	AUTOSAR format module description file
Mcu_Catalog.xml	AUTOSAR format catalog file
anchors.xml	Tresos anchors support file for the MCU driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point
plugin.properties	Tresos plugin support file for the MCU driver
plugin.xml	Tresos plugin support file for the MCU driver

MCU driver**17.1.4 Integration hints**

This section lists the key points that an integrator or user of the MCU driver must consider.

17.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the MCU driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete Ecum module during the integration phase.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Mcu_MemMap.h` file.

The `Mcu_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

MCU driver

```

/* Sample implementation of Mcu_MemMap.h */
***** CONFIGURATION DATA *****/
#if defined MCU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/*user pragma here */
#define MCU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR

#elif defined MCU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/*user pragma here */
#define MCU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR

#elif defined MCU_17_TIMERIP_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_16
/*user pragma here */
#define MCU_17_TIMERIP_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_16
#define MEMMAP_ERROR
#endif

#elif defined MCU_17_TIMERIP_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_16
/*user pragma here */
#define MCU_17_TIMERIP_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_16
#define MEMMAP_ERROR
#endif

***** GLOBAL DATA *****/
#if defined MCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*user pragma here */
#define MCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR

#endif

#if defined MCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*user pragma here */
#define MCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR
#endif

#if defined MCU_17_TIMERIP_START_SEC_VAR_INIT_ASIL_B_GLOBAL_32
/*user pragma here */
#define MCU_17_TIMERIP_START_SEC_VAR_INIT_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR
#endif

#if defined MCU_17_TIMERIP_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_32
/*user pragma here */
#define MCU_17_TIMERIP_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR
#endif

***** CONST DATA *****/
#if defined MCU_17_TIMERIP_START_SEC_CONST_ASIL_B_GLOBAL_UNSPECIFIED
/*user pragma here */
#define MCU_17_TIMERIP_START_SEC_CONST_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR

#endif

#if defined MCU_17_TIMERIP_STOP_SEC_CONST_ASIL_B_GLOBAL_UNSPECIFIED
/*user pragma here */
#define MCU_17_TIMERIP_STOP_SEC_CONST_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR
#endif

```

MCU driver

```
#undef MEMMAP_ERROR

***** CODE ****/
#elif defined MCU_START_SEC_CODE_ASIL_B_GLOBAL
/*user pragma here */
#undef MCU_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#elif defined MCU_STOP_SEC_CODE_ASIL_B_GLOBAL
/*user pragma here */
#undef MCU_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#elif defined MCU_17_TIMERIP_START_SEC_CODE_ASIL_B_GLOBAL
/*user pragma here */
#undef MCU_17_TIMERIP_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#elif defined MCU_17_TIMERIP_STOP_SEC_CODE_ASIL_B_GLOBAL
/*user pragma here */
#undef MCU_17_TIMERIP_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error Mcu MemMap file definition is not correct.
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The MCU driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the MCU driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is a part of the AUTOSAR stack that handles all the production errors reported by the BSW modules. The MCU driver reports all the production errors to the DEM modules through the `Dem_ReportErrorStatus()` API. The user of the MCU driver must process all the production errors (fail / pass) reported to the DEM module through the `Dem_ReportErrorStatus()` API.

The `Dem.h` and `Dem.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DEM module during the integration phase.

- **SchM**

The SchM module is a part of the RTE that manages the Basic Software Scheduler. The MCU driver uses the exclusive areas defined in `SchM_Mcu.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the MCU driver are:

- ATOM AGC registers
- TOM TGC registers

MCU driver

The SchM_Mcu.h and SchM_Mcu.c files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the MCU driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows.

```

void SchM_Enter_Mcu_TomTgcReg(void)
{
    SuspendAllInterrupts();
}

void SchM_Exit_Mcu_TomTgcReg(void)
{
    ResumeAllInterrupts();
}

void SchM_Enter_Mcu_AtomAgcReg(void)
{
    SuspendAllInterrupts();
}

void SchM_Exit_Mcu_AtomAgcReg(void)
{
    ResumeAllInterrupts();
}

```

- **Safety error**

The MCU driver will report all the detected safety errors through the Mcal_ReportSafetyError() API. The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The Mcal_ReportSafetyError() API is provided in the Mcal_SafetyError.c and Mcal_SafetyError.h files as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The MCU driver does not provide any callbacks or notifications.

- **Operating system (OS)**

The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application.

The OS files provided by the MCAL package is only an example code and must be updated by the integrator with the actual OS files for the desired function.

17.1.4.2 Multicore and Resource Manager

The MCU driver supports execution of its runtime APIs simultaneously from all CPU cores (initialization APIs are Mcu_Init(), Mcu_InitClock(), Mcu_DistributePllClock() and Mcu_DeInit()). In general, apart from the initialization APIs of MCU driver, other APIs may be invoked from several CPU cores in parallel with some restrictions, which are also described in this section. The following are the key points to be considered with respect to multicore in the driver:

MCU driver

- Initialization APIs `Mcu_Init()`, `Mcu_InitClock()`, `Mcu_DistributePllClock()` and `Mcu_DeInit()` can only be invoked by the master core.
- DETs will be raised in case APIs are invoked with mismatch of core.
- Locating constants, variables and configuration data to correct memory space should be done by the user. Memory sections are marked GLOBAL (common to all cores). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of the MCU driver is placed under single MemMap section. The executable code can be relocated to any PFlash region.

Data section:

The sections marked as global should be relocated to the non-cached LMU region.

Configuration data and constants:

The sections marked as global should be relocated to the PFlash of the master core.

Note: Relocating code, data or constants to a distant memory region would impact execution timings.

17.1.4.3 MCU support

Not applicable for the MCU driver.

17.1.4.4 Port support

The MCU driver does not use any services provided by the PORT driver.

17.1.4.5 DMA support

The MCU driver does not use any services provided by the DMA driver.

17.1.4.6 Interrupt connections

The MCU driver does not use any interrupt source.

MCU driver

17.1.4.7 Example usage

This section explains an example usage scenario of the MCU Driver for a nominal case.

Configuration of the driver

MCU Driver is configured before usage and configuration files are generated and made available during software build process.

Initialization of driver

Step 1: Include the `Mcu.h` header file, to include definition of the MCU driver configuration data structure..

Step 2: Invoke the `Mcu_Init()` API by passing configuration structure pointer as an input parameter.

Example:

```
#include "Mcu.h"
Mcu_Init (&Mcu_Config); /*Mcu_Config is the configuration structure variable
for MCU */
```

Initialization of PLLs and Clocks

Pre-requisite: The `Mcu_Init()` API must be invoked before this phase.

Step1: Invoke the `Mcu_InitClock()` API by passing the clock configuration index

Example:

```
TempVal = Mcu_InitClock (0); /* 0 is clock setting id */
```

Step2: Wait until the system PLL is locked.

Example: Add a wait loop around the following condition:

```
while (Mcu_GetPllStatus () != MCU_PLL_LOCKED); /* Wait for PLLs to Lock */
```

Step3: Invoke the `Mcu_DistributePllClock()` API to change the clock source as PLL and ramp up/down to the configured clock frequencies.

Example:

```
TempVal = Mcu_DistributePllClock ();
```

De-Initialization of driver

Step1: Invoke the `Mcu_DeInit()` API. The API de-initializes all MCU relevant global configuration registers except for the PLL and clock-related registers.

Using low power modes

The MCU Driver shall be initialized before using low power mode API. Low power mode APIs shall be enabled as per configuration

Step1: Configure the wakeup source before entering into low power modes. Special configurations for STANDBY modes are available in `McuModeSettingConf` container. Ensure the executing core is authorized to perform low power mode transitions, as per the `McuIdleModeCpuCore` and `McuSystemModeCpuCore` configuration parameters.

MCU driver

Step2 (For IDLE/SLEEP): Invoke as shown below. For example for SLEEP mode.

```
Mcu_SetMode (MCU_SLEEP);
```

Step2 (For STANDBY): It is important that wakeup source status flags are cleared on exit of standby mode to ensure further wake-ups from standby state are enabled.

Example sequence during STANDBY entry:

```
Temp_Val = Mcu_GetWakeupCause ();
Mcu_ClearWakeupCause (Temp_Val);
Mcu_SetMode (MCU_STANDBY);
```

17.1.5 Key architectural considerations

17.1.5.1 GTM: usage with complex drivers

The user must consider the following points while using the GTM IP outside of the MCU driver.

- The MCU driver enables the clock for a cluster only if GTM (TIM, TOM or ATOM) channels are reserved inside the McuHardwareResourceAllocationConf for that particular cluster.
- When none of the GTM (TIM, TOM and ATOM) channels are reserved inside the McuHardwareResourceAllocationConf container for a particular cluster, the clock to the TIM, TOM and ATOM modules of that cluster is set to its default value.
- The configurable clocks and the fixed clocks for the clusters are configured as per user configuration.

17.1.5.2 Multicore support for MCU

MCU initialization, de-initialization and clock tree configuration should be carried out by the master core with the following APIs: Mcu_Init, Mcu_DeInit, Mcu_InitClock and Mcu_DistributePllClock. These APIs shall not be invoked from the slave core(s).

[cover parentID MCU={B4FAB0B9-7333-4da0-8A40-59575AEBFF6E}]

17.1.5.3 Usage of Mcu_DeInit API

The Mcu_DeInit API should be called only after all functions have completed their execution in slave cores.

[cover parentID MCU={E02F04BC-B8D2-47c0-83D2-E9BA65207E8E}]

17.1.5.4 Error handling for Timer IP APIs

DETs and DEMs are not reported by the Timer IP APIs. Hence the integrity of the input arguments for these APIs must be done by the user of these APIs.

[cover parentID MCU={46F34BBF-11B7-4ac0-9DA7-73566A300E9D}]

17.1.5.5 User mode support

The MCU Driver supports Supervisor and User-1 modes to write into registers which can be written in the Supervisor mode and User-1 mode.

[cover parentID MCU={E0E98A25-3A4F-478b-B80B-9237918239B5}]

MCU driver

17.1.5.6 Reset reason due to HSM

The Mcu_GetResetReason API does not support application and system resets occurring due to HSM. If an application/system reset due to HSM occurs, then Mcu_GetResetReason returns MCU_RESET_UNDEFINED. In such case, user must use Mcu_GetResetRawValue API to identify the reset reason.

[cover parentID MCU={15307DAC-2ED5-42fe-BDF9-00BC40FCB1FA}]

17.1.5.7 Reset reason due to multiple resets

The Mcu_GetResetReason API does not support multiple reset reasons, unless they are associated with power on reset as there are many other combinations which cannot be covered.

[cover parentID MCU={30EB1B27-5B0A-4581-9C00-05345C1945AB}]

17.1.5.8 Power modes entry

Before entering any power down mode like Idle, sleep or standby, the steps for ramping down the frequencies mentioned in the Power management system (PMS) chapter of the HW Target Specification should be followed.

[cover parentID MCU={3C41313F-F55F-46b3-A2B4-B384C5205D21}]

17.1.5.9 Generic AoUs to users of MCU

Users of the MCU shall ensure to provide valid input parameters for TOM/ATOM, CCU6, GPT12 APIs, MCU De-init and Timer Ip De-init APIs should be called before re-initializing. Modules shall use the APIs provided by the MCU driver to access common resources and to perform a force update of the GTM registers

[cover parentID MCU={AB317AE6-76D0-433d-ADE5-992094CB5901}]

17.1.5.10 Timer channel reservation in MCU hardware resource allocation

For GTM, CCU6, GPT12 and ERU hardware channels, channel reserved and not utilized by any of the drivers, has to be unreserved. Similarly for STM comparators, comparator reserved and not utilized by any of the driver should be unreserved.

17.1.5.11 Usage of Mcu_SetMode API

If the MCU driver is programmed to enter into the sleep or standby mode, where all the CPUs unanimously decide to enter the sleep/standby mode, then the slave cores should enter the respective power down modes first, with the master core being the last CPU to enter the power down mode.

[cover parentID MCU={E8E1B722-AE0A-4bb6-BD92-F79F3A200DA4}]

17.1.5.12 Cluster 0 clock should not be disabled if GTM is to be used

Cluster 0 clock should always be kept enabled in the configuration if the GTM is used as CMU derives its clock from Cluster 0 clock.

[cover parentID MCU={2EDBA464-E77A-423c-A5DB-978106D4819F}]

17.1.5.13 CCU6 and GPT12 initialization is performed only for the kernel/timers reserved by the user

CCU6 and GPT12 initialization is performed only for the kernel/timers reserved by the user.

MCU driver**17.1.5.14 Approximation of frequency to divider calculation**

In MCU clock configuration container `McuClockSettingConfig` the user enters a desired frequency for all the clocks.

The MCU driver automatically calculates the divider for all the clocks based on configured clock frequency and its source frequency (Source Frequency / Configured Clock Frequency).

If the calculated divider is an integer then the exact calculated value for the divider is programmed in the SFR. In case the calculated divider is not an integer but within +/- 0.1 of an integer. Then the closest integer value is considered and programmed.

For example if the `McuClockReferencePointFrequency2` is 200 MHz and `McuI2CFrequency` is configured as 66.6 MHz, the calculated divider value is 3.003.

In this scenario, a value of 3 will be considered to be programmed for the divider value as it is within the threshold of +/- 0.1

17.1.5.15 Timer APIs in the driver

The MCU driver contains a submodule apart from providing its main functionality as described in AUTOSAR. The submodule, `Mcu_17_TimerIp`, contains support functions for GTM, CCU6 and GPT12 timer channels, which may be used by other drivers for initializing, starting and so on of timer channels. The MCU driver through the `Mcu_Init()` API initializes the GTM global configurations such as cluster, clock management unit, time base unit, etc. initializes the clock control for CCU6 and GPT12.

MCU driver

17.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Atomic access using TriCore™ atomic instruction for ERU registers**

User of the ERU shall ensure that all the ERU-specific SFRs are accessed atomically.

[cover parentID MCU={7E9E92CE-7018-4b24-B184-DB24346D9E8A}]

- **ConfigPtr passed to InitCheck**

User of the MCU shall ensure that InitCheck is invoked with the same ConfigPtr that was used during initialization.

[cover parentID MCU={ADE0F1CA-CEC3-423c-AA12-F673593DB8F2}]

- **Correctness of the configuration is generated - ERU**

User of the MCU (ERU) shall ensure that the resource allocation information generated for the ERU channels is as per the configuration in the GUI.

[cover parentID MCU={C4CA831B-4FF9-4d97-A06B-B571161992DE}]

- **Critical section protection with Interrupt enable/disable**

User of the MCU (TOM/ATOM) shall ensure that the critical section protection provided by the MCU for TOM and ATOM shall be implemented to disable interrupts.

[cover parentID MCU={276431BA-062F-47b5-B2E8-270B6095F087}]

- **Freedom from Interference**

It is the onus on the user to provide protection to the MCAL data and SFRs from the QM software to avoid any SFR or memory corruption.

[cover parentID MCU={78293C3C-A3AB-4c45-BE00-30A0D271FF97}]

- **Generic AoUs for the users of the MCU**

- Drivers using the MCU shall ensure that GTM, CCU6 and GPT12 APIs are invoked after completion of the MCU initialization (clock tree initialization). - Drivers using the MCU shall ensure to provide valid input parameters for TOM/ATOM, CCU6 and GPT12 APIs. - MCU de-init and Timerlp de-init APIs shall be called before re-initializing the MCU Timerlp-related initialization services, respectively. - Common resources shall be accessed using the MCU APIs.

[cover parentID MCU={E91C15B4-38E0-485f-ADAA-EBCFFD98D831}]

- **InitCheck sequence**

User shall invoke the Mcu_InitCheck() API to ensure the initialization is done correctly. The McuInitCheckApi parameter shall be enabled and the user of the MCU shall call the InitCheck function before the execution of any runtime API (except GetVersionInfo) but after the completion of the MCU initialization sequence.

[cover parentID MCU={AF9A5DC2-05BA-4b55-8377-D1A640B25832}]

- **Interrupt source needs to be checked by user for GPT12 ISR**

User shall ensure that the intended GPT12 channel is the source of the interrupt to avoid unexpected/spurious interrupts.

[cover parentID MCU={EA111806-7E04-4e56-AD1A-AF63E5648682}]

- **Maximum STM compare duration**

User of the MCU (STM) shall ensure that the maximum compare duration does not exceed the 32-bit compare value.

[cover parentID MCU={22FB290D-B9BC-41ca-81C8-A85E6AF795D5}]

- **Provide correct configuration**

User shall provide the correct configuration values for the configuration parameters.

[cover parentID MCU={1E99EFD8-6D52-4be8-AF7E-8D6C82CC41D5}]

- **RAM section base address**

MCU driver

User shall provide the start address for the RAM section as per the natural memory alignment of the memory type.

[cover parentID MCU={4B92F5E7-BD7A-48eb-805C-8B7C525A3ED7}]

- **Sequence to enter the Sleep or Standby mode using the Mcu_SetMode API**

User shall ensure that when the MCU driver is programmed to enter into the sleep or standby mode where all the CPUs unanimously decide to enter the sleep or standby mode, the slave cores should enter the respective power down modes first, with the master core being the last CPU to enter the power down mode.

[cover parentID MCU={2261FEE8-1D74-46f2-929C-BFA1A65A7541}]

- **Setting same trap again**

User shall ensure that the Mcu_ClearTrapRequest() API is called from the trap handler to clear the trap cause.

[cover parentID MCU={E2582802-9F0C-4794-9EC6-A30E801DFD95}]

- **SMU alarms with clock initialization**

User shall disable the SMU alarms relating to the clock tree before calling the Mcu_InitClock() and Mcu_DistributePllClock() APIs and re-configure to user setting after the successful execution of both the APIs.

[cover parentID MCU={D10AE831-59F1-4bf4-A3D1-F41F9CED6C9B}]

- **Software reset configuration**

User shall ensure that when the Mcu_PerformReset API is called to perform software reset, the McuSWResetConf parameter shall not be configured as no reset.

[cover parentID MCU={34569091-6D4D-4789-BA0E-193A77598D5F}]

- **STM is enabled**

User of the MCAL shall ensure that the STM is enabled and not in the sleep mode before invoking any MCAL APIs.

[cover parentID MCU={944C58EE-586A-49f6-8036-C206C63762E1}]

- **STM same configuration used for check and setup comparator**

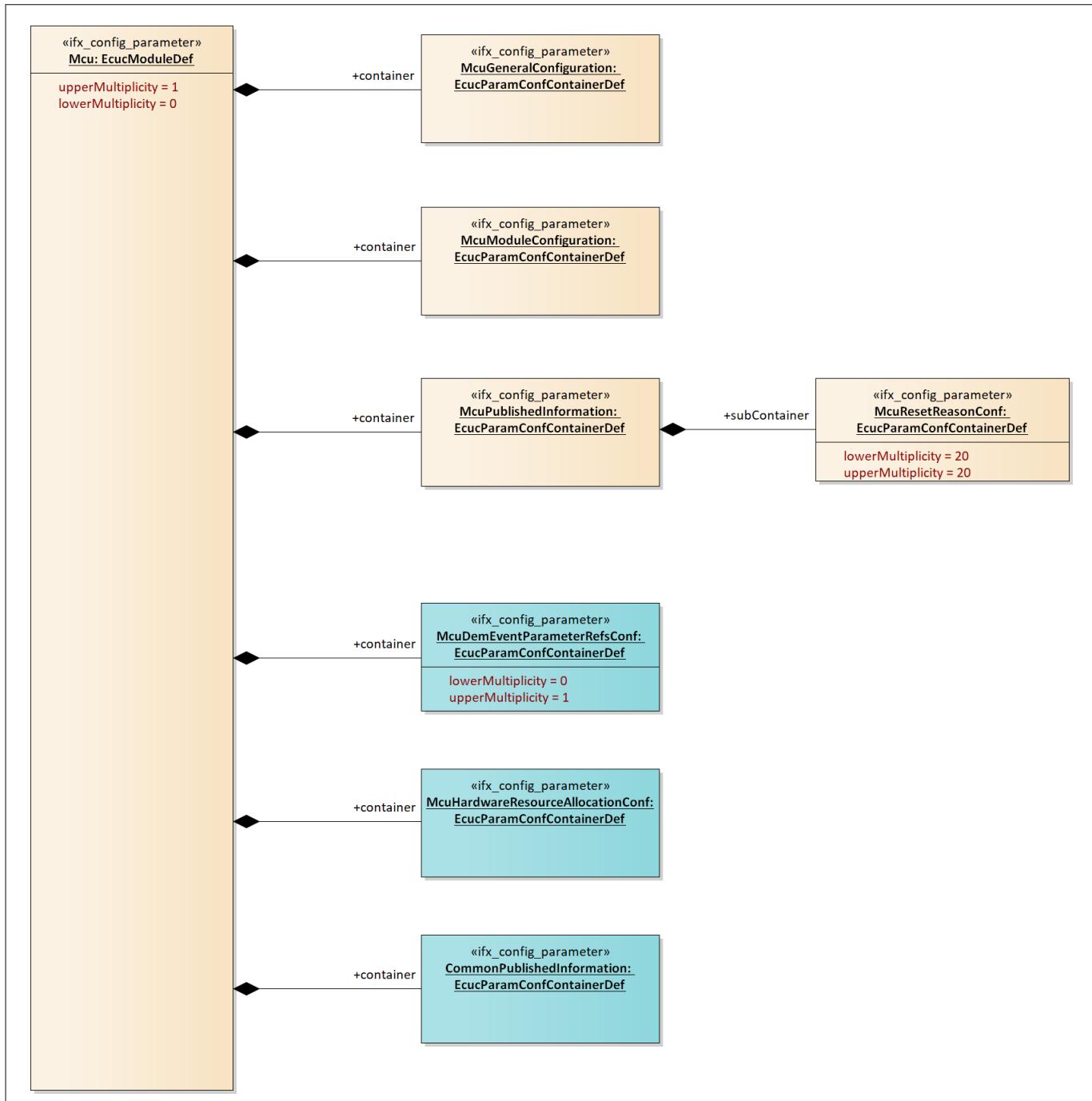
User of the MCU (STM) shall provide same configuration for the SetupComparator() and CheckComparator() APIs.

[cover parentID MCU={3BC33D10-04B2-4b7f-82E7-5F93FDB874E8}]

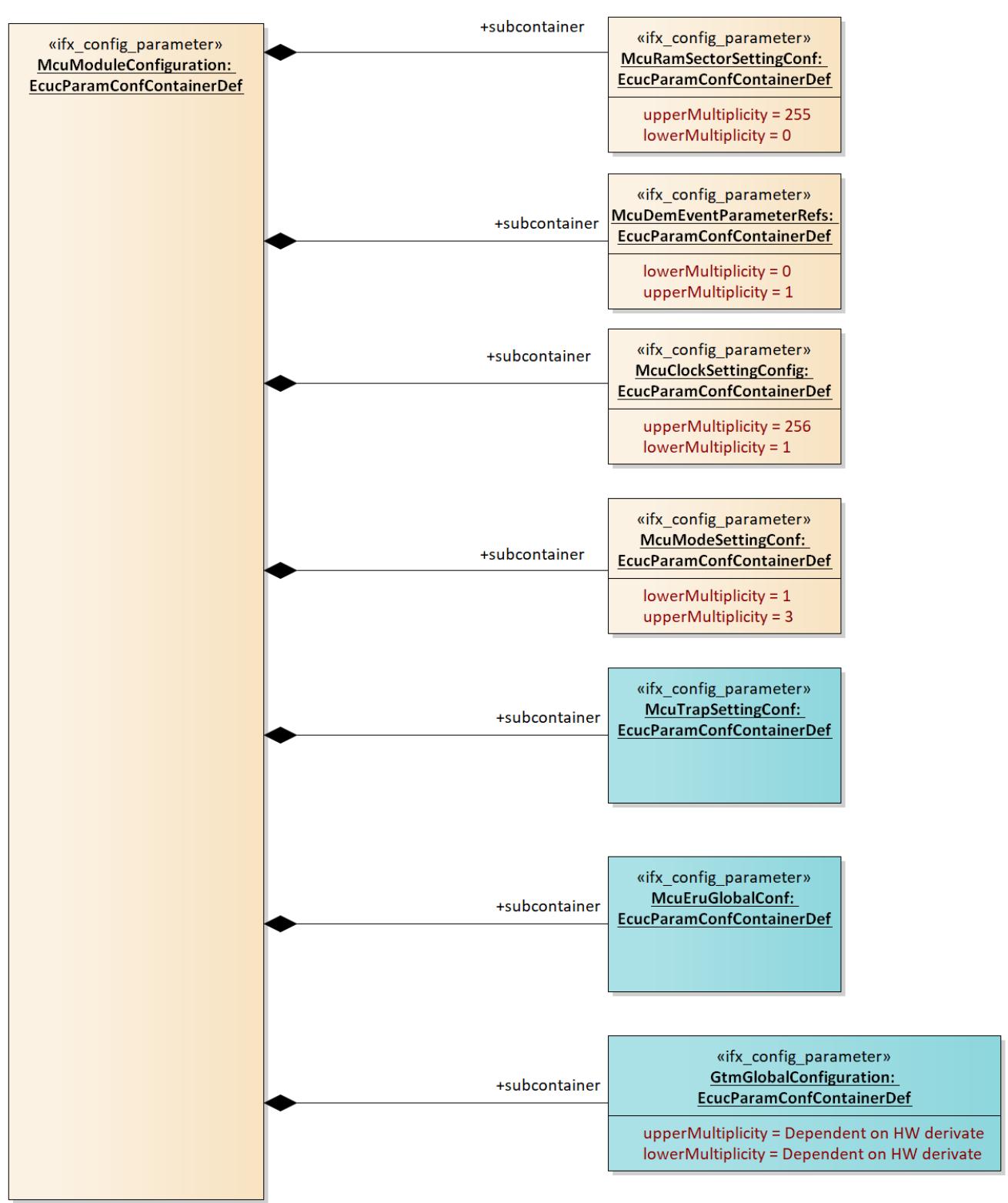
MCU driver

17.3 Reference information

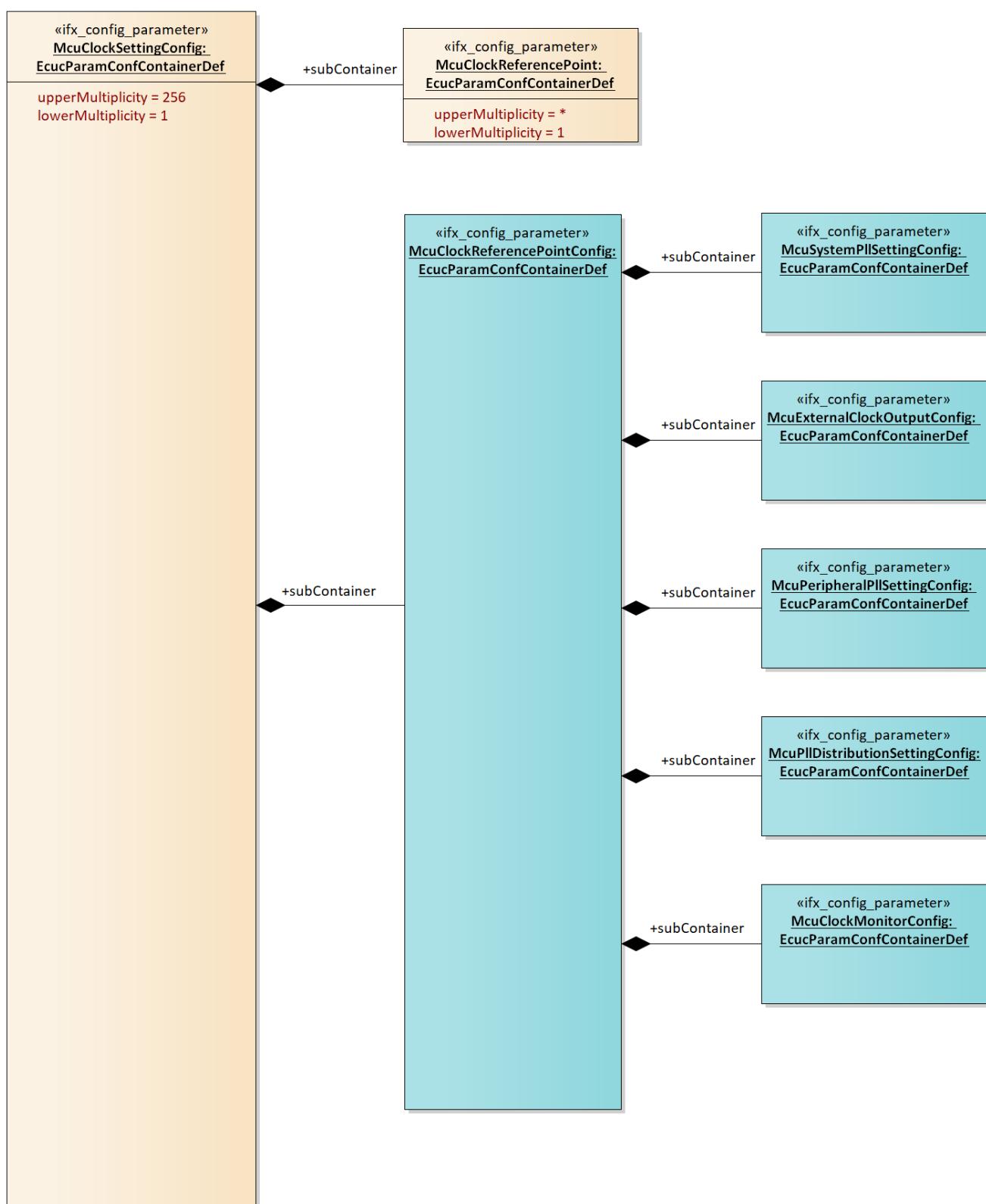
17.3.1 Configuration interfaces

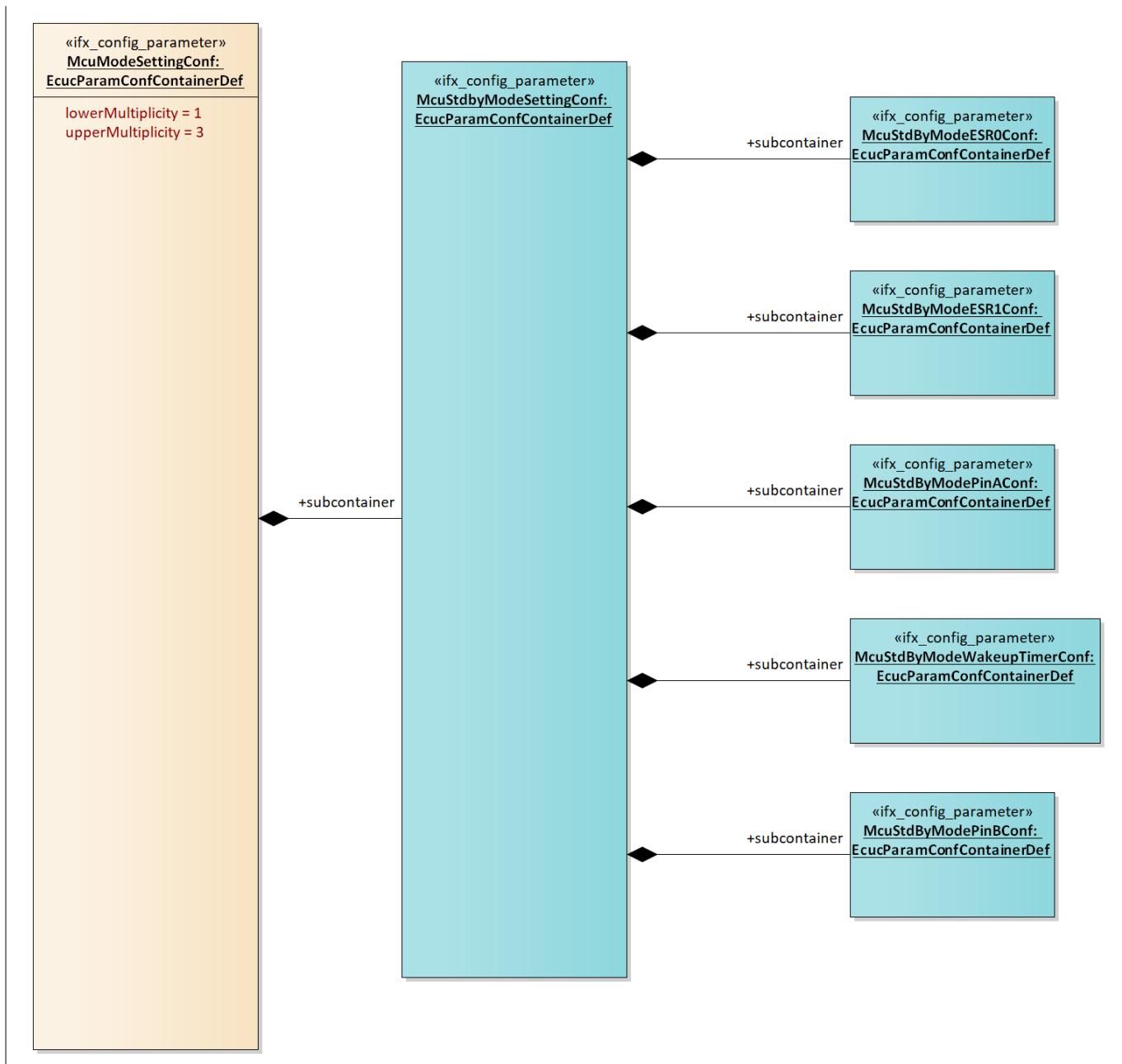


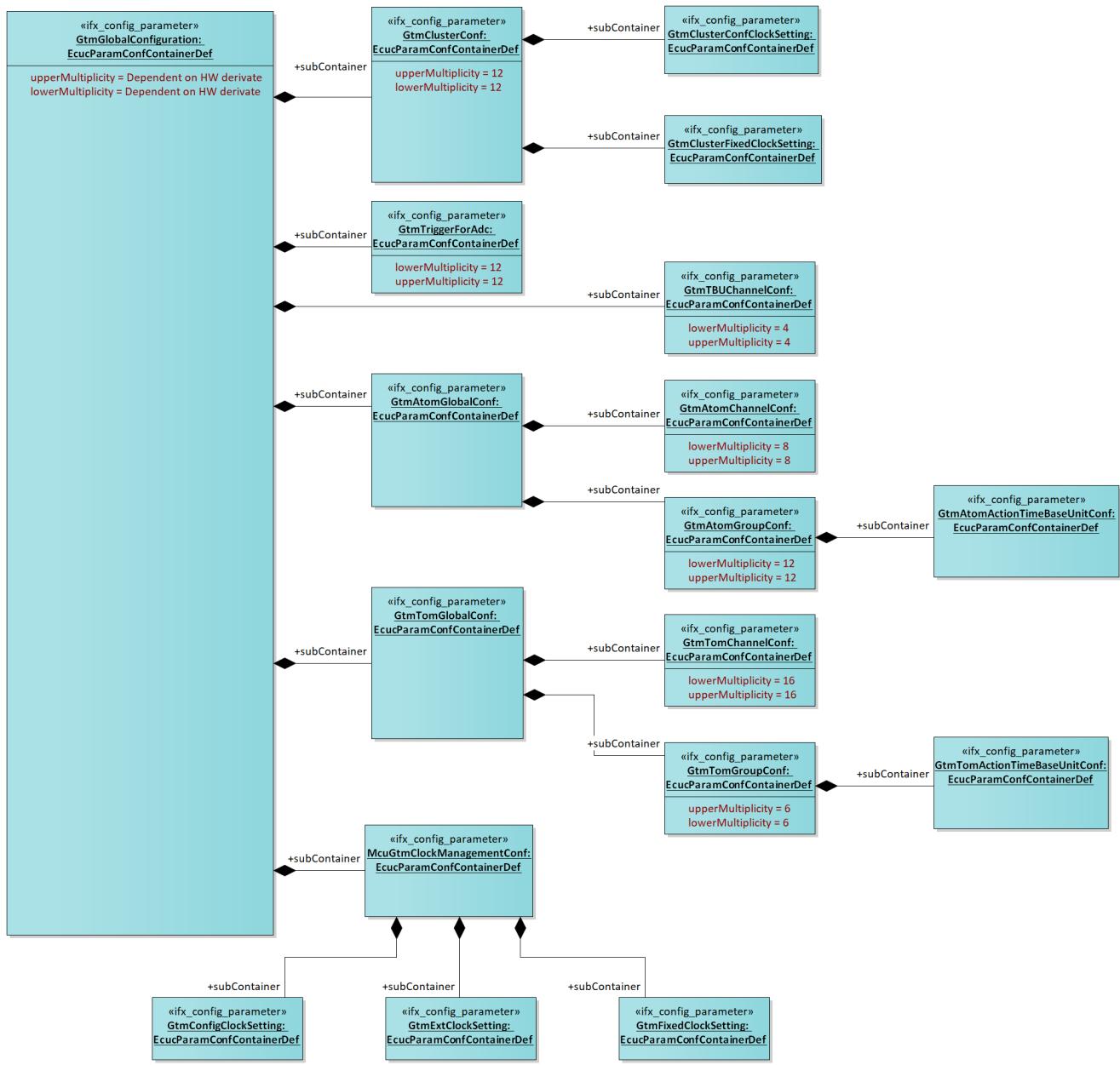
MCU driver

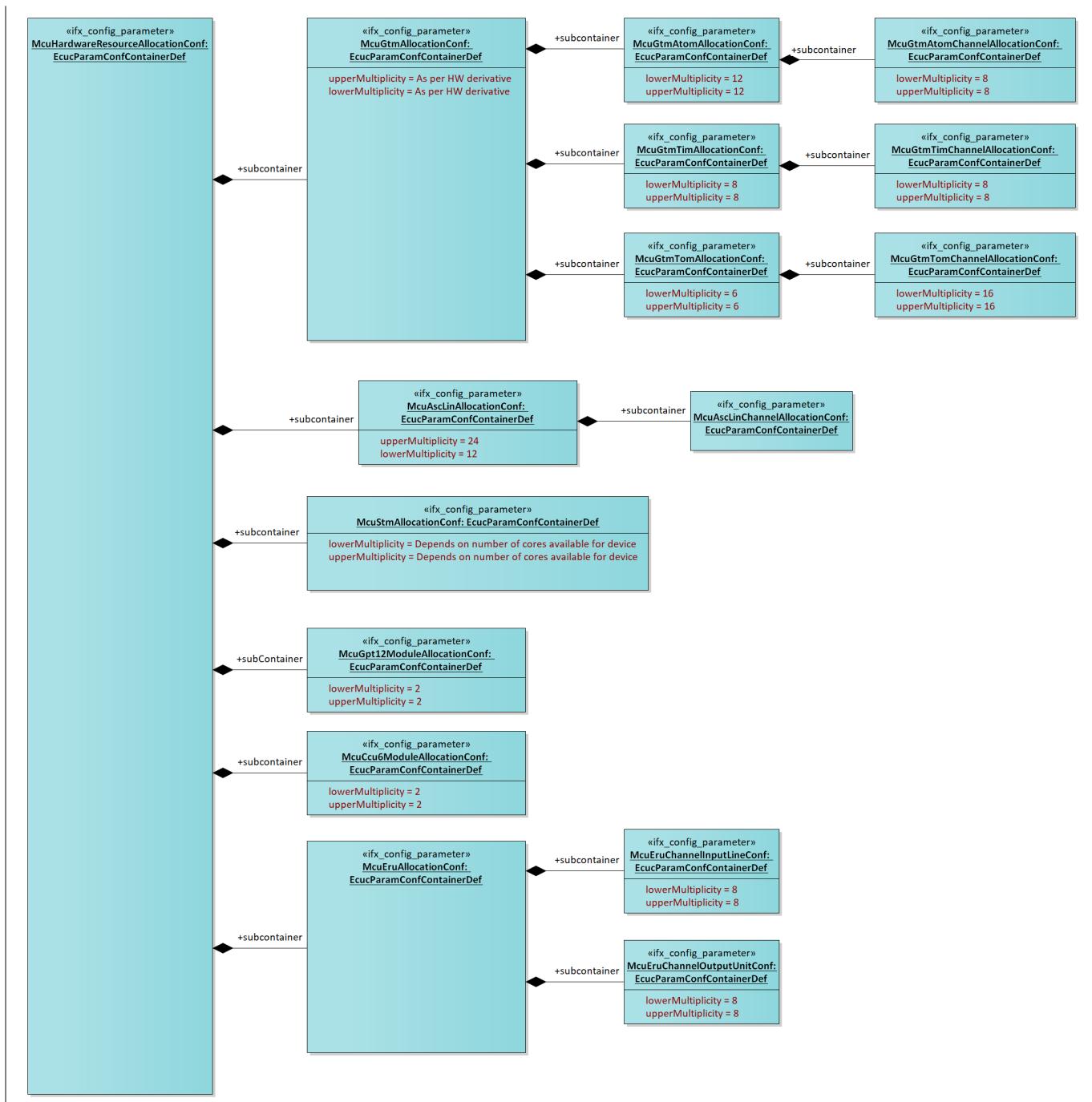


MCU driver



MCU driver

MCU driver


MCU driver

Figure 159 Container hierarchy along with their configuration parameters
17.3.1.1 Container: McuClockMonitorConfig

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

MCU driver

17.3.1.1.1 McuBackupClockMonEnable

Table 1126 Specification for McuBackupClockMonEnable

Name	McuBackupClockMonEnable		
Description	Specifies if the Backup clock monitoring is enabled/disabled. TRUE: Backup clock monitoring is enabled FALSE: Backup clock monitoring is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.1.2 McuBackupClockRangeMonEnable

Table 1127 Specification for McuBackupClockRangeMonEnable

Name	McuBackupClockRangeMonEnable		
Description	Specifies if the Backup clock range monitoring is enabled/disabled. TRUE: Backup clock range monitoring is enabled FALSE: Backup clock range monitoring is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.1.3 McuPll0ClockMonEnable****Table 1128 Specification for McuPll0ClockMonEnable**

Name	McuPll0ClockMonEnable		
Description	Specifies if the PLL0 monitoring is enabled/disabled. TRUE : PLL0 monitoring is enabled FALSE: PLL0 monitoring is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.1.4 McuPll1ClockMonEnable**Table 1129 Specification for McuPll1ClockMonEnable**

Name	McuPll1ClockMonEnable		
Description	Specifies if the PLL1 monitoring is enabled/disabled. TRUE : PLL1 monitoring is enabled FALSE: PLL1 monitoring is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.1.5 McuPll2ClockMonEnable****Table 1130 Specification for McuPll2ClockMonEnable**

Name	McuPll2ClockMonEnable		
Description	Specifies if the PLL2 monitoring is enabled/disabled. TRUE : PLL2 monitoring is enabled FALSE: PLL2 monitoring is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.1.6 McuSpbClockMonEnable**Table 1131 Specification for McuSpbClockMonEnable**

Name	McuSpbClockMonEnable		
Description	Specifies if the SPB clock monitoring is enabled/disabled. TRUE : SPB clock monitoring is enabled FALSE: SPB clock monitoring is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.2 Container: McuGpt12PrescalerConf

This container defines the configuration parameters for the GPT prescalar

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.2.1 Gpt1BlockPrescalerSel

Table 1132 Specification for Gpt1BlockPrescalerSel

Name	Gpt1BlockPrescalerSel		
Description	Specifies the selection for GPT1 block prescalar		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GPT1_BLOCK_NOT_USED: GPT1 Timer Block is not used GPT1_PRESCALING_FACTOR_16: GPT1 Timer Block is clocked at GPT frequency by 16 GPT1_PRESCALING_FACTOR_32: GPT1 Timer Block is clocked at GPT frequency by 32 GPT1_PRESCALING_FACTOR_4: GPT1 Timer Block is clocked at GPT frequency by 4 GPT1_PRESCALING_FACTOR_8: GPT1 Timer Block is clocked at GPT frequency by 8		
Default value	GPT1_BLOCK_NOT_USED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuGpt12ModuleAllocationConf		

17.3.1.2.2 Gpt2BlockPrescalerSel

Table 1133 Specification for Gpt2BlockPrescalerSel

Name	Gpt2BlockPrescalerSel		
Description	Specifies the selection for GPT2 block prescalar		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GPT2_BLOCK_NOT_USED: GPT2 Timer Block is not used GPT2_PRESCALING_FACTOR_16: GPT2 Timer Block is clocked at GPT frequency by 16 GPT2_PRESCALING_FACTOR_2: GPT2 Timer Block is clocked at GPT frequency by 2 GPT2_PRESCALING_FACTOR_4: GPT2 Timer Block is clocked at GPT frequency by 4 GPT2_PRESCALING_FACTOR_8: GPT2 Timer Block is clocked at GPT frequency by 8		
Default value	GPT2_BLOCK_NOT_USED		

MCU driver**Table 1133 Specification for Gpt2BlockPrescalerSel (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuGpt12ModuleAllocationConf		

17.3.1.3 Container: McuStmAllocationConf

This container holds information related to MCU STM resource allocation configuration. User is not allowed to change the name of the parameters in this container.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.3.1 McuStmCmp0RegAllocationConf**Table 1134 Specification for McuStmCmp0RegAllocationConf**

Name	McuStmCmp0RegAllocationConf		
Description	The STM timer compare register 0 usage. Note: Availability of module is based on the Release Notes.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	STM_CMP0_NOT_USED: STM timer compare register 0 is not used. STM_CMP0_USED_BY_STM: STM timer compare register 0 is used by the STM. STM_CMP0_USED_BY_WDG: STM timer compare register 0 is used by the WDG.		
Default value	STM_CMP0_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.3.2 McuStmCmp1RegAllocationConf**Table 1135 Specification for McuStmCmp1RegAllocationConf**

Name	McuStmCmp1RegAllocationConf
Description	The STM timer compare register 1 usage.

MCU driver**Table 1135 Specification for McuStmCmp1RegAllocationConf (continued)**

	Note: Availability of module is based on the Release Notes.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	STM_CMP1_NOT_USED: STM timer compare register 1 is not used. STM_CMP1_USED_BY_STM: STM timer compare register 1 is used by the STM STM_CMP1_USED_BY_WDG: STM timer compare register 1 is used by the WDG		
Default value	STM_CMP1_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.4 Container: CommonPublishedInformation

Container for common published information

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.4.1 ArMajorVersion

Table 1136 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	ArMajorVersion parameter provides the major version of the AUTOSAR specification.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.4.2 ArMinorVersion

Table 1137 Specification for ArMinorVersion

Name	ArMinorVersion		
Description	ArMinorVersion parameter provides the minor version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.4.3 ArPatchVersion

Table 1138 Specification for ArPatchVersion

Name	ArPatchVersion		
Description	ArPatchVersion parameter provides the patch version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.4.4 ModuleId

Table 1139 Specification for ModuleId

Name	ModuleId		
Description	ModuleId provides the Module Id.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		

MCU driver**Table 1139 Specification for ModuleId (continued)**

Default value	101		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.4.5 Release**Table 1140 Specification for Release**

Name	Release		
Description	Release parameter provides the TC3xx derivative used for the implementation		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per HW derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.4.6 SwMajorVersion**Table 1141 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	SwMajorVersion provides the major version of the Software.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	As per driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

MCU driver**Table 1141 Specification for SwMajorVersion (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.4.7 SwMinorVersion**Table 1142 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	SwMinorVersion provides the minor version of the Software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.4.8 SwPatchVersion**Table 1143 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	SwPatchVersion provides the patch version of the Software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.4.9 VendorId

Table 1144 Specification for VendorId

Name	VendorId		
Description	VendorId provides the Vendor Id		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.5 Container: GtmAtomActionTimeBaseUnitConf

This container holds the configuration parameters for the actual TBU setting. The action TBU setting is required to generate a trigger that can copy from shadow register to the actual registers for period, duty cycle and channel clock source.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.5.1 GtmAtomActionTimeBaseSelection

Table 1145 Specification for GtmAtomActionTimeBaseSelection

Name	GtmAtomActionTimeBaseSelection		
Description	Specifies time base selected to compare with the value configured in GtmAtomActionTimeBaseValue		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	ATOM_ACT_TB_TBU_TS0: ATOM group level trigger is generated when GtmAtomActionTimeBaseValue matches TBU_TS0 ATOM_ACT_TB_TBU_TS1: ATOM group level trigger is generated when GtmAtomActionTimeBaseValue matches TBU_TS1 ATOM_ACT_TB_TBU_TS2: ATOM group level trigger is generated when GtmAtomActionTimeBaseValue matches TBU_TS2		
Default value	ATOM_ACT_TB_TBU_TS0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver
Table 1145 Specification for GtmAtomActionTimeBaseSelection (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.5.2 GtmAtomActionTimeBaseValue

Table 1146 Specification for GtmAtomActionTimeBaseValue

Name	GtmAtomActionTimeBaseValue		
Description	Specifies the time base value for the ATOM group channel level trigger. A trigger at the AGC level is raised when TBU_TS[x] (x can be selected through GtmAtomActionTimeBaseSelection) value matches the value configured in this configuration parameter. The trigger request has to be explicitly enabled by the user by setting the ATOM_AGC_ACT_TB.TB_TRIG bitfield.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.6 Container: GtmAtomChannelConf

This container holds the configuration parameters for ATOM channel- level parameters required to be configured globally. Therefore multiplicity is always 8.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.6.1 GtmAtomChInternalTriggerEnable

Table 1147 Specification for GtmAtomChInternalTriggerEnable

Name	GtmAtomChInternalTriggerEnable
Description	Enables/disables internal trigger from channel 0 of the corresponding group channel number.

MCU driver**Table 1147 Specification for GtmAtomChInternalTriggerEnable (continued)**

	<p>If a channel belongs to AGC0 (channel number 0 - 7), setting this configuration parameter for the corresponding channel enables the trigger from channel0.</p> <p>Values:</p> <p>TRUE: enable internal trigger from channel 0 to 7 (based on the AGC a channel belong to). FALSE: disable internal trigger from channel 0 to 7 (based on the AGC a channel belong to).</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.6.2 GtmAtomChResetCn0OnTriggerEnable**Table 1148 Specification for GtmAtomChResetCn0OnTriggerEnable**

Name	GtmAtomChResetCn0OnTriggerEnable		
Description	<p>Enables/disables the ATOM channel counter CN0 value that will be reset on global trigger from any of the trigger sources.</p> <p>Values:</p> <p>TRUE: resetting of ATOM channel CN0 on global trigger from any trigger source is enabled FALSE: resetting of ATOM channel CN0 on global trigger from any trigger source is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

MCU driver

17.3.1.7 Container: GtmAtomGlobalConf

This container holds the configuration parameters for ATOM global parameters. Various instances of ATOM channels can be used by ADC, PWM, GPT and WDG drivers and, therefore the global configuration for these channels within one ATOM group channel (AGC) is taken care of by this container.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.8 Container: GtmAtomGroupConf

This container holds the configuration parameters for ATOM group channel parameters. ATOM module has one group and therefore the multiplicity is 1.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.9 Container: GtmClusterConf

This container holds the cluster configuration. A cluster is organized as a set of GTM sub peripheral instances. As an example, cluster-0 contains one instance of (CMU, TBU, TOM0, ATOM0 TIM0 etc.). This container holds configuration parameters for all cluster configuration modules. User is not allowed to change the name of the configuration parameter in this container

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.9.1 GtmCmuClusterInputClockDividerEnable

Table 1149 Specification for GtmCmuClusterInputClockDividerEnable

Name	GtmCmuClusterInputClockDividerEnable		
Description	<p>Enables/disables the dividing of fGTM to CMU.</p> <p>The configuration value CLS0_CLK_DIV defines the primary input clock period for CMU.</p> <p>If CLS0_CLK_DIV is configured to a value 0b10 (that is clock divider 2), the maximum CMU clock frequency for all other cluster c=1..n is also limited to the configured CMU clock frequency of cluster 0.</p> <p>Note: For the clusters greater than 4, (only 100 MHz capable), the allowed settings for the CLS_CLK_DIV are 00 and 10 (clock divider 2).</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CLS_CLK_CFG_DISABLED_SEL0: cluster x is disabled CLS_CLK_CFG_ENABLED_WITHOUT_DIV_SEL1: cluster x is enabled without clock divider CLS_CLK_CFG_ENABLED_WITH_DIV_SEL2: cluster x is enabled with clock divider		
Default value	CLS_CLK_CFG_ENABLED_WITH_DIV_SEL2		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1149 Specification for GtmCmuClusterInputClockDivideEnable (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.10 Container: GtmClusterConfClockSetting

This container contains the configuration (parameters) for the GTM cluster clock settings

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.10.1 GtmClusterConfClock0Src**Table 1150 Specification for GtmClusterConfClock0Src**

Name	GtmClusterConfClock0Src		
Description	Specifies the input clock source for the current GTM cluster sub- peripheral using configurable clock 0. User is not allowed to change the name of the configuration parameter		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CONF_CLOCK0_SEL0: configurable clock 0 is used for the clock CMU_CONF_CLOCK8_SEL1: configurable clock8 is used for the clock EXT_CAPTURE_SEL2: external capture source is used for the clock.		
Default value	CMU_CONF_CLOCK0_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.10.2 GtmClusterConfClock1Src**Table 1151 Specification for GtmClusterConfClock1Src**

Name	GtmClusterConfClock1Src		
Description	Specifies the input clock source for the current GTM cluster sub peripheral using configurable clock 1.		

MCU driver**Table 1151 Specification for GtmClusterConfClock1Src (continued)**

	User is not allowed to change the name of the configuration parameter.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CONF_CLOCK1_SEL0: configurable clock 1 is used for the clock CMU_CONF_CLOCK8_SEL1: configurable clock8 is used for the clock EXT_CAPTURE_SEL2: external capture source is used for the clock.		
Default value	CMU_CONF_CLOCK1_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.10.3 GtmClusterConfClock2Src**Table 1152 Specification for GtmClusterConfClock2Src**

Name	GtmClusterConfClock2Src		
Description	Specifies the input clock source for the current GTM cluster sub peripheral using configurable clock 2. User is not allowed to change the name of the configuration parameter.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CONF_CLOCK2_SEL0: configurable clock 2 is used for the clock CMU_CONF_CLOCK8_SEL1: configurable clock8 is used for the clock EXT_CAPTURE_SEL2: external capture source is used for the clock.		
Default value	CMU_CONF_CLOCK2_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

MCU driver**17.3.1.10.4 GtmClusterConfClock3Src****Table 1153 Specification for GtmClusterConfClock3Src**

Name	GtmClusterConfClock3Src		
Description	Specifies the input clock source for the current GTM cluster sub peripheral using configurable clock 3. User is not allowed to change the name of the configuration parameter.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CONF_CLOCK3_SEL0: configurable clock 3 is used for the clock CMU_CONF_CLOCK8_SEL1: configurable clock8 is used for the clock EXT_CAPTURE_SEL2: external capture source is used for the clock.		
Default value	CMU_CONF_CLOCK3_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.10.5 GtmClusterConfClock4Src**Table 1154 Specification for GtmClusterConfClock4Src**

Name	GtmClusterConfClock4Src		
Description	Specifies the input clock source for the current GTM cluster sub- peripheral using configurable clock 4. User is not allowed to change the name of the configuration parameter.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CONF_CLOCK4_SEL0: configurable clock 4 is used for the clock CMU_CONF_CLOCK8_SEL1: configurable clock8 is used for the clock EXT_CAPTURE_SEL2: external capture source is used for the clock.		
Default value	CMU_CONF_CLOCK4_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

MCU driver**17.3.1.10.6 GtmClusterConfClock5Src****Table 1155 Specification for GtmClusterConfClock5Src**

Name	GtmClusterConfClock5Src		
Description	Specifies the input clock source for the current GTM cluster sub- peripheral using configurable clock 5. User is not allowed to change the name of the configuration parameter		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CONF_CLOCK5_SEL0: configurable clock 5 is used for the clock CMU_CONF_CLOCK8_SEL1: configurable clock8 is used for the clock EXT_CAPTURE_SEL2: external capture source is used for the clock.		
Default value	CMU_CONF_CLOCK5_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.10.7 GtmClusterConfClock6Src**Table 1156 Specification for GtmClusterConfClock6Src**

Name	GtmClusterConfClock6Src		
Description	Specifies the input clock source for the current GTM cluster sub- peripheral using configurable clock 6. User is not allowed to change the name of the configuration parameter.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CONF_CLOCK6_SEL0: configurable clock 6 is used for the clock CMU_CONF_CLOCK8_SEL1: configurable clock8 is used for the clock EXT_CAPTURE_SEL2: external capture source is used for the clock.		
Default value	CMU_CONF_CLOCK6_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

MCU driver
17.3.1.10.8 GtmClusterConfClock7Src
Table 1157 Specification for GtmClusterConfClock7Src

Name	GtmClusterConfClock7Src		
Description	Specifies the input clock source for the current GTM cluster sub- peripheral using configurable clock 7. User is not allowed to change the name of the configuration parameter.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CONF_CLOCK7_SEL0: configurable clock 7 is used for the clock CMU_CONF_CLOCK8_SEL1: configurable clock8 is used for the clock EXT_CAPTURE_SEL2: external capture source is used for the clock.		
Default value	CMU_CONF_CLOCK7_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.11 Container: GtmClusterFixedClockSetting

GtmClusterFixedClockSetting container contains the configuration (parameters) for GTM cluster fixed clock settings

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.11.1 GtmClusterFixedClockSrc
Table 1158 Specification for GtmClusterFixedClockSrc

Name	GtmClusterFixedClockSrc		
Description	GtmClusterFixedClockSrc parameter specifies the input clock source for GTM cluster-x sub peripherals.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CONF_CLOCK8_SEL1: Configurable clock8 will be used for clock CMU_FIXED_CLOCK0_SEL0: Fixed clock0 will be used for clock		
Default value	CMU_FIXED_CLOCK0_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1158 Specification for GtmClusterFixedClockSrc (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.12 Container: GtmConfigClockSetting

This container contains the configuration (parameters) for the GTM configuration clock settings.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.12.1 GtmCmuConfigClock0Div**Table 1159 Specification for GtmCmuConfigClock0Div**

Name	GtmCmuConfigClock0Div		
Description	Specifies the configurable clock0 divider count value. Defines the count value for the clock divider of clock source CMU_CLK0. Value can only be modified when clock enable EN_CLK0 and EN_ECLK1 are disabled. This configuration parameter is applicable only if the CmuConfigClock0Enable is set to TRUE.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuConfigClock0Enable		

17.3.1.12.2 GtmCmuConfigClock0Enable**Table 1160 Specification for GtmCmuConfigClock0Enable**

Name	GtmCmuConfigClock0Enable
Description	Enables the configurable clock0. Divider for configurable clock0 is defined by GtmCmuConfigClock0Div. Values:

MCU driver**Table 1160 Specification for GtmCmuConfigClock0Enable (continued)**

	TRUE: CMU configurable clock0 is enabled FALSE: CMU configurable clock0 is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.12.3 GtmCmuConfigClock1Div**Table 1161 Specification for GtmCmuConfigClock1Div**

Name	GtmCmuConfigClock1Div		
Description	Specifies the configurable clock1 divider count value. Defines the count value for the clock divider of clock source CMU_CLK1. Value can only be modified when clock enable EN_CLK1 and EN_ECLK1 are disabled. This configuration parameter is applicable only if CmuConfigClock1Enable is set to TRUE.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuConfigClock1Enable		

MCU driver

17.3.1.12.4 GtmCmuConfigClock1Enable

Table 1162 Specification for GtmCmuConfigClock1Enable

Name	GtmCmuConfigClock1Enable		
Description	<p>Enables the configurable clock1.</p> <p>Divider for configurable clock1 is defined by GtmCmuConfigClock1Div.</p> <p>Values:</p> <p>TRUE: CMU configurable clock1 is enabled</p> <p>FALSE: CMU configurable clock1 is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.12.5 GtmCmuConfigClock2Div

Table 1163 Specification for GtmCmuConfigClock2Div

Name	GtmCmuConfigClock2Div		
Description	<p>Specifies the configurable clock2 divider count value.</p> <p>Defines the count value for the clock divider of clock source CMU_CLK2.</p> <p>Value can only be modified when clock enable EN_CLK2 and EN_ECLK1 are disabled.</p> <p>This configuration parameter is applicable only if CmuConfigClock2Enable is set to TRUE.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU

MCU driver**Table 1163 Specification for GtmCmuConfigClock2Div (continued)**

Dependency	GtmCmuConfigClock2Enable
-------------------	--------------------------

17.3.1.12.6 GtmCmuConfigClock2Enable**Table 1164 Specification for GtmCmuConfigClock2Enable**

Name	GtmCmuConfigClock2Enable		
Description	<p>Enables the configurable clock2.</p> <p>Divider for configurable clock2 is defined by GtmCmuConfigClock2Div.</p> <p>Values:</p> <p>TRUE: CMU configurable clock2 is enabled</p> <p>FALSE: CMU configurable clock2 is disabled</p>		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.12.7 GtmCmuConfigClock3Div**Table 1165 Specification for GtmCmuConfigClock3Div**

Name	GtmCmuConfigClock3Div		
Description	<p>Specifies the configurable clock3 divider count value.</p> <p>Defines the count value for the clock divider of clock source CMU_CLK3.</p> <p>Value can only be modified when clock enable EN_CLK3 and EN_ECLK1 are disabled.</p> <p>This configuration parameter is applicable only if CmuConfigClock3Enable is set to TRUE.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1165 Specification for GtmCmuConfigClock3Div (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuConfigClock3Enable		

17.3.1.12.8 GtmCmuConfigClock3Enable**Table 1166 Specification for GtmCmuConfigClock3Enable**

Name	GtmCmuConfigClock3Enable		
Description	Enables the configurable clock3. Divider for configurable clock3 is defined by GtmCmuConfigClock3Div. Values: TRUE: CMU configurable clock3 is enabled FALSE: CMU configurable clock3 is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.12.9 GtmCmuConfigClock4Div**Table 1167 Specification for GtmCmuConfigClock4Div**

Name	GtmCmuConfigClock4Div		
Description	Specifies the configurable clock4 divider count value. Defines the count value for the clock divider of clock source CMU_CLK4. Value can only be modified when clock enable EN_CLK4 and EN_ECLK1 are disabled. This configuration parameter is applicable only if CmuConfigClock4Enable is set to TRUE.		
Multiplicity	1..1	Type	EcucIntegerParamDef

MCU driver**Table 1167 Specification for GtmCmuConfigClock4Div (continued)**

Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuConfigClock4Enable		

17.3.1.12.10 GtmCmuConfigClock4Enable**Table 1168 Specification for GtmCmuConfigClock4Enable**

Name	GtmCmuConfigClock4Enable		
Description	Enables the configurable clock4. Divider for configurable clock4 is defined by GtmCmuConfigClock4Div. Values: TRUE: CMU configurable clock4 is enabled FALSE: CMU configurable clock4 is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.12.11 GtmCmuConfigClock5Div**Table 1169 Specification for GtmCmuConfigClock5Div**

Name	GtmCmuConfigClock5Div
Description	Specifies the configurable clock5 divider count value. Defines the count value for the clock divider of clock source CMU_CLK5.

MCU driver**Table 1169 Specification for GtmCmuConfigClock5Div (continued)**

	Value can only be modified when clock enable EN_CLK5 and EN_ECLK1 are disabled. This configuration parameter is applicable only if CmuConfigClock5Enable is set to TRUE.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuConfigClock5Enable		

17.3.1.12.12 GtmCmuConfigClock5Enable**Table 1170 Specification for GtmCmuConfigClock5Enable**

Name	GtmCmuConfigClock5Enable		
Description	Enables the configurable clock5 Divider for configurable clock5 is defined by GtmCmuConfigClock5Div. Values: TRUE: CMU configurable clock5 is enabled FALSE: CMU configurable clock5 is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

MCU driver**17.3.1.12.13 GtmCmuConfigClock6Div****Table 1171 Specification for GtmCmuConfigClock6Div**

Name	GtmCmuConfigClock6Div		
Description	<p>Specifies the configurable clock6 divider count value.</p> <p>Defines the count value for the clock divider of clock source CMU_CLK6.</p> <p>Value can only be modified when clock enable EN_CLK6 and EN_ECLK1 are disabled.</p> <p>This configuration parameter is applicable only if CmuConfigClock6Enable is set to TRUE.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuConfigClock6Enable		

17.3.1.12.14 GtmCmuConfigClock6Enable**Table 1172 Specification for GtmCmuConfigClock6Enable**

Name	GtmCmuConfigClock6Enable		
Description	<p>Enables the configurable clock6</p> <p>Divider for configurable clock6 is defined by GtmCmuConfigClock6Div.</p> <p>Values:</p> <p>TRUE: CMU configurable clock6 is enabled</p> <p>FALSE: CMU configurable clock6 is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU

MCU driver**Table 1172 Specification for GtmCmuConfigClock6Enable (continued)**

Dependency	-
-------------------	---

17.3.1.12.15 GtmCmuConfigClock7Div**Table 1173 Specification for GtmCmuConfigClock7Div**

Name	GtmCmuConfigClock7Div		
Description	<p>Specifies the configurable clock7 divider count value.</p> <p>Defines the count value for the clock divider of clock source CMU_CLK7.</p> <p>Value can only be modified when clock enable EN_CLK7 and EN_ECLK1 are disabled.</p> <p>This configuration parameter is applicable only if CmuConfigClock7Enable is set to TRUE.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuConfigClock7Enable		

17.3.1.12.16 GtmCmuConfigClock7Enable**Table 1174 Specification for GtmCmuConfigClock7Enable**

Name	GtmCmuConfigClock7Enable		
Description	<p>Enables the configurable clock7</p> <p>Divider for configurable clock7 is defined by GtmCmuConfigClock7Div.</p> <p>Values:</p> <p>TRUE: CMU configurable clock7 is enabled</p> <p>FALSE: CMU configurable clock7 is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver
Table 1174 Specification for GtmCmuConfigClock7Enable (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.13 Container: GtmExtClockSetting

This container contains the configuration (parameters) for the GTM external clock settings.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.13.1 GtmCmuExtClock0Denominator

Table 1175 Specification for GtmCmuExtClock0Denominator

Name	GtmCmuExtClock0Denominator		
Description	Specifies the denominator value for external clock 0. The GtmCmuExtClock0Numerator value should not be less than GtmCmuExtClock0Denominator.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuExtClock0Numerator, GtmCmuExtClock0Enable		

17.3.1.13.2 GtmCmuExtClock0Enable

Table 1176 Specification for GtmCmuExtClock0Enable

Name	GtmCmuExtClock0Enable
Description	Specifies the numerator value for the external clock 0 All other configuration parameters relevant to CMU external clocks are enabled only when this configuration parameter is enabled. Values: TRUE: CMU external configurable clock 0 is enabled FALSE: CMU external configurable clock 0 is disabled

MCU driver
Table 1176 Specification for GtmCmuExtClock0Enable (continued)

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.13.3 GtmCmuExtClock0Numerator

Table 1177 Specification for GtmCmuExtClock0Numerator

Name	GtmCmuExtClock0Numerator		
Description	Specifies the numerator value for external clock 0. The GtmCmuExtClock0Numerator value should not be less than GtmCmuExtClock0Denominator.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuExtClock0Denominator, GtmCmuExtClock0Enable		

17.3.1.13.4 GtmCmuExtClock1Denominator

Table 1178 Specification for GtmCmuExtClock1Denominator

Name	GtmCmuExtClock1Denominator		
Description	Specifies the denominator value for the external clock 1. The GtmCmuExtClock1Numerator value should not be less than GtmCmuExtClock1Denominator.		
Multiplicity	1..1	Type	EcucIntegerParamDef

MCU driver**Table 1178 Specification for GtmCmuExtClock1Denominator (continued)**

Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuExtClock1Numerator, GtmCmuExtClock1Enable		

17.3.1.13.5 GtmCmuExtClock1Enable**Table 1179 Specification for GtmCmuExtClock1Enable**

Name	GtmCmuExtClock1Enable		
Description	<p>Specifies the numerator value for the external clock 1.</p> <p>All other configuration parameters relevant to CMU external clocks are enabled only when this configuration parameter is enabled.</p> <p>Values:</p> <p>TRUE: CMU external configurable clock 1 is enabled</p> <p>FALSE: CMU external configurable clock 1 is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.13.6 GtmCmuExtClock1Numerator**Table 1180 Specification for GtmCmuExtClock1Numerator**

Name	GtmCmuExtClock1Numerator
Description	Specifies the numerator value for the external clock 1.

MCU driver**Table 1180 Specification for GtmCmuExtClock1Numerator (continued)**

	The GtmCmuExtClock1Numerator value should not be less than GtmCmuExtClock1Denominator.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuExtClock1Denominator, GtmCmuExtClock1Enable		

17.3.1.13.7 GtmCmuExtClock2Denominator**Table 1181 Specification for GtmCmuExtClock2Denominator**

Name	GtmCmuExtClock2Denominator		
Description	Specifies the denominator value for the external clock 2. The GtmCmuExtClock2Numerator value should not be less than GtmCmuExtClock2Denominator.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuExtClock2Numerator, GtmCmuExtClock2Enable		

17.3.1.13.8 GtmCmuExtClock2Enable**Table 1182 Specification for GtmCmuExtClock2Enable**

Name	GtmCmuExtClock2Enable
Description	Specifies the numerator value for the external clock 2 All other configuration parameters relevant to CMU external clocks are enabled only when this configuration parameter is enabled.

MCU driver**Table 1182 Specification for GtmCmuExtClock2Enable (continued)**

	Values: TRUE: CMU external configurable clock 2 is enabled FALSE: CMU external configurable clock 2 is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.13.9 GtmCmuExtClock2Numerator**Table 1183 Specification for GtmCmuExtClock2Numerator**

Name	GtmCmuExtClock2Numerator		
Description	Specifies the numerator value for the external clock 2. GtmCmuExtClock2Numerator value should not be less than GtmCmuExtClock2Denominator.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuExtClock2Denominator, GtmCmuExtClock2Enable		

17.3.1.14 Container: GtmFixedClockSetting

This container contains the configuration (parameters) for the GTM fixed clock settings.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

MCU driver**17.3.1.14.1 GtmCmuFixedClockEnable****Table 1184 Specification for GtmCmuFixedClockEnable**

Name	GtmCmuFixedClockEnable		
Description	<p>Enables the fixed clock.</p> <p>The source for fixed clock is defined by GtmCmuFixedClockSel.</p> <p>Values:</p> <p>TRUE: CMU fixed clock is enabled</p> <p>FALSE: CMU fixed clock is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.14.2 GtmCmuFixedClockSel**Table 1185 Specification for GtmCmuFixedClockSel**

Name	GtmCmuFixedClockSel		
Description	Specifies the source for the fixed clock.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>CMU_CLOCK0_SEL1: CMU0 is selected as the source for the fixed clock</p> <p>CMU_CLOCK1_SEL2: CMU1 is selected as the source for the fixed clock</p> <p>CMU_CLOCK2_SEL3: CMU2 is selected as the source for the fixed clock</p> <p>CMU_CLOCK3_SEL4: CMU3 is selected as the source for the fixed clock</p> <p>CMU_CLOCK4_SEL5: CMU4 is selected as the source for the fixed clock</p> <p>CMU_CLOCK5_SEL6: CMU5 is selected as the source for the fixed clock</p> <p>CMU_CLOCK6_SEL7: CMU6 is selected as the source for the fixed clock</p> <p>CMU_CLOCK7_SEL8: CMU7 is selected as the source for the fixed clock</p> <p>CMU_GLOBAL_CLOCK_SEL0: CMU global clock is selected as the source for the fixed clock</p>		
Default value	CMU_GLOBAL_CLOCK_SEL0		

MCU driver
Table 1185 Specification for GtmCmuFixedClockSel (continued)

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.15 Container: GtmGlobalConfiguration

This container holds the global (common) parameters of the GTM hardware. The GTM peripheral is used by multiple drivers. This container is responsible for initializing the common resources used by these drivers. Note: This container is not available for derivatives not having GTM peripheral.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.16 Container: GtmTBUChannelConf

This container holds the configuration parameters for the TBU channels of the GTM. The TBU can be used by TOM or ATOM trigger and TIM channels

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.16.1 GtmTbuChClockSourceSelection
Table 1186 Specification for GtmTbuChClockSourceSelection

Name	GtmTbuChClockSourceSelection		
Description	Selects the configurable clock source selection for the corresponding TBU channel. This parameter is relevant only to the TBU channels 0, 1 and 2. This configuration parameter is applicable only if GtmTbuChannelEnable is set to TRUE.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CMU_CLOCK0_SEL0: TBUX clock source is CMU0 CMU_CLOCK1_SEL1: TBUX clock source is CMU1 CMU_CLOCK2_SEL2: TBUX clock source is CMU2 CMU_CLOCK3_SEL3: TBUX clock source is CMU3 CMU_CLOCK4_SEL4: TBUX clock source is CMU4 CMU_CLOCK5_SEL5: TBUX clock source is CMU5 CMU_CLOCK6_SEL6: TBUX clock source is CMU6 CMU_CLOCK7_SEL7: TBUX clock source is CMU7		
Default value	CMU_CLOCK0_SEL0		

MCU driver**Table 1186 Specification for GtmTbuChClockSourceSelection (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmTbuChannelEnable		

17.3.1.16.2 GtmTbuChMode**Table 1187 Specification for GtmTbuChMode**

Name	GtmTbuChMode		
Description	Selects the timer counting mode. This is applicable only to the TBU channels-1 and 2. This configuration parameter is applicable only if GtmTbuChannelEnable is set to TRUE.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	FORWARD_BACKWARD_SEL1: Forward/backward counter mode FREE_RUNNING_COUNTER_SEL0: Free- running counter mode		
Default value	FREE_RUNNING_COUNTER_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmTbuChannelEnable		

17.3.1.16.3 GtmTbuChModuloCntrSel**Table 1188 Specification for GtmTbuChModuloCntrSel**

Name	GtmTbuChModuloCntrSel		
Description	Selects the channel selector for the modulo counter. This is applicable only to TBU channel 3. This configuration parameter is applicable only if GtmTbuChannelEnable is set to TRUE.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	TBU_CH1_SEL0: TBU_CH1 values used TBU_CH2_SEL1: TBU_CH2 values used		

MCU driver**Table 1188 Specification for GtmTbuChModuloCntrSel (continued)**

Default value	TBU_CH1_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmTbuChannelEnable		

17.3.1.16.4 GtmTbuChResolutionSel**Table 1189 Specification for GtmTbuChResolutionSel**

Name	GtmTbuChResolutionSel		
Description	Selects the resolution of time base values given by TBU_CH0_BASE. This configuration parameter is applicable only if GtmTbuChannelEnable is set to TRUE for the TBU channel0. This configuration parameter is applicable only for the TBU channel0.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	TBU_CH_LOWER_COUNT_BITS_SEL0: 0 to 23 bits of TBU_CH0_BASE is considered TBU_CH_UPPER_COUNT_BITS_SEL1: 3 to 26 bits of TBU_CH0_BASE is considered.		
Default value	TBU_CH_LOWER_COUNT_BITS_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmTbuChannelEnable		

17.3.1.16.5 GtmTbuChannelEnable**Table 1190 Specification for GtmTbuChannelEnable**

Name	GtmTbuChannelEnable
Description	Defines if TBU channels are enabled. All other configuration parameters specific to the TBU channel are disabled if this configuration parameter is set to FALSE. Values: TRUE: Channel is enabled FALSE: Channel is disabled

MCU driver**Table 1190 Specification for GtmTbuChannelEnable (continued)**

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.17 Container: GtmTomActionTimeBaseUnitConf

This container holds the configuration parameters for the actual TBU setting. The action TBU setting is required to generate a trigger that can copy from shadow register to the actual registers for period, duty cycle and channel clock source .

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.17.1 GtmTomActionTimeBaseSelection**Table 1191 Specification for GtmTomActionTimeBaseSelection**

Name	GtmTomActionTimeBaseSelection		
Description	Specifies the time base selected to compare with the value configured in GtmTomActionTimeBaseValue.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	TOM_ACT_TB_TBU_TS0: TOM group level trigger is generated when GtmTomActionTimeBaseValue matches TBU_TS0. TOM_ACT_TB_TBU_TS1: TOM group level trigger is generated when GtmTomActionTimeBaseValue matches TBU_TS1. TOM_ACT_TB_TBU_TS2: TOM group level trigger is generated when GtmTomActionTimeBaseValue matches TBU_TS2.		
Default value	TOM_ACT_TB_TBU_TS0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

MCU driver
Table 1191 Specification for GtmTomActionTimeBaseSelection (continued)

Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.17.2 GtmTomActionTimeBaseValue
Table 1192 Specification for GtmTomActionTimeBaseValue

Name	GtmTomActionTimeBaseValue		
Description	<p>Specifies the time base value for the TOM group channel level trigger.</p> <p>A trigger at the TGC level is raised when TBU_TS[x] (x can be selected through GtmActionTimeBaseSelection) value matches the value configured in this configuration parameter.</p> <p>The trigger request has to be explicitly enabled by the user by setting the TOM_TGC_ACT_TB.TB_TRIG bitfield.</p>		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.18 Container: GtmTomChannelConf

This container holds the configuration parameters for TOM channel-level parameters required to be configured globally

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.18.1 GtmTomChInternalTriggerEnable
Table 1193 Specification for GtmTomChInternalTriggerEnable

Name	GtmTomChInternalTriggerEnable
Description	<p>Enables/disables the internal trigger from channel 0 of the corresponding group channel number.</p> <p>If a channel belong to TGC0 (channel number 0 - 15), setting this configuration parameter for the corresponding channel enables trigger from channel0.</p> <p>Values:</p>

MCU driver**Table 1193 Specification for GtmTomChInternalTriggerEnable (continued)**

	TRUE: enable the internal trigger from channel 0 to 15 (based on the TGC a channel belong to). FALSE: disable the internal trigger from channel 0 to 15 (based on the TGC a channel belong to).		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.18.2 GtmTomChResetCn0OnTriggerEnable**Table 1194 Specification for GtmTomChResetCn0OnTriggerEnable**

Name	GtmTomChResetCn0OnTriggerEnable		
Description	Enables/disables the TOM channel counter CN0 value that is reset by the global trigger from any of the trigger sources. Values: TRUE: resetting of TOM channel CN0 on global trigger from any trigger source is enabled. FALSE: resetting of TOM channel CN0 on global trigger from any trigger source is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

MCU driver**17.3.1.19 Container: GtmTomGroupConf**

This container contains the configuration (parameters) for the GTM TOM group settings

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.20 Container: GtmTriggerForAdc

This container defines the binding between the GTM timers and the ADC trigger lines

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.20.1 GtmAdcTrigger0Select
Table 1195 Specification for GtmAdcTrigger0Select

Name	GtmAdcTrigger0Select		
Description	<p>Defines the GTM timer slice output connected to the adc_trig0 signal.</p> <p>The user is provided with a drop down list of 16 values conforming to the following format.</p> <p>TRIG_'VAL'_NO_TRIGGER indicating that this trigger line is electrically disconnected from possible trigger sources.</p> <p>TRIG_'VAL': 'VAL' is the value programmed into the register. TOMx or ATOMx is the module containing channel which generates the trigger.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>TRIG_0_NO_TRIGGER: No trigger is selected</p> <p>TRIG_10: Trigger 10 is selected</p> <p>TRIG_11: Trigger 11 is selected</p> <p>TRIG_12: Trigger 12 is selected</p> <p>TRIG_13: Trigger 13 is selected</p> <p>TRIG_14: Trigger 14 is selected</p> <p>TRIG_15: Trigger 15 is selected</p> <p>TRIG_1: Trigger 1 is selected</p> <p>TRIG_2: Trigger 2 is selected</p> <p>TRIG_3: Trigger 3 is selected</p> <p>TRIG_4: Trigger 4 is selected</p> <p>TRIG_5: Trigger 5 is selected</p> <p>TRIG_6: Trigger 6 is selected</p> <p>TRIG_7: Trigger 7 is selected</p> <p>TRIG_8: Trigger 8 is selected</p> <p>TRIG_9: Trigger 9 is selected</p>		
Default value	TRIG_0_NO_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1195 Specification for GtmAdcTrigger0Select (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.20.2 GtmAdcTrigger1Select**Table 1196 Specification for GtmAdcTrigger1Select**

Name	GtmAdcTrigger1Select		
Description	<p>Defines the GTM timer slice output connected to the adc_trig1 signal.</p> <p>The user is provided with a drop down list of 16 values conforming to the following format.</p> <p>TRIG_‘VAL’_NO_TRIGGER indicating that this trigger line is electrically disconnected from possible trigger sources.</p> <p>TRIG_‘VAL’: ‘VAL’ is the value programmed into the register. TOMx or ATOMx is the module containing channel which generates the trigger.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>TRIG_0_NO_TRIGGER: No trigger is selected</p> <p>TRIG_10: Trigger 10 is selected</p> <p>TRIG_11: Trigger 11 is selected</p> <p>TRIG_12: Trigger 12 is selected</p> <p>TRIG_13: Trigger 13 is selected</p> <p>TRIG_14: Trigger 14 is selected</p> <p>TRIG_15: Trigger 15 is selected</p> <p>TRIG_1: Trigger 1 is selected</p> <p>TRIG_2: Trigger 2 is selected</p> <p>TRIG_3: Trigger 3 is selected</p> <p>TRIG_4: Trigger 4 is selected</p> <p>TRIG_5: Trigger 5 is selected</p> <p>TRIG_6: Trigger 6 is selected</p> <p>TRIG_7: Trigger 7 is selected</p> <p>TRIG_8: Trigger 8 is selected</p> <p>TRIG_9: Trigger 9 is selected</p>		
Default value	TRIG_0_NO_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

MCU driver**Table 1196 Specification for GtmAdcTrigger1Select (continued)**

Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.20.3 GtmAdcTrigger2Select**Table 1197 Specification for GtmAdcTrigger2Select**

Name	GtmAdcTrigger2Select		
Description	<p>Defines the GTM timer slice output connected to the adc_trig2 signal.</p> <p>The user is provided with a drop down list of 16 values conforming to the following format.</p> <p>TRIG_'VAL'_NO_TRIGGER indicating that this trigger line is electrically disconnected from possible trigger sources.</p> <p>TRIG_'VAL': 'VAL' is the value programmed into the register. TOMx or ATOMx is the module containing channel which generates the trigger.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>TRIG_0_NO_TRIGGER: No Trigger is selected</p> <p>TRIG_10: Trigger 10 is selected</p> <p>TRIG_11: Trigger 11 is selected</p> <p>TRIG_12: Trigger 12 is selected</p> <p>TRIG_13: Trigger 13 is selected</p> <p>TRIG_14: Trigger 14 is selected</p> <p>TRIG_15: Trigger 15 is selected</p> <p>TRIG_1: Trigger 1 is selected</p> <p>TRIG_2: Trigger 2 is selected</p> <p>TRIG_3: Trigger 3 is selected</p> <p>TRIG_4: Trigger 4 is selected</p> <p>TRIG_5: Trigger 5 is selected</p> <p>TRIG_6: Trigger 6 is selected</p> <p>TRIG_7: Trigger 7 is selected</p> <p>TRIG_8: Trigger 8 is selected</p> <p>TRIG_9: Trigger 9 is selected</p>		
Default value	TRIG_0_NO_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

MCU driver

17.3.1.20.4 GtmAdcTrigger3Select**Table 1198 Specification for GtmAdcTrigger3Select**

Name	GtmAdcTrigger3Select		
Description	<p>Defines the GTM timer slice output connected to the adc_trig3 signal.</p> <p>The user is provided with a drop down list of 16 values conforming to the following format.</p> <p>TRIG_'VAL'_NO_TRIGGER indicating that this trigger line is electrically disconnected from possible trigger sources.</p> <p>TRIG_'VAL': 'VAL' is the value programmed into the register. TOMx or ATOMx is the module containing channel which generates the trigger.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>TRIG_0_NO_TRIGGER: No trigger is selected</p> <p>TRIG_10: Trigger 10 is selected</p> <p>TRIG_11: Trigger 11 is selected</p> <p>TRIG_12: Trigger 12 is selected</p> <p>TRIG_13: Trigger 13 is selected</p> <p>TRIG_14: Trigger 14 is selected</p> <p>TRIG_15: Trigger 15 is selected</p> <p>TRIG_1: Trigger 1 is selected</p> <p>TRIG_2: Trigger 2 is selected</p> <p>TRIG_3: Trigger 3 is selected</p> <p>TRIG_4: Trigger 4 is selected</p> <p>TRIG_5: Trigger 5 is selected</p> <p>TRIG_6: Trigger 6 is selected</p> <p>TRIG_7: Trigger 7 is selected</p> <p>TRIG_8: Trigger 8 is selected</p> <p>TRIG_9: Trigger 9 is selected</p>		
Default value	TRIG_0_NO_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.20.5 GtmAdcTrigger4Select**Table 1199 Specification for GtmAdcTrigger4Select**

Name	GtmAdcTrigger4Select
-------------	----------------------

MCU driver**Table 1199 Specification for GtmAdcTrigger4Select (continued)**

Description	Defines the GTM timer slice output connected to the adc_trig4 signal. The user is provided with a drop down list of 16 values conforming to the following format. TRIG_'VAL'_NO_TRIGGER indicating that this trigger line is electrically disconnected from possible trigger sources. TRIG_'VAL': 'VAL' is the value programmed into the register. TOMx or ATOMx is the module containing channel which generates the trigger.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	TRIG_0_NO_TRIGGER: No trigger is selected TRIG_10: Trigger 10 is selected TRIG_11: Trigger 11 is selected TRIG_12: Trigger 12 is selected TRIG_13: Trigger 13 is selected TRIG_14: Trigger 14 is selected TRIG_15: Trigger 15 is selected TRIG_1: Trigger 1 is selected TRIG_2: Trigger 2 is selected TRIG_3: Trigger 3 is selected TRIG_4: Trigger 4 is selected TRIG_5: Trigger 5 is selected TRIG_6: Trigger 6 is selected TRIG_7: Trigger 7 is selected TRIG_8: Trigger 8 is selected TRIG_9: Trigger 9 is selected		
Default value	TRIG_0_NO_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.21 Container: GtmTriggerForDsadc

This container defines the binding between the GTM timers and the DSADC trigger lines

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

MCU driver

17.3.1.21.1 GtmDsadcTrigger0Select**Table 1200 Specification for GtmDsadcTrigger0Select**

Name	GtmDsadcTrigger0Select		
Description	<p>Defines the GTM timer slice output connected to the Dsadc_trig0 signal.</p> <p>The user is provided with a drop down list of 16 values conforming to the following format.</p> <p>TRIG_[VAL]_NO_TRIGGER indicating that this trigger line is electrically disconnected from possible trigger sources.</p> <p>TRIG_[VAL]: [VAL] is the value programmed into the register. TOMx or ATOMx is the module containing channel which generates the trigger.</p> <p>The value of this parameter should be unique across all containers only when DSADC module is configured.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>TRIG_0_NO_TRIGGER: Trigger 0 indicates no trigger is selected</p> <p>TRIG_10: Trigger 10 is selected</p> <p>TRIG_11: Trigger 11 is selected</p> <p>TRIG_12: Trigger 12 is selected</p> <p>TRIG_13: Trigger 13 is selected</p> <p>TRIG_14: Trigger 14 is selected</p> <p>TRIG_15: Trigger 15 is selected</p> <p>TRIG_1: Trigger 1 is selected</p> <p>TRIG_2: Trigger 2 is selected</p> <p>TRIG_3: Trigger 3 is selected</p> <p>TRIG_4: Trigger 4 is selected</p> <p>TRIG_5: Trigger 5 is selected</p> <p>TRIG_6: Trigger 6 is selected</p> <p>TRIG_7: Trigger 7 is selected</p> <p>TRIG_8: Trigger 8 is selected</p> <p>TRIG_9: Trigger 9 is selected</p>		
Default value	TRIG_0_NO_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.21.2 GtmDsadcTrigger1Select**Table 1201 Specification for GtmDsadcTrigger1Select**

Name	GtmDsadcTrigger1Select		
Description	<p>Defines the GTM timer slice output connected to the Dsadc_trig1 signal.</p> <p>The user is provided with a drop down list of 16 values conforming to the following format.</p> <p>TRIG_[VAL]_NO_TRIGGER indicating that this trigger line is electrically disconnected from possible trigger sources.</p> <p>TRIG_[VAL]: [VAL] is the value programmed into the register. TOMx or ATOMx is the module containing channel which generates the trigger.</p> <p>The value of this parameter should be unique across all containers only when DSADC module is configured.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>TRIG_0_NO_TRIGGER: Trigger 0 indicates no trigger is selected</p> <p>TRIG_10: Trigger 10 is selected</p> <p>TRIG_11: Trigger 11 is selected</p> <p>TRIG_12: Trigger 12 is selected</p> <p>TRIG_13: Trigger 13 is selected</p> <p>TRIG_14: Trigger 14 is selected</p> <p>TRIG_15: Trigger 15 is selected</p> <p>TRIG_1: Trigger 1 is selected</p> <p>TRIG_2: Trigger 2 is selected</p> <p>TRIG_3: Trigger 3 is selected</p> <p>TRIG_4: Trigger 4 is selected</p> <p>TRIG_5: Trigger 5 is selected</p> <p>TRIG_6: Trigger 6 is selected</p> <p>TRIG_7: Trigger 7 is selected</p> <p>TRIG_8: Trigger 8 is selected</p> <p>TRIG_9: Trigger 9 is selected</p>		
Default value	TRIG_0_NO_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.21.3 GtmDsadcTrigger2Select**Table 1202 Specification for GtmDsadcTrigger2Select**

Name	GtmDsadcTrigger2Select		
Description	<p>Defines the GTM timer slice output connected to the Dsadc_trig2 signal.</p> <p>The user is provided with a drop down list of 16 values conforming to the following format.</p> <p>TRIG_[VAL]_NO_TRIGGER indicating that this trigger line is electrically disconnected from possible trigger sources.</p> <p>TRIG_[VAL]: [VAL] is the value programmed into the register. TOMx or ATOMx is the module containing channel which generates the trigger.</p> <p>The value of this parameter should be unique across all containers only when DSADC module is configured.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>TRIG_0_NO_TRIGGER: Trigger 0 indicates no trigger is selected</p> <p>TRIG_10: Trigger 10 is selected</p> <p>TRIG_11: Trigger 11 is selected</p> <p>TRIG_12: Trigger 12 is selected</p> <p>TRIG_13: Trigger 13 is selected</p> <p>TRIG_14: Trigger 14 is selected</p> <p>TRIG_15: Trigger 15 is selected</p> <p>TRIG_1: Trigger 1 is selected</p> <p>TRIG_2: Trigger 2 is selected</p> <p>TRIG_3: Trigger 3 is selected</p> <p>TRIG_4: Trigger 4 is selected</p> <p>TRIG_5: Trigger 5 is selected</p> <p>TRIG_6: Trigger 6 is selected</p> <p>TRIG_7: Trigger 7 is selected</p> <p>TRIG_8: Trigger 8 is selected</p> <p>TRIG_9: Trigger 9 is selected</p>		
Default value	TRIG_0_NO_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.21.4 GtmDsadcTrigger3Select****Table 1203 Specification for GtmDsadcTrigger3Select**

Name	GtmDsadcTrigger3Select		
Description	<p>Defines the GTM timer slice output connected to the Dsadc_trig3 signal.</p> <p>The user is provided with a drop down list of 16 values conforming to the following format.</p> <p>TRIG_[VAL]_NO_TRIGGER indicating that this trigger line is electrically disconnected from possible trigger sources.</p> <p>TRIG_[VAL]: [VAL] is the value programmed into the register. TOMx or ATOMx is the module containing channel which generates the trigger.</p> <p>The value of this parameter should be unique across all containers only when DSADC module is configured.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>TRIG_0_NO_TRIGGER: Trigger 0 indicates no trigger is selected</p> <p>TRIG_1: Trigger 1 is selected</p> <p>TRIG_2: Trigger 2 is selected</p> <p>TRIG_3: Trigger 3 is selected</p> <p>TRIG_4: Trigger 4 is selected</p>		
Default value	TRIG_0_NO_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.22 Container: Mcu

Configuration of the Mcu (Microcontroller Unit) module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.22.1 Config Variant**Table 1204 Specification for Config Variant**

Name	Config Variant		
Description	<p>Selects the config-variant for the MCU module.</p> <p>The default value of this parameter is set to VariantPostBuild as per AUTOSAR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef

MCU driver
Table 1204 Specification for Config Variant (continued)

Range	VariantPostBuild:		
Default value	VariantPostBuild		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.23 Container: McuAscLinChannelAllocationConf

This container holds the ASCLIN channel allocation to different MCAL drivers.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.23.1 McuAscLinChannelAllocationConf

Table 1205 Specification for McuAscLinChannelAllocationConf

Name	McuAscLinChannelAllocationConf		
Description	Specifies which driver(s) have used or not used this particular AscLin channel Note: Availability of the module is based on the Release Notes.		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	ASCLIN_CH_NOT_USED: ASCLIN channel is not reserved for any driver ASCLIN_CH_USED_BY_LIN_DRIVER: ASCLIN channel is reserved for the LIN driver ASCLIN_CH_USED_BY_UART_DRIVER: ASCLIN channel is reserved for the UART driver		
Default value	ASCLIN_CH_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.24 Container: McuAscLinAllocationConf

This container holds the ASCLIN channel allocation to different MCAL drivers. Note: Availability of the module is based on the Release Notes

Post-Build Variant Multiplicity: FALSE

MCU driver

Multiplicity Configuration Class: Pre-Compile

17.3.1.25 Container: McuCcu6ModuleAllocationConf

This container holds the CCU6 kernel allocation to different MCAL drivers User is not allowed to change the name of the parameter in this container

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.25.1 McuCcu6ModuleAllocationConf

Table 1206 Specification for McuCcu6ModuleAllocationConf

Name	McuCcu6ModuleAllocationConf		
Description	Specifies which driver have used this particular CCU6 module or this module is not used by any driver (unused). User is not allowed to change the name of the parameter		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CCU6_MODULE_NOT_USED: CCU6 kernel is not used CCU6_MODULE_USED_BY_ADC_DRIVER: CCU6 kernel is reserved for the ADC driver CCU6_MODULE_USED_BY_ICU_DRIVER: CCU6 kernel is reserved for the ICU driver CCU6_MODULE_USED_BY_PWM_DRIVER: CCU6 kernel is reserved for the PWM driver		
Default value	CCU6_MODULE_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.26 Container: McuClockReferencePoint

This container defines a reference point in the MCU clock tree. This container defines the frequency which then can be used by other modules as an input value. Lower multiplicity is 1, as even in the simplest case (only one frequency is used), there is one frequency to be defined.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

17.3.1.26.1 McuClockRefSelection

Table 1207 Specification for McuClockRefSelection

Name	McuClockRefSelection
-------------	----------------------

MCU driver**Table 1207 Specification for McuClockRefSelection (continued)**

Description	Selects the source of clock reference, based on which McuClockReferencePointFrequency is populated with frequency		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_ADAS_FREQUENCY: ADAS frequency MCU_ADC_FREQUENCY: ADC frequency MCU_ASCLINFAST_FREQUENCY: ASCLIN FAST frequency MCU_ASCLINSLOW_FREQUENCY: ASCLIN SLOW frequency MCU_BBB_FREQUENCY: Back Bone Bus frequency MCU_CPU0_FREQUENCY: CPU0 frequency MCU_CPU1_FREQUENCY: CPU1 frequency MCU_CPU2_FREQUENCY: CPU2 frequency MCU_CPU3_FREQUENCY: CPU3 frequency MCU_CPU4_FREQUENCY: CPU4 frequency MCU_CPU5_FREQUENCY: CPU5 frequency MCU_EBU_FREQUENCY: EBU frequency MCU_ERAY_FREQUENCY: ERAY frequency MCU_FSI2_FREQUENCY: FSI2 frequency MCU_FSI_FREQUENCY: FSI frequency MCU_GETH_FREQUENCY: Gigabit Ethernet frequency MCU_GTM_FREQUENCY: GTM frequency MCU_HSCT_FREQUENCY: HSCT frequency MCU_HSPDM160_FREQUENCY: HSPDM160 frequency MCU_HSPDM320_FREQUENCY: HSPDM320 frequency MCU_I2C_FREQUENCY: I2C frequency MCU_MCANH_FREQUENCY: MCANH frequency MCU_MCAN_FREQUENCY: MCAN frequency MCU_MSC_FREQUENCY: MSC frequency MCU_QSPI_FREQUENCY: QSPI frequency MCU_REF_FREQUENCY_1: REFERENCE 1 frequency MCU_REF_FREQUENCY_2: REFERENCE 2 frequency MCU_SOURCE0_FREQUENCY: fSource0 frequency MCU_SOURCE1_FREQUENCY: fSource1 frequency MCU_SOURCE2_FREQUENCY: fSource2 frequency MCU_SPB_FREQUENCY: SPB frequency MCU_SRI_FREQUENCY: SRI frequency MCU_STM_FREQUENCY: STM frequency MCU_USER_DEFINED_FREQUENCY: Frequency defined by user.		
Default value	MCU_USER_DEFINED_FREQUENCY		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver
Table 1207 Specification for McuClockRefSelection (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.26.2 McuClockReferencePointFrequency
Table 1208 Specification for McuClockReferencePointFrequency

Name	McuClockReferencePointFrequency		
Description	<p>Defines the frequency for the specific instance of the McuClockReferencePoint container. The frequency is always expressed in Hertz (Hz).</p> <p>The frequency is already calculated in Infineon defined containers.</p> <p>The value entered here by the user will not be validated and is only for information purpose</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0 - 320000000		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	McuClockRefSelection		

17.3.1.27 Container: McuClockReferencePointConfig

This container holds sub-container for the configuration of the MCU clock tree.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.28 Container: McuClockSettingConfig

This container contains the configuration (parameters) for the clock settings of the MCU.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

17.3.1.28.1 McuClockSettingId
Table 1209 Specification for McuClockSettingId

Name	McuClockSettingId
-------------	-------------------

MCU driver
Table 1209 Specification for McuClockSettingId (continued)

Description	The Id of this parameter is used as an argument for the Mcu_InitClock() API call.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.29 Container: McuDemEventParameterRefs

This is a container for the references to the DemEventParameter elements which are invoked using the Dem_ReportErrorStatus() API in case the corresponding errors occur. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic name. The standardized errors are provided in the container and can be extended by vendor-specific error references.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

17.3.1.29.1 MCU_E_CLOCK_FAILURE

Table 1210 Specification for MCU_E_CLOCK_FAILURE

Name	MCU_E_CLOCK_FAILURE		
Description	Provides the provision to enable or disable the production error event on clock failure reported through DEM. This configuration container is kept disabled, just to conform to AUTOSAR schema model.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.30 Container: McuDemEventParameterRefsConf

This is a container for the references to the DemEventParameter elements which are invoked using the Dem_ReportErrorStatus() API in case the corresponding errors occur. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic name. The standardized errors are provided in the container and can be extended by vendor-specific error references. All DEM event parameters are implemented as pre compile parameters.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

17.3.1.30.1 MCU_E_CCU6_CLC_DISABLE_ERR

Table 1211 Specification for MCU_E_CCU6_CLC_DISABLE_ERR

Name	MCU_E_CCU6_CLC_DISABLE_ERR		
Description	This error is reported when the CCU6 kernel CLC bit cannot be turned OFF within the specified time.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.30.2 MCU_E_CCU6_CLC_ENABLE_ERR

Table 1212 Specification for MCU_E_CCU6_CLC_ENABLE_ERR

Name	MCU_E_CCU6_CLC_ENABLE_ERR		
Description	This error is reported when the CCU6 kernel CLC bit cannot be turned ON within the specified time.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile

MCU driver**Table 1212 Specification for MCU_E_CCU6_CLC_ENABLE_ERR (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.30.3 MCU_E_CCUCON_UPDATE_ERR**Table 1213 Specification for MCU_E_CCUCON_UPDATE_ERR**

Name	MCU_E_CCUCON_UPDATE_ERR		
Description	This error is reported when the LCK bit is not reset within the specified time		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.30.4 MCU_E_CONVCTRL_CLC_DISABLE_ERR**Table 1214 Specification for MCU_E_CONVCTRL_CLC_DISABLE_ERR**

Name	MCU_E_CONVCTRL_CLC_DISABLE_ERR		
Description	This error is reported when the CONVCTRL CLC bit cannot be turned OFF within the specified time.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.30.5 MCU_E_CONVCTRL_CLC_ENABLE_ERR****Table 1215 Specification for MCU_E_CONVCTRL_CLC_ENABLE_ERR**

Name	MCU_E_CONVCTRL_CLC_ENABLE_ERR		
Description	This error is reported if the CONVCTRL CLC bit cannot be turned ON within the specified time.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.30.6 MCU_E_GPT12_CLC_DISABLE_ERR**Table 1216 Specification for MCU_E_GPT12_CLC_DISABLE_ERR**

Name	MCU_E_GPT12_CLC_DISABLE_ERR		
Description	This error is reported if the GPT12 CLC bit cannot be turned OFF within the specified time.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.30.7 MCU_E_GPT12_CLC_ENABLE_ERR**Table 1217 Specification for MCU_E_GPT12_CLC_ENABLE_ERR**

Name	MCU_E_GPT12_CLC_ENABLE_ERR
Description	This error is reported if the GPT12 CLC bit cannot be turned ON within the specified time.

MCU driver**Table 1217 Specification for MCU_E_GPT12_CLC_ENABLE_ERR (continued)**

Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.30.8 MCU_E_GTM_CLC_DISABLE_ERR**Table 1218 Specification for MCU_E_GTM_CLC_DISABLE_ERR**

Name	MCU_E_GTM_CLC_DISABLE_ERR		
Description	This error is reported if the GTM CLC bit cannot be turned OFF within the specified time.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.30.9 MCU_E_GTM_CLC_ENABLE_ERR**Table 1219 Specification for MCU_E_GTM_CLC_ENABLE_ERR**

Name	MCU_E_GTM_CLC_ENABLE_ERR		
Description	This error is reported if the GTM CLC bit cannot be turned ON within the specified time.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		

MCU driver**Table 1219 Specification for MCU_E_GTM_CLC_ENABLE_ERR (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.30.10 MCU_E_OSC_FAILURE**Table 1220 Specification for MCU_E_OSC_FAILURE**

Name	MCU_E_OSC_FAILURE		
Description	<p>This error is reported when the oscillator develops a failure. This error can be reported both at Init as well as run time.</p> <p>MCU_E_OSC_FAILURE can only be enabled if the ClockSourceFailureNotification parameter is enabled provided that the Mcu_InitClock() API is available</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	McuInitClock, McuClockSourceFailureNotification		

17.3.1.30.11 MCU_E_PERIPHERAL_PLL_LOCK_LOSS**Table 1221 Specification for MCU_E_PERIPHERAL_PLL_LOCK_LOSS**

Name	MCU_E_PERIPHERAL_PLL_LOCK_LOSS		
Description	This error is reported at run time when the peripheral PLL develops loss of lock. This error can only be enabled if the parameter ClockSourceFailureNotification is enabled		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		

MCU driver**Table 1221 Specification for MCU_E_PERIPHERAL_PLL_LOCK_LOSS (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	McuClockSourceFailureNotification		

17.3.1.30.12 MCU_E_PERIPHERAL_PLL_TIMEOUT_ERR**Table 1222 Specification for MCU_E_PERIPHERAL_PLL_TIMEOUT_ERR**

Name	MCU_E_PERIPHERAL_PLL_TIMEOUT_ERR		
Description	This error is reported when the peripheral PLL does not lock within the specified time during the clock initialization. This error can only be enabled if the ClockSourceFailureNotification parameter is enabled		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	McuClockSourceFailureNotification		

17.3.1.30.13 MCU_E_PMSWCR_UPDATE_ERR**Table 1223 Specification for MCU_E_PMSWCR_UPDATE_ERR**

Name	MCU_E_PMSWCR_UPDATE_ERR		
Description	This error is reported when the PMSWCRx register cannot be written because the BUSY bit is always set (register update is not allowed)		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE

MCU driver**Table 1223 Specification for MCU_E_PMSWCR_UPDATE_ERR (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.30.14 MCU_E_SYSTEM_PLL_LOCK_LOSS**Table 1224 Specification for MCU_E_SYSTEM_PLL_LOCK_LOSS**

Name	MCU_E_SYSTEM_PLL_LOCK_LOSS		
Description	This error is reported at run time when the system PLL develops loss of lock. This error can only be enabled if the ClockSourceFailureNotification parameter is enabled		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	McuClockSourceFailureNotification		

17.3.1.30.15 MCU_E_SYSTEM_PLL_TIMEOUT_ERR**Table 1225 Specification for MCU_E_SYSTEM_PLL_TIMEOUT_ERR**

Name	MCU_E_SYSTEM_PLL_TIMEOUT_ERR		
Description	This error is reported when the System PLL does not lock within the specified time during clock initialization sequence. This error can only be enabled if the ClockSourceFailureNotification parameter is enabled		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE

MCU driver**Table 1225 Specification for MCU_E_SYSTEM_PLL_TIMEOUT_ERR (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	McuClockSourceFailureNotification		

17.3.1.31 Container: McuEruAllocationConf

This container holds the ownership information of the input(ERS) and the output(OGU) channels of the ERU
User is not allowed to change the name of the parameters in this container

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.32 Container: McuEruChannelInputLineConf

This container holds the ownership information of the input (ERS) channels of the ERU.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.32.1 McuEruChannelInputLineConf**Table 1226 Specification for McuEruChannelInputLineConf**

Name	McuEruChannelInputLineConf		
Description	Specifies the user of this particular ERU input line		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ERU_CHANNEL_INP_NOT_USED: ERU input channel is not used ERU_CHANNEL_INP_USED_BY_ADC_DRIVER: ERU input channel is reserved for the ADC driver ERU_CHANNEL_INP_USED_BY_DSADC_DRIVER: ERU input channel is reserved for the DSADC driver ERU_CHANNEL_INP_USED_BY_ICU_DRIVER: ERU input channel is reserved for the ICU driver		
Default value	ERU_CHANNEL_INP_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

MCU driver

17.3.1.33 Container: McuEruChannelOutputUnitConf

This container holds the ownership information of the output (OGU) channels of the ERU

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.33.1 McuEruChannelOutputUnitConf

Table 1227 Specification for McuEruChannelOutputUnitConf

Name	McuEruChannelOutputUnitConf		
Description	Specifies the user of this particular ERU output line.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ERU_CHANNEL_OUT_NOT_USED: ERU output channel is not used ERU_CHANNEL_OUT_USED_BY_ADC_DRIVER: ERU output channel is reserved for the ADC driver ERU_CHANNEL_OUT_USED_BY_DSADC_DRIVER: ERU output channel is reserved for the DSADC driver ERU_CHANNEL_OUT_USED_BY_ICU_DRIVER: ERU output channel is reserved for the ICU driver		
Default value	ERU_CHANNEL_OUT_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.34 Container: McuEruGlobalConf

This container holds the input filter configuration parameters of the ERU.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.34.1 McuEruInputFilterRegVal

Table 1228 Specification for McuEruInputFilterRegVal

Name	McuEruInputFilterRegVal
Description	Enables/disables the glitch filter and also the glitch filter pre-divider and filters depth. (EIFILT register) A value of zero in this register disables all glitch filtering.

MCU driver
Table 1228 Specification for McuEruInputFilterRegVal (continued)

	In case 0 is passed for bit fields which are reserved according to the Target Specification, the value will be masked out.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4278321151		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.35 Container: McuExternalClockOutputConfig

This container defines the configuration (parameters) for the external clock out of the MCU.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.35.1 McuExtClock0Enable

Table 1229 Specification for McuExtClock0Enable

Name	McuExtClock0Enable		
Description	Enables/disables the EXTCLK0 signal. Values : TRUE: EXTCLK0 signal is available on the external pad FALSE: EXTCLK0 signal is not available on the external pad		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.35.2 McuExtClock1Enable****Table 1230 Specification for McuExtClock1Enable**

Name	McuExtClock1Enable		
Description	Enables/disables the EXTCLK1 signal. Values : TRUE: EXTCLK1 signal is available on the external pad FALSE: EXTCLK1 signal is not available on the external pad		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.35.3 McuExtClock1Inverted**Table 1231 Specification for McuExtClock1Inverted**

Name	McuExtClock1Inverted		
Description	Enables/disables the inversion of EXTCLK1. Values : TRUE: output signal is inverted of the actual signal for the EXTCLK1 FALSE: output signal is not inverted of the actual signal for the EXTCLK1		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCU driver**Table 1231 Specification for McuExtClock1Inverted (continued)**

Dependency	McuExtClock1Enable
-------------------	--------------------

17.3.1.35.4 McuExtClockOutSel0**Table 1232 Specification for McuExtClockOutSel0**

Name	McuExtClockOutSel0		
Description	Specifies the clock source that is selected as the output for EXTCLK0 Note: ALT mode for corresponding port pin must be configured in the PORT driver to observe the output at a port pin.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	BACKUP_EXT_CLOCK0_SEL4: fBACK is selected for EXTCLK0 BBB_EXT_CLOCK0_SEL6: fBBB is selected for EXTCLK0 ERAY_MT0_EXT_CLOCK0_SEL15: fERAY is selected for EXTCLK0 FOUT_EXT_CLOCK0_SEL0: fOUT is selected for EXTCLK0 FSI2_EXT_CLOCK0_SEL14: fFSI2 is selected for EXTCLK0 FSI_EXT_CLOCK0_SEL10: fFSI is selected for EXTCLK0 GTM_EXT_CLOCK0_SEL12: fGTM is selected for EXTCLK0 OSC0_EXT_CLOCK0_SEL3: fOSC0 is selected for EXTCLK0 PLL0_EXT_CLOCK0_SEL1: fPLL0 is selected for EXTCLK0 PLL1_EXT_CLOCK0_SEL2: fPLL1 is selected for EXTCLK0 PLL2_EXT_CLOCK0_SEL5: fPLL2 is selected for EXTCLK0 SPB_EXT_CLOCK0_SEL9: fSPB is selected for EXTCLK0 SRI_EXT_CLOCK0_SEL8: fSRI is selected for EXTCLK0 STM_EXT_CLOCK0_SEL11: fSTM is selected for EXTCLK0 TCK_EXT_CLOCK0_SEL13: fTCK is selected for EXTCLK0		
Default value	FOUT_EXT_CLOCK0_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuExtClock0Enable		

17.3.1.35.5 McuExtClockOutSel1**Table 1233 Specification for McuExtClockOutSel1**

Name	McuExtClockOutSel1
-------------	--------------------

MCU driver**Table 1233 Specification for McuExtClockOutSel1 (continued)**

Description	Specifies the clock source that is selected as the output for EXTCLK1 Note: ALT mode for corresponding port pin must be configured in the PORT driver to observe the output at a port pin.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ADC_EXT_CLOCK1_SEL6: fADC is selected for EXTCLK1 ASCLINF_EXT_CLOCK1_SEL13: fASCLINF is selected for EXTCLK1 ASCLINS_EXT_CLOCK1_SEL14: fASCLINS is selected for EXTCLK1 BACKUP_EXT_CLOCK1_SEL4: fBACK is selected for EXTCLK1 EBU_EXT_CLOCK1_SEL3: fEBU is selected for EXTCLK1 ERAY_EXT_CLOCK1_SEL12: fERAY is selected for EXTCLK1 FOUT_EXT_CLOCK1_SEL0: fOUT is selected for EXTCLK1 I2C_EXT_CLOCK1_SEL10: fI2C is selected for EXTCLK1 MCAN_EXT_CLOCK1_SEL5: fMCAN is selected for EXTCLK1 MSC_EXT_CLOCK1_SEL11: fMSC is selected for EXTCLK1 OSCFL_EXT_CLOCK1_SEL15: fOSCFL is selected for EXTCLK1 PLL0_EXT_CLOCK1_SEL1: fPLL0 is selected for EXTCLK1 PLL1_EXT_CLOCK1_SEL2: fPLL1 is selected for EXTCLK1 QSPI_EXT_CLOCK1_SEL7: fQSPI is selected for EXTCLK1 SPB_EXT_CLOCK1_SEL9: fSPB is selected for EXTCLK1 SRI_EXT_CLOCK1_SEL8: fSRI is selected for EXTCLK1		
Default value	FOUT_EXT_CLOCK1_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build		
Origin	IFX	Scope	LOCAL
Dependency	McuExtClock1Enable		

17.3.1.35.6 McuFoutClockDiv**Table 1234 Specification for McuFoutClockDiv**

Name	McuFoutClockDiv
Description	<p>Determines the divider for fOUT clock (for EXTCLK1 only). The fOUT frequency for EXTCLK1 can be calculated as below:</p> $fOUT = fSPB / \text{McuFoutClockDiv}.$ <p>Note: McuFoutClockDiv value is editable and considered for calculation when McuExtClockOutSel1 is set to FOUT_EXT_CLOCK1_SEL0 and McuExtClock1Enable is set to True.</p>

MCU driver
Table 1234 Specification for McuFoutClockDiv (continued)

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 256		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuExtClockOutSel1, McuExtClock1Enable		

17.3.1.36 Container: McuGeneralConfiguration

This container holds the general configuration parameters of the MCU driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.36.1 McuCCU61SleepModeEnabled

Table 1235 Specification for McuCCU61SleepModeEnabled

Name	McuCCU61SleepModeEnabled		
Description	Specifies whether CCU6 kernel 1 is configured to go to sleep or not. TRUE: CCU6 kernel 1 will go to sleep when system is put to sleep. FALSE: CCU6 kernel 1 will not go to sleep when system is put to sleep.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.36.2 McuCcu60SleepModeEnabled****Table 1236 Specification for McuCcu60SleepModeEnabled**

Name	McuCcu60SleepModeEnabled		
Description	Specifies whether CCU6 kernel 0 is configured to go to sleep or not. TRUE: CCU6 kernel 0 will go to sleep when system is put to sleep. FALSE: CCU6 kernel 0 will not go to sleep when system is put to sleep.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.3 McuClearColdResetStatusApi**Table 1237 Specification for McuClearColdResetStatusApi**

Name	McuClearColdResetStatusApi		
Description	Pre-processor switch to enable/disable the Mcu_ClearColdResetStatus() API Values: TRUE: enables Mcu_ClearColdResetStatus FALSE: disables Mcu_ClearColdResetStatus		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.36.4 McuClockSourceFailureNotification****Table 1238 Specification for McuClockSourceFailureNotification**

Name	McuClockSourceFailureNotification		
Description	Clock failure related DEMs are reported to the application when this parameter is enabled. Values: TRUE: Clock failure-related DEMs are reported FALSE: Clock failure-related DEMs are not reported		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.5 McuDevErrorDetect**Table 1239 Specification for McuDevErrorDetect**

Name	McuDevErrorDetect		
Description	Pre-processor switch for enabling the development error detection and reporting Values: TRUE: Development error detection is enabled FALSE: Development error detection is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL

MCU driver**Table 1239 Specification for McuDevErrorDetect (continued)**

Dependency	-
-------------------	---

17.3.1.36.6 McuGetRamStateApi**Table 1240 Specification for McuGetRamStateApi**

Name	McuGetRamStateApi		
Description	Pre-processor switch to enable/disable the Mcu_GetRamState API. Values: TRUE: Mcu_GetRamState() is enabled FALSE: Mcu_GetRamState() is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.36.7 McuGpt12SleepModeEnabled**Table 1241 Specification for McuGpt12SleepModeEnabled**

Name	McuGpt12SleepModeEnabled		
Description	Specifies whether GPT12 is configured to go to sleep or not. TRUE: GPT12 will go to sleep when system is put to sleep. FALSE: GPT12 will not go to sleep when system is put to sleep.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

MCU driver**Table 1241 Specification for McuGpt12SleepModeEnabled (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.8 McuGtmSleepModeEnabled**Table 1242 Specification for McuGtmSleepModeEnabled**

Name	McuGtmSleepModeEnabled		
Description	Specifies if GTM peripheral has to go into the Sleep mode when the complete system is put into the Sleep mode. TRUE: enables the Sleep mode for the GTM peripheral FALSE : disables the Sleep mode for the GTM peripheral		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.36.9 McuIdleModeCpuCore**Table 1243 Specification for McuIdleModeCpuCore**

Name	McuIdleModeCpuCore		
Description	Defines which core can trigger the Idle mode.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CPU_IDLE_CORE0_SEL1: CPU0 Idle request will send all CPUs in the Idle state CPU_IDLE_CORE1_SEL2: CPU1 Idle request will send all CPUs in the Idle state CPU_IDLE_CORE2_SEL3: CPU2 Idle request will send all CPUs in the Idle state CPU_IDLE_CORE3_SEL4: CPU3 Idle request will send all CPUs in the Idle state CPU_IDLE_CORE4_SEL5: CPU4 Idle request will send all CPUs in the Idle state		

MCU driver**Table 1243 Specification for McuIdleModeCpuCore (continued)**

	CPU_IDLE_CORE5_SEL6: CPU5 Idle request will send all CPUs in the Idle state INDIVIDUAL_IDLE_CORES_SEL0: Entry to the respective Idle mode is decided by each individual CPU		
Default value	INDIVIDUAL_IDLE_CORES_SEL0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.10 McuIfxCpuCcuonApi**Table 1244 Specification for McuIfxCpuCcuonApi**

Name	McuIfxCpuCcuonApi		
Description	Enables/disables the availability of CPU clock configuration register update API defined by Infineon namely Mcu_UpdateCpuCcuonReg Values: TRUE: CPU clock configuration register update API is available FALSE: CPU clock configuration register update API is not available		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.11 McuIfxDeInitApi**Table 1245 Specification for McuIfxDeInitApi**

Name	McuIfxDeInitApi
Description	Enables/disables the availability of MCU de-initialization API, Mcu_DeInit

MCU driver**Table 1245 Specification for McuIfxDeInitApi (continued)**

	Values: TRUE: Mcu_DeInit() API is available FALSE: Mcu_DeInit() API is not available		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.12 McuIfxLpmApi**Table 1246 Specification for McuIfxLpmApi**

Name	McuIfxLpmApi		
Description	Enables/disables the availability of low power mode APIs defined by Infineon namely Mcu_GetCpudleModelInitiator, Mcu_GetCpuState, Mcu_GetWakeupCause and Mcu_ClearWakeupCause. Values: TRUE: Low power mode APIs are available FALSE: Low power mode APIs are not available		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.36.13 McuIfxTrapApi****Table 1247 Specification for McuIfxTrapApi**

Name	McuIfxTrapApi		
Description	Enables/disables the availability of trap related APIs defined by Infineon namely Mcu_GetTrapCause, Mcu_SetTrapRequest and Mcu_ClearTrapRequest Values: TRUE: Trap-related APIs are available FALSE: Trap-related APIs are not available		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.14 McuInitCheckApi**Table 1248 Specification for McuInitCheckApi**

Name	McuInitCheckApi		
Description	Enables/disables the availability of the Mcu_InitCheck() API. Values: TRUE: Mcu_InitCheck() API is available FALSE: Mcu_InitCheck() API is not available		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

MCU driver**Table 1248 Specification for McuInitCheckApi (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.15 McuInitClock**Table 1249 Specification for McuInitClock**

Name	McuInitClock		
Description	If McuInitClock 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. Values: TRUE: Mcu_InitClock() API is available FALSE: Mcu_InitClock() API is not available		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.36.16 McuInitDeInitApiMode**Table 1250 Specification for McuInitDeInitApiMode**

Name	McuInitDeInitApiMode		
Description	Operating modes for MCU initialization/de-initialization APIs		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_MCAL_SUPERVISOR: Initialization APIs are run in the Supervisor mode MCU_MCAL_USER1: Initialization APIs are run in the User 1 mode		
Default value	MCU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-

MCU driver**Table 1250 Specification for McuInitDeInitApiMode (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.17 McuMainOscillatorFrequency**Table 1251 Specification for McuMainOscillatorFrequency**

Name	McuMainOscillatorFrequency		
Description	Denotes the external crystal frequency value in MHz. External crystal frequency value (in MHz): (16 MHz to 40 MHz): External crystal mode is selected (4 MHz to 40 MHz): Direct input mode is selected, if the shaper is not bypassed		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	4MHz - 40MHz		
Default value	20MHz		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.36.18 McuMultiCoreErrorDetect**Table 1252 Specification for McuMultiCoreErrorDetect**

Name	McuMultiCoreErrorDetect		
Description	Pre-processor switch for enabling the multicore error detection and reporting. Values: TRUE: Multicore error detection is enabled FALSE: Multicore error detection is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		

MCU driver**Table 1252 Specification for McuMultiCoreErrorDetect (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.19 McuNoPlI**Table 1253 Specification for McuNoPlI**

Name	McuNoPlI		
Description	<p>McuNoPlI is set to TRUE, if the hardware does not have a system PLL or the system PLL circuitry enabled after the power on without software intervention. In this case MCU_DistributePlIClock should be disabled and MCU_GetPlIStatus should return MCU_PLL_STATUS_UNDEFINED.</p> <p>McuNoPlI is always disabled as the TC3xx micro-controller supports PLL.</p> <p>Values:</p> <p>TRUE: MCU does not have to intervene in the PLL-related setup.</p> <p>FALSE: MCU is responsible to get the PLLs up and running.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.36.20 McuOscAmpRegulationEnable**Table 1254 Specification for McuOscAmpRegulationEnable**

Name	McuOscAmpRegulationEnable
Description	Selects whether oscillator amplitude regulation is enabled or disabled. TRUE: Amplitude regulation is enabled.

MCU driver**Table 1254 Specification for McuOscAmpRegulationEnable (continued)**

	FALSE: Amplitude regulation is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.21 McuOscCapacitance0Enable**Table 1255 Specification for McuOscCapacitance0Enable**

Name	McuOscCapacitance0Enable		
Description	Selects that load capacitance CL0 is enabled or disabled. TRUE: Capacitance CL0 is enabled. FALSE: Capacitance CL0 is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuOscAmpRegulationEnable		

17.3.1.36.22 McuOscCapacitance1Enable**Table 1256 Specification for McuOscCapacitance1Enable**

Name	McuOscCapacitance1Enable
-------------	--------------------------

MCU driver**Table 1256 Specification for McuOscCapacitance1Enable (continued)**

Description	Selects that load capacitance CL1 is enabled or disabled. TRUE: Capacitance CL1 is enabled. FALSE: Capacitance CL1 is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuOscAmpRegulationEnable		

17.3.1.36.23 McuOscCapacitance2Enable**Table 1257 Specification for McuOscCapacitance2Enable**

Name	McuOscCapacitance2Enable		
Description	Selects that load capacitance CL2 is enabled or disabled. TRUE: Capacitance CL2 is enabled. FALSE: Capacitance CL2 is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuOscAmpRegulationEnable		

MCU driver**17.3.1.36.24 McuOscCapacitance3Enable****Table 1258 Specification for McuOscCapacitance3Enable**

Name	McuOscCapacitance3Enable		
Description	Selects that load capacitance CL3 is enabled or disabled. TRUE: Capacitance CL3 is enabled. FALSE: Capacitance CL3 is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuOscAmpRegulationEnable		

17.3.1.36.25 McuOscillatorMode**Table 1259 Specification for McuOscillatorMode**

Name	McuOscillatorMode		
Description	Pre-processor switch to select the oscillator mode		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	EXT_CRYSTAL_CERAMIC_RES_MODE_SEL0: external crystal or ceramic resonator mode is selected EXT_INPUT_CLOCK_MODE_SEL2: external input clock source mode is selected OSC_DISABLED_MODE_SEL3: Oscillator is disabled		
Default value	EXT_CRYSTAL_CERAMIC_RES_MODE_SEL0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.36.26 McuPerformResetApi****Table 1260 Specification for McuPerformResetApi**

Name	McuPerformResetApi		
Description	Pre-processor switch to enable/disable the availability of the Mcu_PerformReset() API Values: TRUE: Mcu_PerformReset() API is available FALSE: Mcu_PerformReset() API is not available		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.36.27 McuRuntimeApiMode**Table 1261 Specification for McuRuntimeApiMode**

Name	McuRuntimeApiMode		
Description	Operating modes for MCU runtime APIs		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_MCAL_SUPERVISOR: run time APIs are run in the Supervisor mode MCU_MCAL_USER1: run time APIs are run in the User 1 mode		
Default value	MCU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.36.28 McuSafetyEnable****Table 1262 Specification for McuSafetyEnable**

Name	McuSafetyEnable		
Description	Enables/disables safety checks and features of the MCU driver Values: TRUE: Safety features are available FALSE: Safety features are disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.29 McuStandbyEntryMode**Table 1263 Specification for McuStandbyEntryMode**

Name	McuStandbyEntryMode		
Description	Pre-processor parameter to select the standby mode entry criteria		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	STANDBY_ENTRY_ESR_SEL4: entry to the standby mode domain is through ESR1/NMI assertion. Configuration of proper ALT selection for the corresponding port pin has to be done in the PORT driver. STANDBY_ENTRY_REQ_SLEEP_SEL0: entry to the standby domain is through PMSWCR1_STBYEV. This can be done by calling Mcu_SetMode (STANDBY_MODE).		
Default value	STANDBY_ENTRY_REQ_SLEEP_SEL0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCU driver**Table 1263 Specification for McuStandbyEntryMode (continued)**

Dependency	-
-------------------	---

17.3.1.36.30 McuSystemModeCpuCore**Table 1264 Specification for McuSystemModeCpuCore**

Name	McuSystemModeCpuCore		
Description	Defines which core can trigger system modes (sleep/standby)		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CPU_SYSTEM_CORE0_SEL0: Only CPU0 can trigger the power down modes CPU_SYSTEM_CORE1_SEL1: Only CPU1 can trigger the power down modes CPU_SYSTEM_CORE2_SEL2: Only CPU2 can trigger the power down modes CPU_SYSTEM_CORE3_SEL3: Only CPU3 can trigger the power down modes CPU_SYSTEM_CORE4_SEL4: Only CPU4 can trigger the power down modes CPU_SYSTEM_CORE5_SEL5: Only CPU5 can trigger the power down modes UNANIMOUS_SYSTEM_ALL_CORES_SEL6: Entry to power down modes is unanimously decided by all the CPUs		
Default value	CPU_SYSTEM_CORE0_SEL0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.36.31 McuVersionInfoApi**Table 1265 Specification for McuVersionInfoApi**

Name	McuVersionInfoApi
Description	<p>Pre-processor switch to enable/disable the API to read out the driver version information.</p> <p>If this parameter is set to TRUE then, following macro is generated.</p> <pre>#define MCU_VERSION_INFO_API (STD_ON) #else #define MCU_VERSION_INFO_API (STD_OFF)</pre> <p>Mcu_GetVersionInfo() is guarded by above generated macro.</p> <p>Values:</p> <ul style="list-style-type: none"> TRUE: Version information API is enabled FALSE: Version information API is disabled

MCU driver**Table 1265 Specification for McuVersionInfoApi (continued)**

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.36.32 McuSysClkFrequency**Table 1266 Specification for McuSysClkFrequency**

Name	McuSysClkFrequency		
Description	Specifies the input signal frequency value in MHz applied at the SYSCLK port pad.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	16 MHz - 40 MHz		
Default value	20 MHz		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.37 Container: GtmTomGlobalConf

This container holds the configuration parameters for the TOM global parameters. Various instances of TOM channels can be used by the ADC, PWM, GPT and WDG drivers and hence the global configuration for these channels within one TOM group channel (TGC) is taken care of by this container

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.38 Container: McuGpt12ModuleAllocationConf

This container holds the GPT timer allocation to the different MCAL drivers.

Post-Build Variant Multiplicity: -

MCU driver

Multiplicity Configuration Class: -

17.3.1.38.1 McuGpt12ModuleAllocationConf

Table 1267 Specification for McuGpt12ModuleAllocationConf

Name	McuGpt12ModuleAllocationConf		
Description	Specifies which driver(s) have used this particular GPT timer or this module is not used by any driver (unused).		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GPT_TIMER_NOT_USED: GPT timer is not used GPT_TIMER_USED_BY_GPT_DRIVER: GPT timer is reserved for the GPT driver GPT_TIMER_USED_BY_ICU_DRIVER: GPT timer is reserved for the ICU driver		
Default value	GPT_TIMER_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	Gpt2BlockPrescalerSel, Gpt1BlockPrescalerSel		

17.3.1.38.2 McuGpt12TimerAllocation

Table 1268 Specification for McuGpt12TimerAllocation

Name	McuGpt12TimerAllocation		
Description	Specifies the timer to be reserved		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GPT_TIMER_2: GPT timer T2 is reserved for the allocation. GPT_TIMER_3: GPT timer T3 is used for resource allocation GPT_TIMER_4: GPT timer T4 is used for resource allocation GPT_TIMER_5: GPT timer T5 is used for resource allocation GPT_TIMER_6: GPT timer T6 is used for resource allocation		
Default value	GPT_TIMER_2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

MCU driver**Table 1268 Specification for McuGpt12TimerAllocation (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.39 Container: McuGtmAllocationConf

This container holds the ownership information of the sub-modules of GTM peripherals such as TOM, ATOM and TIM. The number of instances of the TIM, TOM and ATOM container depends on the underlying derivative.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.40 Container: McuGtmAtomAllocationConf

This container holds the GTM ATOM allocation. Multiplicity of this container depends on the underlying derivative. User is not allowed to change the name of the parameters in this container.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.41 Container: McuGtmAtomChannelAllocationConf

This container holds the GTM ATOM channel allocation.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.41.1 McuAtomChannelEventHandledByDsadc**Table 1269 Specification for McuAtomChannelEventHandledByDsadc**

Name	McuAtomChannelEventHandledByDsadc		
Description	<p>Specifies whether callback of DSADC or the driver reserving the resource will be invoked when an event occurs</p> <p>TRUE : The callback of DSADC is invoked on an event.</p> <p>FALSE: The callback of the module which has configured the channel is invoked on an event."</p> <p>Note: This parameter can only be selected in case the user of ATOM channel is PWM or OCU.</p>		
Multiplicity	1..1	Type	EcucBooleanParamD ef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

MCU driver**Table 1269 Specification for McuAtomChannelEventHandledByDsadc (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.41.2 McuGtmAtomChannelAllocationConf**Table 1270 Specification for McuGtmAtomChannelAllocationConf**

Name	McuGtmAtomChannelAllocationConf		
Description	Specifies which driver(s) have used or not used this particular ATOM channel		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_ATOM_CHANNEL_NOT_USED: ATOM channel is not used GTM_ATOM_CHANNEL_USED_BY_ADC: ATOM channel is reserved for the ADC driver GTM_ATOM_CHANNEL_USED_BY_GPT: ATOM channel is reserved for the GPT driver GTM_ATOM_CHANNEL_USED_BY_OCU: ATOM channel is reserved for the OCU driver GTM_ATOM_CHANNEL_USED_BY_PWM: ATOM channel is reserved for the PWM driver GTM_ATOM_CHANNEL_USED_BY_WDG: ATOM channel is reserved for the WDG driver		
Default value	GTM_ATOM_CHANNEL_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.42 Container: McuGtmClockManagementConf

This container deals with configuration of the CMU parameters

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.42.1 GtmCmuGlobalClockDenominator**Table 1271 Specification for GtmCmuGlobalClockDenominator**

Name	GtmCmuGlobalClockDenominator		
Description	Used to configure the global denominator value for configurable clock and fixed clock GtmCmuGlobalClockNumerator should not be less than GtmCmuGlobalClockDenominator		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 16777215		

MCU driver**Table 1271 Specification for GtmCmuGlobalClockDenominator (continued)**

Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuGlobalClockNumerator		

17.3.1.42.2 GtmCmuGlobalClockNumerator**Table 1272 Specification for GtmCmuGlobalClockNumerator**

Name	GtmCmuGlobalClockNumerator		
Description	Used to configure the global numerator value for configurable clock and fixed clock GtmCmuGlobalClockNumerator should not be less than GtmCmuGlobalClockDenominator		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	GtmCmuGlobalClockDenominator		

17.3.1.43 Container: McuGtmTimAllocationConf

This container holds the GTM TIM allocation. The multiplicity of this container depends on the underlying derivative. User is not allowed to change the name of the configuration parameters in this container.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.44 Container: McuGtmTimChannelAllocationConf

This container holds the GTM TIM channel allocation.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

MCU driver**17.3.1.44.1 McuGtmTimChannelAllocationConf****Table 1273 Specification for McuGtmTimChannelAllocationConf**

Name	McuGtmTimChannelAllocationConf		
Description	Specifies which driver(s) have used or not used this particular TIM channel		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_TIM_CHANNEL_NOT_USED: TIM channel is not used GTM_TIM_CHANNEL_USED_BY_ICU: TIM channel is reserved for the ICU driver		
Default value	GTM_TIM_CHANNEL_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.45 Container: McuGtmTomAllocationConf

This container holds the GTM TOM allocation. The multiplicity of this container depends on the underlying derivative. User is not allowed to change the name of the parameters in this container.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.46 Container: McuGtmTomChannelAllocationConf

This container holds the GTM TOM channel allocation.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.46.1 McuGtmTomChannelAllocationConf**Table 1274 Specification for McuGtmTomChannelAllocationConf**

Name	McuGtmTomChannelAllocationConf		
Description	Specifies which driver(s) have used or not used this particular TOM channel		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_TOM_CHANNEL_NOT_USED: TOM channel is not used GTM_TOM_CHANNEL_USED_BY_ADC: TOM channel is reserved for the ADC driver GTM_TOM_CHANNEL_USED_BY_GPT: TOM channel is reserved for the GPT driver GTM_TOM_CHANNEL_USED_BY_OCU: TOM channel is reserved for the OCU driver		

MCU driver**Table 1274 Specification for McuGtmTomChannelAllocationConf (continued)**

	GTM_TOM_CHANNEL_USED_BY_PWM: TOM channel is reserved for the PWM driver GTM_TOM_CHANNEL_USED_BY_WDG: TOM channel is reserved for the WDG driver		
Default value	GTM_TOM_CHANNEL_NOT_USED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.46.2 McuTomChannelEventHandledByDsadc**Table 1275 Specification for McuTomChannelEventHandledByDsadc**

Name	McuTomChannelEventHandledByDsadc		
Description	Specifies whether callback of DSADC or the driver reserving the resource will be invoked when an event occurs TRUE : The callback of DSADC is invoked on an event. FALSE: The callback of the module which has configured the channel is invoked on an event." Note: This parameter can only be selected in case the user of TOM channel is PWM or OCU.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.47 Container: McuHardwareResourceAllocationConf

This container holds the hardware resource allocation for the peripherals whose unique instances are used by multiple modules such as GTM, ASCLIN, CCU, ADC and ERU.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

MCU driver

17.3.1.48 Container: McuModeSettingConf

This container holds the configuration (parameters) for the mode setting of the MCU.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

17.3.1.48.1 McuEvrcLPMOnSleepReqEnable

Table 1276 Specification for McuEvrcLPMOnSleepReqEnable

Name	McuEvrcLPMOnSleepReqEnable		
Description	<p>Enables EVRC low power mode when the sleep mode is enabled. McuEvrcLPMOnSleepReqEnable is enabled only if McuMode is selected as MCU_SLEEP. TRUE: entering into the low power mode for EVRC on sleep mode request is enabled FALSE: Entering into the low power mode for EVRC on sleep mode request is disabled Caution: When McuEvrcLPMOnSleepReqEnable is enabled, ensure smooth current ramp-down before entering into the Sleep mode. High current jumps during mode transition may lead to unintended device reset.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.48.2 McuMode

Table 1277 Specification for McuMode

Name	McuMode		
Description	<p>Refers to the modes supported other than the RUN mode (for example SLEEP mode, IDLE mode, STANDBY mode) Mcu_SetMode entertains only the configured modes, However for the Sleep or Standby mode, other CPUs are put to Idle mode. For a given ConfigSet of the MCU driver, there could be a maximum of 3 set of modes: 0 - IDLE mode 1 - SLEEP mode 2 - STANDBY mode</p>		

MCU driver
Table 1277 Specification for McuMode (continued)

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 2		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.49 Container: McuModuleConfiguration

McuModuleConfiguration container contains the configuration (parameters) of the MCU driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.49.1 McuClockSrcFailureNotification

Table 1278 Specification for McuClockSrcFailureNotification

Name	McuClockSrcFailureNotification		
Description	Enables/disables the clock source failure notification. This parameter is disabled and is included here for completeness.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DISABLED: clock source failure notification is disabled ENABLED: clock source failure notification is enabled		
Default value	DISABLED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.49.2 McuNumberOfMcuModes

Table 1279 Specification for McuNumberOfMcuModes

Name	McuNumberOfMcuModes
-------------	---------------------

MCU driver**Table 1279 Specification for McuNumberOfMcuModes (continued)**

Description	Represents the number of modes available for the MCU. McuNumberOfMcuModes is disabled and is included here for completeness. Note: The postbuild variant value for the McuNumberOfMcuModes is deviated from AUTOSAR.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 255		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.49.3 McuRamSectors**Table 1280 Specification for McuRamSectors**

Name	McuRamSectors		
Description	Represents the number of RAM sectors available for the MCU. This parameter is disabled and is included here for completeness. Note: The postbuild variant value for the McuRamSectors is deviated from AUTOSAR.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.49.4 McuResetSetting**Table 1281 Specification for McuResetSetting**

Name	McuResetSetting
-------------	-----------------

MCU driver**Table 1281 Specification for McuResetSetting (continued)**

Description	Relates to the MCU specific reset configuration. McuResetSetting is disabled and is included here for completeness. Note: The postbuild variant value for the McuResetSetting is deviated from AUTOSAR.		
Multiplicity	0..1	Type	EcucIntegerParamDef
Range	1 - 255		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.50 Container: McuPeripheralPllSettingConfig

This container contains the configuration (parameters) for the peripheral clock settings.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.50.1 McuClockReferencePointFrequency1**Table 1282 Specification for McuClockReferencePointFrequency1**

Name	McuClockReferencePointFrequency1		
Description	Users have to configure the resulting target frequency after configuring the N, P and K2 dividers for the peripheral PLL. The configured value should be divided by 2 if McuFreqSource1ClockDivSelect is configured with DIV_FACTOR_2_NOT_BYPASSED_SEL1. A calculation button is provided for updating this values (in Hz). The McuClockReferencePointFrequency1 for NORMAL_MODE should be in the range: 20 to 320 MHz. If McuClockDistributionInpClockSel is selected as BACKUP_INPUT_CLOCK_SRC_SELECT, then manually configure this clock to Fback = 100 MHz. fSOURCE1 is McuClockReferencePointFrequency1		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	20000000.0 - 32000000.0		
Default value	160000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1282 Specification for McuClockReferencePointFrequency1 (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel, McuPeripheralPllK2Divider, McuFreqSource1ClockDivSelect, McuPeripheralPllNDivider, McuPeripheralPllPDivide, McuPllInputSrcSelection		

17.3.1.50.2 McuClockReferencePointFrequency2**Table 1283 Specification for McuClockReferencePointFrequency2**

Name	McuClockReferencePointFrequency2		
Description	<p>Users have to configure the resulting target frequency after configuring the N, P and K3 dividers for the peripheral PLL.</p> <p>A configuration button is provided for updating this value (in Hz).</p> <p>The McuClockReferencePointFrequency2 for NORMAL_MODE should be in the range: 20 to 200 MHz. If McuClockDistributionInpClockSel is selected as BACKUP_INPUT_CLOCK_SRC_SELECT, then manually configure this clock to Fback = 100 MHz.</p> <p>fSOURCE2 is McuClockReferencePointFrequency2</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	20000000.0 - 200000000.0		
Default value	200000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuPll2DivSelect, McuClockDistributionInpClockSel, McuPeripheralPllK3Divider, McuPeripheralPllNDivider, McuPeripheralPllPDivide, McuPllInputSrcSelection		

17.3.1.50.3 McuFreqSource1ClockDivSelect**Table 1284 Specification for McuFreqSource1ClockDivSelect**

Name	McuFreqSource1ClockDivSelect		
Description	Specifies whether Fpll1 is divided by a factor of two or divider is bypassed.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

MCU driver**Table 1284 Specification for McuFreqSource1ClockDivSelect (continued)**

Range	DIV_FACTOR_2_BYPASSSED_SEL1: divider factor of two is bypassed. ($F_{PLL1} = F_{source1}$) DIV_FACTOR_2_NOT_BYPASSSED_SEL0: divider factor of two is not bypassed ($F_{PLL1} = F_{source1} / 2$)		
Default value	DIV_FACTOR_2_BYPASSSED_SEL1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.50.4 McuPerPllK2DivStepDownChangeDelay**Table 1285 Specification for McuPerPllK2DivStepDownChangeDelay**

Name	McuPerPllK2DivStepDownChangeDelay		
Description	Delay required to configure the step changes between two consecutive changes in the K2 divider value of the peripheral PLL. This is a common delay used for peripheral PLL1 frequency ramp up sequences through the K2 divider. Note : The value is expressed in microseconds (us).		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	5 - 100		
Default value	10		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.50.5 McuPerPllK2DivStepUpChangeDelay**Table 1286 Specification for McuPerPllK2DivStepUpChangeDelay**

Name	McuPerPllK2DivStepUpChangeDelay		
Description	Delay required to configure the step changes between two consecutive changes in the K2 divider value of the peripheral PLL. This is a common delay used for the peripheral PLL1 frequency ramp up sequences through the K2 divider. Note : The value is expressed in microseconds (us).		

MCU driver**Table 1286 Specification for McuPerPllK2DivStepUpChangeDelay (continued)**

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	5 - 100		
Default value	10		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.50.6 McuPerPllK3DivStepDownChangeDelay**Table 1287 Specification for McuPerPllK3DivStepDownChangeDelay**

Name	McuPerPllK3DivStepDownChangeDelay		
Description	Delay required to configure the step changes between two consecutive changes in the K3 divider value of the peripheral PLL. This is a common delay used for the peripheral PLL2 frequency ramp down sequences through the K3 divider. Note : The value is expressed in microseconds (us).		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	5 - 100		
Default value	10		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.50.7 McuPerPllK3DivStepUpChangeDelay**Table 1288 Specification for McuPerPllK3DivStepUpChangeDelay**

Name	McuPerPllK3DivStepUpChangeDelay		
Description	Delay required to configure the step changes between two consecutive changes in the K3 divider value of the peripheral PLL. This is a common delay used for the peripheral PLL2 frequency ramp up sequences through the K3 divider. Note : The value is expressed in microseconds (us)		

MCU driver**Table 1288 Specification for McuPerPllk3DivStepUpChangeDelay (continued)**

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	5 - 100		
Default value	10		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.50.8 McuPeripheralPllk2Divider**Table 1289 Specification for McuPeripheralPllk2Divider**

Name	McuPeripheralPllk2Divider		
Description	3-bit output divider. Even values are preferred to get 50% duty cycle. Note : Clock equations are incremented by 1 to this parameter. Changing the system operation frequency by changing the value of the K2-divider has a direct coupling to the power consumption of the device. Therefore, this must be done carefully.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 7		
Default value	4		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.50.9 McuPeripheralPllk3Divider**Table 1290 Specification for McuPeripheralPllk3Divider**

Name	McuPeripheralPllk3Divider		
Description	3-bit output divider. Even values are preferred to get 50% duty cycle. Clock equations are incremented by 1 to this parameter.		

MCU driver**Table 1290 Specification for McuPeripheralPllk3Divider (continued)**

	Note: Changing the system operation frequency by changing the value of the K3-divider has a direct coupling to the power consumption of the device. Therefore, this must be done carefully.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 7		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.50.10 McuPeripheralPllNDivider**Table 1291 Specification for McuPeripheralPllNDivider**

Name	McuPeripheralPllNDivider		
Description	7-bit feedback divider value used for generating the system clock. Clock equations are incremented by 1 to this parameter.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 127		
Default value	39		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.50.11 McuPeripheralPllPDivider**Table 1292 Specification for McuPeripheralPllPDivider**

Name	McuPeripheralPllPDivider		
Description	Frequency divider of main oscillator (3 bits) Clock equations are incremented by 1 to this parameter.		
Multiplicity	1..1	Type	EcuIntegerParamDef

MCU driver**Table 1292 Specification for McuPeripheralPllPDivider (continued)**

Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.50.12 McuPll2DivSelect**Table 1293 Specification for McuPll2DivSelect**

Name	McuPll2DivSelect		
Description	Specifies whether divider factor in before the K3 divider is bypassed or not		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_K3_DIV_FACTOR_BYPASSSED_SEL1: divider factor for K3 is bypassed MCU_K3_DIV_FACTOR_NOT_BYPASSSED_SEL0: divider factor for K3 is not bypassed		
Default value	MCU_K3_DIV_FACTOR_BYPASSSED_SEL1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.51 Container: McuPllDistributionSettingConfig

This container holds the configuration (parameters) for PLL distribution and frequencies to various hardware modules within the clock tree.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.51.1 McuAdasFrequency**Table 1294 Specification for McuAdasFrequency**

Name	McuAdasFrequency
-------------	------------------

MCU driver**Table 1294 Specification for McuAdasFrequency (continued)**

Description	Specifies the ADAS peripheral frequency in Hz. The ratio between ADAS frequency and McuClockReferencePointFrequency0 should be within the range as specified in the target specification. In order to facilitate the clearing of SRAM support hardware registers, this frequency is also configurable for non-ADAS devices. However, the default value for such devices is kept to 0, which disables the ADAS clock.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockReferencePointFrequency0, McuLowPowerDivValue		

17.3.1.51.2 McuAdcFrequency**Table 1295 Specification for McuAdcFrequency**

Name	McuAdcFrequency		
Description	Specifies the clock frequency for the ADC peripheral. The ADC clock frequency is always the same as McuClockReferencePointFrequency1. Unit is expressed in Hz. The maximum possible value is taken as the default value for McuAdcFrequency.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	20000000.0 - 160000000.0		
Default value	160000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	McuClockReferencePointFrequency1		

MCU driver**17.3.1.51.3 McuAscLinFastFrequency****Table 1296 Specification for McuAscLinFastFrequency**

Name	McuAscLinFastFrequency		
Description	<p>Specifies the clock frequency for the ASCLIN peripheral for the fast mode.</p> <p>To disable the ASCLIN peripheral frequency for fast mode, a value of 0 should be configured to this configuration parameter.</p> <p>If not disabled, the intended target frequency to be configured should be McuClockReferencePointFrequency2 perfectly divisible by one of the divider values as specified in Target Specification. Unit is in Hz.</p> <p>By default, the ASCLIN fast clock is switched OFF, therefore the default value is taken as 0.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 200000000.0		
Default value	0.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	McuClockReferencePointFrequency2		

17.3.1.51.4 McuAscLinSlowClockSourceSelection**Table 1297 Specification for McuAscLinSlowClockSourceSelection**

Name	McuAscLinSlowClockSourceSelection		
Description	<p>Specifies the input clock source for the ASCLIN peripheral slow frequency</p> <p>Frequency calculation of the ASCLIN is done in the McuAscLinSlowFrequency configuration parameter.</p> <p>By default, the ASCLIN slow clock is switched OFF</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ASCLINS_CLOCK_SOURCE_ASCLINSI_SEL1: McuAscLinSlowFrequency is used as the input clock source for the ASCLIN dividers ASCLINS_CLOCK_SOURCE_DISABLED_SEL0: ASCLIN peripheral frequency is disabled ASCLINS_CLOCK_SOURCE_OSC0_SEL2: McuMainOscillatorFrequency is used as the input clock source for the ASCLIN dividers		
Default value	ASCLINS_CLOCK_SOURCE_DISABLED_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1297 Specification for McuAscLinSlowClockSourceSelection (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.51.5 McuAscLinSlowFrequency**Table 1298 Specification for McuAscLinSlowFrequency**

Name	McuAscLinSlowFrequency		
Description	<p>Specifies the clock frequency for the ASCLIN peripheral for slow mode.</p> <p>To disable the ASCLIN peripheral frequency for slow mode, a value of 0 should be configured to this configuration parameter.</p> <p>If not disabled, the intended target frequency to be configured should be McuClockReferencePointFrequency1 perfectly divisible by one of the divider values as specified in Target Specification. Unit is expressed in Hz.</p> <p>The minimum possible value is taken as the default value for McuAscLinSlowFrequency</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 200000000.0		
Default value	0.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	McuClockReferencePointFrequency1, McuAscLinSlowClockSourceSelection		

17.3.1.51.6 McuBBBFrequency**Table 1299 Specification for McuBBBFrequency**

Name	McuBBBFrequency		
Description	<p>Specifies the Back Bone Bus (BBB) frequency. The BBB frequency output can be stopped by configuring 0 to this configuration parameter.</p> <p>If enabled, the possible divider values are provided in the Target Specification</p> <p>If enabled, the Fbbb must be faster than or equal to Fspb.</p> <p>Unit is expressed in Hz.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 150000000.0		

MCU driver**Table 1299 Specification for McuBBBFrequency (continued)**

Default value	150000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue		

17.3.1.51.7 McuCPU0Frequency**Table 1300 Specification for McuCPU0Frequency**

Name	McuCPU0Frequency		
Description	<p>Specifies the intended target CPU0 frequency. The user should enter the intended target frequency expected for CPU0 operation.</p> <p>McuCPU0Frequency configuration requires adherence to the following formula:</p> $\text{McuCPU0Frequency} = \text{McuSRIFrequency} * (64 - \text{CPU0DIV}) / 64$ <p>Note: Possible range for CPU0DIV is from 0 to 63. Unit is expressed in Hz.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	1.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue, McuSRIFrequency		

17.3.1.51.8 McuCPU1Frequency**Table 1301 Specification for McuCPU1Frequency**

Name	McuCPU1Frequency		
Description	<p>Specifies the intended target CPU1 frequency. The user should enter the intended target frequency expected for CPU1 operation.</p> <p>McuCPU1Frequency configuration requires adherence to the following formula:</p> $\text{McuCPU1Frequency} = \text{McuSRIFrequency} * (64 - \text{CPU1DIV}) / 64$ <p>Note: Possible range for CPU1DIV is from 0 to 63. Unit is expressed in Hz.</p>		

MCU driver**Table 1301 Specification for McuCPU1Frequency (continued)**

Multiplicity	1..1	Type	EcucFloatParamDef
Range	1.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue, McuSRIFrequency		

17.3.1.51.9 McuCPU2Frequency**Table 1302 Specification for McuCPU2Frequency**

Name	McuCPU2Frequency		
Description	<p>Specifies the intended target CPU2 frequency. The user should enter the intended target frequency expected for CPU2 operation.</p> <p>McuCPU2Frequency configuration requires adherence to the following formula:</p> $\text{McuCPU2Frequency} = \text{McuSRIFrequency} * (64 - \text{CPU2DIV}) / 64$ <p>Note: Possible range for CPU2DIV is from 0 to 63. Unit is expressed in Hz.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	1.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue, McuSRIFrequency		

17.3.1.51.10 McuCPU3Frequency**Table 1303 Specification for McuCPU3Frequency**

Name	McuCPU3Frequency		
Description	<p>Specifies the intended target CPU3 frequency. The user should enter the intended target frequency expected for CPU3 operation.</p> <p>McuCPU3Frequency configuration requires adherence to the following formula:</p>		

MCU driver**Table 1303 Specification for McuCPU3Frequency (continued)**

	McuCPU3Frequency = McuSRIFrequency * (64 - CPU3DIV) / 64 Note: Possible range for CPU3DIV is from 0 to 63. Unit is expressed in Hz.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	1.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue, McuSRIFrequency		

17.3.1.51.11 McuCPU4Frequency**Table 1304 Specification for McuCPU4Frequency**

Name	McuCPU4Frequency		
Description	Specifies the intended target CPU4 frequency. The user should enter the intended target frequency expected for CPU1 operation. McuCPU4Frequency configuration requires adherence to the following formula: $\text{McuCPU4Frequency} = \text{McuSRIFrequency} * (64 - \text{CPU4DIV}) / 64$ Note: Possible range for CPU4DIV is from 0 to 63. Unit is expressed in Hz.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	1.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue, McuSRIFrequency		

17.3.1.51.12 McuCPU5Frequency**Table 1305 Specification for McuCPU5Frequency**

Name	McuCPU5Frequency
-------------	------------------

MCU driver**Table 1305 Specification for McuCPU5Frequency (continued)**

Description	Specifies the intended target CPU5 frequency. The user should enter the intended target frequency expected for CPU5 operation. McuCPU5Frequency configuration requires adherence to the following formula: $\text{McuCPU5Frequency} = \text{McuSRIFrequency} * (64 - \text{CPU5DIV}) / 64$ Note: Possible range for CPU5DIV is from 0 to 63. Unit is expressed in Hz.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	1.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue, McuSRIFrequency		

17.3.1.51.13 McuClockDistributionInpClockSel**Table 1306 Specification for McuClockDistributionInpClockSel**

Name	McuClockDistributionInpClockSel		
Description	Specifies the input clock source selection for the clock distribution unit. Either the back up clock or the PLLx can be selected as an input clock source to the clock distribution unit.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	BACKUP_INPUT_CLOCK_SRC_SELECT_SEL0: Backup clock is selected as an input clock source to SPB, reference clock frequency1, reference clock frequency2, BBB, GTM, STM, MSC, MCAN, ASCLINF, ASCLINS, QSPI, ADC, I2C and EBU PLL_INPUT_CLOCK_SRC_SELECT_SEL1: If PLL is selected as an input clock source then, - fSOURCE0 is selected as the clock source for SRI, SPB, CPU0, CPU1, CPU2, CPU3, CPU4, CPU5, FSI, FSI2, reference clock frequency1, BBB, GTM, STM, MCAN, GETH and ADAS - fSRC1 is selected as the clock source for reference clock frequency2, ERAY, MSC, MCAN, ASCLINS, QSPI, ADC, EBU, HSPDM_320 and HSPDM_160 - fSOURCE2 is selected as the clock source for MSC, ASCLINF, QSPI and I2C		
Default value	PLL_INPUT_CLOCK_SRC_SELECT_SEL1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCU driver**Table 1306 Specification for McuClockDistributionInpClockSel (continued)**

Dependency	-
-------------------	---

17.3.1.51.14 McuConvCtrlPhaseSynchConf**Table 1307 Specification for McuConvCtrlPhaseSynchConf**

Name	McuConvCtrlPhaseSynchConf		
Description	Specifies the phase shift frequency divider for the converter control block. McuConvCtrlPhaseSynchConf is included here as it is common across the ADC and DSADC modules		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PHASE_SYNCH_CONST_ACTIVE_SEL0: constant phase signal is active PHASE_SYNCH_PER_FREQ_BY_10_SEL9: phase synchronization is generated at fPER by 10 PHASE_SYNCH_PER_FREQ_BY_11_SEL10: phase synchronization is generated at fPER by 11 PHASE_SYNCH_PER_FREQ_BY_12_SEL11: phase synchronization is generated at fPER by 12 PHASE_SYNCH_PER_FREQ_BY_13_SEL12: phase synchronization is generated at fPER by 13 PHASE_SYNCH_PER_FREQ_BY_14_SEL13: phase synchronization is generated at fPER by 14 PHASE_SYNCH_PER_FREQ_BY_15_SEL14: phase synchronization is generated at fPER by 15 PHASE_SYNCH_PER_FREQ_BY_16_SEL15: phase synchronization is generated at fPER by 16 PHASE_SYNCH_PER_FREQ_BY_2_SEL1: phase synchronization is generated at fPER by 2 PHASE_SYNCH_PER_FREQ_BY_3_SEL2: phase synchronization is generated at fPER by 3 PHASE_SYNCH_PER_FREQ_BY_4_SEL3: phase synchronization is generated at fPER by 4 PHASE_SYNCH_PER_FREQ_BY_5_SEL4: phase synchronization is generated at fPER by 5 PHASE_SYNCH_PER_FREQ_BY_6_SEL5: phase synchronization is generated at fPER by 6 PHASE_SYNCH_PER_FREQ_BY_7_SEL6: phase synchronization is generated at fPER by 7 PHASE_SYNCH_PER_FREQ_BY_8_SEL7: phase synchronization is generated at fPER by 9 PHASE_SYNCH_PER_FREQ_BY_9_SEL8: phase synchronization is generated at fPER by 9		
Default value	PHASE_SYNCH_CONST_ACTIVE_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.51.15 McuEbuClkEnable****Table 1308 Specification for McuEbuClkEnable**

Name	McuEbuClkEnable		
Description	<p>Specifies if the frequency provided for the EBU module, McuEbuFrequency is enabled or not</p> <p>TRUE: McuEbuFrequency is enabled</p> <p>FALSE: McuEbuFrequency is disabled</p> <p>This parameter is enabled if the EBU is available in the hardware</p> <p>By default, the EBU clock is kept disabled. The user can enable the clock when required.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.51.16 McuEbuFrequency**Table 1309 Specification for McuEbuFrequency**

Name	McuEbuFrequency		
Description	<p>Specifies the EBU peripheral frequency.</p> <p>This clock frequency is always the same as McuClockReferencePointFrequency1. Unit is expressed in Hz.</p> <p>The maximum value is taken as the default value for McuEbuFrequency.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 160000000.0		
Default value	160000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockReferencePointFrequency1, McuEbuClkEnable		

MCU driver**17.3.1.51.17 McuErayClkEnable****Table 1310 Specification for McuErayClkEnable**

Name	McuErayClkEnable		
Description	<p>Specifies if the frequency provided for the ERAY module, McuErayFrequency is enabled or not</p> <p>Values:</p> <p>TRUE: McuErayFrequency is enabled.</p> <p>FALSE: McuErayFrequency is disabled.</p> <p>By default, the ERAY clock is disabled. Based on the use case the user can enable it.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.51.18 McuErayFrequency**Table 1311 Specification for McuErayFrequency**

Name	McuErayFrequency		
Description	<p>Specifies the ERAY frequency.</p> <p>The resultant ERAY frequency is always equal to peripheral PLL frequency (McuClockReferencePointFrequency1) divided by fixed divider 2.</p> <p>The ERAY would not be functional when the BACKUP clock is selected as distribution source.</p> <p>Unit is expressed in Hz.</p> <p>The maximum possible value is taken as the default value for McuErayFrequency.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 80000000.0		
Default value	80000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

MCU driver**Table 1311 Specification for McuErayFrequency (continued)**

Origin	IFX	Scope	ECU
Dependency	McuClockReferencePointFrequency1, McuErayClkEnable		

17.3.1.51.19 McuFSI2Frequency**Table 1312 Specification for McuFSI2Frequency**

Name	McuFSI2Frequency		
Description	<p>Specifies the intended target FSI2 frequency. The user should enter the intended target frequency expected for the FSI2.</p> <p>The FSI2 cannot be disabled.</p> <p>FSI2 and SRI should follow:</p> <ul style="list-style-type: none"> -FSI2 can be same as SRI. -If FSI2 is intended to be half of SRI then SRIDIV must be either 1 or 2. -If FSI2 is intended to be one third of SRI then SRIDIV must be either 1 or 2. <p>The user must ensure that points 2 and 3 are taken care of.</p> <p>The possible divider values are available in the Target Specification. Unit is expressed in Hz.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	1.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue		

17.3.1.51.20 McuFSIFrequency**Table 1313 Specification for McuFSIFrequency**

Name	McuFSIFrequency		
Description	<p>Specifies the intended target FSI frequency. The user should enter the intended target frequency expected for the FSI.</p> <p>FSI cannot be disabled</p> <p>FSI and SRI should follow:</p> <ul style="list-style-type: none"> -FSI can be same as SRI. -If FSI is intended to be half of SRI then SRIDIV must be either 1 or 2. -If FSI is intended to be one third of SRI then SRIDIV must be either 1 or 2. <p>The user must ensure that points 2 and 3 are taken care of.</p>		

MCU driver**Table 1313 Specification for McuFSIFrequency (continued)**

	The possible divider values are available in the Target Specification. Unit is expressed in Hz.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	20000000.0 - 100000000.0		
Default value	100000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue		

17.3.1.51.21 McuGEthFrequency**Table 1314 Specification for McuGEthFrequency**

Name	McuGEthFrequency		
Description	Specifies the Gigabit Ethernet peripheral frequency. The Gigabit Ethernet frequency should be divisible by McuClockReferencePointFrequency0 with the divider values specified in Target Specification. Unit is expressed in Hz. The module frequency to Gigabit Ethernet can be disabled by setting McuGEthFrequency to 0. The minimum value of McuGEthFrequency is 150 MHz when GEth is enabled, therefore it is taken as the default value.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 200000000.0		
Default value	150000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	McuLowPowerDivValue		

17.3.1.51.22 McuGTMFrequency**Table 1315 Specification for McuGTMFrequency**

Name	McuGTMFrequency
-------------	-----------------

MCU driver**Table 1315 Specification for McuGTMFrequency (continued)**

Description	<p>Specifies the GTM peripheral frequency. To disable the GTM peripheral frequency, a value of 0 has to be configured to this configuration parameter.</p> <p>The GTM frequency, if enabled, is derived by dividing the fSOURCEGTM frequency by one of the following factors: 1, 2, 3, 4, 5, 6, 8, 10, 12, 15.</p> <p>fSOURCEGTM is derived using following formula:</p> $\text{if GTMDIV} = 1, \text{fSOURCEGTM} = \text{McuSPBFrequency} * 2,$ $\text{otherwise } \text{fSOURCEGTM} = \text{McuClockReferencePointFrequency}_0$ <p>Therefore, GTM should be configured either equal to = McuSPBFrequency * 2 or a fraction of McuClockReferencePointFrequency0. (Valid fraction values are available in Target Specification). Unit is expressed in Hz.</p> <p>The maximum possible value is taken as the default value for McuGTMFrequency.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 200000000.0		
Default value	200000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	McuLowPowerDivValue		

17.3.1.51.23 McuHsctFrequency**Table 1316 Specification for McuHsctFrequency**

Name	McuHsctFrequency		
Description	<p>Specifies the clock frequency for HSCT. The HSCT clock frequency is $(\text{McuMainOscillatorFrequency} * (\text{McuPeripheralNDivider} + 1)) / ((\text{McuPeripheralNDivider} + 1) * 2)$</p> <p>Unit is expressed in Hz.</p> <p>The maximum possible value is taken as the default value of McuHsctFrequency.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 400000000.0		
Default value	400000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

MCU driver**Table 1316 Specification for McuHsctFrequency (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.51.24 McuHspd160Frequency**Table 1317 Specification for McuHspd160Frequency**

Name	McuHspd160Frequency		
Description	Specifies the HSPDM160 peripheral frequency. The HSPDM160 clock frequency is always equal to McuClockReferencePointFrequency1. Unit is expressed in Hz. The maximum value is taken as the default value of McuHspd160Frequency.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	20000000.0 - 160000000.0		
Default value	160000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuHspd1ClkEnable		

17.3.1.51.25 McuHspd1320Frequency**Table 1318 Specification for McuHspd1320Frequency**

Name	McuHspd1320Frequency		
Description	Specifies the HSPDM320 peripheral frequency. The HSPDM320 clock frequency is always equal to fPLL1 or fBACKUP(based on McuClockDistributionInpClockSel). Unit is expressed in Hz. Divider value of 2 is considered as default in this case, therefore 160 MHz is used as the default value.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	20000000.0 - 320000000.0		
Default value	160000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1318 Specification for McuHspd320Frequency (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuHspd320Frequency		

17.3.1.51.26 McuHspd320Frequency**Table 1319 Specification for McuHspd320Frequency**

Name	McuHspd320Frequency		
Description	<p>Specifies if frequencies provided for the HSPDM modules, fHSPDM160 and fHSPDM320 are enabled or not.</p> <p>TRUE : fHSPDM160 and fHSPDM320 are enabled</p> <p>FALSE: fHSPDM160 and fHSPDM320 are disabled</p> <p>McuHspd320Frequency is enabled if the HSPDM is available in the hardware.</p> <p>By default, the HSPDM clock is kept disabled.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.51.27 McuI2CFrequency**Table 1320 Specification for McuI2CFrequency**

Name	McuI2CFrequency		
Description	<p>Specifies the I2C peripheral frequency. The I2C frequency, if enabled, should be divisible by McuClockReferencePointFrequency2 with the divider values specified in the Target Specification. Unit is expressed in Hz.</p> <p>The maximum possible value is taken as the default value for McuI2CFrequency.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef

MCU driver**Table 1320 Specification for McuI2CFrequency (continued)**

Range	0.0 - 100000000.0		
Default value	100000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockReferencePointFrequency2		

17.3.1.51.28 McuLowPowerDivValue**Table 1321 Specification for McuLowPowerDivValue**

Name	McuLowPowerDivValue		
Description	<p>Specifies whether low power divider feature is enabled or disabled.</p> <p>The McuLowPowerDivValue divider is also applicable to the frequencies derived from SRI and SPB.</p> <p>If this parameter is enabled, the configuration of dividers done in the CCUCON register is no longer valid.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>LOW_POWER_DIVIDER_DISABLE_SEL0: low power mode is disabled</p> <p>LOW_POWER_DIVIDE_BY_120_SEL3: low power mode clock divider is set to 120</p> <p>LOW_POWER_DIVIDE_BY_240_SEL4: low power mode clock divider is set to 240</p> <p>LOW_POWER_DIVIDE_BY_30_SEL1: low power mode clock divider is set to 30</p> <p>LOW_POWER_DIVIDE_BY_60_SEL2: low power mode clock divider is set to 60</p>		
Default value	LOW_POWER_DIVIDER_DISABLE_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.51.29 McuMCanClockSourceSelection**Table 1322 Specification for McuMCanClockSourceSelection**

Name	McuMCanClockSourceSelection
-------------	-----------------------------

MCU driver**Table 1322 Specification for McuMCanClockSourceSelection (continued)**

Description	Specifies the input clock source for the MCAN peripheral. The frequency calculation for the MSC peripheral is done in McuMCanFrequency configuration parameter. By, default, the MCAN clock source is disabled.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCAN_CLOCK_SOURCE_DISABLED_SEL0: MCAN frequency is disabled MCAN_CLOCK_SOURCE_MCANI_SEL1: McuMCanFrequency is used as input clock source for the MCAN peripheral MCAN_CLOCK_SOURCE_OSC_SEL2: McuMainOscillatorFrequency is used as input clock source for the MCAN peripheral		
Default value	MCAN_CLOCK_SOURCE_DISABLED_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	-		

17.3.1.51.30 McuMCanFrequency**Table 1323 Specification for McuMCanFrequency**

Name	McuMCanFrequency		
Description	Specifies the clock frequency for the MCAN peripheral. The McuMCanFrequency is applicable only if McuMCANClockSourceSelection is not set to MCAN_CLOCK_SOURCE_DISABLED. The target frequency to be configured should be perfectly divisible by the divider values specified in Target Specification. Unit is expressed in Hz. The minimum value is taken as the default value in case of McuMCanFrequency.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 80000000.0		
Default value	0.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU

MCU driver**Table 1323 Specification for McuMCanFrequency (continued)**

Dependency	McuMCanClockSourceSelection
-------------------	-----------------------------

17.3.1.51.31 McuMcanHFrequency**Table 1324 Specification for McuMcanHFrequency**

Name	McuMcanHFrequency		
Description	Specifies the MCANH peripheral frequency. The MCANH frequency should be divisible by McuClockReferencePointFrequency0 with the divider values specified in the Target Specification. Unit is expressed in Hz. The maximum value is taken as the default value of McuMcanHFrequency		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 100000000.0		
Default value	100000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	McuClockReferencePointFrequency0		

17.3.1.51.32 McuMscClockSourceSelection**Table 1325 Specification for McuMscClockSourceSelection**

Name	McuMscClockSourceSelection		
Description	Specifies the input clock source for the MSC peripheral. The frequency calculation for the MSC peripheral is done in McuMscFrequency configuration parameter. By default, the MSC clock source is disabled.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MSC_CLOCK_SOURCE_DISABLED_SEL0: MSC frequency is disabled MSC_CLOCK_SOURCE_SOURCE1_SEL1: McuClockReferencePointFrequency1 is used as input clock source for the MSC dividers MSC_CLOCK_SOURCE_SOURCE2_SEL2: McuClockReferencePointFrequency2 is used as input clock source for the MSC dividers		
Default value	MSC_CLOCK_SOURCE_DISABLED_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1325 Specification for McuMscClockSourceSelection (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.51.33 McuMscFrequency**Table 1326 Specification for McuMscFrequency**

Name	McuMscFrequency		
Description	<p>Specifies the clock frequency for the MSC peripheral. The McuMscFrequency is applicable only if McuMscClockSourceSelection is not set to MSC_CLOCK_SOURCE_DISABLED.</p> <p>The target frequency to be configured should be perfectly divisible by the divider values specified in the Target Specification. Unit is expressed in Hz.</p> <p>The minimum value is taken as the default value in case of McuMscFrequency.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 200000000.0		
Default value	0.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMscClockSourceSelection		

17.3.1.51.34 McuQspiClockSourceSelection**Table 1327 Specification for McuQspiClockSourceSelection**

Name	McuQspiClockSourceSelection		
Description	<p>Specifies the input clock source for the QSPI peripheral.</p> <p>The frequency calculation for the QSPI peripheral is done in the McuQspiFrequency configuration parameter.</p> <p>By default, the QSPI clock is switched OFF.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	QSPI_CLOCK_SOURCE_DISABLED_SEL0: QSPI peripheral frequency is disabled		

MCU driver**Table 1327 Specification for McuQspiClockSourceSelection (continued)**

	QSPI_CLOCK_SOURCE_SOURCE1_SEL1: McuClockReferencePointFrequency1 is used as input clock source for the QSPI dividers QSPI_CLOCK_SOURCE_SOURCE2_SEL2: McuClockReferencePointFrequency2 is used as input clock source for the QSPI dividers		
Default value	QSPI_CLOCK_SOURCE_DISABLED_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.51.35 McuQspiFrequency**Table 1328 Specification for McuQspiFrequency**

Name	McuQspiFrequency		
Description	Specifies the clock frequency for the QSPI peripheral. The McuQspiFrequency is applicable only if McuQspiClockSourceSelection is not set to QSPI_CLOCK_SOURCE_DISABLED. The target frequency to be configured should be perfectly divisible by one of the dividers mentioned in the Target Specification. Unit is expressed in Hz. Since 100 MHz is the most predominantly used QSPI frequency, therefore, it is taken as the default value for McuQspiFrequency		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 200000000.0		
Default value	100000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	McuQspiClockSourceSelection		

17.3.1.51.36 McuReferenceFrequency1**Table 1329 Specification for McuReferenceFrequency1**

Name	McuReferenceFrequency1
-------------	------------------------

MCU driver**Table 1329 Specification for McuReferenceFrequency1 (continued)**

Description	Specifies the reference frequency 1 for the MCDS. McuReferenceFrequency1 is calculated as follows: $McuReferenceFrequency1 = McuClockReferencePointFrequency0 / 24$ Unit is expressed in Hz.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 100000000.0		
Default value	12500000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockReferencePointFrequency0		

17.3.1.51.37 McuReferenceFrequency2**Table 1330 Specification for McuReferenceFrequency2**

Name	McuReferenceFrequency2		
Description	Specifies the reference frequency 2 for the MCDS. McuReferenceFrequency2 is calculated as follows: $McuReferenceFrequency2 = McuClockReferencePointFrequency1 / 24$ Unit is expressed in Hz.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 100000000.0		
Default value	6666667.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockReferencePointFrequency1		

17.3.1.51.38 McuSPBFrequency**Table 1331 Specification for McuSPBFrequency**

Name	McuSPBFrequency
-------------	-----------------

MCU driver**Table 1331 Specification for McuSPBFrequency (continued)**

Description	Specifies the intended target SPB frequency. The user should enter the intended target frequency expected for the SPB. The SPB should always be proportionate to McuClockReferencePointFrequency0. The possible divider values are available in the Target Specification. Unit is expressed in Hz.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	1.0 - 100000000.0		
Default value	100000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	ECU
Dependency	McuLowPowerDivValue		

17.3.1.51.39 McuSRIFrequency**Table 1332 Specification for McuSRIFrequency**

Name	McuSRIFrequency		
Description	Specifies the intended target SRI frequency. The user should enter the intended target frequency expected for the SRI. The SRI should always be proportionate to McuClockReferencePointFrequency0. The possible divider values are available in the Target Specification. Unit is expressed in Hz.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	1.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuLowPowerDivValue		

17.3.1.51.40 McuSTMFrequency**Table 1333 Specification for McuSTMFrequency**

Name	McuSTMFrequency
-------------	-----------------

MCU driver**Table 1333 Specification for McuSTMFrequency (continued)**

Description	Specifies the STM peripheral frequency. To disable the STM peripheral frequency, a value of 0 has to be configured to this configuration parameter. The STM frequency, if enabled, should be divisible by McuClockReferencePointFrequency0 with the divider values specified in the Target Specification. The STM frequency can be slower or faster or equal to the SPB frequency. Unit is expressed in Hz. The maximum possible value is taken as the default value for McuSTMFrequency.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 100000000.0		
Default value	100000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockReferencePointFrequency0		

17.3.1.52 Container: McuPublishedInformation

This container holds all the MCU-specific published information parameters.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.53 Container: McuRamSectorSettingConf

This container holds the configuration (parameters) for the RAM Sector setting.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

17.3.1.53.1 McuRamDefaultValue**Table 1334 Specification for McuRamDefaultValue**

Name	McuRamDefaultValue		
Description	Preset value used to fill the configured RAM section		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1334 Specification for McuRamDefaultValue (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.53.2 McuRamSectionBaseAddress**Table 1335 Specification for McuRamSectionBaseAddress**

Name	McuRamSectionBaseAddress		
Description	Represents the MCU RAM section base address. The default value for this parameter is CPU0 DSPR0 base address		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	1879048192		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.53.3 McuRamSectionSize**Table 1336 Specification for McuRamSectionSize**

Name	McuRamSectionSize		
Description	Represents the MCU RAM section size in bytes. McuRamSectionBaseAddress+ McuRamSectionSize should not exceed boundary for the RAM section.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	4		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

MCU driver**Table 1336 Specification for McuRamSectionSize (continued)**

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

17.3.1.53.4 McuRamSectorSettingId**Table 1337 Specification for McuRamSectorSettingId**

Name	McuRamSectorSettingId		
Description	Used as an argument for the Mcu_InitRamSection() API call.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.54 Container: McuResetReasonConf

An instance of this multi-instance container publishes one reset reason types available on the microcontroller.
 Reset reasons are provided as a pre-configuration file.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.54.1 McuResetReason**Table 1338 Specification for McuResetReason**

Name	McuResetReason		
Description	Specifies the reset reason types available on the microcontroller. McuResetReason is microcontroller dependent and provided as fixed configuration which is non-modifiable by the user.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-

MCU driver
Table 1338 Specification for McuResetReason (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

17.3.1.55 Container: McuStdByModeESR0Conf

This container defines the configuration (parameters) for the ESR0 in the standby mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.55.1 McuStdbyModeESR0EdgeDetection
Table 1339 Specification for McuStdbyModeESR0EdgeDetection

Name	McuStdbyModeESR0EdgeDetection		
Description	<p>Specifies if the trigger is generated on rising edge detection, falling edge detection, or both.</p> <p>McuStdbyModeESR0EdgeDetection is applicable only if McuMode is 2 (STANDBY) and McuStdbyModeESR0WakeupEnable is set to TRUE.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>ESR0_TRIG_FALLING_EDGE_SEL2: a trigger is generated on the falling edge detection</p> <p>ESR0_TRIG_RISING_EDGE_SEL1: a trigger is generated on the rising edge detection</p> <p>ESR0_TRIG_RISING_FALLING_EDGE_SEL3: a trigger is generated on both the rising edge detection and the falling edge detection</p>		
Default value	ESR0_TRIG_RISING_EDGE_SEL1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModeESR0WakeupEnable, McuMode		

17.3.1.55.2 McuStdbyModeESR0FltEnable
Table 1340 Specification for McuStdbyModeESR0FltEnable

Name	McuStdbyModeESR0FltEnable
Description	Specifies if the digital filter is enabled for the ESR0 to wake up from the standby mode.

MCU driver**Table 1340 Specification for McuStdbyModeESR0FltEnable (continued)**

	<p>McuStdbyModeESR0FltEnable is applicable only if McuMode is 2 (STANDBY) and McuStdbyModeESR0WakeupEnable is set to TRUE.</p> <p>Values:</p> <p>TRUE: digital filter is enabled for the ESR0 wakeup from the standby mode.</p> <p>FALSE: digital filter is disabled for the ESR0 wakeup from the standby mode.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModeESR0WakeupEnable, McuMode		

17.3.1.55.3 McuStdbyModeESR0WakeupEnable**Table 1341 Specification for McuStdbyModeESR0WakeupEnable**

Name	McuStdbyModeESR0WakeupEnable		
Description	<p>Specifies if the wakeup from the standby mode is enabled through ESR0.</p> <p>McuStdbyModeESR0WakeupEnable is applicable only if McuMode is 2 (STANDBY).</p> <p>Values:</p> <p>TRUE: wakeup from the standby mode through ESR0 is enabled</p> <p>FALSE: wakeup from the standby mode through ESR0 is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCU driver**Table 1341 Specification for McuStdbyModeESR0WakeupEnable (continued)**

Dependency	McuMode
-------------------	---------

17.3.1.56 Container: McuStdByModeESR1Conf

This container defines the configuration (parameters) for ESR1 in the standby mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.56.1 McuStdbyModeESR1EdgeDetection**Table 1342 Specification for McuStdbyModeESR1EdgeDetection**

Name	McuStdbyModeESR1EdgeDetection		
Description	Specifies if the trigger is generated on rising edge detection, falling edge detection or both. McuStdbyModeESR1EdgeDetection is applicable only if McuMode is 2 (STANDBY) and McuStdbyModeESR1WakeupEnable is set to TRUE.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ESR1_TRIG_FALLING_EDGE_SEL2: a trigger is generated on the falling edge detection ESR1_TRIG_RISING_EDGE_SEL1: a trigger is generated on the rising edge detection ESR1_TRIG_RISING_FALLING_EDGE_SEL3: a trigger is generated on both the rising edge detection and the falling edge detection		
Default value	ESR1_TRIG_RISING_EDGE_SEL1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModeESR1WakeupEnable, McuMode		

17.3.1.56.2 McuStdbyModeESR1FltEnable**Table 1343 Specification for McuStdbyModeESR1FltEnable**

Name	McuStdbyModeESR1FltEnable
Description	Specifies if the digital filter is enabled for the ESR1 to wake up from the standby mode. McuStdbyModeESR1FltEnable is applicable only if McuMode is 2 (STANDBY) and McuStdbyModeESR1WakeupEnable is set to TRUE. Values: TRUE: digital filter is enabled for ESR1 wakeup from the standby mode.

MCU driver**Table 1343 Specification for McuStdbyModeESR1FltEnable (continued)**

	FALSE: digital filter is disabled for ESR1 wakeup from the standby mode.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModeESR1WakeupEnable, McuMode		

17.3.1.56.3 McuStdbyModeESR1WakeupEnable**Table 1344 Specification for McuStdbyModeESR1WakeupEnable**

Name	McuStdbyModeESR1WakeupEnable		
Description	Specifies if the wakeup from the standby mode is enabled through ESR1. McuStdbyModeESR1WakeupEnable is applicable only if McuMode is 2 (STANDBY). Values: TRUE: wakeup from the standby mode through ESR1 is enabled FALSE: wakeup from the standby mode through ESR1 is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.57 Container: McuStdByModePinAConf

This container contains the configuration (parameters) for the standby PinA mode.

MCU driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.57.1 McuStdbyModePinAEdgeDetection

Table 1345 Specification for McuStdbyModePinAEdgeDetection

Name	McuStdbyModePinAEdgeDetection		
Description	<p>Specifies if the trigger will be generated on rising edge detection, falling edge detection or both.</p> <p>McuStdbyModePinAEdgeDetection is applicable only if McuMode is 2 (STANDBY) and McuStdbyModePinAWakeupEnable is set to TRUE.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PINA_TRIG_FALLING_EDGE_SEL2: a trigger is generated on the falling edge detection PINA_TRIG_RISING_EDGE_SEL1: a trigger is generated on the rising edge detection PINA_TRIG_RISING_FALLING_EDGE_SEL3: a trigger is generated on both the rising edge detection and the falling edge detection		
Default value	PINA_TRIG_RISING_EDGE_SEL1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModePinAWakeupEnable, McuMode		

17.3.1.57.2 McuStdbyModePinAFiltEnable

Table 1346 Specification for McuStdbyModePinAFiltEnable

Name	McuStdbyModePinAFiltEnable		
Description	<p>Specifies if the digital filter is enabled for PinA to wake up from the standby mode.</p> <p>McuStdbyModePinAFiltEnable is applicable only if McuMode is 2 (STANDBY) and McuStdbyModePinAWakeupEnable is set to TRUE.</p> <p>Values:</p> <p>TRUE: digital filter is enabled for PinA wakeup from the standby mode.</p> <p>FALSE: digital filter is disabled for PinA wakeup from the standby mode.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

MCU driver
Table 1346 Specification for McuStdbyModePinAFItEnable (continued)

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModePinAWakeupEnable, McuMode		

17.3.1.57.3 McuStdbyModePinAWakeupEnable
Table 1347 Specification for McuStdbyModePinAWakeupEnable

Name	McuStdbyModePinAWakeupEnable		
Description	<p>Specifies if the wake up from the standby mode is enabled through PinA.</p> <p>McuStdbyModePinAWakeupEnable is applicable only if McuMode is 2 (STANDBY).</p> <p>Values:</p> <p>TRUE: wakeup from the standby mode through PinA is enabled.</p> <p>FALSE: wakeup from the standby mode through PinA is disabled.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.58 Container: McuStdByModePinBConf

This container contains the configuration (parameters) for the standby PinB mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

MCU driver

17.3.1.58.1 McuStdbyModePinBEdgeDetection

Table 1348 Specification for McuStdbyModePinBEdgeDetection

Name	McuStdbyModePinBEdgeDetection		
Description	<p>Sceifies if the trigger will be generated on rising edge detection, falling edge detection or both.</p> <p>McuStdbyModePinBEdgeDetection is applicable only if McuMode is 2 (STANDBY) and McuStdbyModePinBWakeupEnable is set to TRUE.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>PINB_TRIG_FALLING_EDGE_SEL2: a trigger is generated on the falling edge detection</p> <p>PINB_TRIG_RISING_EDGE_SEL1: a trigger is generated on the rising edge detection.</p> <p>PINB_TRIG_RISING_FALLING_EDGE_SEL3: a trigger is generated on both the rising edge detection and the falling edge detection</p>		
Default value	PINB_TRIG_RISING_EDGE_SEL1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModePinBWakeupEnable, McuMode		

17.3.1.58.2 McuStdbyModePinBFiltEnable

Table 1349 Specification for McuStdbyModePinBFiltEnable

Name	McuStdbyModePinBFiltEnable		
Description	<p>Specifies if the digital filter is enabled for Pin B to wake up from the standby mode.</p> <p>McuStdbyModePinBFiltEnable is applicable only if McuMode is 2 (STANDBY) and McuStdbyModePinBWakeupEnable is set to TRUE.</p> <p>Values:</p> <p>TRUE: digital filter is enabled for PinB wakeup from the standby mode.</p> <p>FALSE: digital filter is disabled for PinB wakeup from the standby mode.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1349 Specification for McuStdbyModePinBFltEnable (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModePinBWakeupEnable, McuMode		

17.3.1.58.3 McuStdbyModePinBWakeupEnable**Table 1350 Specification for McuStdbyModePinBWakeupEnable**

Name	McuStdbyModePinBWakeupEnable		
Description	<p>Specifies if the wakeup from the standby mode is enabled through Pin B.</p> <p>McuStdbyModePinBWakeupEnable is applicable only if McuMode is 2 (STANDBY).</p> <p>Values:</p> <p>TRUE: wakeup from the standby mode through Pin B is enabled.</p> <p>FALSE: wakeup from the standby mode through Pin B is disabled.</p>		
Multiplicity	1..1	Type	EcuCBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.59 Container: McuStdByModeWakeUpTimerConf

This container contains the configuration (parameters) for the standby wakeup timer.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.59.1 McuStdbyModeWakeUpTimerClkDiv**Table 1351 Specification for McuStdbyModeWakeUpTimerClkDiv**

Name	McuStdbyModeWakeUpTimerClkDiv
Description	Specifies the wakeup timer clock source selection.

MCU driver**Table 1351 Specification for McuStdbyModeWakeUpTimerClkDiv (continued)**

	McuStdbyModeWakeUpTimerClkDiv is applicable only if McuStdbyModeWakeUpTimerEnable is set to TRUE.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	WUT_100KHZ_DIV_CLK_SEL1: wake up timer runs on 100 kHz frequency divided by 1024 divider value WUT_100KHZ_NO_DIV_CLK_SEL0: wake up timer runs on 100 kHz frequency.		
Default value	WUT_100KHZ_NO_DIV_CLK_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModeWakeUpTimerEnable, McuMode		

17.3.1.59.2 McuStdbyModeWakeUpTimerEnable**Table 1352 Specification for McuStdbyModeWakeUpTimerEnable**

Name	McuStdbyModeWakeUpTimerEnable		
Description	Specifies if the wake up from the standby mode is supported through the wake up timer. If McuStdbyModeWakeUpTimerEnable is set to TRUE, the wake up timer holds the capability to wake up from the standby mode. Values: TRUE: wakeup from the standby mode with the wake up timer is enabled FALSE: wakeup from the standby mode with the wake up timer is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

MCU driver

17.3.1.59.3 McuStdbyModeWakeupTimerMode

Table 1353 Specification for McuStdbyModeWakeupTimerMode

Name	McuStdbyModeWakeupTimerMode		
Description	<p>Specifies the wakeup timer mode.</p> <p>McuStdbyModeWakeupTimerMode is applicable only if McuStdbyModeWakeupTimerEnable is set to TRUE.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>WUT_AUTO_RELOAD_MODE_SEL0: counter starts from McuStdbyModeWakeupTimerValue. On counter underflow, the wakeup counter value is reloaded with McuStdbyModeWakeupTimerValue.</p> <p>WUT_AUTO_STOP_MODE_SEL1: counter starts from McuStdbyModeWakeupTimerValue. On counter underflow, wakeup timer stops.</p>		
Default value	WUT_AUTO_RELOAD_MODE_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode, McuStdbyModeWakeupTimerEnable		

17.3.1.59.4 McuStdbyModeWakeupTimerValue

Table 1354 Specification for McuStdbyModeWakeupTimerValue

Name	McuStdbyModeWakeupTimerValue		
Description	<p>Specifies the wakeup timer reload value.</p> <p>McuStdbyModeWakeupTimerValue is applicable only if McuStdbyModeWakeupTimerEnable is set to TRUE.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16777215		
Default value	16777215		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModeWakeupTimerEnable, McuMode		

MCU driver

17.3.1.60 Container: McuStdbyModeSettingConf

This container contains the configuration (parameters) for the MCU standby mode setting

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.60.1 McuStdbyModeBlankingFilterDelay

Table 1355 Specification for McuStdbyModeBlankingFilterDelay

Name	McuStdbyModeBlankingFilterDelay		
Description	<p>Specifies the delay for the blanking filter. The blanking filter delay ensures that valid event of VEXT rampup is detected as wakeup from the standby mode for a specified time interval.</p> <p>Actual value may be +/- 30% of mentioned value.</p> <p>This parameter is applicable only if McuMode is 2 (STANDBY) and .</p> <p>McuStdbyModeWakeupFromEVR is TRUE.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>DELAY_0_MS_SEL0: 0 ms blanking filter delay</p> <p>DELAY_10240_MS_SEL13: 10240 ms blanking filter delay</p> <p>DELAY_10_MS_SEL3: 10 ms blanking filter delay</p> <p>DELAY_1280_MS_SEL10: 1280 ms blanking filter delay</p> <p>DELAY_160_MS_SEL7: 160 ms blanking filter delay</p> <p>DELAY_20_MS_SEL4: 20 ms blanking filter delay</p> <p>DELAY_2560_MS_SEL11: 2560 ms blanking filter delay</p> <p>DELAY_2_5_MS_SEL1: 2.5 ms blanking filter delay</p> <p>DELAY_320_MS_SEL8: 320 ms blanking filter delay</p> <p>DELAY_40_MS_SEL5: 40 ms blanking filter delay</p> <p>DELAY_5120_MS_SEL12: 5120 ms blanking filter delay</p> <p>DELAY_5_MS_SEL2: 5 ms blanking filter delay</p> <p>DELAY_640_MS_SEL9: 640 ms blanking filter delay</p> <p>DELAY_80_MS_SEL6: 80 ms blanking filter delay</p>		
Default value	DELAY_0_MS_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuStdbyModeWakeupFromEVR, McuMode		

MCU driver

17.3.1.60.2 McuStdbyModeClkSelection

Table 1356 Specification for McuStdbyModeClkSelection

Name	McuStdbyModeClkSelection		
Description	Specifies the active oscillator clock during the standby mode operation. McuStdbyModeClkSelection is applicable only if McuMode is 2 (STANDBY).		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	OSC_CLOCK_100MHZ_ONLY_SEL0: only 100 MHz oscillator clock is active in the standby mode OSC_CLOCK_70KHZ_100MHZ_SEL1: both 70 KHz oscillator clock and 100 MHz oscillator clock are active in the standby mode		
Default value	OSC_CLOCK_100MHZ_ONLY_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.60.3 McuStdbyModeESR0TriStateEnable

Table 1357 Specification for McuStdbyModeESR0TriStateEnable

Name	McuStdbyModeESR0TriStateEnable		
Description	Specifies if the ESR0 is in tristate while in the standby mode. McuStdbyModeESR0TriStateEnable is applicable only if McuMode is 2 (STANDBY). Values: TRUE: tristate is enabled for ESR0 while in the standby mode. FALSE: tristate will be disabled for ESR0 while in the standby mode.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCU driver**Table 1357 Specification for McuStdbyModeESR0TriStateEnable (continued)**

Dependency	McuMode
-------------------	---------

17.3.1.60.4 McuStdbyModePORSTFilterEnable**Table 1358 Specification for McuStdbyModePORSTFilterEnable**

Name	McuStdbyModePORSTFilterEnable		
Description	<p>Specifies if the PORST digital filter is enabled or disabled.</p> <p>If McuStdbyModePORSTFilterEnable is set to FALSE, the PORST configuration delay = Analog PORST pad filter delay.</p> <p>If McuStdbyModePORSTFilterEnable is set to TRUE, the PORST configuration delay = Analog PORST pad filter delay + Digital filter delay.</p> <p>McuStdbyModePORSTFilterEnable is applicable only if McuMode is 2 (STANDBY).</p> <p>Values:</p> <p>TRUE: PORST digital filter is enabled</p> <p>FALSE: PORST digital filter is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.60.5 McuStdbyModePortTriStateEnable**Table 1359 Specification for McuStdbyModePortTriStateEnable**

Name	McuStdbyModePortTriStateEnable		
Description	<p>Specifies if the pads are in tristate while in the standby mode.</p> <p>McuStdbyModePortTriStateEnable is applicable only if McuMode is 2 (STANDBY).</p> <p>Values:</p> <p>TRUE: tristate is enabled for port pins while in the standby mode</p> <p>FALSE: tristate is disabled for port pins while in the standby mode</p>		

MCU driver**Table 1359 Specification for McuStdbyModePortTriStateEnable (continued)**

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.60.6 McuStdbyModeRamEnable**Table 1360 Specification for McuStdbyModeRamEnable**

Name	McuStdbyModeRamEnable		
Description	Selects the LMU blocks which stay powered up during the standby mode of operation. McuStdbyModeRamEnable is applicable only if McuMode is 2 (STANDBY).		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_STANDBYRAM_CPU0_BLK0_BLK1_SEL2: CPU0 dLMU Block0 Block1 is used as StandByRam MCU_STANDBYRAM_CPU0_BLK0_SEL1: CPU0 dLMU Block0 is used as StandByRam MCU_STANDBYRAM_CPU0_CPU1_BLK0_BLK1_SEL7: CPU0, CPU1s dLMU Block0 and Block 1 is used as StandByRam MCU_STANDBYRAM_CPU1_BLK0_BLK1_SEL4: CPU1 dLMU Block0 Block 1 is used as StandByRam MCU_STANDBYRAM_DISABLED_SEL0: StandByRam is disabled		
Default value	MCU_STANDBYRAM_DISABLED_SEL0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

MCU driver

17.3.1.60.7 McuStdbyModeWakeupFromEVR

Table 1361 Specification for McuStdbyModeWakeupFromEVR

Name	McuStdbyModeWakeupFromEVR		
Description	<p>Specifies if the wakeup from the standby mode is enabled through the wakeup timer.</p> <p>McuStdbyModeWakeupFromEVR is applicable only if McuMode is 2 (STANDBY).</p> <p>Values:</p> <p>TRUE: wakeup from the standby mode through EVR is enabled</p> <p>FALSE: wakeup from the standby mode through EVR is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.60.8 McuStdbyModeWakeupFromPORST

Table 1362 Specification for McuStdbyModeWakeupFromPORST

Name	McuStdbyModeWakeupFromPORST		
Description	<p>Specifies if the wakeup from the standby mode is enabled through PORST.</p> <p>McuStdbyModeWakeupFromPORST is applicable only if McuMode is 2 (STANDBY).</p> <p>Values:</p> <p>TRUE: wakeup from the standby mode through PORST is enabled</p> <p>FALSE: wakeup from the standby mode through PORST is disabled</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1362 Specification for McuStdbyModeWakeupFromPORST (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.60.9 McuStdbyModeWakeupFromSCR**Table 1363 Specification for McuStdbyModeWakeupFromSCR**

Name	McuStdbyModeWakeupFromSCR		
Description	<p>Specifies if the wakeup from the standby mode through controller is enabled.</p> <p>McuStdbyModeWakeupFromSCR is applicable only if McuMode is 2 (STANDBY).</p> <p>Values:</p> <p>TRUE: wakeup from the standby mode through the standby mode controller is enabled.</p> <p>FALSE: wakeup from the standby mode through the standby mode controller is disabled.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMode		

17.3.1.61 Container: McuSystemPllSettingConfig

This container holds the configuration (parameters) for the System PLL clock settings.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.61.1 McuClockReferencePointFrequency0**Table 1364 Specification for McuClockReferencePointFrequency0**

Name	McuClockReferencePointFrequency0
Description	User should configure the resulting target frequency after configuring the N, P and K2 divider for system PLL.

MCU driver**Table 1364 Specification for McuClockReferencePointFrequency0 (continued)**

	<p>By using the default value generation option this frequency can be auto-calculated with the configured values of McuMainOscillatorFrequency, McuSystemPlIIPDivide, McuSystemPlINDivide, and McuSystemPlIK2Divide dividers. Unit is expressed in Hz.</p> <p>The McuClockReferencePointFrequency0 for NORMAL_MODE should be in the range from: 20 to 300 MHz. If McuClockDistributionInpClockSel is selected as BACKUP_INPUT_CLOCK_SRC_SELECT then manually configure this clock to Fback = 100 MHz.</p> <p>fSOURCE0 is McuClockReferencePointFrequency0.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	20000000.0 - 300000000.0		
Default value	300000000.0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuMainOscillatorFrequency, McuSystemPlIK2Divide, McuSystemPlINDivide, McuSystemPlIIPDivide, McuPlIISrcSelection		

17.3.1.61.2 McuFMPllModAmp**Table 1365 Specification for McuFMPllModAmp**

Name	McuFMPllModAmp		
Description	<p>McuFMPllModAmp is the percentage value for modulation amplitude for PLL frequency modulation.</p> <p>MODCFG[9:0] bits of SCU_PLLCON2 is used and is equated as</p> $= (64 * \text{McuFMPllModAmp}/100 * \text{McuMainOscillatorFrequency}/\text{McuPlIIPDivide} * \text{McuPlINDivide}/3.6);$ <p>where (McuFMPllModAmp is expressed in percentage and McuMainOscillatorFrequency in MHz).</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0 - 2.0		
Default value	1.25		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCU driver**Table 1365 Specification for McuFmPllModAmp (continued)**

Dependency	McuFmPllEnable, McuClockDistributionInpClockSel
-------------------	---

17.3.1.61.3 McuFmPllEnable**Table 1366 Specification for McuFmPllEnable**

Name	McuFmPllEnable		
Description	Configuration to enable/disable PLL frequency modulation. Values: TRUE: enables PLL frequency modulation. FALSE: disables PLL frequency modulation		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.61.4 McuPllInputSrcSelection**Table 1367 Specification for McuPllInputSrcSelection**

Name	McuPllInputSrcSelection		
Description	Configuration to select the input clock source for both the PLLs. Note: When Backup clock is selected as source to PLL, oscillator watchdog may raise a SMU alarm (OSC clock frequency out of range) since OSC Watchdog can monitor in range of 16-40MHz. The SMU alarm for oscillator watchdog should be disabled when using Backup clock as source to PLLs.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	BACKUP_CLOCK_SRC_SELECT_SEL0: backup clock is selected as an input source for the system and peripheral PLLs OSC_CLOCK_SRC_SELECT_SEL1: oscillator clock is selected as an input source for the system and peripheral PLLs		

MCU driver**Table 1367 Specification for McuPllInputSrcSelection (continued)**

	SYSCLK_SRC_SELECT_SEL2: SYSCLK pin is selected as an input source for the system and peripheral PLLs		
Default value	OSC_CLOCK_SRC_SELECT_SEL1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.61.5 McuSysPllK2DivStepDownChangeDelay**Table 1368 Specification for McuSysPllK2DivStepDownChangeDelay**

Name	McuSysPllK2DivStepDownChangeDelay		
Description	The delay required to configure the step changes between two consecutive changes in the K2 divider value. McuSysPllK2DivStepDownChangeDelay is a common delay used for system Pll0 frequency ramp down sequences through the K2 divider. Note : The value is expressed in microseconds (us).		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	5 - 100		
Default value	10		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.61.6 McuSysPllK2DivStepUpChangeDelay**Table 1369 Specification for McuSysPllK2DivStepUpChangeDelay**

Name	McuSysPllK2DivStepUpChangeDelay		
Description	The delay required to configure the step changes between two consecutive changes in the K2 divider value. McuSysPllK2DivStepUpChangeDelay is a common delay used for system Pll0 frequency ramp up sequences through the K2 divider. Note : The value is expressed in microseconds (us).		
Multiplicity	1..1	Type	EcuIntegerParamDef

MCU driver**Table 1369 Specification for McuSysPllK2DivStepUpChangeDelay (continued)**

Range	5 - 100		
Default value	10		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.61.7 McuSystemPllK2Divider**Table 1370 Specification for McuSystemPllK2Divider**

Name	McuSystemPllK2Divider		
Description	<p>Three bit output divider. Even values are preferred to get 50% duty cycle.</p> <p>Clock equations are incremented by 1 to this parameter.</p> <p>Note : Changing the system operation frequency by changing the value of the K2-divider has a direct coupling to the power consumption of the device. Therefore this should be done carefully.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 7		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.61.8 McuSystemPllNDivider**Table 1371 Specification for McuSystemPllNDivider**

Name	McuSystemPllNDivider		
Description	<p>Seven bit feedback divider value used for the generation of system clock.</p> <p>Clock equations are incremented by 1 to this parameter.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 127		

MCU driver**Table 1371 Specification for McuSystemPllNDivider (continued)**

Default value	29		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.61.9 McuSystemPllPDivider**Table 1372 Specification for McuSystemPllPDivider**

Name	McuSystemPllPDivider		
Description	Frequency divider of main oscillator (3 bits) Clock equations are incremented by 1 to this parameter.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockDistributionInpClockSel		

17.3.1.62 Container: McuResetSettingConf

This container defines the configuration parameters for the reset settings.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.62.1 McuESR0ResetConf**Table 1373 Specification for McuESR0ResetConf**

Name	McuESR0ResetConf		
Description	Refers to the response of the ESR0 reset request.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

MCU driver**Table 1373 Specification for McuESR0ResetConf (continued)**

Range	MCU_ESR0_APPLICATION_RESET_SEL2: application reset request is triggered MCU_ESR0_NO_RESET_SEL0: no reset request is triggered MCU_ESR0_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_ESR0_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.2 McuESR1ResetConf**Table 1374 Specification for McuESR1ResetConf**

Name	McuESR1ResetConf		
Description	Refers to the response of the ESR1 reset request.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_ESR1_APPLICATION_RESET_SEL2: application reset request is triggered MCU_ESR1_NO_RESET_SEL0: no reset request is triggered MCU_ESR1_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_ESR1_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.3 McuSMUResetConf**Table 1375 Specification for McuSMUResetConf**

Name	McuSMUResetConf		
Description	Refers to the response of the SMU reset request.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

MCU driver**Table 1375 Specification for McuSMUResetConf (continued)**

Range	MCU_SMU_APPLICATION_RESET_SEL2: application reset request is triggered MCU_SMU_NO_RESET_SEL0: no reset request is triggered MCU_SMU_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_SMU_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.4 McuSTM0ResetConf**Table 1376 Specification for McuSTM0ResetConf**

Name	McuSTM0ResetConf		
Description	Refers to the response of the STM0 reset request.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_STM0_APPLICATION_RESET_SEL2: application reset request is triggered MCU_STM0_NO_RESET_SEL0: no reset request is triggered MCU_STM0_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_STM0_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.5 McuSTM0ResetOnApplResetEnable**Table 1377 Specification for McuSTM0ResetOnApplResetEnable**

Name	McuSTM0ResetOnApplResetEnable
Description	Refers to the enabling of resetting the value of STM0 when an application reset is requested. TRUE: STM0 is reset when the application reset is triggered. FALSE: STM0 is not reset when the application reset is triggered.

MCU driver**Table 1377 Specification for McuSTM0ResetOnApplResetEnable (continued)**

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.6 McuSTM1ResetConf**Table 1378 Specification for McuSTM1ResetConf**

Name	McuSTM1ResetConf		
Description	Refers to the response of the STM1 reset request. If the STM1 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_STM1_APPLICATION_RESET_SEL2: application reset request is triggered MCU_STM1_NO_RESET_SEL0: no reset request is triggered MCU_STM1_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_STM1_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.7 McuSTM1ResetOnApplResetEnable**Table 1379 Specification for McuSTM1ResetOnApplResetEnable**

Name	McuSTM1ResetOnApplResetEnable
Description	Refers to the enabling of resetting the value of STM1 when an application reset is requested.

MCU driver**Table 1379 Specification for McuSTM1ResetOnApplResetEnable (continued)**

	TRUE: STM1 is reset when the application reset is triggered. FALSE: STM1 is not reset when the application reset is triggered. If the STM1 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.8 McuSTM2ResetConf**Table 1380 Specification for McuSTM2ResetConf**

Name	McuSTM2ResetConf		
Description	Refers to the response of the STM2 reset request. If the STM2 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_STM2_APPLICATION_RESET_SEL2: application reset request is triggered MCU_STM2_NO_RESET_SEL0: no reset request is triggered MCU_STM2_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_STM2_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.62.9 McuSTM2ResetOnApplResetEnable

Table 1381 Specification for McuSTM2ResetOnApplResetEnable

Name	McuSTM2ResetOnApplResetEnable		
Description	Refers to the enabling of resetting the value of STM2 when an application reset is requested. TRUE: STM2 is reset when the application reset is triggered. FALSE: STM2 is not reset when the application reset is triggered. If the STM2 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.10 McuSTM3ResetConf

Table 1382 Specification for McuSTM3ResetConf

Name	McuSTM3ResetConf		
Description	Refers to the response of the STM3 reset request. If the STM3 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_STM3_APPLICATION_RESET_SEL2: application reset request is triggered MCU_STM3_NO_RESET_SEL0: no reset request is triggered MCU_STM3_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_STM3_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.62.11 McuSTM3ResetOnApplResetEnable

Table 1383 Specification for McuSTM3ResetOnApplResetEnable

Name	McuSTM3ResetOnApplResetEnable		
Description	Refers to the enabling of resetting the value of STM3 when an application reset is requested. TRUE: STM3 is reset when the application reset is triggered. FALSE: STM3 is not reset when the application reset is triggered. If the STM3 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.12 McuSTM4ResetConf

Table 1384 Specification for McuSTM4ResetConf

Name	McuSTM4ResetConf		
Description	Refers to the response of the STM4 reset request. If the STM4 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_STM4_APPLICATION_RESET_SEL2: application reset request is triggered MCU_STM4_NO_RESET_SEL0: no reset request is triggered MCU_STM4_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_STM4_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.62.13 McuSTM4ResetOnApplResetEnable

Table 1385 Specification for McuSTM4ResetOnApplResetEnable

Name	McuSTM4ResetOnApplResetEnable		
Description	Refers to the enabling of resetting the value of STM4 when an application reset is requested. TRUE: STM4 is reset when the application reset is triggered. FALSE: STM4 is not reset when the application reset is triggered. If the STM4 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.14 McuSTM5ResetConf

Table 1386 Specification for McuSTM5ResetConf

Name	McuSTM5ResetConf		
Description	Refers to the response of the STM5 reset request. If the STM5 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_STM5_APPLICATION_RESET_SEL2: application reset request is triggered MCU_STM5_NO_RESET_SEL0: no reset request is triggered MCU_STM5_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_STM5_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.62.15 McuSTM5ResetOnApplResetEnable****Table 1387 Specification for McuSTM5ResetOnApplResetEnable**

Name	McuSTM5ResetOnApplResetEnable		
Description	Refers to enabling of resetting the value of STM5 when an application reset is requested. TRUE: STM5 is reset when the application reset is triggered. FALSE: STM5 is not reset when the application reset is triggered. If the STM5 does not exist on the hardware, the parameter is disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.62.16 McuSWResetConf**Table 1388 Specification for McuSWResetConf**

Name	McuSWResetConf		
Description	Refers to the response of the software reset request.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	MCU_SW_APPLICATION_RESET_SEL2: application reset request is triggered MCU_SW_NO_RESET_SEL0: no reset request is triggered MCU_SW_SYSTEM_RESET_SEL1: system reset request is triggered		
Default value	MCU_SW_NO_RESET_SEL0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver

17.3.1.63 Container: McuTrapSettingConf

This container defines the configuration parameters for the trap settings.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

17.3.1.63.1 McuCPU0ESR0TrapEnable

Table 1389 Specification for McuCPU0ESR0TrapEnable

Name	McuCPU0ESR0TrapEnable		
Description	Enables the trap request for CPU0 from the ESR0 source. TRUE: MCU CPU0 trap can be generated from the ESR0 source. FALSE: MCU CPU0 trap cannot be generated from the ESR0 source.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.2 McuCPU0ESR1TrapEnable

Table 1390 Specification for McuCPU0ESR1TrapEnable

Name	McuCPU0ESR1TrapEnable		
Description	Enables the trap request for CPU0 from the ESR1 source. TRUE: MCU CPU0 trap can be generated from the ESR1 source FALSE: MCU CPU0 trap cannot be generated from the ESR1 source		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1390 Specification for McuCPU0ESR1TrapEnable (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.3 McuCPU0SMUTrapEnable**Table 1391 Specification for McuCPU0SMUTrapEnable**

Name	McuCPU0SMUTrapEnable		
Description	Enables the trap request for CPU0 from the SMU source TRUE: MCU CPU0 trap can be generated from the SMU source FALSE: MCU CPU0 trap cannot be generated from the SMU source		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.4 McuCPU0Trap2Enable**Table 1392 Specification for McuCPU0Trap2Enable**

Name	McuCPU0Trap2Enable		
Description	Enables the trap request for CPU0 from the TRAP2 source TRUE: MCU CPU0 trap can be generated from the TRAP2 source FALSE: MCU CPU0 trap cannot be generated from the TRAP2 source		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

MCU driver**Table 1392 Specification for McuCPU0Trap2Enable (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.5 McuCPU1ESR0TrapEnable**Table 1393 Specification for McuCPU1ESR0TrapEnable**

Name	McuCPU1ESR0TrapEnable		
Description	Enables the trap request for CPU1 from the ESR0 source TRUE: MCU CPU1 trap can be generated from the ESR0 source FALSE: MCU CPU1 trap cannot be generated from the ESR0 source If CPU1 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.6 McuCPU1ESR1TrapEnable**Table 1394 Specification for McuCPU1ESR1TrapEnable**

Name	McuCPU1ESR1TrapEnable		
Description	Enables the trap request for CPU1 from the ESR1 source TRUE: MCU CPU1 trap can be generated from the ESR1 source FALSE: MCU CPU1 trap cannot be generated from the ESR1 source If CPU1 is not available on the hardware, this parameter is disabled.		

MCU driver**Table 1394 Specification for McuCPU1ESR1TrapEnable (continued)**

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.7 McuCPU1SMUTrapEnable**Table 1395 Specification for McuCPU1SMUTrapEnable**

Name	McuCPU1SMUTrapEnable		
Description	Enables the trap request for CPU1 from the SMU source TRUE: MCU CPU1 trap can be generated from the SMU source FALSE: MCU CPU1 trap cannot be generated from the SMU source If CPU1 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.8 McuCPU1Trap2Enable**Table 1396 Specification for McuCPU1Trap2Enable**

Name	McuCPU1Trap2Enable
-------------	--------------------

MCU driver**Table 1396 Specification for McuCPU1Trap2Enable (continued)**

Description	Enables the trap request for CPU1 from the TRAP2 source TRUE: MCU CPU1 trap can be generated from the TRAP2 source FALSE: MCU CPU1 trap cannot be generated from the TRAP2 source If CPU1 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.9 McuCPU2ESR0TrapEnable**Table 1397 Specification for McuCPU2ESR0TrapEnable**

Name	McuCPU2ESR0TrapEnable		
Description	Enables the trap request for CPU2 from the ESR0 source TRUE: MCU CPU2 trap can be generated from the ESR0 source FALSE: MCU CPU2 trap cannot be generated from the ESR0 source If CPU2 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.63.10 McuCPU2ESR1TrapEnable****Table 1398 Specification for McuCPU2ESR1TrapEnable**

Name	McuCPU2ESR1TrapEnable		
Description	Enables the trap request for CPU2 from the ESR1 source TRUE: MCU CPU2 trap can be generated from the ESR1 source FALSE: MCU CPU2 trap cannot be generated from the ESR1 source If CPU2 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.11 McuCPU2SMUTrapEnable**Table 1399 Specification for McuCPU2SMUTrapEnable**

Name	McuCPU2SMUTrapEnable		
Description	Enables the trap request for CPU2 from the SMU source TRUE: MCU CPU2 trap can be generated from the SMU source FALSE: MCU CPU2 trap cannot be generated from the SMU source If CPU2 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCU driver**Table 1399 Specification for McuCPU2SMUTrapEnable (continued)**

Dependency	-
-------------------	---

17.3.1.63.12 McuCPU2Trap2Enable**Table 1400 Specification for McuCPU2Trap2Enable**

Name	McuCPU2Trap2Enable		
Description	<p>Enables the trap request for CPU2 from the TRAP2 source</p> <p>TRUE: MCU CPU2 trap can be generated from the TRAP2 source</p> <p>FALSE: MCU CPU2 trap cannot be generated from the TRAP2 source</p> <p>If CPU2 is not available on the hardware, this parameter is disabled.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.13 McuCPU3ESR0TrapEnable**Table 1401 Specification for McuCPU3ESR0TrapEnable**

Name	McuCPU3ESR0TrapEnable		
Description	<p>Enables the trap request for CPU3 from the ESR0 source</p> <p>TRUE: MCU CPU3 trap can be generated from the ESR0 source</p> <p>FALSE: MCU CPU3 trap cannot be generated from the ESR0 source</p> <p>If CPU3 is not available on the hardware, this parameter is disabled.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1401 Specification for McuCPU3ESR0TrapEnable (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.14 McuCPU3ESR1TrapEnable**Table 1402 Specification for McuCPU3ESR1TrapEnable**

Name	McuCPU3ESR1TrapEnable		
Description	Enables the trap request for CPU3 from the ESR1 source TRUE: MCU CPU3 trap can be generated from the ESR1 source FALSE: MCU CPU3 trap cannot be generated from the ESR1 source If CPU3 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.15 McuCPU3SMUTrapEnable**Table 1403 Specification for McuCPU3SMUTrapEnable**

Name	McuCPU3SMUTrapEnable		
Description	Enables the trap request for CPU3 from the SMU source TRUE: MCU CPU3 trap can be generated from the SMU source FALSE: MCU CPU3 trap cannot be generated from the SMU source If CPU3 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE		

MCU driver**Table 1403 Specification for McuCPU3SMUTrapEnable (continued)**

	FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.16 McuCPU3Trap2Enable**Table 1404 Specification for McuCPU3Trap2Enable**

Name	McuCPU3Trap2Enable		
Description	Enables the trap request for CPU3 from the TRAP2 source TRUE: MCU CPU3 trap can be generated from the TRAP2 source FALSE: MCU CPU3 trap cannot be generated from the TRAP2 source If CPU3 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.17 McuCPU4ESR0TrapEnable**Table 1405 Specification for McuCPU4ESR0TrapEnable**

Name	McuCPU4ESR0TrapEnable		
Description	Enables the trap request for CPU4 from the ESR0 source TRUE: MCU CPU4 trap can be generated from the ESR0 source FALSE: MCU CPU4 trap cannot be generated from the ESR0 source		

MCU driver**Table 1405 Specification for McuCPU4ESR0TrapEnable (continued)**

	If CPU4 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.18 McuCPU4ESR1TrapEnable**Table 1406 Specification for McuCPU4ESR1TrapEnable**

Name	McuCPU4ESR1TrapEnable		
Description	Enables the trap request for CPU4 from the ESR1 source TRUE: MCU CPU4 trap can be generated from the ESR1 source FALSE: MCU CPU4 trap cannot be generated from the ESR1 source If CPU4 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

MCU driver**17.3.1.63.19 McuCPU4SMUTrapEnable****Table 1407 Specification for McuCPU4SMUTrapEnable**

Name	McuCPU4SMUTrapEnable		
Description	Enables the trap request for CPU4 from the SMU source TRUE: MCU CPU4 trap can be generated from the SMU source FALSE: MCU CPU4 trap cannot be generated from the SMU source If CPU4 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.20 McuCPU4Trap2Enable**Table 1408 Specification for McuCPU4Trap2Enable**

Name	McuCPU4Trap2Enable		
Description	Enables the trap request for CPU4 from the TRAP2 source TRUE: MCU CPU4 trap can be generated from the TRAP2 source FALSE: MCU CPU4 trap cannot be generated from the TRAP2 source If CPU4 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

MCU driver**Table 1408 Specification for McuCPU4Trap2Enable (continued)**

Dependency	-
-------------------	---

17.3.1.63.21 McuCPU5ESR0TrapEnable**Table 1409 Specification for McuCPU5ESR0TrapEnable**

Name	McuCPU5ESR0TrapEnable		
Description	<p>Enables the trap request for CPU5 from the ESR0 source</p> <p>TRUE: MCU CPU5 trap can be generated from the ESR0 source</p> <p>FALSE: MCU CPU5 trap cannot be generated from the ESR0 source</p> <p>If CPU5 is not available on the hardware, this parameter is disabled.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.22 McuCPU5ESR1TrapEnable**Table 1410 Specification for McuCPU5ESR1TrapEnable**

Name	McuCPU5ESR1TrapEnable		
Description	<p>Enables the trap request for CPU5 from the ESR1 source</p> <p>TRUE: MCU CPU5 trap can be generated from the ESR1 source</p> <p>FALSE: MCU CPU5 trap cannot be generated from the ESR1 source</p> <p>If CPU5 is not available on the hardware, this parameter is disabled.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

MCU driver**Table 1410 Specification for McuCPU5ESR1TrapEnable (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.23 McuCPU5SMUTrapEnable**Table 1411 Specification for McuCPU5SMUTrapEnable**

Name	McuCPU5SMUTrapEnable		
Description	Enables the trap request for CPU5 from the SMU source TRUE: MCU CPU5 trap can be generated from the SMU source FALSE: MCU CPU5 trap cannot be generated from the SMU source If CPU5 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.1.63.24 McuCPU5Trap2Enable**Table 1412 Specification for McuCPU5Trap2Enable**

Name	McuCPU5Trap2Enable		
Description	Enables the trap request for CPU5 from the TRAP2 source TRUE: MCU CPU5 trap can be generated from the TRAP2 source FALSE: MCU CPU5 trap cannot be generated from the TRAP2 source If CPU5 is not available on the hardware, this parameter is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE		

MCU driver**Table 1412 Specification for McuCPU5Trap2Enable (continued)**

	FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

17.3.2 Functions - Type definitions**17.3.2.1 Mcu_17_Ccu6_TimerChIntType****Table 1413 Specification for Mcu_17_Ccu6_TimerChIntType**

Syntax	Mcu_17_Ccu6_TimerChIntType		
Type	Structure		
File	Mcu_17_TimerIp.h		
Range	Mcu_17_Ccu6_TimerChIdentifierType	CCU6 Timer Id	
	TimerId		
	uint8 IEnBitPos	Bit position of interrupt to be enabled	
	uint8 IEnLen	Length of interrupt to be enabled	
Description	uint8 RegVal	Value to be written in register	
	Data type for configuring interrupts in CCU6.		
	Source		
	IFX		

17.3.2.2 Mcu_17_Eru_SrcIdentifierType**Table 1414 Specification for Mcu_17_Eru_SrcIdentifierType**

Syntax	Mcu_17_Eru_SrcIdentifierType		
Type	uint8		
File	Mcu_17_TimerIp.h		
Range	0-255	Range of uint8	
Description	Data type for user of ERU		
Source	IFX		

MCU driver**17.3.2.3 Mcu_17_Gpt12_ClkPrescalarType****Table 1415 Specification for Mcu_17_Gpt12_ClkPrescalarType**

Syntax	Mcu_17_Gpt12_ClkPrescalarType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	0 - MCU_GPT12_GPT1_CLOCK_DIV8	GPT1 block clock divider 8
	1 - MCU_GPT12_GPT1_CLOCK_DIV4	GPT1 block clock divider 4
	2 - MCU_GPT12_GPT1_CLOCK_DIV32	GPT1 block clock divider 32
	3 - MCU_GPT12_GPT1_CLOCK_DIV16	GPT1 block clock divider 16
	0 - MCU_GPT12_GPT2_CLOCK_DIV4	GPT2 block clock divider 4
	1 - MCU_GPT12_GPT2_CLOCK_DIV2	GPT2 block clock divider 2
	2 - MCU_GPT12_GPT2_CLOCK_DIV16	GPT2 block clock divider 16
	3 - MCU_GPT12_GPT2_CLOCK_DIV8	GPT2 block clock divider 8
Description	This type indicates clock divider value for fGPT for a particular block.	
Source	IFX	

17.3.2.4 Mcu_17_Gpt12_TimerBlockType**Table 1416 Specification for Mcu_17_Gpt12_TimerBlockType**

Syntax	Mcu_17_Gpt12_TimerBlockType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	MCU_GPT12_GPT1_BLOCK	GPT1 block
	MCU_GPT12_GPT2_BLOCK	GPT2 block
Description	This type indicates whether the GPT timer block is - GPT1 or GPT2.	
Source	IFX	

17.3.2.5 Mcu_17_Gtm_AtomCh**Table 1417 Specification for Mcu_17_Gtm_AtomCh**

Syntax	Mcu_17_Gtm_AtomCh	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	lfx_GTM_ATOM_CH CH	ATOM channels
	uint8 Reserved1[20]	Reserved bits
Description	Structure of ATOM channels	

MCU driver**Table 1417 Specification for Mcu_17_Gtm_AtomCh (continued)**

Source	IFX
---------------	-----

17.3.2.6 Mcu_17_Gtm_AtomChArray**Table 1418 Specification for Mcu_17_Gtm_AtomChArray**

Syntax	Mcu_17_Gtm_AtomChArray	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	Mcu_17_Gtm_AtomCh ATOM_CHANNEL[8]	ATOM channel array
Description	Array of size of number of ATOM channels	
Source	IFX	

17.3.2.7 Mcu_17_Gtm_MappedPortTimerOutType**Table 1419 Specification for Mcu_17_Gtm_MappedPortTimerOutType**

Syntax	Mcu_17_Gtm_MappedPortTimerOutType	
Type	Enumeration	
File	Mcu_17_TimerIp.h	
Range	0 - 0-MCU_OUT_TIMER_MAPPED_COL_A	Timer output mapped to column A
	1 - 1-MCU_OUT_TIMER_MAPPED_COL_B	Timer output mapped to column B
	2 - 2-MCU_OUT_TIMER_MAPPED_COL_C	Timer output mapped to column C
	3 - 3-MCU_OUT_TIMER_MAPPED_COL_D	Timer output mapped to column D
	4 - 4-MCU_OUT_TIMER_MAPPED_COL_E	Timer output mapped to column E
	5 - 5-MCU_OUT_TIMER_MAPPED_COL_F	Timer output mapped to column F
	6 - 6-MCU_OUT_TIMER_MAPPED_COL_G	Timer output mapped to column G
	7 - 7-MCU_OUT_TIMER_MAPPED_COL_H	Timer output mapped to column H
	8 - 8-MCU_OUT_TIMER_MAPPED_COL_I	Timer output mapped to column I
	9 - 9-MCU_OUT_TIMER_MAPPED_COL_J	Timer output mapped to column J
	10 - 10-MCU_OUT_TIMER_MAPPED_COL_K	Timer output mapped to column K
	11 - 11-MCU_OUT_TIMER_MAPPED_COL_L	Timer output mapped to column L
Description	Mcu_17_Gtm_MappedPortTimerOutType enumerates the column series to connect the GTM timers TOM/ATOM to port pins	
Source	IFX	

MCU driver**17.3.2.8 Mcu_17_Gtm_TimCh****Table 1420 Specification for Mcu_17_Gtm_TimCh**

Syntax	Mcu_17_Gtm_TimCh	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	Ifx_GTM_TIM_CH CH	TIM channel
	uint8 Reserved1[64]	Reserved bits
Description	Structure of TIM channels	
Source	IFX	

17.3.2.9 Mcu_17_Gtm_TimChArray**Table 1421 Specification for Mcu_17_Gtm_TimChArray**

Syntax	Mcu_17_Gtm_TimChArray	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	Mcu_17_Gtm_TimCh TIM_CHANNEL[8]	TIM channel array
Description	Array of size of number of TIM channels	
Source	IFX	

17.3.2.10 Mcu_17_Gtm_TimerEnableType**Table 1422 Specification for Mcu_17_Gtm_TimerEnableType**

Syntax	Mcu_17_Gtm_TimerEnableType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	MCU_GTM_TIMER_DISABLE	GTM Timer is disabled
	MCU_GTM_TIMER_ENABLE	GTM Timer is enabled
Description	This type identifies if the GTM output timer is either enabled or disabled	
Source	IFX	

17.3.2.11 Mcu_17_Gtm_TimerEnTriggerType**Table 1423 Specification for Mcu_17_Gtm_TimerEnTriggerType**

Syntax	Mcu_17_Gtm_TimerEnTriggerType	
Type	Enumeration	
File	Mcu_17_TimerIp.h	

MCU driver**Table 1423 Specification for Mcu_17_Gtm_TimerEnTriggerType (continued)**

Range	0 - MCU_NOCHANGE_ON_TRIGGER	No change on trigger
	1 - MCU_DISABLE_ON_TRIGGER	Disable on trigger
	2 - MCU_ENABLE_ON_TRIGGER	Enable on trigger
Description	Data type for enabling channel on trigger	
Source	IFX	

17.3.2.12 Mcu_17_Gtm_TimerOutputEnableType**Table 1424 Specification for Mcu_17_Gtm_TimerOutputEnableType**

Syntax	Mcu_17_Gtm_TimerOutputEnableType	
Type	Enumeration	
File	Mcu_17_TimerIp.h	
Range	0 - MCU_GTM_TIMER_OUT_DISABLE	Disable timer output
	1 - MCU_GTM_TIMER_OUT_ENABLE	Enable timer output
Description	This type indicates if the timer output is connected or not to the rest of the controller.	
Source	IFX	

17.3.2.13 Mcu_17_Gtm_TimerOutputEnTriggerType**Table 1425 Specification for Mcu_17_Gtm_TimerOutputEnTriggerType**

Syntax	Mcu_17_Gtm_TimerOutputEnTriggerType	
Type	Enumeration	
File	Mcu_17_TimerIp.h	
Range	0 - MCU_NOCHANGE_OUT_ON_TRIGGER	No change in output on trigger
	1 - MCU_DISABLE_OUT_ON_TRIGGER	Disable output on trigger
	2 - MCU_ENABLE_OUT_ON_TRIGGER	Enable output on trigger
Description	Data type for enabling the timer output on a trigger	
Source	IFX	

17.3.2.14 Mcu_17_Gtm_TimerUpdateEnableType**Table 1426 Specification for Mcu_17_Gtm_TimerUpdateEnableType**

Syntax	Mcu_17_Gtm_TimerUpdateEnableType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	MCU_GTM_TIMER_UPDATE_DISABLE	GTM Timer update is disabled

MCU driver**Table 1426 Specification for Mcu_17_Gtm_TimerUpdateEnableType (continued)**

	MCU_GTM_TIMER_UPDATE_ENABLE	GTM Timer update is enabled
Description	Mcu_17_Gtm_TimerUpdateEnableType specifies whether timer update is enabled or disabled	
Source	IFX	

17.3.2.15 Mcu_17_Gtm_TomCh**Table 1427 Specification for Mcu_17_Gtm_TomCh**

Syntax	Mcu_17_Gtm_TomCh	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	Ifx_GTM_TOM_CH CH	TOM channels
	uint8 Reserved1[20]	Reserved bits
Description	Structure of TOM channels	
Source	IFX	

17.3.2.16 Mcu_17_Gtm_TomChArray**Table 1428 Specification for Mcu_17_Gtm_TomChArray**

Syntax	Mcu_17_Gtm_TomChArray	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	Mcu_17_Gtm_TomCh TOM_CHANNEL[16]	Tom channel array
Description	Array of size of number of TOM channels	
Source	IFX	

17.3.2.17 Mcu_17_Gtm_TomTgc**Table 1429 Specification for Mcu_17_Gtm_TomTgc**

Syntax	Mcu_17_Gtm_TomTgc	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	Ifx_GTM_TOM_TGC_GLB_CTRL TGC_GLB_CTRL	TOM global control
	Ifx_GTM_TOM_TGC_ACT_TB TGC_ACT_TB	TOM time base
	Ifx_GTM_TOM_TGC_FUPD_CTRL TGC_FUPD_CTRL	TOM force update control

MCU driver**Table 1429 Specification for Mcu_17_Gtm_TomTgc (continued)**

	Ifx_GTM_TOM_TGC_INT_TRIG TGC_INT_TRIG	Internal trigger
	uint8 Reserved2[48]	Reserved bits
	Ifx_GTM_TOM_TGC_ENDIS_CTRL TGC_ENDIS_CTRL	Enable/disable control
	Ifx_GTM_TOM_TGC_ENDIS_STAT TGC_ENDIS_STAT	Enable/disable status
	Ifx_GTM_TOM_TGC_OUTEN_CTRL TGC_OUTEN_CTRL	TOM output enable control
	Ifx_GTM_TOM_TGC_OUTEN_STAT TGC_OUTEN_STAT	TOM output enable status
	uint8 Reserved3[432]	
Description	Data type for TOM TGC	
Source	IFX	

17.3.2.18 Mcu_17_Gtm_TomTgcArray**Table 1430 Specification for Mcu_17_Gtm_TomTgcArray**

Syntax	Mcu_17_Gtm_TomTgcArray	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	uint8 Reserved1[48]	Reserved bits
	Mcu_17_Gtm_TomTgc TOM_TGC	TOM global control register
Description	Array of type of TOM TGC	
Source	IFX	

17.3.2.19 Mcu_17_Stm_ComIntEnableType**Table 1431 Specification for Mcu_17_Stm_ComIntEnableType**

Syntax	Mcu_17_Stm_ComIntEnableType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	0-255	Range of uint8
	Data type for interrupt of STM compare match	
Source	IFX	

MCU driver**17.3.2.20 Mcu_17_Stm_StmCmpIdentifierType****Table 1432 Specification for Mcu_17_Stm_StmCmpIdentifierType**

Syntax	Mcu_17_Stm_StmCmpIdentifierType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	0-255	Range of Uint8
Description	Data type to identify STM comparator type	
Source	IFX	

17.3.2.21 Mcu_17_Stm_StmIdentifierType**Table 1433 Specification for Mcu_17_Stm_StmIdentifierType**

Syntax	Mcu_17_Stm_StmIdentifierType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	0-255	Range of uint8
Description	Data type for STM timers	
Source	IFX	

17.3.2.22 Mcu_17_Stm_TimerConfigType**Table 1434 Specification for Mcu_17_Stm_TimerConfigType**

Syntax	Mcu_17_Stm_TimerConfigType	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	uint32 CompareRegVal	Compare register value
	unsigned_int StmTimerId	STM Timer
	unsigned_int CMPRegId	Compare register ID
	unsigned_int CmconRegVal	Compare match control register value
	unsigned_int reserved	Reserved
Description	Configuration structure for STM configuration	
Source	IFX	

17.3.2.23 Mcu_17_Timer_CallbackFuncPtrType**Table 1435 Specification for Mcu_17_Timer_CallbackFuncPtrType**

Syntax	Mcu_17_Timer_CallbackFuncPtrType
---------------	----------------------------------

MCU driver**Table 1435 Specification for Mcu_17_Timer_CallbackFuncPtrType (continued)**

Type	Pointer to a function of type void Function_Name (const uint8 LogicalChId, const uint8 IsrStatus)
File	Mcu_17_TimerIp.h
Description	Function pointer type for the call back functions, associated with TIM/TOM/ATOM. The input parameter for the callback function is the logical channel ID of the GTM timer channel.
Source	IFX

17.3.2.24 Mcu_RamStateType**Table 1436 Specification for Mcu_RamStateType**

Syntax	Mcu_RamStateType	
Type	Enumeration	
File	Mcu.h	
Range	0 - MCU_RAMSTATE_INVALID	Ram contents got corrupted in last power down.
	1 - MCU_RAMSTATE_VALID	Ram contents are valid after last power down.
Description	Return type for Mcu_GetRamState. MCU_RAMSTATE_INVALID: RAM contents got corrupted MCU_RAMSTATE_VALID: RAM contents are valid	
Source	AUTOSAR	

17.3.2.25 Mcu_CpuIdType**Table 1437 Specification for Mcu_CpuIdType**

Syntax	Mcu_CpuIdType	
Type	Enumeration	
File	Mcu.h	
Range	0 - MCU_CPU0	CPU0 identifier
	1 - MCU_CPU1	CPU1 identifier
	2 - MCU_CPU2	CPU2 identifier
	3 - MCU_CPU3	CPU3 identifier
	4 - MCU_CPU4	CPU4 identifier
	5 - MCU_CPU5	CPU5 identifier
Description	Identification for CPU core id	
Source	IFX	

MCU driver**17.3.2.26 Mcu_CpuModeType****Table 1438 Specification for Mcu_CpuModeType**

Syntax	Mcu_CpuModeType	
Type	Enumeration	
File	Mcu.h	
Range	1 - MCU_CPU_NORMAL_MODE	CPU is in normal state.
	2 - MCU_CPU_IDLE_MODE_REQ	CPU is in idle mode requested state.
	3 - MCU_CPU_IDLE_MODE_ACK	CPU is in idle mode acknowledged state.
	4 - MCU_CPU_SLEEP_MODE_REQ	CPU is in sleep mode requested state
	6 - MCU_CPU_STBY_MODE_REQ	CPU is in standby mode requested state
	255 - MCU_CPU_UNDEFINED_MODE	CPU mode is undefined
Description	Type to specify the current CPU power mode	
Source	IFX	

17.3.2.27 Mcu_TrapRequestType**Table 1439 Specification for Mcu_TrapRequestType**

Syntax	Mcu_TrapRequestType	
Type	Enumeration	
File	Mcu.h	
Range	0 - MCU_TRAP_ESR0	ESR0 trap request
	1 - MCU_TRAP_ESR1	ESR1 trap request
	2 - MCU_TRAP_TRAP2	TRAP bit 2 trap request
	3 - MCU_TRAP_SMU	SMU trap request
	4 - MCU_TRAP_INVALID	Invalid trap source request
Description	Type to specify the TRAP type	
Source	IFX	

17.3.2.28 Mcu_ConfigType**Table 1440 Specification for Mcu_ConfigType**

Syntax	Mcu_ConfigType	
Type	Structure	
File	Mcu.h	
Range	-	The elements of the data structure are specific to the microcontroller.

MCU driver**Table 1440 Specification for Mcu_ConfigType (continued)**

Description	A pointer to such a structure is provided to the MCU initialization routines for configuration.
Source	AUTOSAR

17.3.2.29 Mcu_PlStatusType**Table 1441 Specification for Mcu_PlStatusType**

Syntax	Mcu_PlStatusType	
Type	Enumeration	
File	Mcu.h	
Range	0 - MCU_PLL_LOCKED	The status of both the PLLs is locked.
	1 - MCU_PLL_UNLOCKED	The status of system and/or peripheral PLL is unlocked.
	2 - MCU_PLL_STATUS_UNDEFINED	The status of PLLs is not known.
Description	This is a status value returned by the Mcu_GetPlStatus function of the MCU module. This type provides the status of PLL lock.	
Source	AUTOSAR	

17.3.2.30 Mcu_ClockType**Table 1442 Specification for Mcu_ClockType**

Syntax	Mcu_ClockType	
Type	uint32	
File	Mcu.h	
Range	0 - 255	The range is dependent on the number of different clock settings provided in the configuration structure.
Description	Identification for the clock setting, which is configured in the configuration structure	
Source	AUTOSAR	

17.3.2.31 Mcu_ResetType**Table 1443 Specification for Mcu_ResetType**

Syntax	Mcu_ResetType	
Type	Enumeration	
File	Mcu.h	
Range	0 - MCU_ESR0_RESET	The previous reset type is ESR0 reset

MCU driver**Table 1443 Specification for Mcu_ResetType (continued)**

	1 - MCU_ESR1_RESET	The previous reset type is ESR1 reset
	2 - MCU_SMU_RESET	The previous reset type is SMU reset
	3 - MCU_SW_RESET	The previous reset type is software reset
	4 - MCU_STM0_RESET	The previous reset type is STM 0 reset
	5 - MCU_STM1_RESET	The previous reset type is STM 1 reset
	6 - MCU_STM2_RESET	The previous reset type is STM 2 reset
	7 - MCU_STM3_RESET	The previous reset type is STM 3 reset
	8 - MCU_STM4_RESET	The previous reset type is STM 4 reset
	9 - MCU_STM5_RESET	The previous reset type is STM 5 reset
	10 - MCU_POWER_ON_RESET	The previous reset type is power on reset
	11 - MCU_CB0_RESET	The previous reset type is CB0 reset
	12 - MCU_CB1_RESET	The previous reset type is CB1 reset
	13 - MCU_CB3_RESET	The previous reset type is CB3 reset
	14 - MCU_EVRC_RESET	The previous reset type is EVRC reset
	15 - MCU_EVR33_RESET	The previous reset type is EVR 3.3V reset
	16 - MCU_SUPPLY_WDOG_RESET	The previous reset type is Supply Watchdog reset
	17 - MCU_STBYR_RESET	The previous reset type is Standby Mode reset
	18 - MCU_LBIST_RESET	The previous reset type is reset from LBIST completion
	254 - MCU_RESET_MULTIPLE	There were multiple resets reasons, on which power on reset is one
	255 - MCU_RESET_UNDEFINED	The previous reset type is undefined
Description	This type provides the reset reason types	
Source	AUTOSAR	

17.3.2.32 Mcu_RawResetType**Table 1444 Specification for Mcu_RawResetType**

Syntax	Mcu_RawResetType
Type	uint32
File	Mcu.h
Range	0 - 0xFFFFFFFF
Description	This type specifies the reset reason in raw register format read from a reset status register. For the range, bitfields [31], [17], [15-11], [2] are always zero.

MCU driver**Table 1444 Specification for Mcu_RawResetType (continued)**

Source	AUTOSAR
---------------	---------

17.3.2.33 Mcu_RamSectionType**Table 1445 Specification for Mcu_RamSectionType**

Syntax	Mcu_RamSectionType	
Type	uint32	
File	Mcu.h	
Range	0 - (Number of Ram sections - 1)	The range is dependent on the number of RAM sections provided in the configuration structure.
Description	Identification for RAM section, which is configured in the configuration structure	
Source	AUTOSAR	

17.3.2.34 Mcu_ModeType**Table 1446 Specification for Mcu_ModeType**

Syntax	Mcu_ModeType	
Type	uint8	
File	Mcu.h	
Range	0 - 2	TC3xx supports 3 power modes: Idle, Sleep and Standby modes
Description	Identification for MCU mode, which is configured in the configuration structure	
Source	AUTOSAR	

17.3.2.35 Mcu_17_Gtm_TimChConfigType**Table 1447 Specification for Mcu_17_Gtm_TimChConfigType**

Syntax	Mcu_17_Gtm_TimChConfigType	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	Mcu_17_Gtm_TimerChIdentifierType TimerId	Tim channel user identifier.
	uint32 TimChCtrlReg	Tim channel control registers value.
	uint32 TimChExtendedCtrlReg	Tim channel extended control register value
	uint32 TimChFltRisingEdge	Tim channel filter rising edge parameter.

MCU driver**Table 1447 Specification for Mcu_17_Gtm_TimChConfigType (continued)**

	uint32 TimChFltFallingEdge	Tim channel filter falling edge parameter.
	uint8 TimChIntEnMode	Tim channel interrupt enable and interrupt mode values are encoded in this structure member Bit 0 specifies new value interrupt enable Bit 1 specifies ECNT overflow interrupt enable Bit 2 specifies CNT overflow interrupt enable Bit 3 specifies GPR overflow interrupt enable Bit 4 specifies timeout detection interrupt enable Bit 5 specifies glitch detection interrupt enable Bits [6,7] specifies interrupt mode configured for the channel and are encoded as: 00- Level Mode, 01-Pulse Mode, 10- Pulse Notify Mode, 11- Single Pulse Mode
Description	This structure holds the TIM channel specific parameters details required for the TIM channel initialization.	
Source	IFX	

17.3.2.36 Mcu_17_Gtm_TimerChIdentifierType**Table 1448 Specification for Mcu_17_Gtm_TimerChIdentifierType**

Syntax	Mcu_17_Gtm_TimerChIdentifierType	
Type	uint32	
File	Mcu_17_TimerIp.h	
Range	0 - 0xFFFFFFFF	Range of uint32
Description	Contains the information on the user of the channel. Bit[15:8] - Module number Bit[7:0] - Channel number	
Source	IFX	

17.3.2.37 Mcu_17_Gtm_TimerOutType**Table 1449 Specification for Mcu_17_Gtm_TimerOutType**

Syntax	Mcu_17_Gtm_TimerOutType	
Type	Enumeration	
File	Mcu_17_TimerIp.h	
Range	0 - MCU_GTM_TIMER_TOM	Tom channel
	1 - MCU_GTM_TIMER_ATOM	Atom channel
Description	This type identifies if the GTM output timer is either TOM or ATOM type	

MCU driver**Table 1449 Specification for Mcu_17_Gtm_TimerOutType (continued)**

Source	IFX
---------------	-----

17.3.2.38 Mcu_17_Gtm_TomAtomChConfigType**Table 1450 Specification for Mcu_17_Gtm_TomAtomChConfigType**

Syntax	Mcu_17_Gtm_TomAtomChConfigType	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	Mcu_17_Gtm_TimerOutType TimerType	TOM or ATOM channel ID
	Mcu_17_Gtm_TimerChIdentifierType TimerId	TOM/ATOM channel user identifier
	uint32 TimerChCtrlReg	TOM/ATOM channel control registers value
	uint32 TimerChCN0Reg	TOM/ATOM channel CN0 register value
	uint32 TimerChCM0Reg	TOM/ATOM channel CM0 register value
	uint32 TimerChCM1Reg	TOM/ATOM channel CM1 register value
	uint32 TimerChSR0Reg	TOM/ATOM channel SR0 register value
	uint32 TimerChSR1Reg	TOM/ATOM channel SR1 register value
	uint32 TimerChPortOutConfig	TOM/ATOM to port configuration Bits : [31:16] - Column select value [15:0] - TOUT value
	uint8 TimerChIntEnMode	TOM/ATOM channel interrupt enable and interrupt mode values are encoded in this structure member Bit 0 specifies CCU0 interrupt enable Bit 1 specifies CCU1 interrupt enable Bits [7, 6] specifies interrupt mode configured for the channel and are encoded as: 00- Level Mode, 01-Pulse Mode, 10- Pulse Notify Mode, 11- Single Pulse Mode
Description	This structure holds the TOM/ATOM channel-specific initialization parameters	
Source	IFX	

17.3.2.39 Mcu_17_Gtm_TimerStatusType**Table 1451 Specification for Mcu_17_Gtm_TimerStatusType**

Syntax	Mcu_17_Gtm_TimerStatusType
Type	Enumeration
File	Mcu_17_TimerIp.h

MCU driver**Table 1451 Specification for Mcu_17_Gtm_TimerStatusType (continued)**

Range	0 - MCU_GTM_TIMER_STOPPED	GTM timer channel is stopped
	1 - MCU_GTM_TIMER_RUNNING	GTM timer channel is enabled/running
Description	This type informs the running state of the GTM timer channel	
Source	IFX	

17.3.2.40 Mcu_17_Ccu6_ComparatorType**Table 1452 Specification for Mcu_17_Ccu6_ComparatorType**

Syntax	Mcu_17_Ccu6_ComparatorType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	MCU_CCU6_COMPARATOR_CCU60	CCU60 Comparator
	MCU_CCU6_COMPARATOR_CCU61	CCU61 Comparator
	MCU_CCU6_COMPARATOR_CCU62	CCU62 Comparator
	MCU_CCU6_COMPARATOR_CCU63	CCU63 Comparator
Description	This type identifies the CCU6 comparator used for a kernel	
Source	IFX	

17.3.2.41 Mcu_17_Ccu6_KernelIdentifierType**Table 1453 Specification for Mcu_17_Ccu6_KernelIdentifierType**

Syntax	Mcu_17_Ccu6_KernelIdentifierType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	CCU6_KERNEL_0	CCU6 Kernel 0
	CCU6_KERNEL_1	CCU6 Kernel 1
Description	This type identifies the CCU6 kernel used	
Source	IFX	

17.3.2.42 Mcu_17_Ccu6_TimerChIdentifierType**Table 1454 Specification for Mcu_17_Ccu6_TimerChIdentifierType**

Syntax	Mcu_17_Ccu6_TimerChIdentifierType	
Type	uint32	
File	Mcu_17_TimerIp.h	
Range	0 - 0xFFFFFFF	

MCU driver**Table 1454 Specification for Mcu_17_Ccu6_TimerChIdentifierType (continued)**

Description	This type provides the user information of the CCU6 timer channel Bits[7:0] - Kernel used Bits[15:8] - T12/T13 used Bits[23:16] - Comparator used
Source	IFX

17.3.2.43 Mcu_17_Ccu6_TimerConfigType**Table 1455 Specification for Mcu_17_Ccu6_TimerConfigType**

Syntax	Mcu_17_Ccu6_TimerConfigType	
Type	Structure	
File	Mcu_17_TimerIp.h	
Range	Mcu_17_Ccu6_TimerChIdentifierType TimerId	CCU6 timer channel user identifier
	uint32 TimerCtrlReg0	CCU6 Timer channel control register 0 contents For T12 - [2-0] - Timer T12 Input Clock Select [3] - Timer T12 Prescaler Bit [7] - T12 Operating Mode For T13 - [10-8] - Timer T13 Input Clock Select [11] - Timer T13 Prescaler Bit
	uint32 ModCtrlReg	For T12 - [1-0] - Timer T12 modulation enable for comparator For T13 - [2] - Enable Compare Timer T13 Output
	uint32 PasStateLvlReg	For T12 - [1-0] - Compare Outputs Passive State Level of comparator For T13 - [2] - Passive State Level of Output COUT63
	uint32 TimerCntReg	CCU6 timer channel counter channel contents
	uint32 TimerPeriodReg	CCU6 timer channel period register contents
	uint32 Ccu6ShadowReg	CCU6 timer channel shadow register contents
	uint8 TimerModeSelectReg	CCU6 timer mode select register contents for the input kernel
	uint8 PortInSelReg0	Port Input Select register contents for a kernel
	uint8 IntEnReg	CCU6 timer channel interrupt enable register contents For T12 timer Bits [2] - CCU6 Falling edge Bits [1] - CCU6 Rising edge Bits [0] - T12 Period match For T13 timer Bits [1] - T13 Compare match Bits [0] - T13 Period match

MCU driver**Table 1455 Specification for Mcu_17_Ccu6_TimerConfigType (continued)**

	uint8 IntNodePointerReg	Interrupt Node Pointer register contents. [3:2] - T12/T13 Interrupt node pointer contents [1:0] - CC6x Interrupt node pointer contents
Description	This structure holds the CCU6 timer channel specific initialization parameters.	
Source	IFX	

17.3.2.44 Mcu_17_Ccu6_TimerType**Table 1456 Specification for Mcu_17_Ccu6_TimerType**

Syntax	Mcu_17_Ccu6_TimerType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	MCU_CCU6_TIMER_T12	CCU6 T12 timer
	MCU_CCU6_TIMER_T13	CCU6 T13 timer
Description	This type identifies if the CCU6 timer is T12 or T13	
Source	IFX	

17.3.2.45 Mcu_17_Gpt12_TimerChIdentifierType**Table 1457 Specification for Mcu_17_Gpt12_TimerChIdentifierType**

Syntax	Mcu_17_Gpt12_TimerChIdentifierType	
Type	uint8	
File	Mcu_17_TimerIp.h	
Range	MCU_GPT12_TIMER2	T2 timer of GPT12
	MCU_GPT12_TIMER3	T3 timer of GPT12
	MCU_GPT12_TIMER4	T4 timer of GPT12
	MCU_GPT12_TIMER5	T5 timer of GPT12
	MCU_GPT12_TIMER6	T6 timer of GPT12
Description	This type identifies the GPT12 timer used	
Source	IFX	

17.3.2.46 Mcu_17_Gpt12_TimerConfigType**Table 1458 Specification for Mcu_17_Gpt12_TimerConfigType**

Syntax	Mcu_17_Gpt12_TimerConfigType
Type	Structure

MCU driver
Table 1458 Specification for Mcu_17_Gpt12_TimerConfigType (continued)

File	Mcu_17_TimerIp.h	
Range	Mcu_17_Gpt12_TimerChIdentifierType TimerId	GPT12 user identifier
	uint32 TimerCtrlReg	GPT Timer control register contents
	uint32 TimerCntReg	GPT timer counter register contents
	uint8 PortInSelReg	Port Input Select Register Contents for the input GPT timer Bits[3:2] - Input select for TxEUD Bits[1:0] - Input select for TxIN
Description	This structure holds the GPT12 timer channel-specific initialization parameters	
Source	IFX	

17.3.3 Functions - APIs

This section lists all the APIs of the MCU driver.

17.3.3.1 Mcu_GetRamState

Table 1459 Specification for Mcu_GetRamState API

Syntax	Mcu_RamStateType Mcu_GetRamState (void)	
Service ID	0x0A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Mcu_RamStateType	Enumeration depicting state of RAM after a power down cycle
Description	Mcu_GetRamState returns the RAM state. MCU_RAMSTATE_INVALID: RAM contents got corrupted. MCU_RAMSTATE_VALID: RAM contents are valid.	
Source	AUTOSAR	
Error handling	DET:	

MCU driver**Table 1459 Specification for Mcu_GetRamState API (continued)**

	<p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	None

17.3.3.2 Mcu_Init**Table 1460 Specification for Mcu_Init API**

Syntax	<pre>void Mcu_Init (const Mcu_ConfigType * const ConfigPtr)</pre>	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the MCU driver configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Mcu_Init initializes the MCU driver. Mcu_Init initializes the power modes, reset, trap and timer global configurations registers.</p> <p>If the interface Mcu_ClearColdResetStatus is unavailable, then Mcu_Init clears the reset status bit-fields. It also initializes the module clock for GTM, CCU6, GPT12 and Converter control block. Apart from module clock it also initializes cluster clocks, GTM triggers to ADC and DSADC and block pre-scalers for GPT12</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>MCU_E_PARAM_CONFIG: ConfigPtr passed to Mcu_Init is NULL</p> <p>MCU_E_INIT_FAILED: Error is reported when Mcu_Init() API is called when it is already initialized</p> <p>MCU_E_CORE_MISMATCH: API is called from a core which is not the master core</p> <p>Runtime Errors: None</p>	

MCU driver**Table 1460 Specification for Mcu_Init API (continued)**

	<p>DEM:</p> <p>MCU_E_GTM_CLC_ENABLE_ERR: Inability to turn ON the GTM Clock enable bit</p> <p>MCU_E_CCU6_CLC_ENABLE_ERR: Inability to turn ON the CCU6 kernel Clock enable bit</p> <p>MCU_E_GPT12_CLC_ENABLE_ERR: Inability to turn ON the GPT12 Clock enable bit</p> <p>MCU_E_CONVCTRL_CLC_ENABLE_ERR: Inability to turn ON the CONVCTRL Clock enable bit</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	McuClearColdResetStatusApi
User hints	-

17.3.3.3 Mcu_InitRamSection**Table 1461 Specification for Mcu_InitRamSection API**

Syntax	<pre>Std_ReturnType Mcu_InitRamSection (const Mcu_RamSectionType RamSection)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other RAM sections	
Parameters (in)	RamSection	Selects RAM memory section provided in the configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK – RAM successfully initialized E_NOT_OK – RAM initialization failed
Description	Mcu_InitRamSection initializes the specified RAM section.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>MCU_E_PARAM_RAMSECTION : RamSection parameter does not match the corresponding data in the Mcu_ConfigType object</p> <p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p>	

MCU driver**Table 1461 Specification for Mcu_InitRamSection API (continued)**

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	Protection of the RAM initialization through MPU protection for the RAM address passed in configuration shall be responsibility of the user.

17.3.3.4 Mcu_InitClock**Table 1462 Specification for Mcu_InitClock API**

Syntax	<pre>Std_ReturnType Mcu_InitClock (const Mcu_ClockType ClockSetting)</pre>	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non re-entrant	
Parameters (in)	ClockSetting	Clock setting ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Clock successfully initialized E_NOT_OK: Clock not initialized
Description	Mcu_InitClock initializes the system PLL, peripheral PLL and other MCU specific clock options (peripheral clock selection and dividers).	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>MCU_E_PARAM_CLOCK : ClockSetting parameter does not match the corresponding data in the Mcu_ConfigType object</p> <p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>MCU_E_CORE_MISMATCH: API is called from a core which is not the master core</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>MCU_E_OSC_FAILURE: Inability of the oscillator to deliver correct clock</p> <p>MCU_E_PERIPHERAL_PLL_TIMEOUT_ERR: DEM is raised due to inability of the peripheral PLL K2/K3 dividers and power mode to be updated within the specified time</p> <p>MCU_E_SYSTEM_PLL_TIMEOUT_ERR: DEM is raised due to inability of the system PLL K2 divider and power mode to be updated within the specified time</p>	

MCU driver
Table 1462 Specification for Mcu_InitClock API (continued)

	<p>MCU_E_CCUCON_UPDATE_ERR: Inability to update the CCUCON register</p> <p>Safety Errors:</p> <p>MCU_E_PHSCFG_UPDATE_ERR: Error is raised when phase configuration register of Converter Control update fails</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	McuInitClock
User hints	For low power divider configuration scenario, user shall verify the validity of configured clock values as per inter-relationship between different clocks and configuration generation script will not perform data integrity checks for this configuration scenario.

17.3.3.5 Mcu_DistributePllClock
Table 1463 Specification for Mcu_DistributePllClock API

Syntax	Std_ReturnType Mcu_DistributePllClock (void)	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Clock distribution successful. E_NOT_OK: Clock distribution unsuccessful.
Description	Mcu_DistributePllClock switches the clock source to PLL output.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>MCU_E_PLL_NOT_LOCKED: Either the system or peripheral PLL is not locked</p> <p>MCU_E_CORE_MISMATCH: API is called from a core which is not the master core</p> <p>Runtime Errors: None</p> <p>DEM:</p>	

MCU driver
Table 1463 Specification for `Mcu_DistributePllClock API` (continued)

	<p>MCU_E_SYSTEM_PLL_TIMEOUT_ERR: DEM is raised due to inability of the system PLL K2 divider and power mode to be updated within the specified time</p> <p>MCU_E_CCUCON_UPDATE_ERR: Inability to update the CCUCON register</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	McuNoPll
User hints	<p>Upper layer calls Distribute PLL Clock API, in case MCU module needs a separate request to activate the system PLL and peripheral PLL clock after the system PLL and peripheral PLL is locked.</p> <p>Status of the system and peripheral PLL lock as locked, is checked by the upper layer before calling this API.</p>

17.3.3.6 `Mcu_GetPllStatus`
Table 1464 Specification for `Mcu_GetPllStatus API`

Syntax	Mcu_PllStatusType Mcu_GetPllStatus (void)	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Mcu_PllStatusType	A 32-bit enumerator denoting status of PLL
Description	Mcu_GetPllStatus provides the lock status of system and peripheral PLL.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

MCU driver**Table 1464 Specification for `Mcu_GetPllStatus` API (continued)**

Configuration dependencies	-
User hints	-

17.3.3.7 `Mcu_GetResetReason`**Table 1465 Specification for `Mcu_GetResetReason` API**

Syntax	Mcu_ResetType Mcu_GetResetReason (void)	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Mcu_ResetType	A 32-bit enumerator denoting the cause of reset
Description	Mcu_GetResetReason reads the reset type from the hardware	
Source	AUTOSAR	
Error handling	DET: MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.8 `Mcu_GetResetRawValue`**Table 1466 Specification for `Mcu_GetResetRawValue` API**

Syntax	Mcu_RawResetType Mcu_GetResetRawValue (
---------------	--

MCU driver**Table 1466 Specification for Mcu_GetResetRawValue API (continued)**

	void)	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Mcu_RawResetType	32-bit unsigned integer denoting raw reset value
Description	Mcu_GetResetRawValue reads the reset type from the hardware register	
Source	AUTOSAR	
Error handling	DET: MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.9 Mcu_PerformReset**Table 1467 Specification for Mcu_PerformReset API**

Syntax	void Mcu_PerformReset (void)	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-

MCU driver**Table 1467 Specification for Mcu_PerformReset API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_PerformReset performs a microcontroller reset(software reset).	
Source	AUTOSAR	
Error handling	DET: MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called Runtime Errors: None DEM: None Safety Errors: MCU_E_SW_RESET_FAILED: Error is reported when software reset fails after calling the Mcu_PerformReset API <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	McuPerformResetApi	
User hints	-	

17.3.3.10 Mcu_SetMode**Table 1468 Specification for Mcu_SetMode API**

Syntax	<pre>void Mcu_SetMode (const Mcu_ModeType McuMode)</pre>	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Concurrency Safe for IDLE mode transition requests and non re-entrant for other transitions	
Parameters (in)	McuMode	Set different MCU power modes configured in the configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_SetMode activates the MCU power modes. The 3 power modes supported are Idle, Sleep and StandBy.	

MCU driver**Table 1468 Specification for Mcu_SetMode API (continued)**

	The API is re-entrant and concurrency safe for Idle mode, but for Sleep and Stand By mode, it is not concurrency safe and non - reentrant
Source	AUTOSAR
Error handling	<p>DET:</p> <p>MCU_E_PARAM_MODE : McuMode parameter does not match the corresponding data in the Mcu_ConfigType object</p> <p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>MCU_E_UNAUTHORIZED_REQUESTER: Power down mode entry is requested by an unauthorized CPU</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>MCU_E_PERIPHERAL_PLL_TIMEOUT_ERR: DEM is raised due to inability of the peripheral PLL K2/K3 dividers and power mode to be updated within the specified time</p> <p>MCU_E_SYSTEM_PLL_TIMEOUT_ERR: DEM is raised due to inability of the system PLL K2 divider and power mode to be updated within the specified time</p> <p>MCU_E_PMSWCR_UPDATE_ERR: Inability to update the PMSWCRx register</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	<p>The API Mcu_SetMode assumes that all interrupts are disabled prior to the call of API by the calling instance.</p> <p>For SLEEP or STANDBY modes, user shall start a timer with notification before calling Mcu_SetMode(), such that the timer expires and provides notification, if system has not entered SLEEP or STANDBY mode.</p>

17.3.3.11 Mcu_GetVersionInfo**Table 1469 Specification for Mcu_GetVersionInfo API**

Syntax	<pre>void Mcu_GetVersionInfo (const Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	versioninfo	Pointer to where to store the version information of this module.

MCU driver**Table 1469 Specification for Mcu_GetVersionInfo API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_GetVersionInfo returns the version information of this module.	
Source	AUTOSAR	
Error handling	DET: MCU_E_PARAM_POINTER: Versioninfo pointer passed to Mcu_GetVersionInfo is NULL Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	McuVersionInfoApi	
User hints	-	

17.3.3.12 Mcu_ClearColdResetStatus**Table 1470 Specification for Mcu_ClearColdResetStatus API**

Syntax	<pre>void Mcu_ClearColdResetStatus (void)</pre>	
Service ID	0x50	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non-reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_ClearColdResetStatus is used to clear the cause of the cold reset	
Source	IFX	
Error handling	DET:	

MCU driver**Table 1470 Specification for Mcu_ClearColdResetStatus API (continued)**

	<p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	McuClearColdResetStatusApi
User hints	-

17.3.3.13 Mcu_DeInit**Table 1471 Specification for Mcu_DeInit API**

Syntax	<pre>void Mcu_DeInit (void)</pre>	
Service ID	0x51	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non-reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Mcu_DeInit de-initializes the MCU driver. Mcu_DeInit puts all the resources used by the MCU for reset configuration and power management in the reset state. PLL is not de-initialized by this function.</p> <p>Mcu_DeInit also de-initializes the module clock for GTM, CCU6, GPT12 and Converter control block.</p> <p>Mcu_DeInit also resets all the global variables to uninitialized state</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>MCU_E_CORE_MISMATCH: API is called from a core which is not the master core</p> <p>Runtime Errors: None</p> <p>DEM:</p>	

MCU driver**Table 1471 Specification for Mcu_DeInit API (continued)**

	MCU_E_GTM_CLC_DISABLE_ERR: Inability to turn OFF the GTM clock disable bit MCU_E_GPT12_CLC_DISABLE_ERR: Inability to turn OFF the GPT12 clock disable bit MCU_E_CCU6_CLC_DISABLE_ERR: Inability to turn OFF the CCU6 kernel clock disable bit MCU_E_CONVCTRL_CLC_DISABLE_ERR: Inability to turn OFF the CONVCTRL Clock disable bit Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	MculfxDelinitApi
User hints	-

17.3.3.14 Mcu_GetCpuIdleModeInitiator**Table 1472 Specification for Mcu_GetCpuIdleModeInitiator API**

Syntax	<pre>uint32 Mcu_GetCpuIdleModeInitiator (void)</pre>	
Service ID	0x52	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	CPU Id in case a CPU is setup as initiator of idle mode 0xFFFFFFFFFU in case each CPU is responsible for its power state transition 7U in case idle mode is not configured
Description	The CPU responsible for initiating the idle mode entry of other CPUs is returned by the interface.	
Source	IFX	
Error handling	DET: MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called Runtime Errors: None DEM: None Safety Errors: None	

MCU driver**Table 1472 Specification for Mcu_GetCpuIdleModeInitiator API (continued)**

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	MculfxLpmApi
User hints	-

17.3.3.15 Mcu_GetCpuState**Table 1473 Specification for Mcu_GetCpuState API**

Syntax	Mcu_CpuModeType Mcu_GetCpuState (const Mcu_CpuIdType CpuId)	
Service ID	0x53	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Cpuld	Cpuld CPU Identifier
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Mcu_CpuModeType	Cpu state for the input Cpu ID
Description	A valid power state is returned by the interface for valid CPUs. MCU_CPU_UNDEFINED_MODE is returned as a power state for invalid CPUs OR when CPU state is indicating reserved states.	
Source	IFX	
Error handling	DET: MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called MCU_E_PARAM_CPUID: Input argument for CPU Id passed with an invalid core index Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	MculfxLpmApi	
User hints	-	

MCU driver**17.3.3.16 Mcu_GetWakeupCause****Table 1474 Specification for Mcu_GetWakeupCause API**

Syntax	uint32 Mcu_GetWakeupCause (void)	
Service ID	0x54	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Standby mode wakeup cause
Description	A bit-mask indicating events responsible for wakeup from the standby mode is returned back to the caller. In case the API is called prior to MCU initialization, it returns a value of 0xFFFFFFFF.	
Source	IFX	
Error handling	DET: MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	McufxLpmApi	
User hints	-	

17.3.3.17 Mcu_ClearWakeupCause**Table 1475 Specification for Mcu_ClearWakeupCause API**

Syntax	void Mcu_ClearWakeupCause (const uint32 WakeupCause)
Service ID	0x55
Sync/Async	Synchronous

MCU driver**Table 1475 Specification for Mcu_ClearWakeupCause API (continued)**

ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	WakeupCause	Wakeup causes to be cleared by this API
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_ClearWakeupCause clears the reason for wakeup from the standby mode. The input parameter passed is masked accordingly and written in the register to clear the standby wake up cause.	
Source	IFX	
Error handling	DET: MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	McuLfxLpmApi	
User hints	User should ensure that the wake-up cause(s) which triggered wakeup during STANDBY, shall be cleared explicitly before next STANDBY entry.	

17.3.3.18 Mcu_GetTrapCause**Table 1476 Specification for Mcu_GetTrapCause API**

Syntax	<pre>uint32 Mcu_GetTrapCause (void)</pre>	
Service ID	0x56	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-

MCU driver**Table 1476 Specification for Mcu_GetTrapCause API (continued)**

Parameters (in - out)	-	-
Return	uint32	Returns the reason for the occurrence of the trap
Description	A bit-mask indicating events responsible for the current trap/last trap serviced is returned back to the caller. In case the API is called prior to MCU initialization, it returns a value of 0xFFFFFFFF.	
Source	IFX	
Error handling	DET: MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	McufxTrapApi	
User hints	-	

17.3.3.19 Mcu_SetTrapRequest**Table 1477 Specification for Mcu_SetTrapRequest API**

Syntax	void Mcu_SetTrapRequest (const Mcu_TrapRequestType TrapRequestId)	
Service ID	0x57	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other Trap Ids	
Parameters (in)	TrapRequestId	Type of the trap request to be set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_SetTrapRequest is used to manually assert the specified trap request.	
Source	IFX	
Error handling	DET: MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called	

MCU driver**Table 1477 Specification for Mcu_SetTrapRequest API (continued)**

	<p>MCU_E_PARAM_TRAPID: Trap-related read or write with an invalid trap source id</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	MculfxTrapApi
User hints	-

17.3.3.20 Mcu_ClearTrapRequest**Table 1478 Specification for Mcu_ClearTrapRequest API**

Syntax	<pre>void Mcu_ClearTrapRequest (const Mcu_TrapRequestType TrapRequestId)</pre>	
Service ID	0x58	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other Trap IDs	
Parameters (in)	TrapRequestId	Type of the trap request to be cleared
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_ClearTrapRequest is used to clear the trap status currently set.	
Source	IFX	
Error handling	<p>DET:</p> <p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>MCU_E_PARAM_TRAPID: Trap-related read or write with an invalid trap source id</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	MculfxTrapApi	

MCU driver**Table 1478 Specification for Mcu_ClearTrapRequest API (continued)**

User hints	-
------------	---

17.3.3.21 Mcu_UpdateCpuCcuconReg**Table 1479 Specification for Mcu_UpdateCpuCcuconReg API**

Syntax	<pre>void Mcu_UpdateCpuCcuconReg (const Mcu_CpuIdType CpuId, const uint8 DivVal, const uint8 Delay)</pre>	
Service ID	0x59	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other cores	
Parameters (in)	Cpuld DivVal Delay	Cpuld of core-x to update its CCUCONx divider value New divider value for update Delay in microseconds after CCUCONx register update
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_UpdateCpuCcuconReg is used to update the CCUCONx divider value of CPUx to the user provided value.	
Source	IFX	
Error handling	<p>DET:</p> <p>MCU_E_UNINIT: Error is reported if the API is called before Mcu_Init is called</p> <p>MCU_E_PARAM_CPUID: Input argument for CPU Id passed with an invalid core index</p> <p>MCU_E_PARAM_DIV_VAL: CpuCcucon divider update requested with value higher than maximum possible divider value</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	McufxCpuCcuconApi	
User hints	-	

MCU driver**17.3.3.22 Mcu_InitCheck****Table 1480 Specification for Mcu_InitCheck API**

Syntax	<pre>Std_ReturnType Mcu_InitCheck (const Mcu_ConfigType * const ConfigPtr)</pre>	
Service ID	0x5A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non-reentrant	
Parameters (in)	ConfigPtr	Pointer to MCU driver configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Initcheck is successful E_NOT_OK: Initcheck failed
Description	Mcu_InitCheck verifies the initialization done by the MCU driver in Mcu_Init(), Mcu_InitClock() and Mcu_DistributePllClock() APIs.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	McuInitCheckApi	
User hints	None	

17.3.3.23 Mcu_17_Gtm_AtomChannelInit**Table 1481 Specification for Mcu_17_Gtm_AtomChannelInit API**

Syntax	<pre>void Mcu_17_Gtm_AtomChannelInit (const Mcu_17_Gtm_TomAtomChConfigType * const ConfigPtr)</pre>
Service ID	0x64
Sync/Async	Synchronous
ASIL Level	B

MCU driver**Table 1481 Specification for Mcu_17_Gtm_AtomChannelInit API (continued)**

Re-entrancy	Reentrant for other channels	
Parameters (in)	ConfigPtr	Pointer to the configuration data of an ATOM channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChannelInit configures an instance of an ATOM channel. User of an ATOM channel invokes this interface at the time of initialization.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.24 Mcu_17_Gtm_AtomChInitCheck**Table 1482 Specification for Mcu_17_Gtm_AtomChInitCheck API**

Syntax	<pre>Std_ReturnType Mcu_17_Gtm_AtomChInitCheck (const Mcu_17_Gtm_TomAtomChConfigType * const ConfigPtr)</pre>	
Service ID	0x7B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	ConfigPtr	Configuration of the ATOM channel that is to be verified
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: ATOM initcheck is successful E_NOT_OK: ATOM initcheck failed

MCU driver**Table 1482 Specification for Mcu_17_Gtm_AtomChInitCheck API (continued)**

Description	Mcu_17_Gtm_AtomChInitCheck verifies the initialization done by the MCU driver in the Mcu_17_Gtm_AtomChannelInit() API for the input ATOM channel
Source	IFX
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

17.3.3.25 Mcu_17_Gtm_AtomChannelDeInit**Table 1483 Specification for Mcu_17_Gtm_AtomChannelDeInit API**

Syntax	<pre>void Mcu_17_Gtm_AtomChannelDeInit (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x66	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module	ATOM module number
	Channel	ATOM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChannelDeInit resets an ATOM channel to reset values.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

MCU driver**Table 1483 Specification for Mcu_17_Gtm_AtomChannelDeInit API (continued)**

Configuration dependencies	-
User hints	-

17.3.3.26 Mcu_17_Gtm_AtomChannelEnable**Table 1484 Specification for Mcu_17_Gtm_AtomChannelEnable API**

Syntax	<pre>void Mcu_17_Gtm_AtomChannelEnable (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerOutputEnableType TimerOutputEn)</pre>	
Service ID	0x6A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel TimerOutputEn	ATOM module number ATOM channel number ATOM output enable configuration
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChannelEnable starts the specified timer. Applications that use the timer slice for PWM functionality must enable the output (TimerOutPutEn = 1). Applications that use the timer for counting (timebase) purpose can disable the output.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

MCU driver**17.3.3.27 Mcu_17_Gtm_AtomChannelDisable****Table 1485 Specification for Mcu_17_Gtm_AtomChannelDisable API**

Syntax	<pre>void Mcu_17_Gtm_AtomChannelDisable (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x6B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel	ATOM module number ATOM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChannelDisable stops the specified timer. The timer output is unconditionally disabled.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.28 Mcu_17_Gtm_IsAtomChannelEnabled**Table 1486 Specification for Mcu_17_Gtm_IsAtomChannelEnabled API**

Syntax	<pre>Mcu_17_Gtm_TimerStatusType Mcu_17_Gtm_IsAtomChannelEnabled (const uint8 Module, const uint8 Channel)</pre>
Service ID	0x6F
Sync/Async	Synchronous

MCU driver**Table 1486 Specification for Mcu_17_Gtm_IsAtomChannelEnabled API (continued)**

ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Module Channel	ATOM module number ATOM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Mcu_17_Gtm_TimerStatus Type	MCU_GTM_TIMER_RUNNING : Timer is running. MCU_GTM_TIMER_STOPPED : Timer is stopped
Description	Mcu_17_Gtm_IsAtomChannelEnabled confirms whether or not the specified timer slice is running.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.29 Mcu_17_Gtm_AtomChannelShadowTransfer**Table 1487 Specification for Mcu_17_Gtm_AtomChannelShadowTransfer API**

Syntax	<pre>void Mcu_17_Gtm_AtomChannelShadowTransfer (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x65	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Module Channel	ATOM module number ATOM channel number
Parameters (out)	-	-

MCU driver**Table 1487 Specification for Mcu_17_Gtm_AtomChannelShadowTransfer API (continued)**

Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChannelShadowTransfer is used to initiate a copy of values in shadow registers (compare, period and clock source) of the specified ATOM channel of a specified ATOM module to its main timer registers	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.30 Mcu_17_Gtm_AtomChUpdateEnDis**Table 1488 Specification for Mcu_17_Gtm_AtomChUpdateEnDis API**

Syntax	<pre>void Mcu_17_Gtm_AtomChUpdateEnDis (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerUpdateEnableType UpEnVal)</pre>	
Service ID	0x7C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel UpEnVal	Specifies the module used Specifies the GTM channel used Specifies if GTM timer update is enabled or disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChUpdateEnDis is used to update the value of the ATOM Channel Update Enable/ Disable Control register	
Source	IFX	

MCU driver**Table 1488 Specification for Mcu_17_Gtm_AtomChUpdateEnDis API (continued)**

Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

17.3.3.31 Mcu_17_Gtm_AtomChEndisStatUpdate**Table 1489 Specification for Mcu_17_Gtm_AtomChEndisStatUpdate API**

Syntax	<pre>void Mcu_17_Gtm_AtomChEndisStatUpdate (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerEnableType TimerEnDis)</pre>	
Service ID	0x80	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel TimerEnDis	Specifies the module used Specifies the GTM channel used Specifies whether timer is enabled or disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChEndisStatUpdate is used by applications to enable or disable the ATOM channel directly	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	

MCU driver**Table 1489 Specification for Mcu_17_Gtm_AtomChEndisStatUpdate API (continued)**

User hints	-
-------------------	---

17.3.3.32 Mcu_17_Gtm_AtomChEndisCtrlUpdate**Table 1490 Specification for Mcu_17_Gtm_AtomChEndisCtrlUpdate API**

Syntax	<pre>void Mcu_17_Gtm_AtomChEndisCtrlUpdate (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerEnTriggerType TimerEnDis)</pre>	
Service ID	0x7F	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel TimerEnDis	Specifies the module being used Specifies the GTM channel being used Enable/disable the ATOM channel on a trigger
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChEndisCtrlUpdate is used by applications to enable or disable the ATOM channel on a trigger	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

MCU driver**17.3.3.33 Mcu_17_Gtm_AtomChOutEnStatUpdate****Table 1491 Specification for Mcu_17_Gtm_AtomChOutEnStatUpdate API**

Syntax	<pre>void Mcu_17_Gtm_AtomChOutEnStatUpdate (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerOutputEnableType TimerOutputEnDis)</pre>	
Service ID	0x7E	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel TimerOutputEnDis	Specifies the module used Specifies the GTM channel used Specifies whether GTM timer output is enabled or disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChOutEnStatUpdate is used by applications to enable or disable the output of an ATOM channel directly	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.34 Mcu_17_Gtm_AtomChOutEnCtrlUpdate**Table 1492 Specification for Mcu_17_Gtm_AtomChOutEnCtrlUpdate API**

Syntax	<pre>void Mcu_17_Gtm_AtomChOutEnCtrlUpdate (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerOutputEnTriggerType TimerOutputEnDis)</pre>
---------------	--

MCU driver**Table 1492 Specification for Mcu_17_Gtm_AtomChOutEnCtrlUpdate API (continued)**

Service ID	0x7D	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel TimerOutputEnDis	Specifies the module being used Specifies the GTM channel being used Enable/disable the ATOM channel output on a trigger
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChOutEnCtrlUpdate is used by applications to enable or disable the output of an ATOM channel on a trigger	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.35 Mcu_17_Gtm_AtomTriggerRequest**Table 1493 Specification for Mcu_17_Gtm_AtomTriggerRequest API**

Syntax	<pre>void Mcu_17_Gtm_AtomTriggerRequest (const uint8 Module, const uint16 TriggerChannels)</pre>	
Service ID	0x7A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other AGC	
Parameters (in)	Module TriggerChannels	ATOM Module ID Mask for the channels to be triggered

MCU driver**Table 1493 Specification for Mcu_17_Gtm_AtomTriggerRequest API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Interface is used by applications to enable or disable the ATOM channel on a trigger	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.36 Mcu_17_Gtm_TomChannelInit**Table 1494 Specification for Mcu_17_Gtm_TomChannelInit API**

Syntax	<pre>void Mcu_17_Gtm_TomChannelInit (const Mcu_17_Gtm_TomAtomChConfigType * const ConfigPtr)</pre>	
Service ID	0x60	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	ConfigPtr	Pointer to the configuration data of a TOM channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomChannelInit configures an instance of the TOM channel. User of a TOM channel invokes this interface at the time of initialization.	
Source	IFX	
Error handling	DET: None Runtime Errors: None	

MCU driver**Table 1494 Specification for Mcu_17_Gtm_TomChannelInit API (continued)**

	DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

17.3.3.37 Mcu_17_Gtm_TomChInitCheck**Table 1495 Specification for Mcu_17_Gtm_TomChInitCheck API**

Syntax	Std_ReturnType Mcu_17_Gtm_TomChInitCheck (const Mcu_17_Gtm_TomAtomChConfigType * const ConfigPtr)	
Service ID	0x74	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	ConfigPtr	Configuration of the TOM channel that is to be verified
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: TOM initcheck is successful E_NOT_OK: TOM initcheck failed
Description	Mcu_17_Gtm_TomChInitCheck verifies the initialization done by the MCU driver in the Mcu_17_Gtm_TomChannelInit() API for the input TOM channel	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

MCU driver**17.3.3.38 Mcu_17_Gtm_TomChannelDeInit****Table 1496 Specification for Mcu_17_Gtm_TomChannelDeInit API**

Syntax	<pre>void Mcu_17_Gtm_TomChannelDeInit (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x63	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel	TOM module number TOM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomChannelDeInit resets a TOM channel to reset values.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.39 Mcu_17_Gtm_TomChannelEnable**Table 1497 Specification for Mcu_17_Gtm_TomChannelEnable API**

Syntax	<pre>void Mcu_17_Gtm_TomChannelEnable (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerOutputEnableType TimerOutputEn)</pre>
Service ID	0x63
Sync/Async	Synchronous

MCU driver**Table 1497 Specification for Mcu_17_Gtm_TomChannelEnable API (continued)**

ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel TimerOutputEn	TOM module number TOM channel number TOM output enable configuration
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomChannelEnable starts the specified timer. Applications which use the timer slice for the PWM functionality must enable the output (TimerOutPutEn = 1). Applications which use the timer for counting (timebase) purpose can disable the output.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.40 Mcu_17_Gtm_TomChannelDisable**Table 1498 Specification for Mcu_17_Gtm_TomChannelDisable API**

Syntax	<pre>void Mcu_17_Gtm_TomChannelDisable (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x69	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel	TOM module number TOM channel number
Parameters (out)	-	-

MCU driver**Table 1498 Specification for Mcu_17_Gtm_TomChannelDisable API (continued)**

Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomChannelDisable stops the specified timer. The timer output is unconditionally disabled.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.41 Mcu_17_Gtm_IsTomChannelEnabled**Table 1499 Specification for Mcu_17_Gtm_IsTomChannelEnabled API**

Syntax	Mcu_17_Gtm_TimerStatusType Mcu_17_Gtm_IsTomChannelEnabled (const uint8 Module, const uint8 Channel)	
Service ID	0x68	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Module	TOM module number
	Channel	TOM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Mcu_17_Gtm_TimerStatus Type	MCU_GTM_TIMER_RUNNING : Timer is running. MCU_GTM_TIMER_STOPPED : Timer is stopped
Description	Mcu_17_Gtm_IsTomChannelEnabled confirms whether or not the specified timer slice is running.	
Source	IFX	
Error handling	DET: None	

MCU driver**Table 1499 Specification for Mcu_17_Gtm_IsTomChannelEnabled API (continued)**

	<p>Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

17.3.3.42 Mcu_17_Gtm_TomChannelShadowTransfer**Table 1500 Specification for Mcu_17_Gtm_TomChannelShadowTransfer API**

Syntax	<pre>void Mcu_17_Gtm_TomChannelShadowTransfer (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x61	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Module Channel	TOM module number TOM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomChannelShadowTransfer is used to initiate a copy of values in the shadow registers (compare, period and clock Source) of the specified TOM channel of a specified TOM module to the main timer registers	
Source	IFX	
Error handling	<p>DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

MCU driver**17.3.3.43 Mcu_17_Gtm_TomChUpdateEnDis****Table 1501 Specification for Mcu_17_Gtm_TomChUpdateEnDis API**

Syntax	<pre>void Mcu_17_Gtm_TomChUpdateEnDis (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerUpdateEnableType UpEnVal)</pre>	
Service ID	0x75	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel UpEnVal	Specifies the module being used Specifies the GTM channel being used Specifies if the GTM timer update is enabled or disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomChUpdateEnDis is used to update the value of the TOM Channel update enable/disable control register	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.44 Mcu_17_Gtm_TomChOutEnStatUpdate**Table 1502 Specification for Mcu_17_Gtm_TomChOutEnStatUpdate API**

Syntax	<pre>void Mcu_17_Gtm_TomChOutEnStatUpdate (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerOutputEnableType TimerOutputEnDis)</pre>
---------------	--

MCU driver**Table 1502 Specification for Mcu_17_Gtm_TomChOutEnStatUpdate API (continued)**

Service ID	0x77	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other Channels	
Parameters (in)	Module Channel TimerOutputEnDis	Specifies the module being used Specifies the GTM channel being used Specifies if the timer output is enabled or disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomChOutEnStatUpdate is used to update the value of the TOM Channel Output Enable/ Disable Status register	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.45 Mcu_17_Gtm_TomChOutEnCtrlUpdate**Table 1503 Specification for Mcu_17_Gtm_TomChOutEnCtrlUpdate API**

Syntax	<pre>void Mcu_17_Gtm_TomChOutEnCtrlUpdate (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerOutputEnTriggerType TimerOutputEnDis)</pre>	
Service ID	0x76	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel	Specifies the module being used Specifies the GTM channel being used

MCU driver**Table 1503 Specification for Mcu_17_Gtm_TomChOutEnCtrlUpdate API (continued)**

	TimerOutputEnDis	Enable/disable the TOM channel output on a trigger
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomChOutEnCtrlUpdate is used to update the value of the TOM Channel Output Enable/ Disable Control register	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.46 Mcu_17_Gtm_TomChEndisStatUpdate**Table 1504 Specification for Mcu_17_Gtm_TomChEndisStatUpdate API**

Syntax	<pre>void Mcu_17_Gtm_TomChEndisStatUpdate (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerEnableType TimerEnDis)</pre>	
Service ID	0x79	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel TimerEnDis	Specifies the module being used Specifies the GTM channel being used Specifies if the timer is enabled or disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

MCU driver**Table 1504 Specification for Mcu_17_Gtm_TomChEndisStatUpdate API (continued)**

Description	Mcu_17_Gtm_TomChEndisStatUpdate is used to update the value of the TOM channel enable/disable status register
Source	IFX
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

17.3.3.47 Mcu_17_Gtm_TomChEndisCtrlUpdate**Table 1505 Specification for Mcu_17_Gtm_TomChEndisCtrlUpdate API**

Syntax	<pre>void Mcu_17_Gtm_TomChEndisCtrlUpdate (const uint8 Module, const uint8 Channel, const Mcu_17_Gtm_TimerEnTriggerType TimerEnDis)</pre>	
Service ID	0x78	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel TimerEnDis	Specifies the module being used TOM channel used Enable/disable the TOM channel on a trigger
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomChEndisCtrlUpdate is used to update the value of the ATOM Channel Enable/ Disable Control register	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None	

MCU driver**Table 1505 Specification for Mcu_17_Gtm_TomChEndisCtrlUpdate API (continued)**

	Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

17.3.3.48 Mcu_17_Gtm_TomTriggerRequest**Table 1506 Specification for Mcu_17_Gtm_TomTriggerRequest API**

Syntax	<pre>void Mcu_17_Gtm_TomTriggerRequest (const uint8 Module, const uint8 TomTgcIndex, const uint16 TriggerChannels)</pre>	
Service ID	0x73	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other TGC	
Parameters (in)	Module TomTgcIndex TriggerChannels	TOM Module ID TOM TGC ID Channels to be triggered
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TomTriggerRequest is used by applications to enable or disable multiple TOM channels	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

MCU driver**17.3.3.49 Mcu_17_Gtm_TimChannelInit****Table 1507 Specification for Mcu_17_Gtm_TimChannelInit API**

Syntax	<pre>void Mcu_17_Gtm_TimChannelInit (const Mcu_17_Gtm_TimChConfigType * const ConfigPtr)</pre>	
Service ID	0x62	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	ConfigPtr	Pointer to the configuration data of a TIM channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TimChannelInit configures an instance of a TIM channel. Consumer of a TIM channel invokes this interface at the time of initialization.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.50 Mcu_17_Gtm_TimChInitCheck**Table 1508 Specification for Mcu_17_Gtm_TimChInitCheck API**

Syntax	<pre>Std_ReturnType Mcu_17_Gtm_TimChInitCheck (const Mcu_17_Gtm_TimChConfigType * const ConfigPtr)</pre>	
Service ID	0x81	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	

MCU driver**Table 1508 Specification for Mcu_17_Gtm_TimChInitCheck API (continued)**

Parameters (in)	ConfigPtr	Configuration of the TIM channel that is to be verified
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: TIM initcheck is successful E_NOT_OK: TIM initcheck failed
Description	Mcu_17_Gtm_TimChInitCheck verifies the initialization done by the MCU driver in the Mcu_17_Gtm_TimChannelInit API for the input TIM channel	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

17.3.3.51 Mcu_17_Gtm_TimChannelDeInit**Table 1509 Specification for Mcu_17_Gtm_TimChannelDeInit API**

Syntax	<pre>void Mcu_17_Gtm_TimChannelDeInit (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x67	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module	TIM module number
	Channel	TIM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

MCU driver**Table 1509 Specification for Mcu_17_Gtm_TimChannelDeInit API (continued)**

Description	Mcu_17_Gtm_TimChannelDeInit resets a TIM channel to default values
Source	IFX
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

17.3.3.52 Mcu_17_Gtm_TimChannelEnable**Table 1510 Specification for Mcu_17_Gtm_TimChannelEnable API**

Syntax	<pre>void Mcu_17_Gtm_TimChannelEnable (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x6C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module	TIM module number
	Channel	TIM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TimChannelEnable starts the specified timer	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	

MCU driver**Table 1510 Specification for Mcu_17_Gtm_TimChannelEnable API (continued)**

User hints	-
-------------------	---

17.3.3.53 Mcu_17_Gtm_TimChannelDisable**Table 1511 Specification for Mcu_17_Gtm_TimChannelDisable API**

Syntax	<pre>void Mcu_17_Gtm_TimChannelDisable (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x6D	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Module Channel	TIM module number TIM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TimChannelDisable stops the specified timer	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.54 Mcu_17_Gtm_IsTimChannelEnabled**Table 1512 Specification for Mcu_17_Gtm_IsTimChannelEnabled API**

Syntax	<pre>Mcu_17_Gtm_TimerStatusType Mcu_17_Gtm_IsTimChannelEnabled (const uint8 Module,</pre>
---------------	--

MCU driver**Table 1512 Specification for Mcu_17_Gtm_IsTimChannelEnabled API (continued)**

	const uint8 Channel)	
Service ID	0x70	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Module Channel	TIM module number TIM channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Mcu_17_Gtm_TimerStatus Type	MCU_GTM_TIMER_RUNNING: Timer is running MCU_GTM_TIMER_STOPPED: Timer is stopped
Description	Mcu_17_Gtm_IsTimChannelEnabled confirms whether or not the specified timer slice is running	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.55 Mcu_17_Gtm_ConnectPortPinToTim**Table 1513 Specification for Mcu_17_Gtm_ConnectPortPinToTim API**

Syntax	void Mcu_17_Gtm_ConnectPortPinToTim (const uint8 Module, const uint8 Channel, const uint8 TimerChselValue)
Service ID	0x72
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for other TIM modules

MCU driver**Table 1513 Specification for Mcu_17_Gtm_ConnectPortPinToTim API (continued)**

Parameters (in)	Module Channel TimerChselValue	TIM module number TIM channel number Timer input select register CHxSEL bit-field value
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_ConnectPortPinToTim is used to connect a port pin to an input GTM channel (TIM)	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.56 Mcu_17_Gtm_ConnectTimerOutToPortPin**Table 1514 Specification for Mcu_17_Gtm_ConnectTimerOutToPortPin API**

Syntax	<pre>void Mcu_17_Gtm_ConnectTimerOutToPortPin (const uint16 Tout_IndexNumber, const Mcu_17_Gtm_MappedPortTimerOutType TimerOutColumnSelect)</pre>	
Service ID	0x71	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different port pins	
Parameters (in)	Tout_IndexNumber TimerOutColumnSelect	Timer output index number Represents mapped column for the table GTM output to Port Connection in the hardware manual
Parameters (out)	-	-
Parameters (in - out)	-	-

MCU driver**Table 1514 Specification for Mcu_17_Gtm_ConnectTimerOutToPortPin API (continued)**

Return	void	-
Description	Mcu_17_Gtm_ConnectTimerOutToPortPin is used to connect an output GTM channel(TOM/ATOM) to a port pin. The selected port pin is based on Tout_IndexNumber value and channel is based on TimerOutColumnSelect parameter.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.57 Mcu_17_Ccu6_TimerInit**Table 1515 Specification for Mcu_17_Ccu6_TimerInit API**

Syntax	<pre>void Mcu_17_Ccu6_TimerInit (const Mcu_17_Ccu6_TimerConfigType * const ConfigPtr)</pre>	
Service ID	0x82	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	ConfigPtr	Ccu6 timer channel initialization contents
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Ccu6_TimerInit configures an instance of a CCU6 timer channel. User of the CCU6 channel invokes this interface at the time of channel's initialization.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None	

MCU driver**Table 1515 Specification for Mcu_17_Ccu6_TimerInit API (continued)**

	<i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

17.3.3.58 Mcu_17_Ccu6_TimerInitCheck**Table 1516 Specification for Mcu_17_Ccu6_TimerInitCheck API**

Syntax	Std_ReturnType Mcu_17_Ccu6_TimerInitCheck (const Mcu_17_Ccu6_TimerConfigType * const ConfigPtr)	
Service ID	0x89	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Configuration of the CCU6 comparator channel that is to be verified
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: CCU6 initcheck is successful E_NOT_OK: CCU6 initcheck failed
Description	Mcu_17_Ccu6_TimerInitCheck verifies the initialization done by the MCU driver in the Mcu_17_Ccu6_TimerInit() API for the input CCU6 comparator	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	None	

MCU driver**17.3.3.59 Mcu_17_Ccu6_TimerDeInit****Table 1517 Specification for Mcu_17_Ccu6_TimerDeInit API**

Syntax	<pre>void Mcu_17_Ccu6_TimerDeInit (const Mcu_17_Ccu6_TimerChIdentifierType TimerId)</pre>	
Service ID	0x83	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	TimerId	CCU6 timer to be de-initialized
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Ccu6_TimerDeInit de-initializes the CCU6 timer channel to default values	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.60 Mcu_17_Ccu6_TimerStart**Table 1518 Specification for Mcu_17_Ccu6_TimerStart API**

Syntax	<pre>void Mcu_17_Ccu6_TimerStart (const Mcu_17_Ccu6_TimerChIdentifierType TimerId)</pre>	
Service ID	0x84	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	

MCU driver**Table 1518 Specification for Mcu_17_Ccu6_TimerStart API (continued)**

Parameters (in)	TimerId	CCU6 timer channel to be enabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Ccu6_TimerStart starts the specified CCU6 timer.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.61 Mcu_17_Ccu6_TimerStop**Table 1519 Specification for Mcu_17_Ccu6_TimerStop API**

Syntax	<pre>void Mcu_17_Ccu6_TimerStop (const Mcu_17_Ccu6_TimerChIdentifierType TimerId)</pre>	
Service ID	0x85	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	TimerId	CCU6 timer channel to be disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Ccu6_TimerStop stops the specified CCU6 timer.	
Source	IFX	
Error handling	DET: None	

MCU driver**Table 1519 Specification for Mcu_17_Ccu6_TimerStop API (continued)**

	<p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

17.3.3.62 Mcu_17_Ccu6_TimerIntEnDis**Table 1520 Specification for Mcu_17_Ccu6_TimerIntEnDis API**

Syntax	<pre>void Mcu_17_Ccu6_TimerIntEnDis (const Mcu_17_Ccu6_TimerChIntType Ccu6IntConfig)</pre>	
Service ID	0x87	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	Ccu6IntConfig	CCU6 timer channel interrupt configuration
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Ccu6_TimerIntEnDis enables/disables the specified interrupt of the CCU6 timer	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

MCU driver**17.3.3.63 Mcu_17_Ccu6_TimerShadowTransfer****Table 1521 Specification for Mcu_17_Ccu6_TimerShadowTransfer API**

Syntax	<pre>void Mcu_17_Ccu6_TimerShadowTransfer (const Mcu_17_Ccu6_TimerChIdentifierType TimerId)</pre>	
Service ID	0x86	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other CCU6 timers	
Parameters (in)	TimerId	CCU6 timer channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Ccu6_TimerShadowTransfer enables the shadow transfer for the specified CCU6 timer channel, that is, to copy contents from the shadow register to the main register	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.64 Mcu_17_Gpt12_TimerInit**Table 1522 Specification for Mcu_17_Gpt12_TimerInit API**

Syntax	<pre>void Mcu_17_Gpt12_TimerInit (const Mcu_17_Gpt12_TimerConfigType * const ConfigPtr)</pre>	
Service ID	0x8A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	

MCU driver**Table 1522 Specification for Mcu_17_Gpt12_TimerInit API (continued)**

Parameters (in)	ConfigPtr	GPT12 timer channel initialization contents
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gpt12_TimerInit configures an instance of a GPT12 timer channel. User of a GPT12 channel invokes this interface at the time of former's initialization.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.65 Mcu_17_Gpt12_TimerInitCheck**Table 1523 Specification for Mcu_17_Gpt12_TimerInitCheck API**

Syntax	<pre>Std_ReturnType Mcu_17_Gpt12_TimerInitCheck (const Mcu_17_Gpt12_TimerConfigType * const ConfigPtr)</pre>	
Service ID	0x8B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	ConfigPtr	Configuration of the GPT12 timer channel that is to be verified
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: GPT12 initcheck is successful E_NOT_OK: GPT12 initcheck failed

MCU driver**Table 1523 Specification for Mcu_17_Gpt12_TimerInitCheck API (continued)**

Description	Mcu_17_Gpt12_TimerInitCheck verifies the initialization done by the MCU driver in the Mcu_17_Gpt12_TimerInit() API for the input GPT timer channel
Source	IFX
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	None

17.3.3.66 Mcu_17_Gpt12_TimerDeInit**Table 1524 Specification for Mcu_17_Gpt12_TimerDeInit API**

Syntax	<pre>void Mcu_17_Gpt12_TimerDeInit (const Mcu_17_Gpt12_TimerChIdentifierType TimerId)</pre>	
Service ID	0x8C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	TimerId	GPT12 timer to be de-initialized
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gpt12_TimerDeInit de-initializes the input GPT12 timer channel to default values	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	

MCU driver**Table 1524 Specification for Mcu_17_Gpt12_TimerDeInit API (continued)**

User hints	-
-------------------	---

17.3.3.67 Mcu_17_Gpt12_TimerStart**Table 1525 Specification for Mcu_17_Gpt12_TimerStart API**

Syntax	void Mcu_17_Gpt12_TimerStart (const Mcu_17_Gpt12_TimerChIdentifierType TimerId)	
Service ID	0x8D	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	TimerId	GPT12 timer channel to be enabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gpt12_TimerStart starts the specified GPT12 timer	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.68 Mcu_17_Gpt12_TimerStop**Table 1526 Specification for Mcu_17_Gpt12_TimerStop API**

Syntax	void Mcu_17_Gpt12_TimerStop (const Mcu_17_Gpt12_TimerChIdentifierType TimerId)	
Service ID	0x8E	

MCU driver**Table 1526 Specification for Mcu_17_Gpt12_TimerStop API (continued)**

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other channels	
Parameters (in)	TimerId	GPT12 timer channel to be disabled
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gpt12_TimerStop stops the specified GPT12 timer	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.3.69 Mcu_17_Stm_SetupComparator**Table 1527 Specification for Mcu_17_Stm_SetupComparator API**

Syntax	<pre>void Mcu_17_Stm_SetupComparator (const Mcu_17_Stm_TimerConfigType * const ConfigPtr)</pre>	
Service ID	0x90	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other STM comparators	
Parameters (in)	ConfigPtr	STM Timer Compare operation contents
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

MCU driver**Table 1527 Specification for Mcu_17_Stm_SetupComparator API (continued)**

Description	Mcu_17_Stm_SetupCompareOperation configures the compare register of the STM timer
Source	IFX
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	-
User hints	-

17.3.3.70 Mcu_17_Stm_CheckComparator**Table 1528 Specification for Mcu_17_Stm_CheckComparator API**

Syntax	<pre>void Mcu_17_Stm_CheckComparator (const Mcu_17_Stm_TimerConfigType * const ConfigPtr)</pre>	
Service ID	0x91	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other STM Timers	
Parameters (in)	ConfigPtr	STM Timer channel initialization contents
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Stm_CheckCompareRegContent checks the configuration of the compare register against the passed configuration	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	

MCU driver
Table 1528 Specification for Mcu_17_Stm_CheckComparator API (continued)

User hints	User should verify the value of the Compare register as its value can change at the run-time
-------------------	--

17.3.3.71 Mcu_17_Stm_ComparatorIntDisable
Table 1529 Specification for Mcu_17_Stm_ComparatorIntDisable API

Syntax	<pre>void Mcu_17_Stm_ComparatorIntDisable (const uint8 StmTimerId, const uint8 StmComparatorId)</pre>	
Service ID	0x88	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other STM Timers	
Parameters (in)	StmTimerId StmComparatorId	STM Timer Id STM Comparator Id
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Stm_ComparatorIntDisable disables the comparator interrupt	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.4 Notifications and callbacks

This section lists all the notification and callbacks of the MCU driver.

MCU driver**17.3.4.1 Mcu_ClockFailureNotification****Table 1530 Specification for Mcu_ClockFailureNotification API**

Syntax	<pre>void Mcu_ClockFailureNotification (void)</pre>	
Service ID	0xFF	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Mcu_ClockFailureNotification can be invoked to know the source of the clock failure, after such an occurrence. Mcu_ClockFailureNotification reports any one of MCU_E_SYSTEM_PLL_LOCK_LOSS, MCU_E_PERIPHERAL_LOCK_LOSS and MCU_E_OSC_FAILURE DEMs.</p> <p>If the root cause of a PLL loss of lock is an oscillator failure, then MCU_E_OSC_FAILURE DEM is reported.</p> <p>Availability of this function is controlled by the McuClockSourceFailureNotification parameter.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>MCU_E_SYSTEM_PLL_LOCK_LOSS: This DEM is raised when Loss of System PLL lock occurs</p> <p>MCU_E_PERIPHERAL_PLL_LOCK_LOSS: This DEM is raised when Loss of Peripheral PLL lock occurs</p> <p>MCU_E_OSC_FAILURE: Inability of the oscillator to deliver correct clock</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	McuClockSourceFailureNotification	
User hints	-	

17.3.5 Scheduled functions

The driver does not support any scheduled functions.

MCU driver

17.3.6 Interrupt service routines

This section lists all the interrupt handlers of the driver.

17.3.6.1 Mcu_17_Ccu6_ChannelIsr

Table 1531 Specification for Mcu_17_Ccu6_ChannelIsr API

Syntax	<pre>void Mcu_17_Ccu6_ChannelIsr (const Mcu_17_Ccu6_KernelIdentifierType Kernel, const Mcu_17_Ccu6_ComparatorType Comparator)</pre>	
Service ID	0x95	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Kernel Comparator	CCU6 Kernel CCU6 Comparator type
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Mcu_17_Ccu6_ChannelIsr is the interrupt service routine of a CCU6 timer channel and is invoked by the interrupt frame installed in the interrupt vector table.</p> <p>Mcu_17_Ccu6_ChannelIsr identifies the user of the specified channel and invokes a known call back function associated with the user.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>MCU_E_INVALID_ISR: Error is reported if an ISR is invoked on a spurious interrupt</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

MCU driver**17.3.6.2 Mcu_17_Eru_GatingIsr****Table 1532 Specification for Mcu_17_Eru_GatingIsr API**

Syntax	<pre>void Mcu_17_Eru_GatingIsr (const Mcu_17_Eru_SrcIdentifierType EruSrcId)</pre>	
Service ID	0x98	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	EruSrcId	Input Channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Eru_GatingIsr is the interrupt service routine of the ERU and is invoked by the interrupt frame installed in the interrupt vector table. It identifies the user of the specified ERU channel and invokes a known call back function associated with the user.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCU_E_INVALID_ISR: Error is reported if an ISR is invoked on a spurious interrupt <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	The value of parameter IrqFlag is always zero as it is checked and passed. This parameter is just to maintain the consistency	

17.3.6.3 Mcu_17_Gpt12_ChannelIsr**Table 1533 Specification for Mcu_17_Gpt12_ChannelIsr API**

Syntax	<pre>void Mcu_17_Gpt12_ChannelIsr (const Mcu_17_Gpt12_TimerChIdentifierType Timer)</pre>	
---------------	--	--

MCU driver**Table 1533 Specification for Mcu_17_Gpt12_ChannelIsr API (continued)**

Service ID	0x96	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Timer	GPT12 timer
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gpt12_ChannelIsr is the interrupt service routine of a GPT12 timer channel and is invoked by the interrupt frame installed in the interrupt vector table. Mcu_17_Gpt12_ChannelIsr identifies the user of the specified channel and invokes a known call back function associated with the user.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.6.4 Mcu_17_Gtm_AtomChannelIsr**Table 1534 Specification for Mcu_17_Gtm_AtomChannelIsr API**

Syntax	<pre>void Mcu_17_Gtm_AtomChannelIsr (const uint8 Module, const uint8 Channel)</pre>
Service ID	0x93
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant for different channels

MCU driver**Table 1534 Specification for Mcu_17_Gtm_AtomChannelIsr API (continued)**

Parameters (in)	Module Channel	ATOM module number ATOM channel number (it should always be an even number since two channels are mapped to the same interrupt node)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_AtomChannelIsr is the interrupt service routine of an ATOM channel and is invoked by the interrupt frame installed in the interrupt vector table.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCU_E_INVALID_ISR: Error is reported if an ISR is invoked on a spurious interrupt <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.6.5 Mcu_17_Gtm_TimChannelIsr**Table 1535 Specification for Mcu_17_Gtm_TimChannelIsr API**

Syntax	<pre>void Mcu_17_Gtm_TimChannelIsr (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x94	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Module Channel	TIM module number TIM channel number
Parameters (out)	-	-

MCU driver**Table 1535 Specification for Mcu_17_Gtm_TimChannelIsr API (continued)**

Parameters (in - out)	-	-
Return	void	-
Description	Mcu_17_Gtm_TimChannelIsr is the interrupt service routine of a TIM channel and is invoked by the interrupt frame installed in the interrupt vector table.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: MCU_E_INVALID_ISR: Error is reported if an ISR is invoked on a spurious interrupt <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

17.3.6.6 Mcu_17_Gtm_TomChannelIsr**Table 1536 Specification for Mcu_17_Gtm_TomChannelIsr API**

Syntax	<pre>void Mcu_17_Gtm_TomChannelIsr (const uint8 Module, const uint8 Channel)</pre>	
Service ID	0x92	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Module Channel	TOM module number TOM channel number (it should always be an even number since two channels are mapped to the same interrupt node)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

MCU driver**Table 1536 Specification for Mcu_17_Gtm_TomChannelIsr API (continued)**

Description	Mcu_17_Gtm_TomChannelIsr is the interrupt service routine of a TOM channel and is invoked by the interrupt frame installed in the interrupt vector table.
Source	IFX
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>MCU_E_INVALID_ISR: Error is reported if an ISR is invoked on a spurious interrupt</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

17.3.6.7 Mcu_17_Stm_CompareMatchIsr**Table 1537 Specification for Mcu_17_Stm_CompareMatchIsr API**

Syntax	<pre>void Mcu_17_Stm_CompareMatchIsr (const Mcu_17_Stm_StmIdentifierType StmTimerId, const Mcu_17_Stm_StmCmpIdentifierType StmCmpId)</pre>	
Service ID	0x97	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for other STM timers	
Parameters (in)	StmTimerId	STM timer ID
	StmCmpId	STM comparator ID
Parameters (out)	-	
Parameters (in - out)	-	
Return	void	-
Description	Mcu_17_Stm_CompareMatchIsr is the interrupt service routine of a STM timer and is invoked by the interrupt frame installed in the interrupt vector table. It identifies the user of the specified STM timer and invokes a known call back function associated with the user.	
Source	IFX	
Error handling	DET: None	

MCU driver
Table 1537 Specification for Mcu_17_Stm_CompareMatchIsr API (continued)

	<p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>MCU_E_INVALID_ISR: Error is reported if an ISR is invoked on a spurious interrupt</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

17.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

17.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Note: The following error IDs are also reported as safety errors.

Table 1538 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
API is called from a core which is not the master core.	IFX	MCU_E_CORE_MISMATCH=0x68	Mcu_InitClock, Mcu_Init, Mcu_DistributePllClock, Mcu_DeInit
Error is reported when Mcu_Init() API is called when it is already initialized.	AUTOSAR	MCU_E_INIT_FAILED=0x11	Mcu_Init
ClockSetting parameter does not match the corresponding data in the Mcu_ConfigType object .	AUTOSAR	MCU_E_PARAM_CLOCK =0x0B	Mcu_InitClock
ConfigPtr passed to Mcu_Init is NULL.	AUTOSAR	MCU_E_PARAM_CONFIG=0x0A	Mcu_Init
Input argument for CPU Id passed with an invalid core index.	IFX	MCU_E_PARAM_CPUID=0x13	Mcu_GetCpuState, Mcu_UpdateCpuCcuconReg
CpuCcucon divider update requested with value higher than maximum possible divider value.	IFX	MCU_E_PARAM_DIV_VAL=0x15	Mcu_UpdateCpuCcuconReg
McuMode parameter does not match the	AUTOSAR	MCU_E_PARAM_MODE =0x0C	Mcu_SetMode

MCU driver**Table 1538 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
corresponding data in the Mcu_ConfigType object.			
Versioninfo pointer passed to Mcu_GetVersionInfo is NULL.	AUTOSAR	MCU_E_PARAM_POINTER=0x10	Mcu_GetVersionInfo
RamSection parameter does not match the corresponding data in the Mcu_ConfigType object.	AUTOSAR	MCU_E_PARAM_RAMSECTION =0x0D	Mcu_InitRamSection
Trap-related read or write with an invalid trap source id.	IFX	MCU_E_PARAM_TRAPID=0x14	Mcu_SetTrapRequest, Mcu_ClearTrapRequest
Either the system or peripheral PLL is not locked.	AUTOSAR	MCU_E_PLL_NOT_LOCKED=0x0E	Mcu_DistributePllClock
Power down mode entry is requested by an unauthorized CPU.	IFX	MCU_E_UNAUTHORIZED_REQUESTER=0x12	Mcu_SetMode
Error is reported if the API is called before Mcu_Init is called.	AUTOSAR	MCU_E_UNINIT=0x0F	Mcu_GetRamState, Mcu_DeInit, Mcu_InitRamSection, Mcu_InitClock, Mcu_DistributePllClock, Mcu_GetPllStatus, Mcu_GetResetReason, Mcu_GetResetRawValue, Mcu_PerformReset, Mcu_SetMode, Mcu_ClearColdResetStatus, Mcu_GetCpudleModelInitiator, Mcu_ClearWakeupCause, Mcu_GetCpuState, Mcu_GetWakeupCause, Mcu_GetTrapCause, Mcu_SetTrapRequest, Mcu_ClearTrapRequest, Mcu_UpdateCpuCcuconReg

17.3.7.2 Production errors

The following table lists all the production errors reported by the driver.

MCU driver**Table 1539 Description of production errors reported**

Description	Source	Error code and value	Applicable APIs
Inability to turn OFF the CCU6 kernel clock disable bit.	IFX	MCU_E_CCU6_CLC_DISABLE_ERR=Assigned by DEM	Mcu_DeInit
Inability to turn ON the CCU6 kernel Clock enable bit.	IFX	MCU_E_CCU6_CLC_ENABLE_ERR=Assigned by DEM	Mcu_Init
Inability to update the CCUCON register.	IFX	MCU_E_CCUCON_UPDATE_ERR=Assigned by DEM	Mcu_DistributePllClock, Mcu_InitClock
Inability to turn OFF the CONVCTRL Clock disable bit.	IFX	MCU_E_CONVCTRL_CLC_DISABLE_ERR=Assigned by DEM	Mcu_DeInit
Inability to turn ON the CONVCTRL Clock enable bit.	IFX	MCU_E_CONVCTRL_CLC_ENABLE_ERR=Assigned by DEM	Mcu_Init
Inability to turn OFF the GPT12 clock disable bit.	IFX	MCU_E_GPT12_CLC_DISABLE_ERR=Assigned by DEM	Mcu_DeInit
Inability to turn ON the GPT12 Clock enable bit.	IFX	MCU_E_GPT12_CLC_ENABLE_ERR=Assigned by DEM	Mcu_Init
Inability to turn OFF the GTM clock disable bit.	IFX	MCU_E_GTM_CLC_DISABLE_ERR=Assigned by DEM	Mcu_DeInit
Inability to turn ON the GTM Clock enable bit.	IFX	MCU_E_GTM_CLC_ENABLE_ERR=Assigned by DEM	Mcu_Init
Inability of the oscillator to deliver correct clock.	IFX	MCU_E_OSC_FAILURE=Assigned by DEM	Mcu_ClockFailureNotification, Mcu_InitClock
This DEM is raised when Loss of Peripheral PLL lock occurs.	IFX	MCU_E_PERIPHERAL_PLL_LOCK_LOSS=Assigned by DEM	Mcu_ClockFailureNotification
DEM is raised due to inability of the peripheral PLL K2/K3 dividers and power mode to be updated within the specified time.	IFX	MCU_E_PERIPHERAL_PLL_TIMEOUT_ERR=Assigned by DEM	Mcu_InitClock, Mcu_SetMode
Inability to update the PMSWCRx register.	IFX	MCU_E_PMSWCR_UPDATE_ERR=Assigned by DEM	Mcu_SetMode
This DEM is raised when Loss of System PLL lock occurs.	IFX	MCU_E_SYSTEM_PLL_LOCK_LOSS=Assigned by DEM	Mcu_ClockFailureNotification
DEM is raised due to inability of the system PLL K2 divider and power	IFX	MCU_E_SYSTEM_PLL_TIMEOUT_ERR=Assigned by DEM	Mcu_InitClock, Mcu_DistributePllClock, Mcu_SetMode

MCU driver
Table 1539 Description of production errors reported (continued)

Description	Source	Error code and value	Applicable APIs
mode to be updated within the specified time.			

17.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 1540 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
Error is reported if an ISR is invoked on a spurious interrupt.	IFX	MCU_E_INVALID_ISR=202	Mcu_17_Ccu6_Channelsr, Mcu_17_Stm_CompareMatchIsr, Mcu_17_Eru_GatingIsr, Mcu_17_Gtm_AtomChannelsr, Mcu_17_Gtm_TomChannelsr, Mcu_17_Gtm_TimChannelsr
Error is raised when phase configuration register of Converter Control update fails.	IFX	MCU_E_PHSCFG_UPDATE_ERR=203	Mcu_InitClock
Error is reported when software reset fails after calling the Mcu_PerformReset API.	IFX	MCU_E_SW_RESET_FAILED=201	Mcu_PerformReset

17.3.7.4 Runtime errors

The driver does not report any runtime errors.

17.3.8 Deviations and limitations

The section describes the deviations and limitations of the MCU driver.

17.3.8.1 Deviations

The section describes the deviations from software specification.

Table 1541 Known deviations

Reference	Deviation
Safety error for unintended service request	Refer to Reporting of unintended service requests .

MCU driver**17.3.8.2 Limitations**

There are no deviations for the driver.

17.3.9 Unsupported hardware features

The MCU driver does not support the following features of the SCU:

- Emergency stop
- Power management control

OCU driver

18 OCU driver

18.1 User information

18.1.1 Description

The OCU driver is responsible for triggering an event on a free- running counter compare match. The event can either be a pin level change, a DMA trigger, an ADC trigger or just a notification. The services provided by the OCU driver are:

- Start/stop a channel
- Configure the compare match values
- Set the pin action on the sub-sequent compare matches
- Enable/disable notifications and to set the pin level on a stopped channel

The TOM and ATOM slices of the GTM will be used to achieve the configured functionality.

Note: The OCU driver does not support GTM-less device (for example, TC35x).

18.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

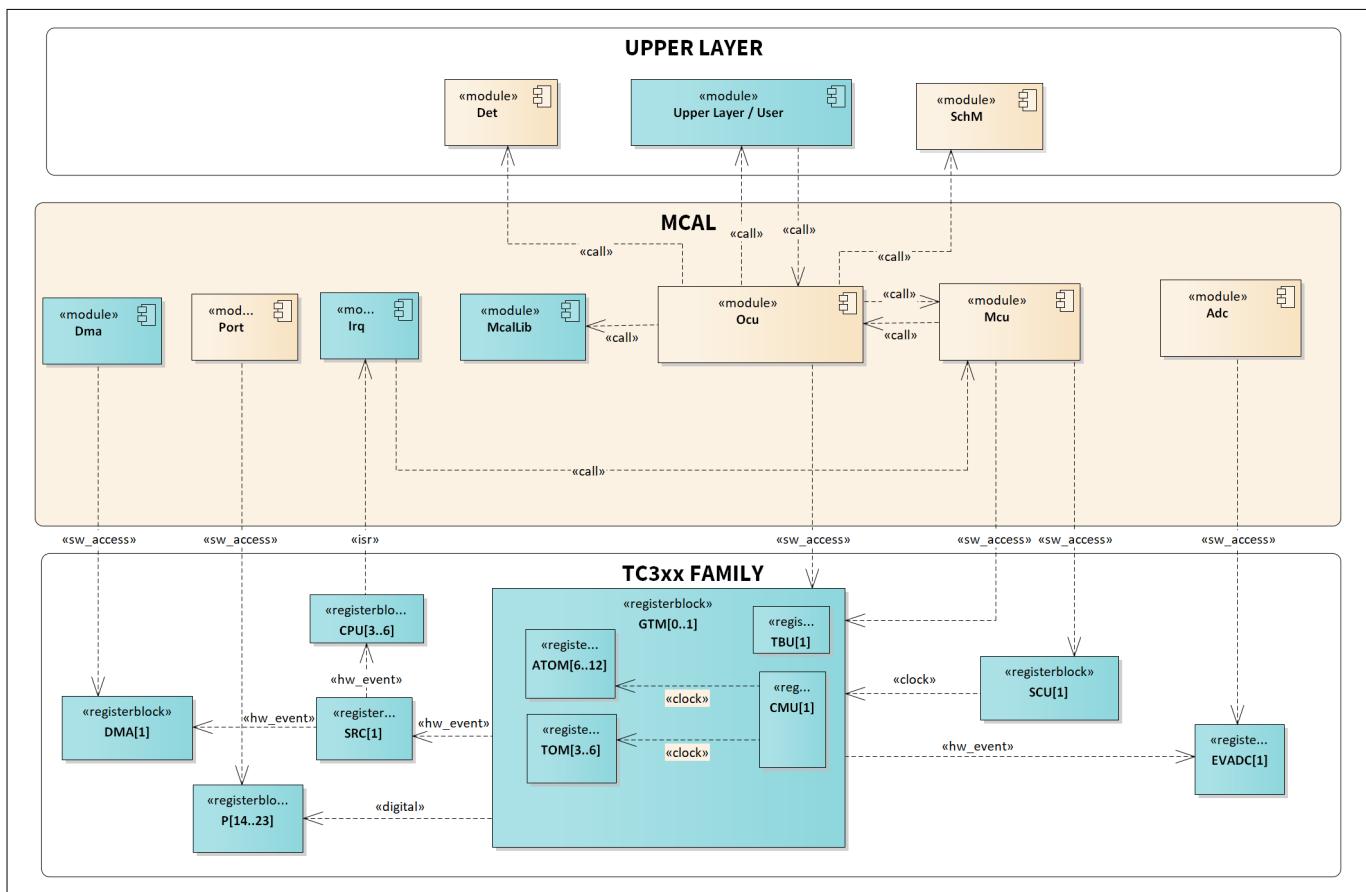


Figure 160

Mapping of hardware-software interfaces

OCU driver

18.1.2.1 **GTM: primary hardware peripheral.**

Hardware functional features

The OCU driver uses the GTM IP (TOM/ATOM) for realizing the following features:

- TOM CM1 compare match
- ATOM CM1 compare match in SOMP mode
- ATOM CM0/CM1 compare match in SOMC mode
- ATOM in SOMC mode pin level changes, ADC trigger
- ATOM in SOMP mode DMA trigger
- TOM/ATOM notification
- TOM DMA trigger

The key hardware functional features used by the driver are:

- Channel clock source configuration
- TOUT configuration for the Port selection
- Cyclic comparators for the compare match(ATOM SOMC mode - shared TBU clock)
- Signaling mechanisms: Global enable/disable mechanism, output enable mechanism
- ATOM operation mode are SOMP, SOMC

The unsupported features of the GTM are:

- Global force update signaling mechanism
- Continuous down counting down mode, one shot up mode, one shot down mode
- ATOM operational mode: SOMI or SOMS
- TOM/ATOM - SOMP mode triggering port pin or ADC configuration

Users of the hardware

The CMU functional block of the GTM peripheral and TBU_TS[x] timer are exclusively handled by the MCU driver. The MCU driver provides APIs to program the GTM SFRs. The OCU driver uses these APIs to write the GTM SFRs. Additionally, updates to the channel-specific SFRs are performed by the OCU driver. Since these channels are exclusively reserved for the OCU driver, access to the channel-specific SFRs from other drivers or user software is not allowed.

The GTM TOM/ATOM functional blocks are used by PWM, GPT and OCU driver.

Note: A TOM/ATOM channel of the GTM is exclusively used by its user. While users of TOM/ATOM channels are many, a channel is allocated to and used by exactly one driver.

Hardware diagnostic features

Not applicable.

Hardware events

The OCU module uses the compare match events generated by the timer channel to provide notification of events to application and perform an action (pin level change, ADC trigger or DMA trigger) without software intervention. Mcu module is responsible for handling interrupts and invoking the callback function to service the interrupt.

18.1.2.2 **SCU: dependent hardware peripheral**

Hardware functional features

The OCU driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB and fGTM clock signals for functioning.

Users of the hardware

OCU driver

The SCU IP supplies clock for all the peripherals and the MCU driver, and is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the OCU driver.

Hardware events

Hardware events from the SCU are not used by the OCU driver.

18.1.2.3 Port: dependent hardware peripheral**Hardware functional features**

The digital output signals are mapped/routed to port pads through TOUT configurations. TOUT configuration are performed in the OCU driver.

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from port pads are not used by the ICU driver.

18.1.3 File structure**18.1.3.1 C file structure**

OCU driver

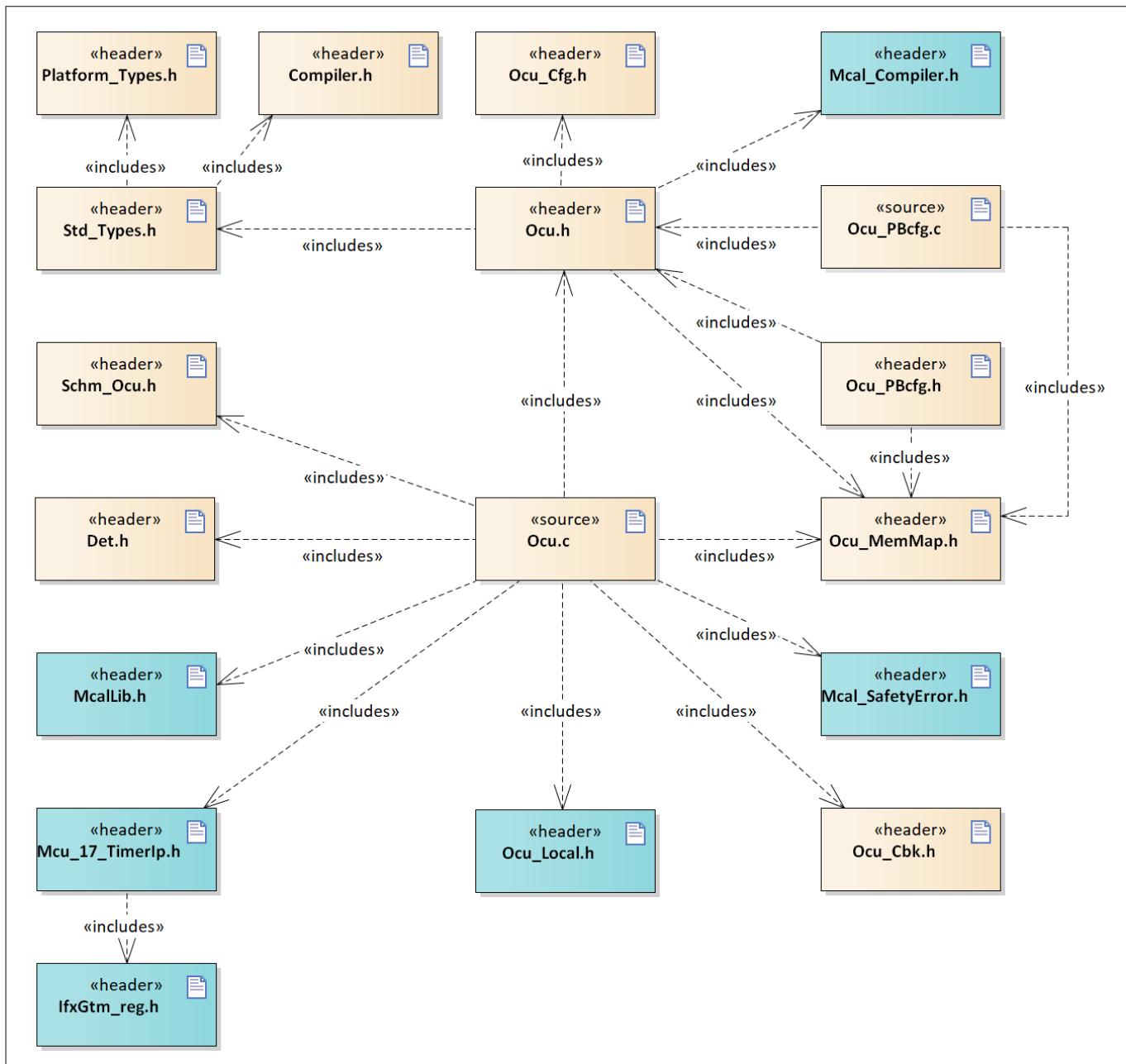


Figure 161 C file structure

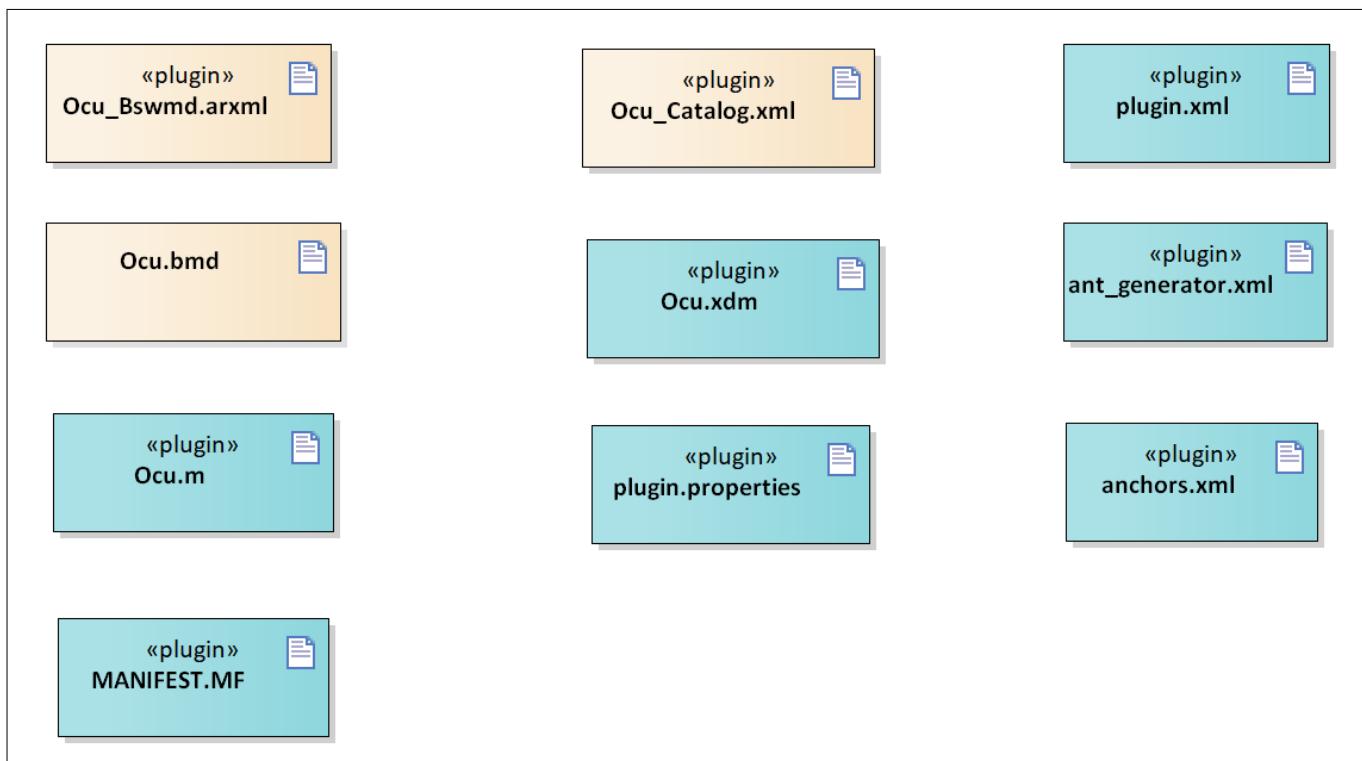
Table 1542 C file structure

File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Det.h	Provides the exported interfaces of Development Error Tracer
IfxGtm_reg.h	SFR header file for GTM
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
Mcal_Compiler.h	Provides abstraction for TriCore™-intrinsic instruction
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors

OCU driver
Table 1542 C file structure (continued)

File name	Description
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Ocu.c	File (static) containing implementation of APIs
Ocu.h	Header file (static) defining prototypes of configuration data structures and APIs
Ocu_Cbk.h	Includes callback header definition
Ocu_Cfg.h	Header file (generated) containing constants and pre-processor macros
Ocu_Local.h	Header file defining type definition of global data and inline APIs, which can be used across source files
Ocu_MemMap.h	File (static) containing the memory section definitions used by the OCU driver
Ocu_PBcfg.c	File (generated) containing objects to data structures
Ocu_PBcfg.h	File (generated) containing declaration of the post-build configuration data structures
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
Schm_Ocu.h	The header file contains the definitions of OCU critical sections
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

18.1.3.2 Code generator plugin files


Figure 162 Code generator plugin files

OCU driver

Table 1543 Code generator plugin files

File name	Description
MANIFEST.MF	Tresos plugin support file containing the metadata for OCU driver
Ocu.m	Code template macro file for OCU driver
Ocu.xdm	Tresos format XML data model schema file
Ocu_Bswmd.arxml	AUTOSAR format module description file
Ocu_Catalog.xml	AUTOSAR format catalog file
anchors.xml	Tresos anchors support file for the OCU driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point
plugin.properties	Tresos plugin support file for the OCU driver
plugin.xml	Tresos plugin support file for the OCU driver

18.1.4 Integration hints

This section lists the key points that an integrator or user of the OCU driver must consider. The OCU features and hardware configurations as follows:

Table 1544 OCU channel features supported different HW channels

Feature	TOM channel	ATOM channel in SOMP mode	ATOM channel in SOMC mode
Counter type	Exclusive	Exclusive	Shared (TBU_TS0/1/2)
Notifications	Yes	Yes	Yes
Pin	No	No	Yes
DMA trigger	Yes	Yes	No
ADC trigger	No	No	Yes

18.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the OCU driver.

- **ECU State Manager (EcuM)**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of the MCAL, the EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

-Initialization of OCU:

The user of OCU driver may use APIs of EcuM to initialize the driver. The initialization of the driver should be invoked from each CPU core, which intends to use the services of the OCU driver. All cores can execute initialization simultaneously.

-De-initialization of OCU:

OCU driver

The user of OCU driver may use APIs of EcuM to de-initialize the driver. The de-initialization of the driver should be invoked from each CPU core that used the services of the OCU driver. All cores can execute de-initialization simultaneously.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Ocu_MemMap.h` file.

OCU driver

The `Ocu_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

```
***** GLOBAL RAM DATA -- NON-CACHED LMU *****/
#if defined OCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*User pragmas here for Non-cached LMU*****/
#undef OCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR

#elif defined OCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*User pragmas here for Non-cached LMU*****/
#undef OCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR

***** CORE[x] RAM DATA -- DSPR ****/ /*[x]=0..5*/
#elif defined OCU_START_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
/*User pragmas here for CORE[x] DSPR*****/
#undef OCU_START_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
#undef MEMMAP_ERROR

#elif defined OCU_STOP_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
/*User pragmas here for CORE[x] DSPR*****/
#undef OCU_STOP_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
#undef MEMMAP_ERROR

***** GLOBAL CONST DATA -- PF[x] ****/
#elif defined OCU_START_SEC_CONST_ASIL_B_GLOBAL_32
/*User pragmas here for PF[x]*****/
#undef OCU_START_SEC_CONST_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR

#elif defined OCU_STOP_SEC_CONST_ASIL_B_GLOBAL_32
/*User pragmas here for PF[x]*****/
#undef OCU_STOP_SEC_CONST_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR

***** GLOBAL CONFIG DATA -- PF[x] ****/
#elif defined OCU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/*User pragmas here for PF[x]*****/
#undef OCU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

#elif defined OCU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/*User pragmas here for PF[x]*****/
#undef OCU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

***** CORE[x] CONFIG DATA -- PF[x] ****/ /*[x]=0..5*/
#elif defined OCU_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
/*User pragmas here for PF[x]*****/
#undef OCU_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
#undef MEMMAP_ERROR

#elif defined OCU_STOP_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
/*User pragmas here for PF[x]*****/
#undef OCU_STOP_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED

```

OCU driver

```
#undef MEMMAP_ERROR

/***** CODE -- PF[x] *****/
#elif defined OCU_START_SEC_CODE_ASIL_B_GLOBAL
/*****User pragmas here for PF[x]*****/
#undef OCU_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#elif defined OCU_STOP_SEC_CODE_ASIL_B_GLOBAL
/*****User pragmas here for PF[x]*****/
#undef OCU_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Ocu_MemMap.h, wrong pragma command"
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The OCU driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the OCU driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

DEM module is not required for integrating the OCU driver.

- **BSW Scheduler Module (SchM)**

The SchM module is a part of the RTE that manages the BSW Scheduler. The OCU driver uses the exclusive areas defined in the `SchM_Ocu.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the OCU driver are:

- SetPinAction
- SetThresholdValue

The `SchM_Ocu.h` and `SchM_Ocu.c` files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the OCU driver as **suspend /**

OCU driver

resume of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows:

```
***** Sample implementation of SchM_Ocu.c *****
#include "Os.h"

/* Disable the interrupts for entering critical section */
void SchM_Enter_Ocu_SetPinAction(void)
{
    SuspendAllInterrupts();
}

/* Re-enable the interrupt for exiting the critical section */
void SchM_Exit_Ocu_SetPinAction(void)
{
    ResumeAllInterrupts();
}

/* Disable the interrupts for entering critical section */
void SchM_Enter_Ocu_SetThresholdValue(void)
{
    SuspendAllInterrupts();
}

/* Re-enable the interrupt for exiting the critical section */
void SchM_Exit_Ocu_SetThresholdValue(void)
{
    ResumeAllInterrupts();
}
```

- **Safety error**

The OCU driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API.

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The OCU driver does not implement any notifications. However, the driver reports the compare match event through notification functions. These notification functions can be configured by the user in EB Tresos for each OCU channel.

The OCU does not expect any call-backs from application. But the OCU driver needs the callback ISR from the MCU driver.

- **Operating system (OS)**

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or the application.

The OS files provided by MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

OCU driver

18.1.4.2 Multicore and Resource Manager

The OCU driver supports execution of its APIs simultaneously from all CPU cores. The user has to allocate each channel of the OCU driver to CPU cores at pre-compile time using the Resource Manager module. The following are the key points to be considered with respect to multicore in the OCU driver:

- Each OCU channel can be allocated to any core using the Resource Manager.
- Interrupts raised by an OCU channel must be serviced by the CPU core to which the channel has been allocated to, if DMA triggering for that channel is not configured.
- OCU channels using GTM-ATOM, channel GTM-ATOM[i]_CH[2x] and ATOM[i]_CH[2x+1], must be allocated to same core as these two channels share same interrupt line.
- OCU channels using GTM-TOM, channel GTM-TOM[i]_CH[2x] and TOM[i]_CH[2x+1], must be allocated to same core as these two channels share same interrupt line.
- Locating constants, variables and configuration data to correct memory space should be done by the user. Memory sections are marked GLOBAL (common to all cores) and CORE[x](specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of the OCU driver is placed under single MemMap section. It can be relocated to any PFlash/DFlash region.

Data section:

The RAM variable memory sections marked as specific to a core should be re-located to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region.

Configuration data and constants:

The configuration data sections marked as specific to a core should be re-located to the PFlash/DFlash of the same core. The sections marked as global should be relocated to the PFlash/DFlash of the master core.

Note: *Relocating code, data or constants to a distant memory region would impact execution timings.*

Note: *If the driver operates from single(master) core, all the sections may be relocated to the PFLASH/DSPR/ DLMU of the same CPU core.*

18.1.4.3 MCU support

The OCU driver is dependent on the MCU driver for clock configuration and timer-IP related services. The initialization of the OCU driver must be started only after completing the MCU initialization. The following must be considered while configuring the MCU driver in EB Tresos:

- The GTM TOM/ATOM timers used by the OCU driver must be reserved in the MCU configuration for exclusive use by the OCU.

Access of Shared GTM Sfrs

If channels of the same TOM/ATOM module is shared between the application and the OCU driver, user shall ensure shared register of TOM/ATOM modules is accessed using the MCU timer-IP library APIs.

18.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the OCU driver through the PORT configuration as output and initialize the port pins prior to invoking of OCU initialization.

OCU driver

18.1.4.5 DMA support

An OCU channel compare match event can trigger a DMA channel. User is responsible to configure and initialize the DMA channel to accept the hardware trigger from the OCU channel.

18.1.4.6 Interrupt connections

The interrupt connections of the OCU driver are described in this section.

User should enable interrupts of the corresponding OCU channel in interrupt configuration register. The interrupt configuration registers of different hardware used by OCU channels are as follows:

Table 1545 GTM interrupts

GTM hardware used	SRC register
GTM-TOM	SRC_GTMTOOMwx (w= TOM module; shared between 2x and 2x+1 TOM channel)
GTM-ATOM	SRC_GTMATOMwx (w= ATOM module; shared between 2x and 2x+1 ATOM channel)

- For OCU channels not triggering DMA

The priority(SRPN) is chosen by the user but the Type Of Service(TOS) should be the CPU to which the OCU channel is allocated.

- For OCU channels triggering DMA

The priority(SRPN) should be the DMA channel which needs to be triggered on a compare match event of that OCU channel and Type Of Service(TOS) should be DMA.

OCU driver

18.1.4.7 Example usage

Configuration:

User can configre either a TOM or an ATOM channel for each OCU channel.

The mode of the ATOM channel is selected during the generation phase depending on the timer selected for that channel. If one of the TBU timer is selected, ATOM channel is configured for the SOMC mode else channel is configured in the SOMP mode.

TOM channel supports only one mode and no selection is required. Also, TBU timer cannot be used as a timer for TOM channel.

DMA triggers:

Each OCU channel using TOM or ATOM(in SOMP) can trigger one DMA channel(configured as SRPN). The compare match event of the OCU channel is routed as DMA trigger. Hence if an OCU channel is triggering a DMA channel, notifications cannot be issued by the OCU channel.

```

Ocu_Init(<Configuration pointer>);

Ocu_StartChannel(<logical channel ID>);

/* DMA gets triggered on each compare match of default threshold until the
call to Ocu_SetAbsoluteThreshold/Ocu_SetRelativeThreshold */

<compare value write in/out interval> = Ocu_SetAbsoluteThreshold(<logical
channel ID>, <reference value>, <absolute threshold value>);
/* Return value indicates if the new compare value is written within the
reference interval or outside */

/* DMA gets triggered on each compare match of new absolute threshold set by
the previous call of Ocu_SetAbsoluteThreshold */

<compare value write in/out interval> = Ocu_SetRelativeThreshold(<logical
channel ID>, <relative threshold value>);
/* Return value indicates if the new compare value is written within the
reference interval or outside */

/* DMA gets triggered on each compare match of new threshold set by the
previous call of Ocu_SetRelativeThreshold */

Ocu_StopChannel(<logical channel ID>);

/* DMA triggering is stopped */

Ocu_DeInit();

```

ADC triggering:

The output level of the ATOM channel (in SOMC mode) is routed to the ADC as a trigger line.

When the OCU channel is configured for both pin and ADC triggering and the compare match events are set, both ADC trigger and pin state are affected.

Note: When the `Ocu_SetPinState` API is called, both pin and ADC trigger (if configured) are affected. ADC triggering is an unintended consequence when this API is invoked.

OCU driver

The user of the ADC driver can configure the trigger from the OCU driver either as a level trigger or as an edge trigger with the following behavioral considerations:

- For OCU channel configured for both pin and ADC triggering, default pin level selected by the user is the level of the ADC trigger line after initialization. The trigger line level can be changed on a compare match event using the `Ocu_SetPinAction` API. The change of pin level gets reflected in the ADC trigger line.

```

Ocu_Init(<Configuration pointer>);

Ocu_StartChannel(<logical channel ID>);

/* ADC trigger line is unchanged */

Ocu_SetPinAction(<logical channel ID>, OCU_SET_HIGH);

/* ADC trigger line is set to high on the next compare match of default
threshold */

<compare value write in/out interval> = Ocu_SetRelativeThreshold(<logical
channel ID>, <relative threshold value>);
Ocu_SetPinAction(<logical channel ID>, OCU_LOW); /* Applicable if pin is
configured*/
/* Return value indicates if the new compare value is written within the
reference interval or outside */

/* ADC trigger line is set to low on the next compare match of new threshold */

Ocu_StopChannel(<logical channel ID>);

/* ADC trigger line is unchanged */

Ocu_SetPinState(<logical channel ID>, OCU_HIGH);

/* ADC trigger line is set to high */

Ocu_DeInit();

```

- For OCU channel configured for ADC triggering but not supporting pin, default ADC trigger line level is LOW after initialisation. The ADC trigger line gets toggled on every compare match event. Hence, for an edge

OCU driver

triggered ADC, user should enable both edge triggering(in ADC) to trigger ADC on each compare match. For a level trigger, user can configure threshold values to start/stop the ADC triggering.

```
Ocu_Init(<Configuration pointer>);

Ocu_StartChannel(<logical channel ID>);

/* ADC trigger line is toggled on each compare match(default threshold) */

<compare value write in/out interval> = Ocu_SetRelativeThreshold(<logical
channel ID>, <relative threshold value>);

/* Return value indicates if the new compare value is written within the
reference interval or outside */

/* ADC trigger line is toggled on each compare match(new threshold) */

Ocu_StopChannel(<logical channel ID>);

/* ADC trigger line is unchanged */

Ocu_DeInit();
```

Notification:

OCU driver

Each OCU channel can issue a notification callback to the application on a compare match event.

```

Ocu_Init(<Configuration pointer>);

Ocu_StartChannel(<logical channel ID>);

/* Notifications are issued on each compare match(default threshold) */

<compare value write in/out interval> = Ocu_SetAbsoluteThreshold(<logical
channel ID>, <reference value>, <absolute threshold value>);
/* Return value indicates if the new compare value is written within the
reference interval or outside */

/* Notifications are issued on each compare match(new threshold) */

<compare value write in/out interval> = Ocu_SetRelativeThreshold(<logical
channel ID>, <relative threshold value>);
/* Return value indicates if the new compare value is written within the
reference interval or outside */

/* Notifications are issued on each compare match(new threshold) */

Ocu_StopChannel(<logical channel ID>);

/* Notifications are no longer issued */

Ocu_DeInit();

```

18.1.5 Key architectural considerations

There are no key architectural considerations for the OCU driver.

18.1.5.1 DMA support

Each OCU channel (using TOM/ATOM in SOMP) is capable of triggering a DMA transfer with the following restrictions:

- To enable a DMA transaction on a compare match, user shall configure the Type Of Service (TOS) of the corresponding OCU channel service node (TOM or ATOM) as **DMA** and **SRPN** with the DMA channel number, which needs to be triggered.
- The hardware resource that shares the interrupt line with the OCU channel, which supports the DMA should not be used.

TOM(2[x]) and TOM(2[x]+1) share the same interrupt line. Where, x is in the range from 0 to 7.

ATOM(2[x]) and ATOM(2[x]+1) share the same interrupt line. Where, x is in the range from 0 to 3.

Hence, if TOM0 is used for OCU channel triggering DMA, TOM1 should not be used for any other OCU channel/module.

18.1.5.2 Default pin action

The pin action after initialization is set to OCU_DISABLE. This is done to ensure that there is no unintended pin level changes on starting a channel after initialization.

OCU driver

Note: *If ADC triggering with no pin support is configured, the OCU channel output toggles on each compare match.*

18.1.5.3 Default pin level

If OcuOutputPinDefaultState is not configured, the default state of the OCU channel pin is set to OCU_LOW.

18.1.5.4 Threshold values at Start channel

The OCU_StartChannel does not update the threshold values, but uses the previously programmed threshold values.

OCU driver

18.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **ADC trigger on Ocu_SetPinState**

User shall be aware of, on configuring both ADC and port pin as a trigger on the OCU compare match. On Ocu_SetPinState API/function call, changing the port pin state may cause a change in the channel output signal. This would potentially cause an unexpected ADC trigger based on trigger edge configured for the ADC channel.

[cover parentID OCU={2A98C475-744B-45fb-8CC4-FE7AE42DBB0A}]

- **Code generation**

User shall ensure the generated configuration structures are correct against the intended GUI configurations.

[cover parentID OCU={20319FAC-437C-4362-B9AE-FBF21CD9E3B0}]

- **ADC trigger on both edges**

In case the OCU channel is configured only to support the ADC, the default compare match action is set to Toggle.

Rationale: As Ocu_SetPinAction and Ocu_SetPinState functions are pre-compile time configurable API (Ocu_SetPinAction and Ocu_SetPinState APIs are disabled if pin actions are not configured on compare match). Therefore, users do not have an have option to set/change the Edge level (falling/raising) for triggering the ADC on a compare match. Hence, the compare match action is set to Toggle.

[cover parentID OCU={C5B71056-8154-4379-A5A7-471EF2C5A58D}]

- **Correctness of configuration pointer**

User shall ensure the correct configuration pointer is being passed for initializing the Ocu driver.

[cover parentID OCU={7A50611B-46D9-43a3-9342-18DC2364D603}]

- **EVADC MUX configurations**

User shall ensure that EVADC MUX configurations (that is, ADC channel connected to ATOM through EVADC MUX) are according to the group selection in the OCU configuration.

[cover parentID OCU={3C177397-9C0D-4b61-816B-B8DC52734D6D}]

- **Compare match event between Ocu_StartChannel and Ocu_EnableNotification**

User shall be aware that if the compare match event occurs after Ocu_StartChannel but before Ocu_EnableNotification, no call back will be reported for that event. However, future compare match events will be notified through the callback.

[cover parentID OCU={6BF58900-A642-4fb8-AC36-E726504DC406}]

- **No ADC conversion**

In case OCU channel is configured to support both ADC and PORT, the default compare match action is set to NO ACTION. As PORT is enabled/configured to get triggered on compare match through the Ocu_SetPinAction and Ocu_SetPinState APIs, Compare match action and current ADC signal level can be modified. Hence, user can configure ADC conversion on raising edge or falling edge or both the edges.

Scenario: When the initial state (ADC trigger signal) is not set to a desired value, the first compare match event may generate an edge, which may not trigger ADC conversion.

For example, ADC configured for conversion on the falling edge: the initial state is already low and the signal is toggled on the next compare match (that is, the next compare match is the raising edge). Hence, user shall ensure output level is initialized before the OCU start channel.

[cover parentID OCU={3316FCFA-E4C8-4105-A79B-EF93BFC70982}]

- **No ADC trigger**

In case the OCU channel is configured to support both ADC and PORT, the default compare-match action is set to NO ACTION. As PORT is enabled/configured to get triggered on compare match, through the

OCU driver

Ocu_SetPinAction and Ocu_SetPinState APIs, the compare-match action and the current ADC signal level can be modified. Hence, user can configure ADC conversion on raising edge or falling edge or both the edges.

Scenario: When the initial state (ADC trigger signal) is not set to a desired value, the first compare match event may not generate an edge required to trigger ADC conversion.

For example, ADC configured for the falling edge: the initial state is already low and on the next compare match the signal is configured to set LOW. The user shall ensure output level is initialized before the start channel.

[cover parentID OCU={7EB6515B-2749-4e6c-9788-25265C24AE80}]

- **Pin/ADC signal state on stop channel**

User shall be aware that on calling the Ocu_StopChannel API only compare-match event notifications are disabled, but the PIN state or ADC signal levels are unchanged. For example, when the ADC is configured as level trigger, on calling the Ocu_StopChannle API the ADC trigger levels are not changed.

[cover parentID OCU={0D2BF3D1-A029-4ed5-84C9-DEC6B75554CD}]

- **Port Pin state on deinit**

User shall ensure, port pin is SET to the required state in GPIO mode before invoking the deinit API.

Rational: When deinit is invoked, all SFRs are placed to power on reset state and hence there may be a change on the output state of the channel.

[cover parentID OCU={7128C809-9265-40dd-8FB8-7D099E979BCC}]

- **Return value of Ocu_SetAbsoluteThreshold/ Ocu_SetRelativeThreshold**

In the Ocu_SetAbsoluteThreshold/ Ocu_SetRelativeThreshold API, the OCU_CM_OUT_REF_INTERVAL is returned even in case of an error and it may lead to wrong system reaction. Hence, user shall perform an explicit check of DET or safety error to validate the return value.

[cover parentID OCU={2392C9C6-5EEC-45be-9971-4040BC89C527}]

- **Software sequence on Ocu_Init**

User shall ensure that the Mcu_Init and Port_Init APIs are called before the Ocu_Init.

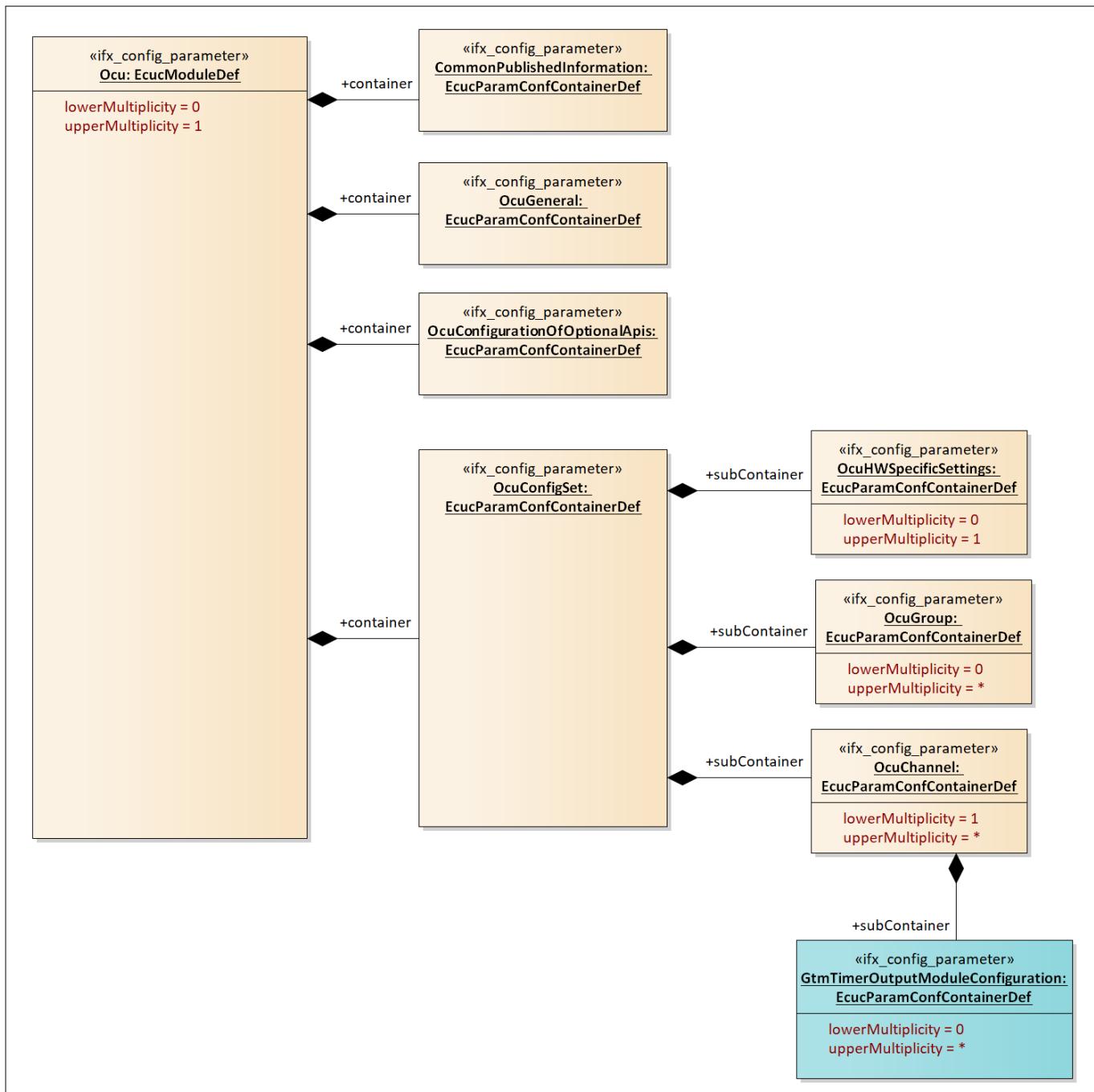
[cover parentID OCU={058E1602-6790-4304-AF7B-C0D06CE9D262}]

OCU driver

18.3 Reference information

18.3.1 Configuration interfaces

OCU driver



OCU driver

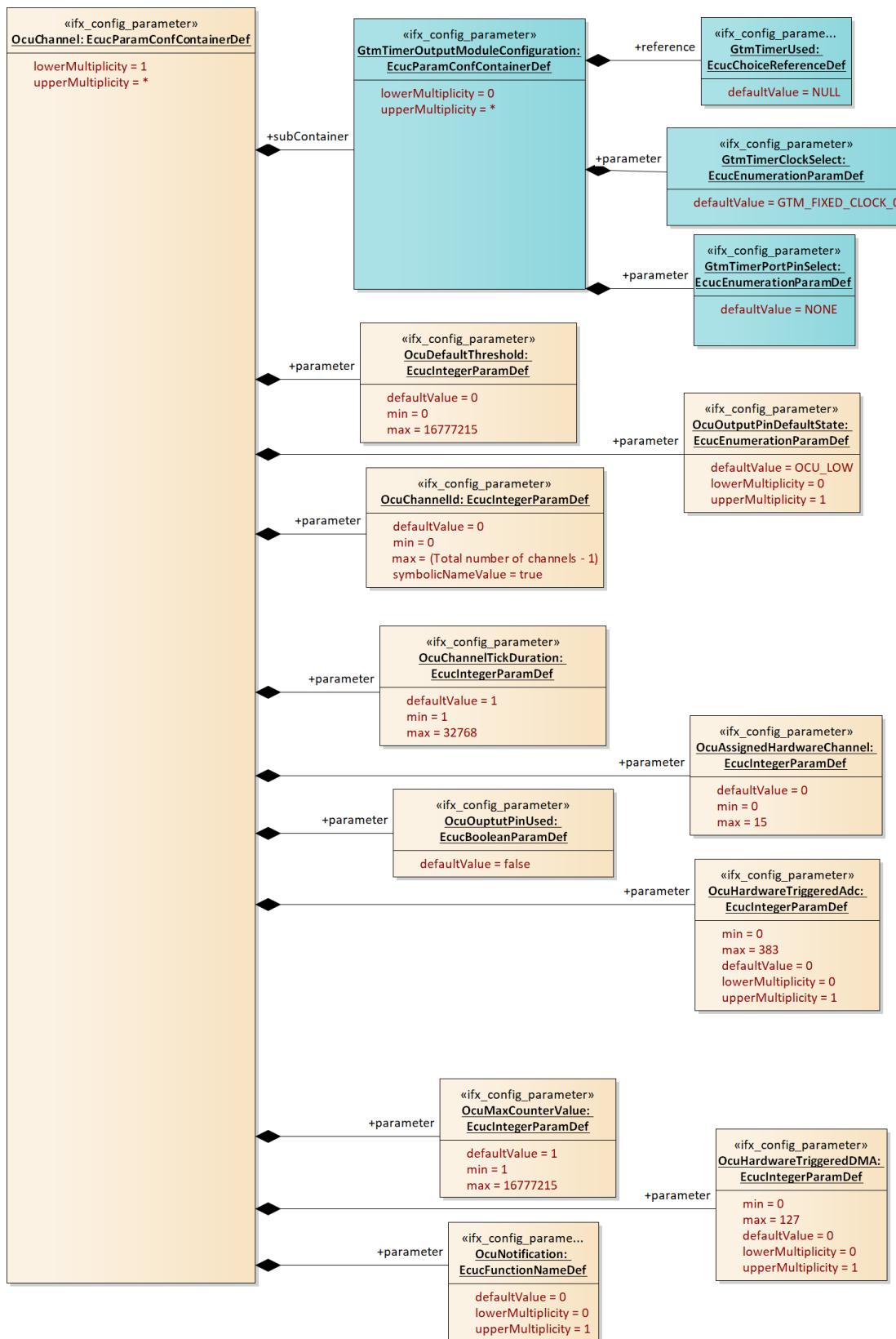


Figure 163 Container hierarchy along with their configuration parameters

OCU driver

18.3.1.1 Container: GtmTimerOutputModuleConfiguration

This container contains the parameters for configuring the selected TOM/ATOM channel.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

18.3.1.1.1 GtmTimerClockSelect

Table 1546 Specification for GtmTimerClockSelect

Name	GtmTimerClockSelect		
Description	This parameter decides the Clock Source for TOM/ATOM timer.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_CONFIGURABLE_CLOCK_0: Configurable Clock 0 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_1: Configurable Clock 1 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_2: Configurable Clock 2 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_3: Configurable Clock 3 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_4: Configurable Clock 4 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_5: Configurable Clock 5 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_6: Configurable Clock 6 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_7: Configurable Clock 7 is selected for ATOM channel. GTM_FIXED_CLOCK_0: Fixed Clock 0 is selected for TOM channel. GTM_FIXED_CLOCK_1: Fixed Clock 1 is selected for TOM channel. GTM_FIXED_CLOCK_2: Fixed Clock 2 is selected for TOM channel. GTM_FIXED_CLOCK_3: Fixed Clock 3 is selected for TOM channel. GTM_TBU_TS0: TBU_TS0 is selected for ATOM channel. GTM_TBU_TS1: TBU_TS1 is selected for ATOM channel. GTM_TBU_TS2: TBU_TS2 is selected for ATOM channel.		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerUsed		

18.3.1.1.2 GtmTimerPortPinSelect

Table 1547 Specification for GtmTimerPortPinSelect

Name	GtmTimerPortPinSelect
Description	This parameter decides the output port and pin for the GTM timer.

OCU driver**Table 1547 Specification for GtmTimerPortPinSelect (continued)**

	Range: Available port pins depends on target device and selected timer unit. refer to target specification for more information.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NONE: No port pin is selected. TOUT<w>_SEL<x>_PORT<y>_PIN<z>; w: Tout Number x: Alternate selection (A-L) y:Port Number z:Pin Number		
Default value	NONE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerClockSelect, GtmTimerUsed		

18.3.1.1.3 GtmTimerUsed**Table 1548 Specification for GtmTimerUsed**

Name	GtmTimerUsed		
Description	The TOM/ATOM Channel resource assigned to the Ocu channel. The HW resource used should be unique in a configuration set of OCU. This parameter is list of all the GTM timer channels (TOM/ATOM) used by OCU Driver. Referred timer channel in MCU should have TomChannelUsage/ AtomChannelUsage as USED_BY_OCU_DRIVER		
Multiplicity	1..1	Type	EcucChoiceReferenceDef
Range	Reference to Node: McuGtmTomChannelAllocationConf, McuGtmAtomChannelAllocationConf		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

OCU driver
Table 1548 Specification for GtmTimerUsed (continued)

Dependency	-
-------------------	---

18.3.1.2 Container: Ocu

Configuration of Ocu (Output Compare Unit) module

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

18.3.1.2.1 Config Variant

Table 1549 Specification for Config Variant

Name	Config Variant		
Description	Selects the config-variant for the OCU module Note: Default value is set to post build as OCU support to post build value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VariantPostBuild: Post-build variant supported.		
Default value	VariantPostBuild		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

18.3.1.3 Container: OcuChannel

Configuration of an individual OCU channel.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

18.3.1.3.1 OcuAssignedHardwareChannel

Table 1550 Specification for OcuAssignedHardwareChannel

Name	OcuAssignedHardwareChannel		
Description	The physical hardware channel that is assigned to this logical channel. NOTE: This parameter is non-editable and not used. The hardware channel is configured through "GtmTimerUsed" parameter.		
Multiplicity	1..1	Type	EcucIntegerParamDef

OCU driver**Table 1550 Specification for OcuAssignedHardwareChannel (continued)**

Range	0 - 15		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.3.2 OcuChannelId**Table 1551 Specification for OcuChannelId**

Name	OcuChannelId		
Description	Channel Id of the OCU channel. This value will be assigned to the symbolic name derived from the OcuChannel container short name. It defines the assignment of the channel to the physical OCU hardware channel. The value of the parameter should be unique in a configuration set. NOTE: Minimum value is set as default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - (Total number of channels - 1)		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.3.3 OcuChannelTickDuration**Table 1552 Specification for OcuChannelTickDuration**

Name	OcuChannelTickDuration		
Description	Specifies the tick duration of the counter of the channel. This parameter is the number of the input clock edges (rising edges or falling edges exclusively) counted each time to increase the counter by one unit. Note: This parameter is not-used,not editable and default value is set to 1.		
Multiplicity	1..1	Type	EcuIntegerParamDef

OCU driver**Table 1552 Specification for OcuChannelTickDuration (continued)**

Range	1 - 32768		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.3.4 OcuDefaultThreshold**Table 1553 Specification for OcuDefaultThreshold**

Name	OcuDefaultThreshold		
Description	Value of comparison threshold used for Initialization.(in ticks) The value should be less than the maximum counter value. For ATOM channel, OcuDefaultThreshold scaled up by amount of ticks should be in 24-bit range. For TOM channel, OcuDefaultThreshold scaled up by amount of ticks should be in 16-bit range. NOTE: Minimum value is set as default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	GtmTimerUsed, OcuChannelTickDuration, OcuMaxCounterValue		

18.3.1.3.5 OcuHardwareTriggeredAdc**Table 1554 Specification for OcuHardwareTriggeredAdc**

Name	OcuHardwareTriggeredAdc
Description	This parameter is used to allow the OCU channel to trigger an ADC channel upon compare match. The value of the parameter represents the ADC Group to trigger. NOTE: Minimum value is set as default value.

OCU driver**Table 1554 Specification for OcuHardwareTriggeredAdc (continued)**

Multiplicity	0..1	Type	EcucIntegerParamDef
Range	0 - 383		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.3.6 OcuHardwareTriggeredDMA**Table 1555 Specification for OcuHardwareTriggeredDMA**

Name	OcuHardwareTriggeredDMA		
Description	This parameter is used to allow the OCU channel to trigger a DMA channel upon compare match. The value of the parameter represents the DMA physical channel to trigger. NOTE: Minimum value is set as default value.		
Multiplicity	0..1	Type	EcucIntegerParamDef
Range	0 - 127		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	OcuNotification		

18.3.1.3.7 OcuMaxCounterValue**Table 1556 Specification for OcuMaxCounterValue**

Name	OcuMaxCounterValue
Description	Maximum value in ticks, the counter of the OCU channel is able to count. For ATOM channel, OcuDefaultThreshold scaled up by amount of ticks should be in 24-bit range. For TOM channel, OcuDefaultThreshold scaled up by amount of ticks should be in 16-bit range.

OCU driver**Table 1556 Specification for OcuMaxCounterValue (continued)**

	NOTE: Minimum value is set as default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	GtmTimerUsed, OcuChannelTickDuration		

18.3.1.3.8 OcuNotification**Table 1557 Specification for OcuNotification**

Name	OcuNotification		
Description	<p>The parameter is used by the OCU driver to invoke the user-defined function if the Compare match is detected. The parameter can be configured as a name or an address(numeric value) of the notification function.</p> <p>Note1: By default, the notification parameter will be NULL , to remove dependency from user defined functions.</p> <p>Note2: The OCU driver does not validate the configured function name or address for correctness and hence the responsibility falls on the user.</p>		
Multiplicity	0..1	Type	EcuFunctionNameDef
Range	String		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	OcuHardwareTriggeredDMA		

18.3.1.3.9 OcuOutputPinUsed**Table 1558 Specification for OcuOutputPinUsed**

Name	OcuOutputPinUsed
-------------	------------------

OCU driver**Table 1558 Specification for OcuOutputPinUsed (continued)**

Description	Information about the usage of an output pin on this channel. True: the channel uses an output pin. False: the channel does not use an output pin. NOTE: As the default HW used by the channel is set to TOM, and TOM cannot support any pin functionality, the default value is set to false.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	GtmTimerClockSelect		

18.3.1.3.10 OcuOutputPinDefaultState**Table 1559 Specification for OcuOutputPinDefaultState**

Name	OcuOutputPinDefaultState		
Description	The parameter OcuOutputPinDefaultState represents the state that a pin associated with a channel shall be set to after initialization. NOTE: OCU_LOW is set as default value as it represents the minimum numeric value.		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	OCU_HIGH: The OCU channel output pin will be set to high (3 or 5 V) when requested. OCU_LOW: The OCU channel output pin will be set to low (0V) when requested.		
Default value	OCU_LOW		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	OcuOutputPinUsed		

OCU driver

18.3.1.4 Container: OcuConfigSet

This container is the base of a Configuration Set, which contains the configured OCU channels. This way, different configuration sets can be defined for post-build process.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

18.3.1.4.1 OcuCountdirection

Table 1560 Specification for OcuCountdirection

Name	OcuCountdirection		
Description	This parameter indicates the count direction for the whole OCU driver. The parameter is non-editable and always configured as "OCU_UPCOUNTING".		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	OCU_UPCOUNTING: The OCU counter will reckon from the minimum to the maximum value.		
Default value	OCU_UPCOUNTING		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.5 Container: OcuConfigurationOfOptionalApis

Configuration of optional APIs NOTE: By default all the optional API's except InitCheck will not be available to optimise the executable size. InitCheck will be available as Safety error reporting is enabled by default.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

18.3.1.5.1 OcuDeInitApi

Table 1561 Specification for OcuDeInitApi

Name	OcuDeInitApi		
Description	Adds / removes the service Ocu_DeInit() from the code.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

OCU driver**Table 1561 Specification for OcuDeInitApi (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.5.2 OcuGetCounterApi**Table 1562 Specification for OcuGetCounterApi**

Name	OcuGetCounterApi		
Description	Adds / removes the service Ocu_GetCounter() from the code.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.5.3 OcuInitCheckApi**Table 1563 Specification for OcuInitCheckApi**

Name	OcuInitCheckApi		
Description	Adds / removes the service Ocu_InitCheck() from the code.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

OCU driver**Table 1563 Specification for OcuInitCheckApi (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

18.3.1.5.4 OcuNotificationSupported**Table 1564 Specification for OcuNotificationSupported**

Name	OcuNotificationSupported		
Description	Adds / removes the services Ocu_EnableNotification() and Ocu_DisableNotification() from the code.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.5.5 OcuSetAbsoluteThresholdApi**Table 1565 Specification for OcuSetAbsoluteThresholdApi**

Name	OcuSetAbsoluteThresholdApi		
Description	Adds / removes the service Ocu_SetAbsoluteThreshold() from the code.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

OCU driver**Table 1565 Specification for OcuSetAbsoluteThresholdApi (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.5.6 OcuSetPinActionApi**Table 1566 Specification for OcuSetPinActionApi**

Name	OcuSetPinActionApi		
Description	Adds / removes the service Ocu_SetPinAction() from the code.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.5.7 OcuSetPinStateApi**Table 1567 Specification for OcuSetPinStateApi**

Name	OcuSetPinStateApi		
Description	Adds / removes the service Ocu_SetPinState() from the code.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

OCU driver**Table 1567 Specification for OcuSetPinStateApi (continued)**

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.5.8 OcuSetRelativeThresholdApi**Table 1568 Specification for OcuSetRelativeThresholdApi**

Name	OcuSetRelativeThresholdApi		
Description	Adds / removes the service Ocu_SetRelativeThreshold() from the code.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.5.9 OcuVersionInfoApi**Table 1569 Specification for OcuVersionInfoApi**

Name	OcuVersionInfoApi		
Description	Switch to indicate that the Ocu_GetVersionInfo() is supported.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

OCU driver

18.3.1.6 Container: OcuGeneral

This container contains the module-wide configuration parameters of the OCU Driver. Note: By default all the error reporting (Development, Safety and Multi-core) are enabled, to ensure proper driver functionality.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

18.3.1.6.1 OcuDevErrorDetect

Table 1570 Specification for OcuDevErrorDetect

Name	OcuDevErrorDetect		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. true: enabled (ON) false: disabled (OFF)		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.6.2 OcuMultiCoreErrorDetect

Table 1571 Specification for OcuMultiCoreErrorDetect

Name	OcuMultiCoreErrorDetect		
Description	The parameter enables or disables the multi core related default error tracer (DET) detection and reporting. It is applicable only when DETs are enabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

OCU driver**Table 1571 Specification for OcuMultiCoreErrorDetect (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	OcuDevErrorDetect		

18.3.1.6.3 OcuSafetyEnable**Table 1572 Specification for OcuSafetyEnable**

Name	OcuSafetyEnable		
Description	Pre-processor switch for enabling the safety features of OCU driver.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

18.3.1.7 Container: OcuGroup

This container contains the parameters for configuring an OCU group. NOTE: The container is not supported. But the parameter will be maintained nonetheless to maintain AUTOSAR compatibility.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

18.3.1.7.1 OcuGroupDefinition**Table 1573 Specification for OcuGroupDefinition**

Name	OcuGroupDefinition		
Description	Assignment of OcuChannels to an OcuGroup.		
Multiplicity	1..*	Type	EcucReferenceDef
Range	Reference to Node: OcuChannel		
Default value	NULL		

OCU driver**Table 1573 Specification for OcuGroupDefinition (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.7.2 OcuGroupId**Table 1574 Specification for OcuGroupId**

Name	OcuGroupId		
Description	Numeric ID of the group. This parameter is the symbolic name of the group.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.8 Container: OcuHWSpecificSettings

This container contains Ocu-specific parameters for selecting the clock source and setting optional prescalers.
 NOTE: The container is not supported. But the parameter will be maintained nonetheless to maintain AUTOSAR compatibility.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

18.3.1.8.1 OcuClockSource**Table 1575 Specification for OcuClockSource**

Name	OcuClockSource
Description	The OCU driver specific clock input for the unit can statically be configured to select different clock sources. NOTE: The parameter is not supported as the clock source selection will be configured using MCU. But the parameter will be maintained nonetheless to maintain AUTOSAR compatibility.

OCU driver**Table 1575 Specification for OcuClockSource (continued)**

Multiplicity	0..1	Type	EcucEnumerationParamDef
Range			
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.8.2 OcuPrescale**Table 1576 Specification for OcuPrescale**

Name	OcuPrescale		
Description	Optional OCU driver specific clock prescale factor. NOTE: The parameter is not supported as the clock prescale factor will be configured using MCU. But the parameter will be maintained nonetheless to maintain AUTOSAR compatibility.		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range			
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.9 Container: CommonPublishedInformation

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

OCU driver

18.3.1.9.1 ArMajorVersion

Table 1577 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	AUTOSAR major version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.9.2 ArMinorVersion

Table 1578 Specification for ArMinorVersion

Name	ArMinorVersion		
Description	AUTOSAR minor version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.9.3 ArPatchVersion

Table 1579 Specification for ArPatchVersion

Name	ArPatchVersion		
Description	AUTOSAR patch version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		

OCU driver**Table 1579 Specification for ArPatchVersion (continued)**

Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.9.4 ModuleId**Table 1580 Specification for ModuleId**

Name	ModuleId		
Description	Parameter to provide the module identifier.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	125		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.9.5 Release**Table 1581 Specification for Release**

Name	Release		
Description	Aurix derivative used for the implementation.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per the configuration.		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

OCU driver**Table 1581 Specification for Release (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

18.3.1.9.6 SwMajorVersion**Table 1582 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Module major version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.9.7 SwMinorVersion**Table 1583 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Module minor version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

OCU driver

18.3.1.9.8 SwPatchVersion

Table 1584 Specification for SwPatchVersion

Name	SwPatchVersion		
Description	Module patch version.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.1.9.9 Vendor ID

Table 1585 Specification for Vendor ID

Name	Vendor ID		
Description			
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

18.3.2 Functions - Type definitions

18.3.2.1 Ocu_ChannelType

Table 1586 Specification for Ocu_ChannelType

Syntax	Ocu_ChannelType
Type	uint8

OCU driver**Table 1586 Specification for Ocu_ChannelType (continued)**

File	Ocu.h	
Range	0-[{maximum TOM channels}+{maximum ATOM channels}]	Maximum value depends on the device.
Description	Numeric identifier of an OCU channel. As the maximum number for HW channels available is 192(96 TOM and 96 ATOM) the data type chosen is 8-bit (maximum possible of 255).	
Source	AUTOSAR	

18.3.2.2 Ocu_ValueType**Table 1587 Specification for Ocu_ValueType**

Syntax	Ocu_ValueType	
Type	uint32	
File	Ocu.h	
Range	0-16777215	To support both TOM and ATOM, data type is chosen as a 32-bit variable.
Description	Type for reading the counter and writing the threshold values (in number of ticks). ATOM channels use 24-bit wide compare registers are TOM channels use 16-bit wide compare registers. Hence to support both 32-bit data type is chosen.	
Source	AUTOSAR	

18.3.2.3 Ocu_PinStateType**Table 1588 Specification for Ocu_PinStateType**

Syntax	Ocu_PinStateType	
Type	Enumeration	
File	Ocu.h	
Range	0 - OCU_LOW	The pin associated to an OCU channel is in low state.
	1 - OCU_HIGH	The pin associated to an OCU channel is in high state.
Description	Output state of the pin linked to an OCU channel.	
Source	AUTOSAR	

18.3.2.4 Ocu_PinActionType**Table 1589 Specification for Ocu_PinActionType**

Syntax	Ocu_PinActionType
---------------	-------------------

OCU driver**Table 1589 Specification for Ocu_PinActionType (continued)**

Type	Enumeration	
File	Ocu.h	
Range	0 - OCU_DISABLE	The channel pin will remain at its current level upon compare match.
	1 - OCU_SET_HIGH	The channel pin will be set HIGH upon compare match.
	2 - OCU_SET_LOW	The channel pin will be set LOW upon compare match.
	3 - OCU_TOGGLE	The channel pin will be set to the opposite of its current level upon compare match.
Description	Automatic action (by hardware) to be performed on a pin attached to an OCU channel.	
Source	AUTOSAR	

18.3.2.5 Ocu_ConfigType**Table 1590 Specification for Ocu_ConfigType**

Syntax	Ocu_ConfigType	
Type	Structure	
File	Ocu.h	
Range	Hardware dependent[]	The contents of the initialization data structure are hardware specific.
Description	Defines the type of data structure containing the set of configuration parameters required for initializing the OCU driver.	
Source	IFX	

18.3.2.6 Ocu_ReturnType**Table 1591 Specification for Ocu_ReturnType**

Syntax	Ocu_ReturnType	
Type	Enumeration	
File	Ocu.h	
Range	0 - OCU_CM_OUT_REF_INTERVAL	The compare match will not occur inside the current Reference Interval.
	1 - OCU_CM_IN_REF_INTERVAL	The compare match will occur inside the current Reference Interval.
Description	Return information after setting a new threshold value.	
Source	AUTOSAR	

OCU driver**18.3.2.7 Ocu_NotifiPtrType****Table 1592 Specification for Ocu_NotifiPtrType**

Syntax	Ocu_NotifiPtrType
Type	Pointer to a function of type void Function_Name (void)
File	Ocu.h
Description	Channel notification function pointer
Source	IFX

18.3.3 Functions - APIs**18.3.3.1 Ocu_Init****Table 1593 Specification for Ocu_Init API**

Syntax	<pre>void Ocu_Init (const Ocu_ConfigType * const ConfigPtr)</pre>	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The purpose of the API is to initialize all relevant registers configured hardware (AssignedHWUnit) with the values of structure referenced by the parameter ConfigPtr. The API will also disable all notifications and the OCU channel status will be set to OCU_STOPPED.</p> <p>For multicore, the function will initialize those channels allocated to the core in which this function is invoked. Additionally for master core, the function will initialize the resources which are shared among cores.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>OCU_E_ALREADY_INITIALIZED: Error code is reported when OCU driver is already initialized and Ocu_Init() is called.</p> <p>OCU_E_INIT_FAILED: Error code is reported when OCU initialization has failed. Example, selected configuration set does not exist.</p>	

OCU driver**Table 1593 Specification for Ocu_Init API (continued)**

	<p>OCU_E_CORE_NOT_CONFIGURED: Error code is reported when OCU module is not configured for the core in which an API is invoked.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

18.3.3.2 Ocu_DeInit**Table 1594 Specification for Ocu_DeInit API**

Syntax	<pre>void Ocu_DeInit (void)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The purpose of this API is to de-initialize the OCU driver. The used peripherals/registers will be set to power-on reset state. The API will disable all used interrupts and notifications.</p> <p>For multicore, the function will de-initialize those channels allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization.</p> <p>OCU_E_PARAM_INVALID_STATE: Error code is reported when API Ocu_SetPinState() called with an invalid pin state or when the channel is in the RUNNING state. Also in case, API Ocu_DeInit called when at least one of the OCU channel is in RUNNING state.</p> <p>Runtime Errors: None</p>	

OCU driver**Table 1594 Specification for Ocu_DeInit API (continued)**

	DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	OcuDeInitApi
User hints	-

18.3.3.3 Ocu_EnableNotification**Table 1595 Specification for Ocu_EnableNotification API**

Syntax	void Ocu_EnableNotification (const Ocu_ChannelType ChannelNumber)	
Service ID	0x0B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to enable notifications from an OCU channel. For multicore, the OCU channel shall be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	DET: OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization. OCU_E_PARAM_INVALID_CHANNEL: Error code is reported when OCU API used with an invalid channel identifier. OCU_E_NO_VALID_NOTIF: Error code is reported on usage of Ocu_DisableNotification() or Ocu_EnableNotification() on a channel where a NULL pointer is configured as the notification function. OCU_E_CORE_CHANNEL_MISMATCH: Error code is reported when an API is called with the channel not allocated to executing core. Runtime Errors: None DEM: None	

OCU driver**Table 1595 Specification for Ocu_EnableNotification API (continued)**

	Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	OcuNotificationSupported
User hints	None

18.3.3.4 Ocu_DisableNotification**Table 1596 Specification for Ocu_DisableNotification API**

Syntax	void Ocu_DisableNotification (const Ocu_ChannelType ChannelNumber)	
Service ID	0x0A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to disable notifications from an OCU channel. For multicore, the OCU channel should be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	DET: OCU_E_PARAM_INVALID_CHANNEL: Error code is reported when OCU API used with an invalid channel identifier. OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization. OCU_E_NO_VALID_NOTIF: Error code is reported on usage of Ocu_DisableNotification() or Ocu_EnableNotification() on a channel where a NULL pointer is configured as the notification function. OCU_E_CORE_CHANNEL_MISMATCH: Error code is reported when an API is called with the channel not allocated to executing core. Runtime Errors: None DEM: None	

OCU driver**Table 1596 Specification for Ocu_DisableNotification API (continued)**

	Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	OcuNotificationSupported
User hints	None

18.3.3.5 Ocu_GetCounter**Table 1597 Specification for Ocu_GetCounter API**

Syntax	Ocu_ValueType Ocu_GetCounter (const Ocu_ChannelType ChannelNumber)	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Ocu_ValueType	Current counter value
Description	The purpose of the API is to read the current value of the counter. For multicore, the OCU channel shall be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	DET: OCU_E_PARAM_INVALID_CHANNEL: Error code is reported when OCU API used with an invalid channel identifier. OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization. OCU_E_CORE_CHANNEL_MISMATCH: Error code is reported when an API is called with the channel not allocated to executing core. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	

OCU driver**Table 1597 Specification for Ocu_GetCounter API (continued)**

Configuration dependencies	OcuGetCounterApi
User hints	None

18.3.3.6 Ocu_SetAbsoluteThreshold**Table 1598 Specification for Ocu_SetAbsoluteThreshold API**

Syntax	<pre>Ocu_ReturnType Ocu_SetAbsoluteThreshold (const Ocu_ChannelType ChannelNumber, const Ocu_ValueType ReferenceValue, const Ocu_ValueType AbsoluteValue)</pre>	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber ReferenceValue AbsoluteValue	Numeric identifier of the OCU channel Value given by the upper layer and used as a base to determine whether, writing the threshold value to the compare register was within or outside the reference Interval. Value to compare with the content of the counter. This value is in ticks.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Ocu_ReturnType	To indicate whether, writing the threshold value to the compare register was within or outside the reference Interval.
Description	<p>The purpose of the API is to set the value of the channel threshold using an absolute input data.</p> <p>For multicore, the OCU channel should be allocated to the core in which the function is invoked.</p> <p>Decision: Ocu_SetAbsoluteThreshold API will return OCU_CM_OUT_REF_INTERVAL if a DET/Safety Error is reported.</p> <p>Rationale : As the enumeration value of OCU_CM_OUT_REF_INTERVAL is numerically minimum (ZERO), it is chosen as a default return value if an error is identified.</p>	
Source	AUTOSAR	
Error handling	DET: OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization.	

OCU driver**Table 1598 Specification for Ocu_SetAbsoluteThreshold API (continued)**

	<p>OCU_E_PARAM_INVALID_CHANNEL: Error code is reported when OCU API used with an invalid channel identifier.</p> <p>OCU_E_CORE_CHANNEL_MISMATCH: Error code is reported when an API is called with the channel not allocated to executing core.</p> <p>OCU_E_PARAM_REF_VALUE: If the reference value is greater than the maximum threshold value then function Ocu_SetAbsoluteThreshold shall raise the error "OCU_E_PARAM_REF_VALUE".</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>OCU_E_PARAM_COMPARE_VALUE: "OCU_E_PARAM_COMPARE_VALUE" error is reported to indicate invalid input compare values.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	OcuSetAbsoluteThresholdApi
User hints	None

18.3.3.7 Ocu_SetPinAction**Table 1599 Specification for Ocu_SetPinAction API**

Syntax	<pre>void Ocu_SetPinAction (const Ocu_ChannelType ChannelNumber, const Ocu_PinActionType PinAction)</pre>	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber PinAction	Numeric identifier of the OCU OCU_SET_LOW, OCU_SET_HIGH, OCU_TOGGLE, OCU_DISABLE
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The purpose of the API is to indicate to the driver the action to be taken by the hardware upon compare match. The action will be disable, high, low or toggle.</p> <p>For multicore, the OCU channel should be allocated to the core in which the function is invoked.</p>	

OCU driver**Table 1599 Specification for Ocu_SetPinAction API (continued)**

Source	AUTOSAR
Error handling	<p>DET:</p> <p>OCU_E_PARAM_NO_PIN: Error code is reported when Ocu_SetPinState() or Ocu_SetPinAction() called for a channel that does not have an associated output pin.</p> <p>OCU_E_PARAM_INVALID_ACTION: Error code is reported when API Ocu_SetPinAction() called with an invalid pin action.</p> <p>OCU_E_PARAM_INVALID_CHANNEL: Error code is reported when OCU API used with an invalid channel identifier.</p> <p>OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization.</p> <p>OCU_E_CORE_CHANNEL_MISMATCH: Error code is reported when an API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	OcuSetPinActionApi
User hints	None

18.3.3.8 Ocu_SetPinState**Table 1600 Specification for Ocu_SetPinState API**

Syntax	<pre>void Ocu_SetPinState (const Ocu_ChannelType ChannelNumber, const Ocu_PinStateType PinState)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber PinState	Numeric identifier of the OCU State of the pin to set. OCU_HIGH or OCU_LOW
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

OCU driver**Table 1600 Specification for `Ocu_SetPinState` API (continued)**

Description	The purpose of the API is to set the level of the pin associated to an OCU channel to high or low. For multicore, the OCU channel should be allocated to the core in which the function is invoked.
Source	AUTOSAR
Error handling	<p>DET:</p> <p><code>OCU_E_PARAM_NO_PIN</code>: Error code is reported when <code>Ocu_SetPinState()</code> or <code>Ocu_SetPinAction()</code> called for a channel that does not have an associated output pin.</p> <p><code>OCU_E_PARAM_INVALID_CHANNEL</code>: Error code is reported when OCU API used with an invalid channel identifier.</p> <p><code>OCU_E_UNINIT</code>: Error code is reported when any OCU API services other than <code>Ocu_GetVersionInfo()</code> and <code>Ocu_Init()</code> are used without module initialization.</p> <p><code>OCU_E_PARAM_INVALID_STATE</code>: Error code is reported when API <code>Ocu_SetPinState()</code> called with an invalid pin state or when the channel is in the RUNNING state. Also in case, API <code>Ocu_DeInit</code> called when at least one of the OCU channel is in RUNNING state.</p> <p><code>OCU_E_CORE_CHANNEL_MISMATCH</code>: Error code is reported when an API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	<code>OcuSetPinStateApi</code>
User hints	None

18.3.3.9 `Ocu_SetRelativeThreshold`**Table 1601 Specification for `Ocu_SetRelativeThreshold` API**

Syntax	<pre>Ocu_ReturnType Ocu_SetRelativeThreshold (const Ocu_ChannelType ChannelNumber, const Ocu_ValueType RelativeValue)</pre>	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber RelativeValue	Numeric identifier of the OCU channel Value to use for computing the new threshold.

OCU driver**Table 1601 Specification for Ocu_SetRelativeThreshold API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Ocu_ReturnType	To indicate whether, writing the threshold value to the compare register was within or outside the reference Interval.
Description	<p>The purpose of the API is to set the value of the channel threshold to the relative current value of the counter.</p> <p>For multicore, the OCU channel should be allocated to the core in which the function is invoked.</p> <p>Decision: Ocu_SetRelativeThreshold API will return OCU_CM_OUT_REF_INTERVAL if a DET/Safety Error is reported.</p> <p>Rationale : As the enumeration value of OCU_CM_OUT_REF_INTERVAL is numerically minimum (ZERO), it is chosen as a default return value if an error is identified.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>OCU_E_PARAM_INVALID_CHANNEL: Error code is reported when OCU API used with an invalid channel identifier.</p> <p>OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization.</p> <p>OCU_E_CORE_CHANNEL_MISMATCH: Error code is reported when an API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>OCU_E_PARAM_COMPARE_VALUE: "OCU_E_PARAM_COMPARE_VALUE" error is reported to indicate invalid input compare values.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	OcuSetRelativeThresholdApi	
User hints	None	

18.3.3.10 Ocu_StartChannel**Table 1602 Specification for Ocu_StartChannel API**

Syntax	void Ocu_StartChannel (const Ocu_ChannelType ChannelNumber)
Service ID	0x02

OCU driver**Table 1602 Specification for Ocu_StartChannel API (continued)**

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The purpose of the API is to start the OCU channel.</p> <p>For multicore, the OCU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>OCU_E_BUSY: Error code is reported when API Ocu_StartChannel() called on a channel that is in state RUNNING.</p> <p>OCU_E_PARAM_INVALID_CHANNEL: Error code is reported when OCU API used with an invalid channel identifier.</p> <p>OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization.</p> <p>OCU_E_CORE_CHANNEL_MISMATCH: Error code is reported when an API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

18.3.3.11 Ocu_StopChannel**Table 1603 Specification for Ocu_StopChannel API**

Syntax	void Ocu_StopChannel (const Ocu_ChannelType ChannelNumber)
Service ID	0x03
Sync/Async	Synchronous

OCU driver**Table 1603 Specification for Ocu_StopChannel API (continued)**

ASIL Level	B	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The purpose of the API is to stop the OCU channel.</p> <p>For multicore, the OCU channel should be allocated to the core in which the function is invoked.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>OCU_E_PARAM_INVALID_CHANNEL: Error code is reported when OCU API used with an invalid channel identifier.</p> <p>OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization.</p> <p>OCU_E_CORE_CHANNEL_MISMATCH: Error code is reported when an API is called with the channel not allocated to executing core.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

18.3.3.12 Ocu_InitCheck**Table 1604 Specification for Ocu_InitCheck API**

Syntax	Std_ReturnType Ocu_InitCheck (const Ocu_ConfigType * const ConfigPtr)
Service ID	0x0C
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant

OCU driver**Table 1604 Specification for Ocu_InitCheck API (continued)**

Parameters (in)	ConfigPtr	Pointer to the configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK - if initialization comparison is success. E_NOT_OK - if initialization comparison fails.
Description	<p>The purpose of the API is to check the values set during the OCU driver initialization are as per the config structure. The API does not modify any SFR/variable and only a read operation is performed.</p> <p>The API is called after Ocu_Init() is done to check for the correctness of initialization. In case any failure is observed in comparison, the API returns E_NOT_OK.</p>	
Source	IFX	
Error handling	<p>DET: None Runtime Errors: None DEM: None Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	OcuInitCheckApi	
User hints	-	

18.3.3.13 Ocu_GetVersionInfo**Table 1605 Specification for Ocu_GetVersionInfo API**

Syntax	<pre>void Ocu_GetVersionInfo (Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to store the version information of the module
Parameters (in - out)	-	-

OCU driver**Table 1605 Specification for Ocu_GetVersionInfo API (continued)**

Return	void	-
Description	The purpose of the API is to return the version information of the OCU driver. The version information includes: Module ID, Vendor ID., vendor specific version numbers. This function is available only if the OCU_VERSION_INFO_API is ON.	
Source	AUTOSAR	
Error handling	DET: OCU_E_PARAM_POINTER: Error code is reported when a NULL pointer is passed an input parameter. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	OcuVersionInfoApi	
User hints	The API can be called before OCU initialization.	

18.3.4 Notifications and Callbacks**18.3.4.1 Ocu_Timer_Isr****Table 1606 Specification for Ocu_Timer_Isr API**

Syntax	<pre>void Ocu_Timer_Isr (const Ocu_ChannelType Channel, const uint32 Flags)</pre>	
Service ID	0x20	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	Channel	Logical channel identifier.
	Flags	Interrupt flags responsible for ISR
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Callback function from MCU to service timer (GTM-TOM and GTM-ATOM) interrupts.	
Source	IFX	

OCU driver

Table 1606 Specification for `Ocu_Timer_Isr` API (continued)

Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p><code>OCU_E_INVALID_ISR_UNINIT</code>: If the ISR is invoked before <code>OCU_Init</code> then Ocu driver shall report a safety error "<code>OCU_E_INVALID_ISR_UNINIT</code>".</p> <p><code>OCU_E_INVALID_ISR_INACTIVE_CHANNEL</code>: If the ISR is invoked when the channel is not active then Ocu driver shall report a safety error "<code>OCU_E_INVALID_ISR_INACTIVE_CHANNEL</code>"</p> <p><code>OCU_E_INVALID_ISR_COMP_INVALID</code>: The Ocu driver shall report a safety error, if ISR is invoked with wrong flags (Compare match notification flag) indicating unexpected compare match.</p> <p><code>OCU_E_INVALID_ISR_CHANNEL_INVALID</code>: The Ocu driver shall report Safety error "<code>OCU_E_INVALID_ISR_CHANNEL_INVALID</code>", if the ISR is invoked when the passed channel ID is not configured.</p> <p><code>OCU_E_INVALID_ISR_CHANNEL_CORE_MISMATCH</code>: The Ocu driver shall report DET/Safety error "<code>OCU_E_INVALID_ISR_CHANNEL_CORE_MISMATCH</code>", if the ISR is invoked when the passed channel and core id mismatch.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	None

18.3.5 Scheduled functions

The OCU driver does not provide any scheduled functions.

18.3.6 Interrupt service routines

Not applicable for the OCU driver.

18.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

18.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Table 1607 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
Error code is reported when any OCU API services other than <code>Ocu_GetVersionInfo()</code> and <code>Ocu_Init()</code> are used	AUTOSAR	<code>OCU_E_UNINIT=0x02</code>	<code>Ocu_GetCounter</code> , <code>Ocu_SetAbsoluteThreshold</code> , <code>Ocu_EnableNotification</code> , <code>Ocu_DisableNotification</code> ,

OCU driver**Table 1607 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
without module initialization.			Ocu_StopChannel, Ocu_SetRelativeThreshold, Ocu_StartChannel, Ocu_DelInit, Ocu_SetPinAction, Ocu_SetPinState
Error code is reported when OCU API used with an invalid channel identifier.	AUTOSAR	OCU_E_PARAM_INVALID_CHANNEL=0x03	Ocu_GetCounter, Ocu_SetAbsoluteThreshold, Ocu_EnableNotification, Ocu_DisableNotification, Ocu_StopChannel, Ocu_SetRelativeThreshold, Ocu_StartChannel, Ocu_SetPinAction, Ocu_SetPinState
Error code is reported when API Ocu_SetPinState() called with an invalid pin state or when the channel is in the RUNNING state. Also in case, API Ocu_DelInit called when at least one of the OCU channel is in RUNNING state.	AUTOSAR	OCU_E_PARAM_INVALID_STATE=0x04	Ocu_DelInit, Ocu_SetPinState
Error code is reported when API Ocu_SetPinAction() called with an invalid pin action.	AUTOSAR	OCU_E_PARAM_INVALID_ACTION=0x05	Ocu_SetPinAction
Error code is reported on usage of Ocu_DisableNotification() or Ocu_EnableNotification() on a channel where a NULL pointer is configured as the notification function.	AUTOSAR	OCU_E_NO_VALID_NOTIF=0x06	Ocu_EnableNotification, Ocu_DisableNotification
Error code is reported when OCU driver is already initialized and Ocu_Init() is called.	AUTOSAR	OCU_E_ALREADY_INITIALIZED=0x07	Ocu_Init
Error code is reported when OCU initialization has failed. Example,	AUTOSAR	OCU_E_INIT_FAILED=0x0B	Ocu_Init

OCU driver**Table 1607 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
selected configuration set does not exist.			
Error code is reported when API Ocu_StartChannel() called on a channel that is in state RUNNING.	AUTOSAR	OCU_E_BUSY=0x09	Ocu_StartChannel
Error code is reported when a NULL pointer is passed an input parameter.	AUTOSAR	OCU_E_PARAM_POINTER=0x08	Ocu_GetVersionInfo
Error code is reported when Ocu_SetPinState() or Ocu_SetPinAction() called for a channel that does not have an associated output pin.	AUTOSAR	OCU_E_PARAM_NO_PIN=0x0A	Ocu_SetPinAction, Ocu_SetPinState
Error code is reported when an API is called with the channel not allocated to executing core.	IFX	OCU_E_CORE_CHANNEL_MISMATCH =0x65	Ocu_EnableNotification, Ocu_GetCounter, Ocu_SetAbsoluteThreshold, Ocu_DisableNotification, Ocu_StopChannel, Ocu_SetRelativeThreshold, Ocu_StartChannel, Ocu_SetPinAction, Ocu_SetPinState
Error code is reported when OCU module is not configured for the core in which an API is invoked.	IFX	OCU_E_CORE_NOT_CONFIGURED=0x64	Ocu_Init
If the reference value is greater than the maximum threshold value then function Ocu_SetAbsoluteThreshold shall raise the error "OCU_E_PARAM_REF_VALUE".	IFX	OCU_E_PARAM_REF_VALUE=201	Ocu_SetAbsoluteThreshold

18.3.7.2 Production errors

The driver does not report any production errors.

OCU driver**18.3.7.3 Safety errors**

The following table lists all the safety errors reported by the driver.

Table 1608 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
"OCU_E_PARAM_COMPARE_VALUE" error is reported to indicate invalid input compare values.	IFX	OCU_E_PARAM_COMPARE_VALUE=200	Ocu_SetRelativeThreshold, Ocu_SetAbsoluteThreshold
If the ISR is invoked before OCU_Init then Ocu driver shall report a safety error "OCU_E_INVALID_ISR_UNINIT".	IFX	OCU_E_INVALID_ISR_UNINIT=202	Ocu_Timer_Isr
If the ISR is invoked when the channel is not active then Ocu driver shall report a safety error "OCU_E_INVALID_ISR_INACTIVE_CHANNEL".	IFX	OCU_E_INVALID_ISR_INACTIVE_CHANNEL=203	Ocu_Timer_Isr
The Ocu driver shall report a safety error, if ISR is invoked with wrong flags (Compare match notification flag) indicating unexpected compare match.	IFX	OCU_E_INVALID_ISR_COMP_INVALID=204	Ocu_Timer_Isr
The Ocu driver shall report Safety error "OCU_E_INVALID_ISR_CHANNEL_INVALID", if the ISR is invoked when the passed channel ID is not configured.	IFX	OCU_E_INVALID_ISR_CHANNEL_INVALID=205	Ocu_Timer_Isr
The Ocu driver shall report DET/Safety error "OCU_E_INVALID_ISR_CHANNEL_CORE_MISMATCH", if the ISR is invoked when the passed channel and core id mismatch.	IFX	OCU_E_INVALID_ISR_CHANNEL_CORE_MISMATCH=206	Ocu_Timer_Isr

18.3.7.4 Runtime errors

The driver does not report any runtime errors.

OCU driver

18.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

18.3.8.1 Deviations

There are no deviations in the OCU driver.

18.3.8.2 Limitations

The section describes the limitations from software specification.

Table 1609 Known limitations

Reference	Limitation
Ocu_EnableNotification, Ocu_DisableNotification	If an OCU channel is configured to trigger a DMA channel, notifications cannot be issued by the OCU channel.
Ocu_SetPinState	Unintended ADC triggering might be issued when the Ocu_SetPinState API is invoked on a channel which is configured for both ADC triggering and pin.
Ocu_Init, ConfigPtr	User shall ensure, Ocu_Init() APIs input parameter ConfigPtr (Pointer to the configuration set) should be same across all the CPU CORES.

18.3.9 Unsupported hardware features

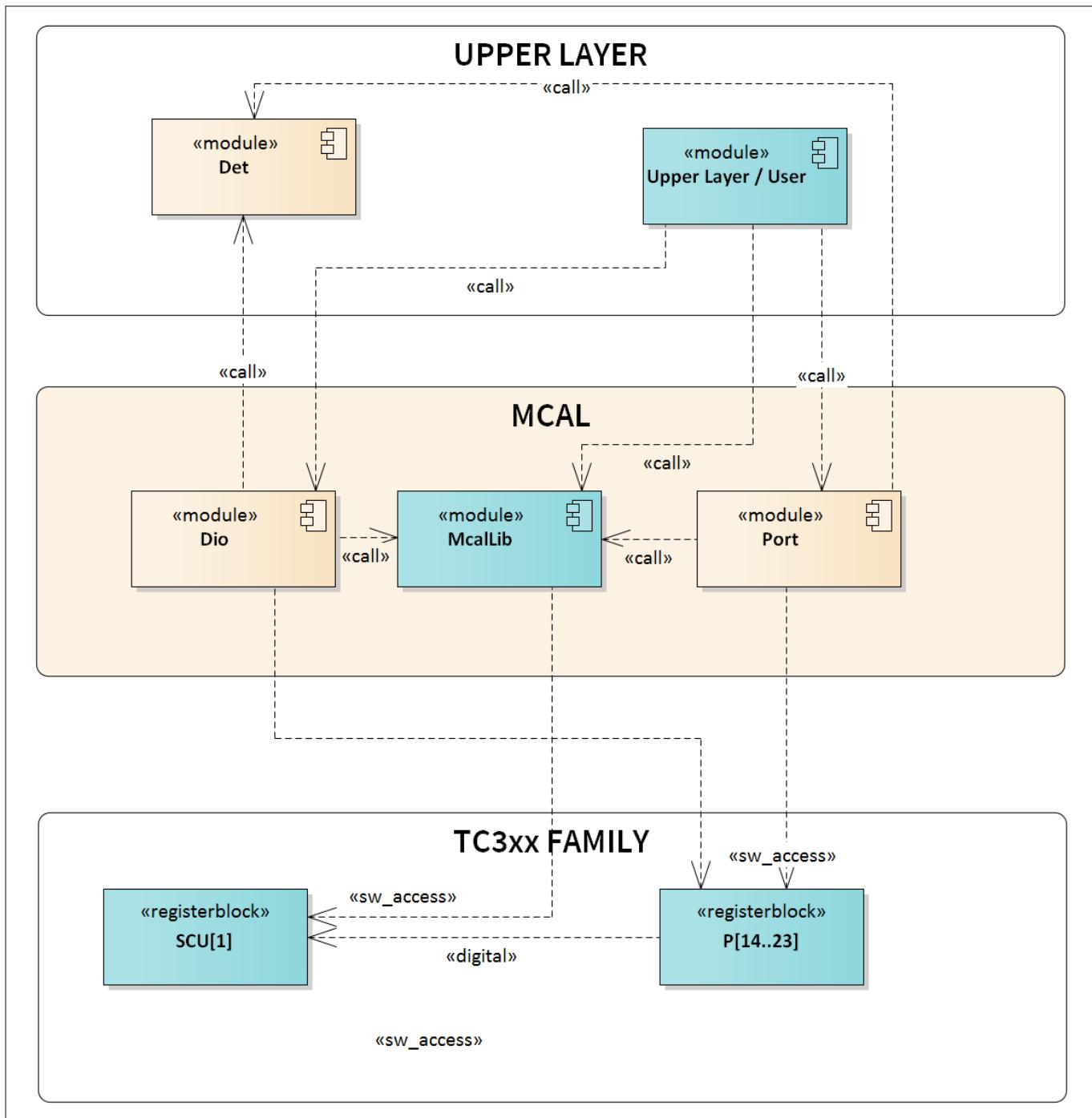
Not applicable to the OCU driver.

PORt driver**19 PORT driver****19.1 User information****19.1.1 Description**

The PORT driver helps user in assignment of Port pins to peripherals and configuration of characteristics/features provided by the underlying hardware. Being limited in availability, these port pins are shared amongst several on chip peripherals. However, at any given point in time, a port pin is assigned to and used exactly by one peripheral.

19.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

PORT driver
Figure 164 **Mapping of hardware-software interfaces**

19.1.2.1 Port: primary hardware peripheral

The following features of the underlying hardware PORT module are configurable.

- Pin direction
- Alternate pin mode
- Pad driver strength
- Pad level
- Pull up/pull down

PORt driver

- Digital/Analog mode
- Low-voltage differential signaling (LVDS) characteristic
- Switching between CMOS and LVDS modes
- Transmit and receive features of LVDS mode
- Emergency stop Enable/Disable

Users of the Hardware

Port pins are used by many MCAL modules but configuring the port pins is done through the PORT driver. The DIO driver also writes into the registers of the Port hardware to toggle/set or clear a port pin. Both the drivers program the registers atomically to avoid corruption due to concurrent updates.

Hardware diagnostic features

None.

Hardware events

None.

Unsupported hardware features

None.

19.1.2.2 SCU: dependent hardware peripheral

SCU is needed for CLOCK for the registers, and ENDINIT functionality is used to update certain registers.

Hardware Functional Features

The PORT driver depends on the SCU for the clock, ENDINIT and reset functionalities.

Users of the hardware

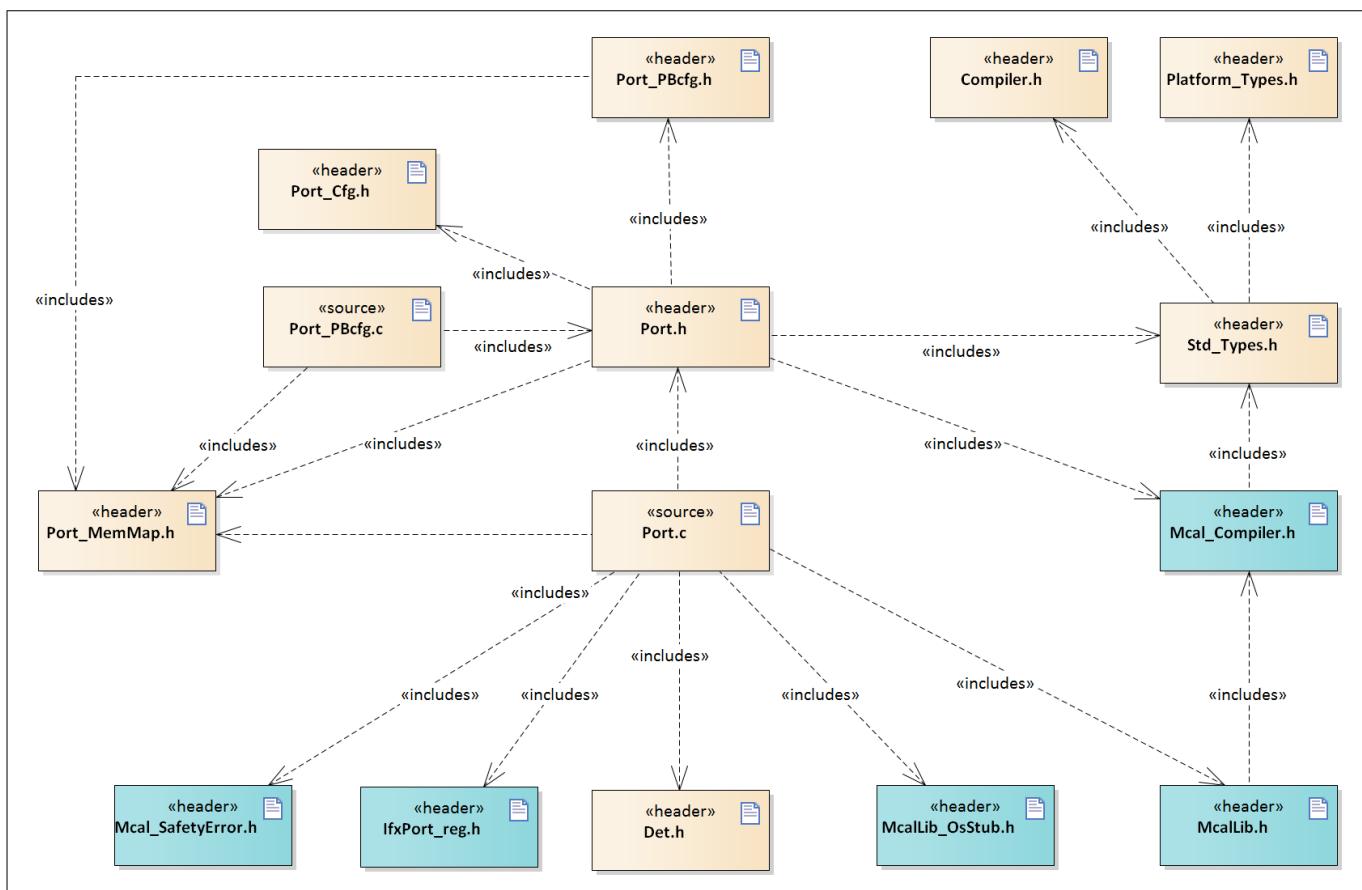
The SCU module supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. In order to avoid conflicts, update to the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU are not monitored by the PORT driver.

Hardware events

Hardware events from the SCU are not used by the PORT driver.

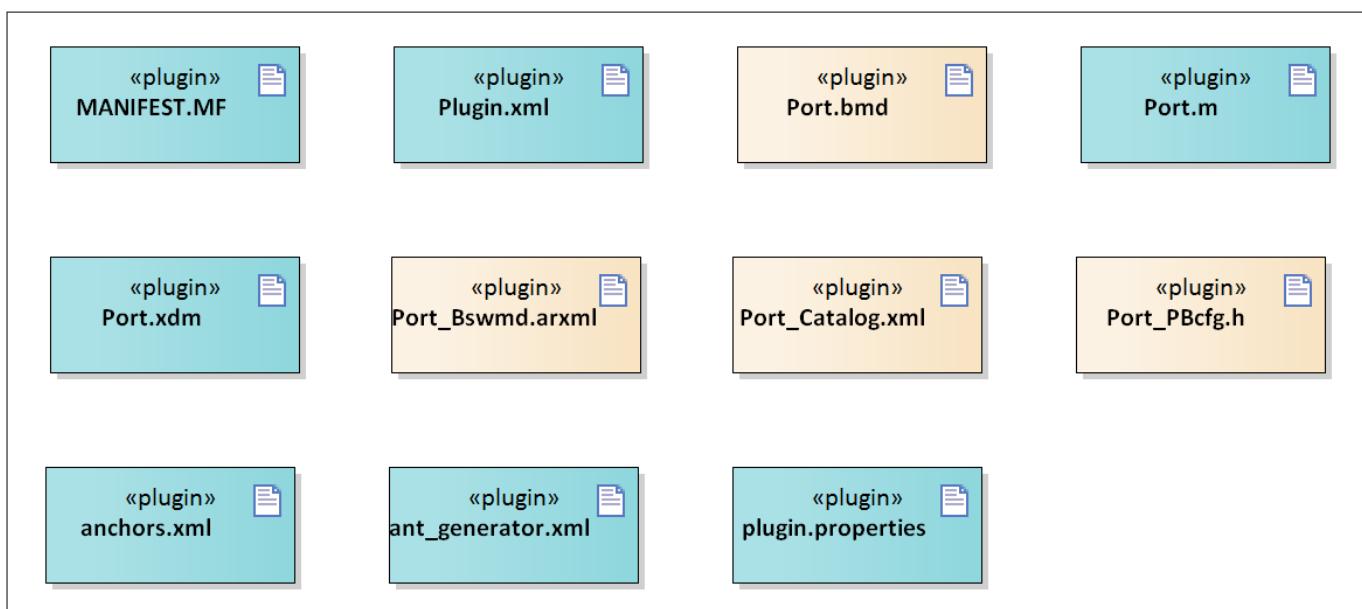
PORT driver**19.1.3 File structure****19.1.3.1 C file structure****Figure 165 C file structure****Table 1610 C file structure**

File name	Description
Port_Cfg.h	Header file (generated) containing constants and pre-processor macros
Port_MemMap.h	File (static) containing the memory section definitions used by the PORT driver
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
Det.h	Provides the exported interfaces of Development Error Tracer
Port.h	Header file (static) defining prototypes of data structures and APIs
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform
Port_PBcfg.c	File (generated) containing post-build configuration data structures
Port.c	File (static) containing implementation of APIs
Compiler.h	Provides abstraction from compiler-specific keywords
IfxPort_Reg.h	SFR header file for the PORT driver

PORT driver**Table 1610 C file structure (continued)**

File name	Description
McalLib.h	Static header file defining prototypes of data structure and APIs exported by MCALLIB
Port_PBcfg.h	File (generated) containing declaration of the post-build configuration data structures
Mcal_Compiler.h	Provides abstraction for the TriCore™ intrinsic instruction
McalLib_OsStub.h	Provides macros to support user mode of the TriCore™. This shall be included by other drivers to call OS APIs.
Mcal_SafetyError.h	Header file containing the prototype for the API for reporting safety-related errors

19.1.3.2 Code generator plugin file

**Figure 166 Code generator plugin file****Table 1611 Code generator plugin file**

File name	Description
Port.m	Code template macro file for the PORT driver
Port.bmd	AUTOSAR format XML data model schema file (for each device)
Port.xdm	Tresos format XML data model schema file
Port_Bswmd.arxml	AUTOSAR format module description file
Port_Catalog.xml	AUTOSAR format catalog file as per catalog_V3_0_0.ml.xsd
MANIFEST.MF	Tresos plugin support file containing the metadata for the PORT driver
anchors.xml	Tresos anchors support file for the PORT driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point

PORT driver
Table 1611 Code generator plugin file (continued)

File name	Description
plugin.properties	Tresos plugin support file for the PORT driver
Plugin.xml	Tresos plugin support file for the PORT driver
Port_PBcfg.h	File (generated) containing declaration of the post-build configuration data structures

19.1.4 Integration hints

This section lists the key points that an integrator or user of the PORT driver must consider.

19.1.4.1 Integration with AUTOSAR stack

This section lists the modules that are not part of the MCAL but are required to integrate the PORT driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Port_MemMap.h` API.

The `Port_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements

PORt driver

are relocated to the correct memory region. A sample implementation listing the memory-section macros is depicted below.

```
#if defined PORT_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*your pragma here*/
#undef PORT_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32

#elif defined PORT_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*your pragma here*/
#undef PORT_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32

#elif defined PORT_START_SEC_CODE_ASIL_B_GLOBAL
/*your pragma here*/
#undef PORT_START_SEC_CODE_ASIL_B_GLOBAL

#elif defined PORT_STOP_SEC_CODE_ASIL_B_GLOBAL
/*your pragma here*/
#undef PORT_STOP_SEC_CODE_ASIL_B_GLOBAL
#endif

#if defined MEMMAP_ERROR
#error "Port_MemMap.h, wrong pragma command"
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The Port driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the Port driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is not required for the integration of the PORT driver.

- **SchM**

The SchM is not required for the integration of the PORT driver.

- **Safety error**

The PORT driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API. The driver performs only detection and reporting of the safety errors. Handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The driver does not provide any callbacks or notifications.

- **Operating system**

The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application. Operating

PORT driver

system files provided by MCAL package is only an example code and must be updated by the integrator with the actual OS files for the desired function.

19.1.4.2 Multicore and Resource Manager

The PORT module supports the multicore concept. All the APIs except for the `Port_Init` API can be accessed from any core. The `Port_Init` API needs to be called only once from any core.

19.1.4.3 MCU support

The PORT driver does not use any services provided by MCU driver.

19.1.4.4 Port support

Not applicable for the driver.

19.1.4.5 DMA support

The PORT driver does not use any services provided by the DMA driver.

19.1.4.6 Interrupt connections

The Port driver does not use any interrupt source.

PORt driver

19.1.4.7 Example usage

Initialization of PORT driver

Initialization of PORT driver is done by calling the `Port_Init` API.

```
/* Include Port.h to access configuration file*/
#include "Port.h"

/* MCU initializations */
Mcu_Init(&Mcu_Config);

Mcu_Init(&Mcu_Config);
(void)Mcu_InitClock( 0 );
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED)
{
}

/* Port Initialization */
Port_Init(&Port_Config);
```

User must take care that the `Port_Init` API is called before using any other API provided by the PORT driver. Development error is reported if user calls other PORT driver API before calling `Port_Init`.

Changing the direction of the pin

```
/* Port Initialization */
Port_Init(&Port_Config);

Port_SetPinDirection(PortConf_PortContainer_0_PORT_0_PIN_0, PORT_PIN_OUT);
```

The PORT driver provides the API to change the direction of the pin during run time. User must enable `PortPinDirectionChangeable` in the configuration tool for the required pin. This API is available only when the `PortSetPinDirectionApi` is switched ON.

Refreshing direction of all configured pins

The PORT driver provides the API to refresh the direction of all the configured pins. This API does not refresh the direction of pins, where pin direction is configured as changeable.

```
/* Port Initialization */
Port_Init(&Port_Config);

Port_RefreshPortDirection();
```

Set port pin mode to another alternate mode

PORt driver

The PORT driver provides this API to set the port pin mode of referenced pin at run-time. This API is available only when the `PortSetPinModeApi` is switched ON.

```
/* Port Initialization */
Port_Init(&Port_Config);

Port_SetPinMode(PortConf_PortContainer_0_PORT_0_PIN_0, PORT_PIN_MODE_ALT3);
```

Port_InitCheck

The PORT driver provides `Port_InitCheck` to check the initialization values is correct after PORT is initialized. It should be called after `Port_Init`. The API returns `E_OK` or `E_NOT_OK`.

```
/* Port Initialization */
Port_Init(&Port_Config);

/*Complete other module initialization*/

result= Port_InitCheck (&Port_Config);
```

19.1.5 Key architectural considerations

19.1.5.1 Pin support for Ethernet

The `PortPinOutputPadDriveStrength` configuration parameter can be set to `PORT_PIN_RGMII_DRIVER` for port pins that are used for GETH.

19.1.5.2 LCK bit not checked before writing to PCSR register by Port_Init API

All the pins must be configured and initialized under the purview of the PORT driver. If some pins are configured and initialized outside the PORT driver, the LCK bit must not be set before calling the `Port_Init` API. If the LCK bit is set before, the `Port_Init` API cannot configure the respective PCSR register.

PORt driver

19.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Configuration check**

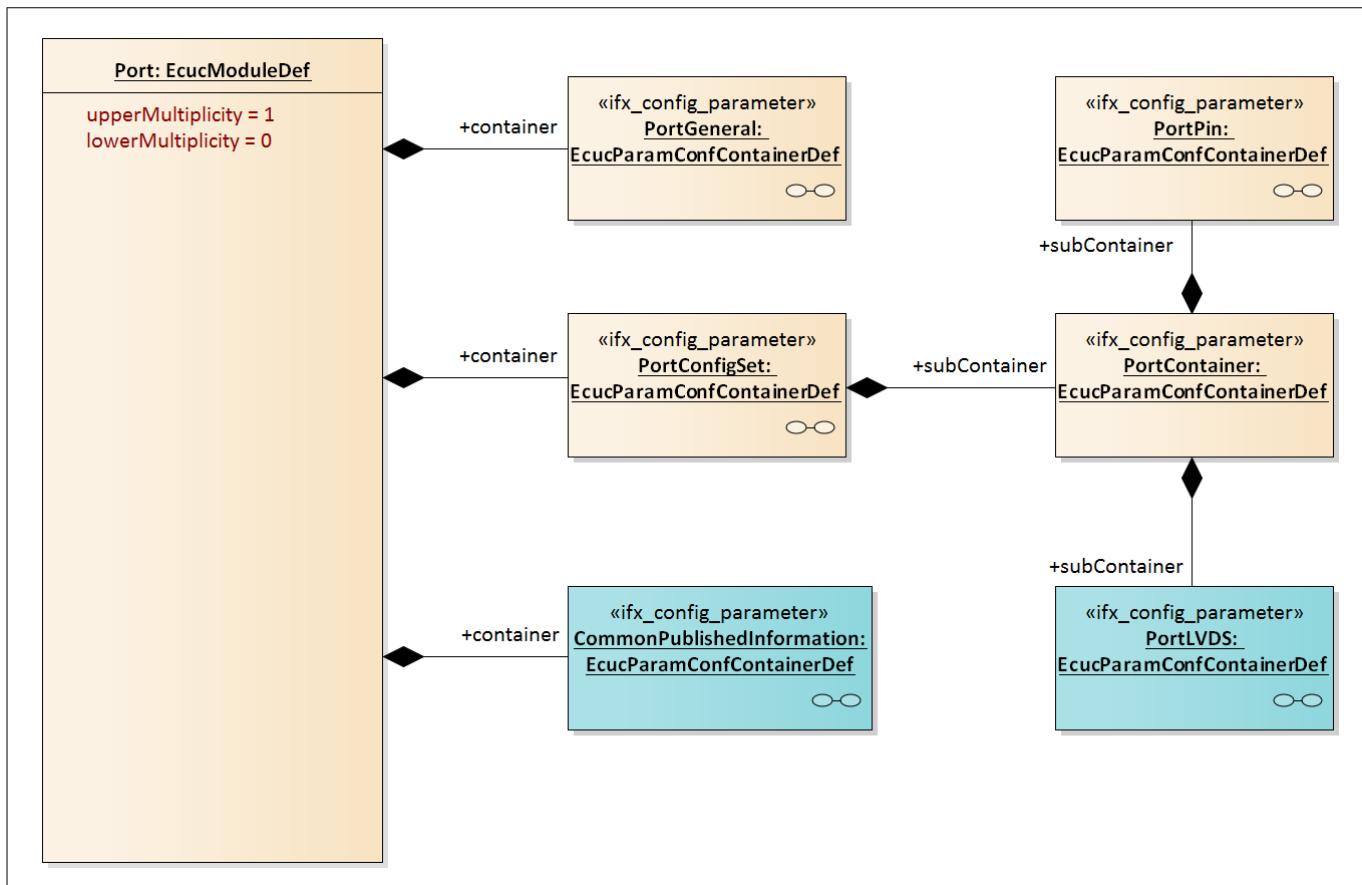
User shall check that the generated configuration code is correct as per GUI configuration.

[cover parentID PORT={4AF3131A-7647-47c5-9AE4-AFD673334476}]

- **InitCheck**

If configured, the Port_InitCheck API shall be invoked by the application or user to validate successful initialization of the Port_Init API. The Port_InitCheck API shall be invoked before starting any functionality of PORT.

[cover parentID PORT={2F1F7426-F632-45ed-B43B-D922AF98DC75}]

PORT driver**19.3 Reference information****19.3.1 Configuration interfaces****Figure 167 Container hierarchy along with their configuration parameters****19.3.1.1 Container: CommonPublishedInformation**

This container holds all the published information of the PORT driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

19.3.1.1.1 ArMajorVersion**Table 1612 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Major version number of the AUTOSAR specification on which the implementation is based on.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	4		

PORT driver**Table 1612 Specification for ArMajorVersion (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.1.2 ArMinorVersion**Table 1613 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	Minor version number of the AUTOSAR specification on which the implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.1.3 ArPatchVersion**Table 1614 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	Patch version number of the AUTOSAR specification on which the implementation is based on.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

PORT driver**Table 1614 Specification for ArPatchVersion (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.1.4 ModuleId**Table 1615 Specification for ModuleId**

Name	ModuleId		
Description	Module ID of the PORT driver.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 65535		
Default value	124		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.1.5 Release**Table 1616 Specification for Release**

Name	Release		
Description	Specifies the derivative for which the configuration project is created.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per hardware derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

PORT driver**19.3.1.1.6 SwMajorVersion****Table 1617 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Specifies the major version of the driver software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.1.7 SwMinorVersion**Table 1618 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Specifies the minor version of the driver software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.1.8 SwPatchVersion**Table 1619 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Specifies the patch version of the driver software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		

PORT driver**Table 1619 Specification for SwPatchVersion (continued)**

Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.1.9 VendorId**Table 1620 Specification for VendorId**

Name	VendorId		
Description	Vendor ID for Infineon.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.2 Container: PortConfigSet

This container contains the configuration parameters and sub containers of the PORT driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

19.3.1.3 Container: PortContainer

This container holds the configuration parameters related to all the port pins.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

PORT driver**19.3.1.3.1 PortNumber****Table 1621 Specification for PortNumber**

Name	PortNumber		
Description	Specifies the port number currently being configured. Note: The values configured here are device dependent and retrieved from device property files. It cannot be edited by the user.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - max port available		
Default value	Device dependent. Starting value is retrieved from device property files.		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.3.2 PortNumberOfPortPins**Table 1622 Specification for PortNumberOfPortPins**

Name	PortNumberOfPortPins		
Description	Specifies the number of port pins available for the selected port. Note: The values configured here are device dependent and retrieved from device property files. It cannot be edited by the user.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 16		
Default value	Device dependent. Starting value is retrieved from device property files.		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

19.3.1.4 Container: PortGeneral

The container holds the module wide configuration parameters.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

PORT driver**19.3.1.4.1 PortDevErrorDetect****Table 1623 Specification for PortDevErrorDetect**

Name	PortDevErrorDetect		
Description	Enables / Disables the detection and reporting of Development Error Detection. TRUE: DET is enabled. FALSE: DET is disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

19.3.1.4.2 PortInitApiMode**Table 1624 Specification for PortInitApiMode**

Name	PortInitApiMode		
Description	Specifies the privilege mode in which the initialization API will operate. Note: The driver accesses the SFRs, it is more efficient to operate the PORT driver in supervisor mode. Hence, the default mode of operation is supervisor.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PORT_MCAL_SUPERVISOR: The selected privilege mode is SUPERVISOR PORT_MCAL_USER1: The selected privilege mode is USER1		
Default value	PORT_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

PORT driver**19.3.1.4.3 PortInitCheckApi****Table 1625 Specification for PortInitCheckApi**

Name	PortInitCheckApi		
Description	<p>Enables/disables the Port_InitCheck API.</p> <p>TRUE: PortInitCheckApi API is enabled.</p> <p>FALSE: PortInitCheckApi API is disabled.</p> <p>Note: The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.4.4 PortSafetyEnable**Table 1626 Specification for PortSafetyEnable**

Name	PortSafetyEnable		
Description	<p>Specifies whether the safety checks mandated by safety standards are enabled or disabled.</p> <p>TRUE: Safety checks are enabled.</p> <p>FALSE: Safety checks are disabled.</p> <p>Note: The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

PORT driver**Table 1626 Specification for PortSafetyEnable (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.4.5 PortSetPinDirectionApi**Table 1627 Specification for PortSetPinDirectionApi**

Name	PortSetPinDirectionApi		
Description	<p>Pre-processor switch for enabling the API Port_SetPinDirection () which sets the port pin direction of the referenced pin during runtime.</p> <p>TRUE: Port_SetPinDirection API is available.</p> <p>FALSE:Port_SetPinDirection API is not available.</p> <p>Note: The API is disabled by default to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

19.3.1.4.6 PortSetPinModeApi**Table 1628 Specification for PortSetPinModeApi**

Name	PortSetPinModeApi		
Description	<p>Pre-processor switch to enable / disable the use of the API Port_SetPinMode().</p> <p>TRUE: Port_SetPinMode() API is available.</p> <p>FALSE:Port_SetPinMode() API is not available.</p> <p>Note: The API is disabled by default to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		

PORT driver**Table 1628 Specification for PortSetPinModeApi (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

19.3.1.4.7 PortVersionInfoApi**Table 1629 Specification for PortVersionInfoApi**

Name	PortVersionInfoApi		
Description	Pre-processor switch to enable / disable the API to read out the modules version information. TRUE: Port_GetVersionInfo API is enabled. FALSE: Port_GetVersionInfo API is disabled. Note: The APIs is disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

19.3.1.5 Container: PortLVDS

This container holds all the configuration parameters for LVDS port pin pairs.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

19.3.1.5.1 PortLVDSMode**Table 1630 Specification for PortLVDSMode**

Name	PortLVDSMode
-------------	--------------

PORT driver**Table 1630 Specification for PortLVDSMode (continued)**

Description	Specifies the frequency mode for the Rx pads. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	LVDSH: High frequency mode is selected LVDSM: Reduced frequency mode is selected		
Default value	LVDSH		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.5.2 PortLVDSPadSupply**Table 1631 Specification for PortLVDSPadSupply**

Name	PortLVDSPadSupply		
Description	Specifies the supply voltage for both Tx and Rx pads Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	V3_3: 3.3 V is selected V5_0: 5.0 V is selected		
Default value	3_3V		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.5.3 PortLVDSPinPair**Table 1632 Specification for PortLVDSPinPair**

Name	PortLVDSPinPair
-------------	-----------------

PORT driver**Table 1632 Specification for PortLVDSPinPair (continued)**

Description	Publishes the port pin pair which supports the LVDS feature. The value is automatically retrieved by the tool from the device property files. Note: This parameter cannot be edited by the user.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	Pin_x_Pin_y [x and y indicate the pin pair for LVDS]		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.5.4 PortLVDSRxEnController**Table 1633 Specification for PortLVDSRxEnController**

Name	PortLVDSRxEnController		
Description	Specifies the controller of the LVDS enable/disable function. It is applicable only for the Rx LVDS pair. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	HSCT_CONTROLLED: If HSCT_CONTROLLED is selected as the controller, the user must enable/disable the LVDS in registers related to HSCT module. PORT_CONTROLLED: If PORT_CONTROLLED is selected, then LVDS enable/disable is performed by the means of PORT registers.		
Default value	PORT_CONTROLLED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

PORT driver**19.3.1.5.5 PortLVDSRxPathEnable****Table 1634 Specification for PortLVDSRxPathEnable**

Name	PortLVDSRxPathEnable		
Description	<p>Specifies whether the pin-Pair is in LVDS mode or CMOS mode.</p> <p>TRUE- Enables the LVDS transceiver and disables the CMOS mode</p> <p>FALSE -Disables the LVDS transceiver and enables the CMOS mode</p> <p>The parameter is applicable for only Rx LVDS pair and when PortLVDSRxEnController is holding a value PORT_CONTROLLED.</p> <p>Note: The default value of this parameter is set to the reset value of the corresponding SFR</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.5.6 PortLVDSRxTerminationMode**Table 1635 Specification for PortLVDSRxTerminationMode**

Name	PortLVDSRxTerminationMode		
Description	<p>Selects the terminal load resistor for the port pin .This parameter is applicable only for the Rx LVDS pair.</p> <p>Note: The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>EXTERNAL_TERMINATION: External load resistor is selected (external termination on the PCB).</p> <p>INTERNAL_TERMINATION: Internal load resistor of a 100 ohms is selected.</p>		
Default value	INTERNAL_TERMINATION		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

PORT driver**Table 1635 Specification for PortLVDSRxTerminationMode (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.5.7 PortLVDSTxEnController**Table 1636 Specification for PortLVDSTxEnController**

Name	PortLVDSTxEnController		
Description	Specifies the controller of the LVDS enable/disable function .It is applicable only for Tx LVDS pair. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	HSCT_CONTROLLED: HSCT_CONTROLLED is selected as the controller, the user should take care of enabling/disabling the LVDS in registers related to HSCT module. PORT_CONTROLLED: PORT_CONTROLLED is selected, then LVDS enable/disable can be done by means of PORT registers.		
Default value	PORT_CONTROLLED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.5.8 PortLVDSTxPathEnable**Table 1637 Specification for PortLVDSTxPathEnable**

Name	PortLVDSTxPathEnable		
Description	Specifies whether the Pin-Pair is in LVDS mode or CMOS mode TRUE :Enables the LVDS transceiver FALSE :Disable the LVDS transceiver. The parameter is applicable for TX LVDS pair and only when PortLVDSTxEnController is holding a value PORT_CONTROLLED. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		

PORT driver**Table 1637 Specification for PortLVDSTxPathEnable (continued)**

Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.5.9 PortLVDSTxPowerDownPullDown**Table 1638 Specification for PortLVDSTxPowerDownPullDown**

Name	PortLVDSTxPowerDownPullDown		
Description	Specifies the state of the Transmit Pull down resistor FALSE-Disables the Tx Power down - pull down resistor, TRUE- Enables the Tx Power down - pull down resistor. This parameter is applicable only for the transmitting LVDS pair. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

19.3.1.6 Container: PortPin

Configuration of the individual port pins.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

PORT driver**19.3.1.6.1 PortPinControllerSelect****Table 1639 Specification for PortPinControllerSelect**

Name	PortPinControllerSelect		
Description	The parameter enables / disables whether SCR controls the port pins configuration and data. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DISABLE: Tricore is selected for data and control of the pin. ENABLE : SCR/VADC/GETH/SMU selected for data and control of the pin.		
Default value	DISABLE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PortNumber		

19.3.1.6.2 PortPinDirection**Table 1640 Specification for PortPinDirection**

Name	PortPinDirection		
Description	Specifies the direction for the port pin. Note: The configuration of this parameter is not valid for analog input ports. Note:: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PORT_PIN_IN: Port Pin direction set as input PORT_PIN_OUT: Port Pin direction set as output		
Default value	PORT_PIN_IN		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PortNumber		

PORT driver**19.3.1.6.3 PortPinDirectionChangeable****Table 1641 Specification for PortPinDirectionChangeable**

Name	PortPinDirectionChangeable		
Description	<p>Specifies whether the pin direction can be changed at run-time for the current port pin.</p> <p>The configuration of this parameter is not valid for analog or digital input ports.</p> <p>Note: The optional features are disabled by default to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PortNumber, PortSetPinDirectionApi		

19.3.1.6.4 PortPinEmergencyStop**Table 1642 Specification for PortPinEmergencyStop**

Name	PortPinEmergencyStop		
Description	<p>With this feature enabled, a pin configured as output pin will be automatically be reconfigured as an input , when an emergency condition is encountered.</p> <p>FALSE: Disable emergency stop function for the selected pin.</p> <p>TRUE: Enables emergency stop function for the selected pin .</p> <p>Note: The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

PORT driver**Table 1642 Specification for PortPinEmergencyStop (continued)**

Dependency	PortPinDirection
-------------------	------------------

19.3.1.6.5 PortPinEnableAnalogInputOnly**Table 1643 Specification for PortPinEnableAnalogInputOnly**

Name	PortPinEnableAnalogInputOnly		
Description	Certain digital PORT pads additionally support analog input functionality. It is possible to completely disable the digital functions of the pad in order to use the pad for analog input. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PORT_PIN_ANALOG_INPUT_DISABLE: Analog Input functionality is disabled PORT_PIN_ANALOG_INPUT_ENABLE : Analog Input functionality is enabled.		
Default value	PORT_PIN_ANALOG_INPUT_DISABLE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PortNumber		

19.3.1.6.6 PortPinId**Table 1644 Specification for PortPinId**

Name	PortPinId		
Description	Published an Id for each port pins. Note: The default value of this parameter is to be retained. It is derived from property files. This is a read-only parameter		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0x0000 - max port pin ID		
Default value	Device Dependent. Starting value is retrieved from device property files.		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL

PORT driver**Table 1644 Specification for PortPinId (continued)**

Dependency	-
-------------------	---

19.3.1.6.7 PortPinInitialMode**Table 1645 Specification for PortPinInitialMode**

Name	PortPinInitialMode		
Description	<p>The parameter allows to configure the operating mode i.e. the different alternate functionality of each pin. The selected operating mode is programmed to registers during initialization.</p> <p>Note: Editable only if the port pin direction is set to PORT_PIN_OUT.</p> <p>Note: Tooltip for PortPinInitialMode gives the list of all available ALT mode.</p> <p>Note: The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ALT1: ALT1 mode ALT2: ALT2 mode ALT3: ALT3 mode ALT4: ALT4 mode ALT5: ALT5 mode ALT6: ALT6 mode ALT7: ALT7 mode GPIO: GPIO mode		
Default value	GPIO		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PortPinDirection		

19.3.1.6.8 PortPinInputPadLevel**Table 1646 Specification for PortPinInputPadLevel**

Name	PortPinInputPadLevel		
Description	<p>The parameter allows to configure the voltage level for the selected port pin.</p> <p>Note: The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef

PORT driver**Table 1646 Specification for PortPinInputPadLevel (continued)**

Range	PORT_INPUT_LEVEL_TTL_3_3V: PORT_INPUT_LEVEL_TTL_3_3V:TTL level for 3.3V PORT_INPUT_LEVEL_TTL_5_0V: PORT_INPUT_LEVEL_TTL_5_0V:TTL level for 5V. PORT_INPUT_LEVEL_CMOS_AUTOMOTIVE: PORT_INPUT_LEVEL_CMOS_AUTOMOTIVE:CMOS automotive level		
Default value	PORT_INPUT_LEVEL_CMOS_AUTOMOTIVE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PortPinDirection		

19.3.1.6.9 PortPinInputPullResistor**Table 1647 Specification for PortPinInputPullResistor**

Name	PortPinInputPullResistor		
Description	The parameter allows to configure the internal Pull resistor [up/down] for the selected port pin. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PORT_PIN_IN_NO_PULL: PORT_PIN_IN_NO_PULL:Input pull is not connected and pin operates in tristate mode. PORT_PIN_IN_PULL_DOWN: PORT_PIN_IN_PULL_DOWN: Pull-down is connected. PORT_PIN_IN_PULL_UP: PORT_PIN_IN_PULL_UP: Pull-up is connected.		
Default value	PORT_PIN_IN_PULL_UP		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PortPinDirectionChangeable, PortPinDirection		

19.3.1.6.10 PortPinLevelValue**Table 1648 Specification for PortPinLevelValue**

Name	PortPinLevelValue
-------------	-------------------

PORT driver**Table 1648 Specification for PortPinLevelValue (continued)**

Description	The parameter is to configure the initial pin level for each pin. This level is set to the output latch of the port pin during initialization of the port driver, irrespective of the direction configured to the pin. Note: For analog or digital input ports, the configuration of this parameter is not valid. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PORT_PIN_LEVEL_HIGH: Port Pin level is High. PORT_PIN_LEVEL_LOW: Port Pin level is LOW.		
Default value	PORT_PIN_LEVEL_LOW		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PortPinDirection		

19.3.1.6.11 PortPinMode**Table 1649 Specification for PortPinMode**

Name	PortPinMode		
Description	The parameter allows to configure all the possible alternate modes for the current port pin. Note: Refer to PortPinInitialMode tooltip for the ALT modes available. Note: The default value is PORT_PIN_MODE_ALL, which are the bits set according to the modes that are available for that Port pin.		
Multiplicity	1..8	Type	EcucEnumerationParamDef
Range	PORT_PIN_MODE_ALL: All the alternate modes are supported. PORT_PIN_MODE_ALT1: Alternate mode 1 is supported. PORT_PIN_MODE_ALT2: Alternate mode 2 is supported. PORT_PIN_MODE_ALT3: Alternate mode 3 is supported. PORT_PIN_MODE_ALT4: Alternate mode 4 is supported. PORT_PIN_MODE_ALT5: Alternate mode 5 is supported. PORT_PIN_MODE_ALT6: Alternate mode 6 is supported. PORT_PIN_MODE_ALT7: Alternate mode 7 is supported. PORT_PIN_MODE_GPIO: Alternate mode GPIO is supported.		
Default value	PORT_PIN_MODE_ALL		

PORT driver**Table 1649 Specification for PortPinMode (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PortPinModeChangeable, PortPinDirection, PortSetPinModeApi		

19.3.1.6.12 PortPinModeChangeable**Table 1650 Specification for PortPinModeChangeable**

Name	PortPinModeChangeable		
Description	<p>The parameter allows to configure whether mode for a port pin is allowed to be changed at run-time</p> <p>The configuration of this parameter is not valid for analog or digital input ports.</p> <p>FALSE: Mode is not changeable at the runtime for the selected pin.</p> <p>TRUE: Mode is changeable at the runtime for the selected pin.</p> <p>Note: The optional features are disabled by default to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PortPinDirectionChangeable, PortNumber, PortSetPinModeApi		

19.3.1.6.13 PortPinOutputPadDriveStrength**Table 1651 Specification for PortPinOutputPadDriveStrength**

Name	PortPinOutputPadDriveStrength		
Description	<p>This parameter configures the output drive strength and slew rate for each pin.</p> <p>Note: The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef

PORT driver**Table 1651 Specification for PortPinOutputPadDriveStrength (continued)**

Range	PORT_PIN_DEFAULT_DRIVER: Default driver strength. PORT_PIN_MEDIUM_DRIVER: Medium driver strength applicable for RFast ,Fast and Slow pads PORT_PIN_MEDIUM_DRIVER_SHARP_EDGE: Medium driver strength with sharp edge applicable only for slow pads. PORT_PIN_RGMII_DRIVER: RGMII driver applicable only for RFast pads PORT_PIN_STRONG_DRIVER_MEDIUM_EDGE: This option is selectable for RFast and Fast pads. PORT_PIN_STRONG_DRIVER_SHARP_EDGE: Strong driver strength with sharp edge applicable for RFast and Fast pads.		
Default value	PORT_PIN_DEFAULT_DRIVER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PortPinDirectionChangeable, PortPinDirection		

19.3.1.6.14 PortPinOutputPinDriveMode**Table 1652 Specification for PortPinOutputPinDriveMode**

Name	PortPinOutputPinDriveMode		
Description	This parameter allows to configure between open-drain or push pull. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PORT_PIN_OUT_OPENDRAIN: PORT_PIN_OUT_OPENDRAIN :Open drain configuration is selected for the pin. PORT_PIN_OUT_PUSH_PULL: PORT_PIN_OUT_PUSH_PULL : Push pull configuration is selected for the pin.		
Default value	PORT_PIN_OUT_PUSH_PULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PortPinDirectionChangeable, PortPinDirection		

PORT driver**19.3.1.6.15 PortPinSymbolicName****Table 1653 Specification for PortPinSymbolicName**

Name	PortPinSymbolicName		
Description	<p>This parameter is a user defined name for the port pin under consideration.</p> <p>The user of the PORT Driver module can use the enumerator to identify a Port-Pin pair rather than using an absolute number.</p>		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	PORT_x_PIN_Y		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PortPinId		

19.3.2 Functions - Type definitions**19.3.2.1 Port_ConfigType****Table 1654 Specification for Port_ConfigType**

Syntax	Port_ConfigType		
Type	Structure		
File	Port.h		
Description	Data type for the post-build configuration structure		
Source	AUTOSAR		

19.3.2.2 Port_PinDirectionType**Table 1655 Specification for Port_PinDirectionType**

Syntax	Port_PinDirectionType		
Type	Enumeration		
File	Port.h		
Range	0 - PORT_PIN_IN	Sets port pin as input.(0x00)	
	128 - PORT_PIN_OUT	Sets port pin as output.(0x80)	
Description	The type defines PORT pin direction		

PORT driver**Table 1655 Specification for Port_PinDirectionType (continued)**

Source	AUTOSAR
---------------	---------

19.3.2.3 Port_PinModeType**Table 1656 Specification for Port_PinModeType**

Syntax	Port_PinModeType	
Type	uint8	
File	Port.h	
Range		
	PORT_PIN_MODE_ALT1	Alternate mode 1
	PORT_PIN_MODE_ALT2	Alternate mode 2.
	PORT_PIN_MODE_ALT3	Alternate mode 3.
	PORT_PIN_MODE_ALT4	Alternate mode 4.
	PORT_PIN_MODE_ALT5	Alternate mode 5.
	PORT_PIN_MODE_ALT6	Alternate mode 6.
	PORT_PIN_MODE_ALT7	Alternate mode 7.
	PORT_PIN_MODE_GPIO	GPIO mode.
Description	The type defines PORT pin mode	
Source	AUTOSAR	

19.3.2.4 Port_PinType**Table 1657 Specification for Port_PinType**

Syntax	Port_PinType	
Type	uint16	
File	Port.h	
Range	0-Number of available port pins	
Description	This type defines numeric ID for port pins.	
Source	AUTOSAR	

19.3.3 Functions - APIs**19.3.3.1 Port_GetVersionInfo****Table 1658 Specification for Port_GetVersionInfo API**

Syntax	void Port_GetVersionInfo (
---------------	-------------------------------

PORT driver**Table 1658 Specification for Port_GetVersionInfo API (continued)**

	Std_VersionInfoType * const versioninfo)	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where to store the version information of the module.
Parameters (in - out)	-	-
Return	void	-
Description	Returns the version information of PORT driver.	
Source	AUTOSAR	
Error handling	DET: PORT_E_PARAM_POINTER : This error code is reported if a null pointer is passed as a parameter. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	PortVersionInfoApi	
User hints	-	

19.3.3.2 Port_Init**Table 1659 Specification for Port_Init API**

Syntax	void Port_Init (const Port_ConfigType * const ConfigPtr)
Service ID	0x00
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Non Reentrant

PORT driver**Table 1659 Specification for Port_Init API (continued)**

Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Initializes the PORT driver module as per the configuration set passed.	
Source	AUTOSAR	
Error handling	DET: PORT_E_INIT_FAILED : This error code is reported if initialization is invoked with incorrect configuration pointer. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

19.3.3.3 Port_InitCheck**Table 1660 Specification for Port_InitCheck API**

Syntax	<pre>Std_ReturnType Port_InitCheck (const Port_ConfigType ConfigPtr)</pre>	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : Driver is initialized correctly . E_NOT_OK : Driver is not initialized correctly .

PORT driver**Table 1660 Specification for Port_InitCheck API (continued)**

Description	Performs the initialization check for the PORT driver.
Source	IFX
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	-
User hints	-

19.3.3.4 Port_RefreshPortDirection**Table 1661 Specification for Port_RefreshPortDirection API**

Syntax	void Port_RefreshPortDirection (void)	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Refreshes port direction.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>PORT_E_UNINIT : This error code is reported if any API is invoked prior to the module initialization.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

PORT driver**Table 1661 Specification for Port_RefreshPortDirection API (continued)**

Configuration dependencies	-
User hints	-

19.3.3.5 Port_SetPinDirection**Table 1662 Specification for Port_SetPinDirection API**

Syntax	<pre>void Port_SetPinDirection (const Port_PinType Pin, const Port_PinDirectionType Direction)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different pins independent of port.	
Parameters (in)	Pin Direction	Port pin ID Port pin direction
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API sets the PORT pin to the specified direction.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>PORT_E_UNINIT : This error code is reported if any API is invoked prior to the module initialization.</p> <p>PORT_E_DIRECTION_UNCHANGEABLE: This error code is reported if a port pin for which direction is not changeable is passed as a parameter.</p> <p>PORT_E_PARAM_PIN: This error code is reported if the port pin ID passed is invalid.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	PortSetPinDirectionApi	
User hints	-	

PORT driver**19.3.3.6 Port_SetPinMode****Table 1663 Specification for Port_SetPinMode API**

Syntax	<pre>void Port_SetPinMode (const Port_PinType Pin, const Port_PinModeType Mode)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different pins independent of port.	
Parameters (in)	Pin Mode	Port pin ID number whose mode has to be set Port pin mode to be set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API sets the port pin mode during runtime as per the passed mode.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>PORTE_UNINIT : This error code is reported if any API is invoked prior to the module initialization.</p> <p>PORTE_PARAM_INVALID_MODE : This error code is reported if the passed mode is invalid.</p> <p>PORTE_MODE_UNCHANGEABLE : This error code is reported if a port pin for which the mode is not changeable is passed as a parameter.</p> <p>PORTE_PARAM_PIN: This error code is reported if the port pin ID passed is invalid.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	PortSetPinModeApi	
User hints	-	

19.3.4 Notifications and callbacks

Not applicable for the PORT driver.

19.3.5 Scheduled functions

Not applicable for the PORT driver.

PORT driver

19.3.6 Interrupt service routines

Not applicable for the PORT driver.

19.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

19.3.7.1 Development errors

The table below lists all the development errors reported by the driver. Note: The below error IDs are also reported as safety errors.

Table 1664 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
This error code is reported if a port pin for which direction is not changeable is passed as a parameter.	AUTOSAR	PORT_E_DIRECTION_UNCHANGEABLE=0x0B	Port_SetPinDirection
This error code is reported if initialization is invoked with incorrect configuration pointer.	AUTOSAR	PORT_E_INIT_FAILED = 0x0C	Port_Init
This error code is reported if a port pin for which the mode is not changeable is passed as a parameter.	AUTOSAR	PORT_E_MODE_UNCHANGEABLE =0x0E	Port_SetPinMode
This error code is reported if the passed mode is invalid.	AUTOSAR	PORT_E_PARAM_INVALID_MODE =0x0D	Port_SetPinMode
This error code is reported if the port pin ID passed is invalid.	AUTOSAR	PORT_E_PARAM_PIN=0x0A	Port_SetPinDirection , Port_SetPinMode
This error code is reported if a null pointer is passed as a parameter.	AUTOSAR	PORT_E_PARAM_POINTER =0x10	Port_GetVersionInfo
This error code is reported if any API is invoked prior to the module initialization.	AUTOSAR	PORT_E_UNINIT =0x0F	Port_SetPinMode, Port_RefreshPortDirection , Port_SetPinDirection

19.3.7.2 Production errors

The module does not report any production errors.

19.3.7.3 Safety errors

The module does not report any safety errors

PORT driver

19.3.7.4 Run time errors

The module does not report any runtime errors.

19.3.8 Deviations and limitations

19.3.8.1 Deviations

There are no deviations for the driver.

19.3.8.2 Limitations

The section describes the deviations from software specification.

Table 1665 Known limitations

Reference	Limitation
Range check for input parameter direction for the Port_SetPinDirection API.	The range check for input parameter direction is not performed for the Port_SetPinDirection API. As a workaround, the user shall ensure that PORT_PIN_IN (0) or PORT_PIN_OUT (0x80) is passed as the input parameter Direction for the Port_SetPinDirection API.
Exporting the configuration using EB tresos tool and then importing the configuration in another AUTOSAR configuration tool results in PortPinMode configuration not updated for read only port pins.	PortPinMode configuration parameter is restricted to configure for read-only ports in EB Tresos tool. Exporting the configuration from EB Tresos tool to AUTOSAR based arxml format, the PortPinMode parameter will not be populated in the arxml file for the read-only ports. As a workaround user can manually add the PortPinMode parameter for the read-only ports (for example, TC39x: Port 40, 41 and 20.2) after importing the arxml configuration in the AUTOSAR configuration tool. Note: PortPinMode is unused for read only ports and needed to be added only for schema consistency

19.3.9 Unsupported hardware features

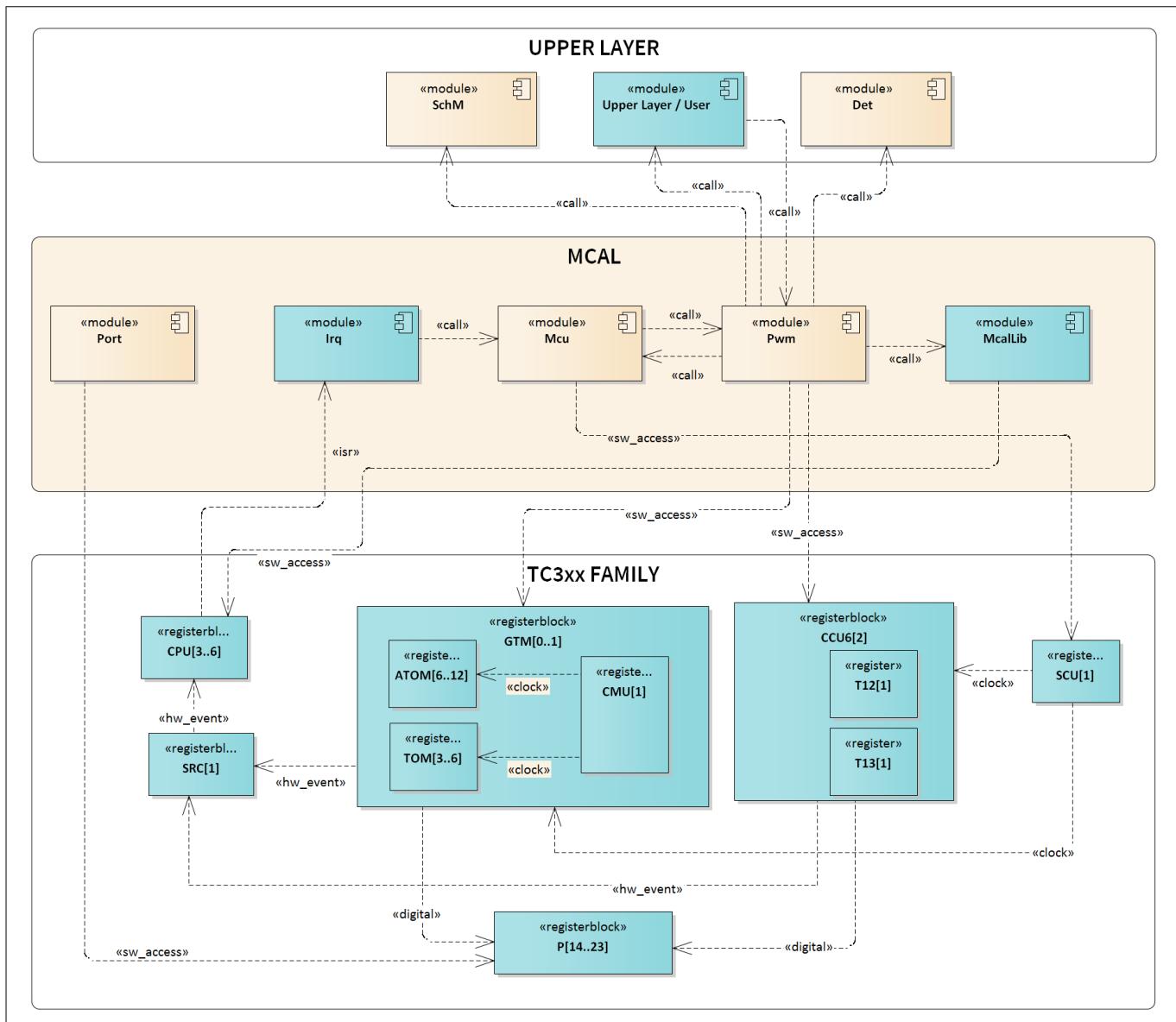
All the features of the PORT driver are supported.

Pwm_17_GtmCcu6 driver**20 Pwm_17_GtmCcu6 driver****20.1 User information****20.1.1 Description**

The pulse width modulation (PWM) driver is responsible for providing standard services related to the PWM signal generation specified by AUTOSAR. The underlying timer engine of a PWM channel is a GTM (TOM or ATOM slice) or CCU6 (T12 or T13 slice) timer channel. The PWM driver provides UI options to configure the driver parameters as described in the AUTOSAR PWM specification (Version AS 4.2.2) and additional parameters to configure the timer engine. The parameters of the GTM or CCU6 timer slice that must be configured are described in the MCU driver chapter.

20.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

Pwm_17_GtmCcu6 driver

Figure 168 **Mapping of hardware-software interfaces**

20.1.2.1 GTM: primary hardware peripheral

Hardware functional features

The PWM driver uses the TOM/ATOM sub-module of GTM for generating output signal for a configured duty cycle and period.

The PWM driver uses the TOM/ATOM sub-module for period, duty cycle and polarity updates.

The key hardware GTM features used by the driver are:

- Synchronous start: Host trigger feature is used if a synchronous start is desired
- Signaling mechanisms: Global enable/disable mechanism, output enable mechanism
- Trigger mechanisms: Host CPU, the internal trigger signal (from current channel)
- Continuous counting up mode
- ATOM operation mode: Signal output mode PWM (SOMP)

The unsupported features of the GTM are:

Pwm_17_GtmCcu6 driver

- Global force update signaling mechanism
- The TBU time stamp trigger mechanism
- Continuous counting up-down mode, one shot up mode, one shot down mode
- ATOM operational mode: SOMI/SOMC/SOMS/SOMB
- Pulse count modulation mode
- Trigger generation (shadow register is used to define additional output signal and interrupt trigger event)

Users of the hardware

The TOM/ATOM channel of the GTM is exclusively used by the PWM driver. The GTM TOM or ATOM channel can be used by the PWM, GPT, ADC, OCU, DSADC or WDG driver.

The MCU driver initializes the GTM clocks and provides APIs to program the GTM SFRs. The PWM driver uses these APIs to write the GTM SFRs. Additionally, updates to channel-specific SFRs are performed by the PWM driver. Since these channels are exclusively reserved for the PWM driver, access to the channel-specific SFRs from other drivers or user software is not allowed. The MCU driver is responsible to route the GTM interrupt to the PWM driver.

Hardware diagnostic features

Not applicable.

Hardware events

The PWM driver uses the following hardware events from the TOM/ATOM sub-module of the GTM IP:

- period match events generated by the timer channel
- compare match events generated by the timer channel

20.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The PWM driver depends on the SCU IP for the clock functionality. The driver requires the fSPB, fGTM and fCCU6 clock signals for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver, and is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the PWM driver.

Hardware events

Hardware events from the SCU are not used by the PWM driver.

20.1.2.3 Port: dependent hardware peripheral

Hardware functional features

The output signals are routed to the GTM and CCU6 through the port pads. The port pads are configured and enabled through the PORT driver.

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Not applicable.

Pwm_17_GtmCcu6 driver

20.1.2.4 CCU6: primary hardware peripheral

Hardware functional features

The PWM driver uses the CCU6 for generating output signal for a configured duty cycle and period.

The main features of the CCU6 functional block configured and accessed by the PWM driver are period, duty cycle and polarity update.

The key hardware CCU6 features used by the driver are:

- Edge-aligned operating mode
- Compare mode
- Synchronous start: Host trigger feature is used if a synchronous start is required a. for T12 channels b. from T12 to T13 channel

The unsupported features of the CCU6 are:

- Hall sensor mode
- Trap handling of CCU6
- Capture mode for T12 and T13
- Multi-channel mode

Users of the hardware

The T12/T13 channel of the CCU6 is exclusively used by the PWM driver. The CCU6 T12 or T13 channel can be used by the PWM, ICU driver.

The MCU driver provides APIs to program the CCU6 SFRs. The PWM driver uses these APIs to write the CCU6 SFRs. Additionally, updates to channel-specific SFRs are performed by the PWM driver. Since these channels are exclusively reserved for the PWM driver, access to the channel-specific SFRs from other drivers or user software is not allowed. The MCU driver is responsible to route the CCU6 interrupt to the PWM driver.

Hardware diagnostic features

Not applicable.

Hardware events

The PWM driver uses the following hardware events from the CCU6 IP:

- period match events generated by the timer channel
- compare match events generated by the timer channel

20.1.3 File structure

20.1.3.1 C file structure

The section provides details on the C files of the PWM driver.

Pwm_17_GtmCcu6 driver

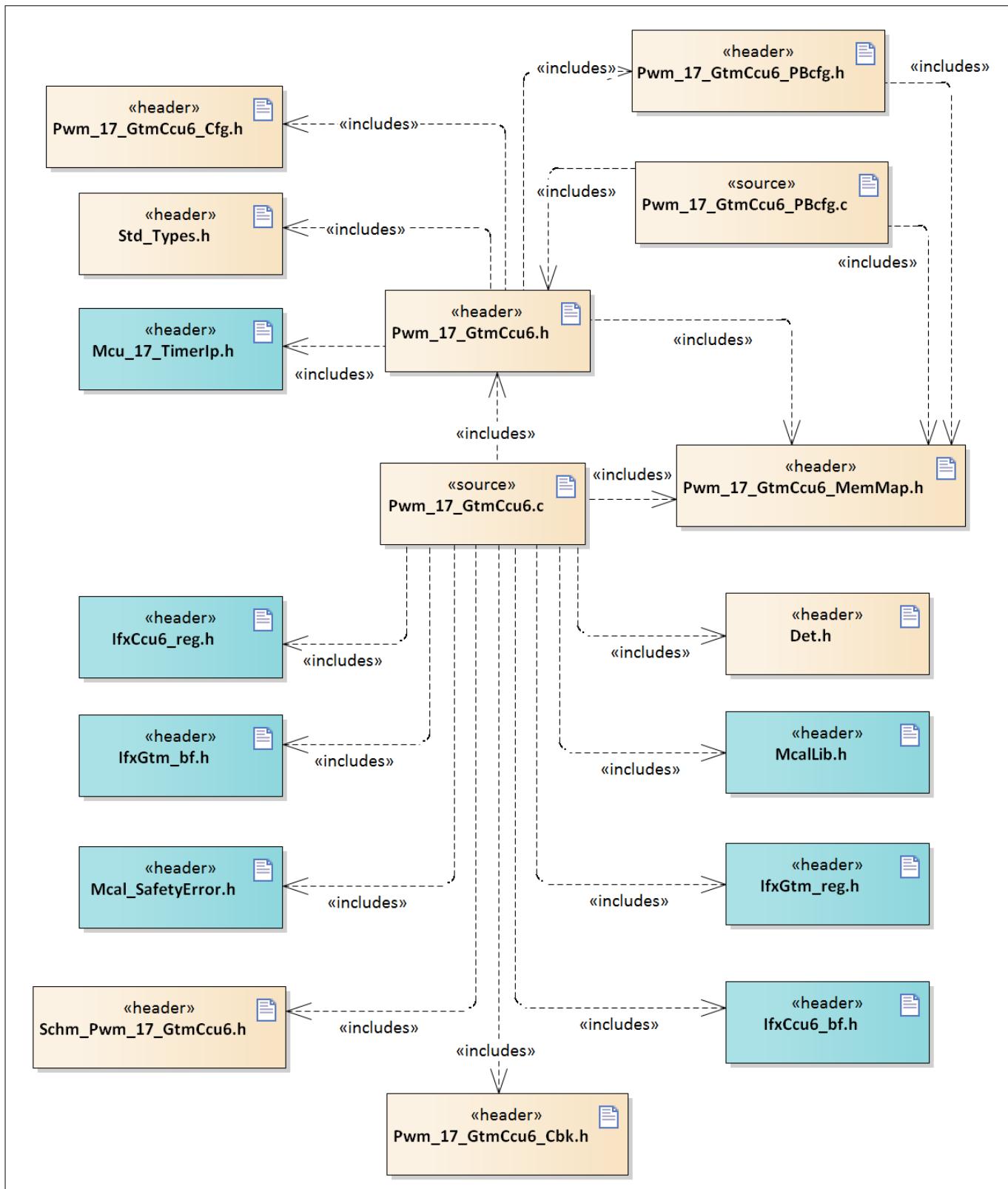


Figure 169

C file structure

Pwm_17_GtmCcu6 driver

Table 1666 C file structure

File name	Description
Det.h	Provides the exported interfaces of Development Error Tracer
IfxCcu6_bf.h	SFR header file for CCU6
IfxCcu6_reg.h	SFR header file for CCU6
IfxGtm_bf.h	SFR header file for GTM
IfxGtm_reg.h	SFR header file for GTM
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Pwm_17_GtmCcu6.c	Static source code containing API definition
Pwm_17_GtmCcu6.h	Header file (Static) defining prototypes of data structures and APIs. Header file implements all pre-processor directives.
Pwm_17_GtmCcu6_Cbk.h	Includes callback header definition
Pwm_17_GtmCcu6_Cfg.h	Header file (Generated) containing constants and pre-processor macros
Pwm_17_GtmCcu6_MemMap.h	Mapping of code and data (variables, constant variables) to specific memory sections
Pwm_17_GtmCcu6_PBcfg.c	File (Generated) containing objects to data structures
Pwm_17_GtmCcu6_PBcfg.h	Code template header file for the PWM driver that should be included by the application
Schm_Pwm_17_GtmCcu6.h	PWM critical sections are declared in this file
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

20.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the PWM driver.

Pwm_17_GtmCcu6 driver

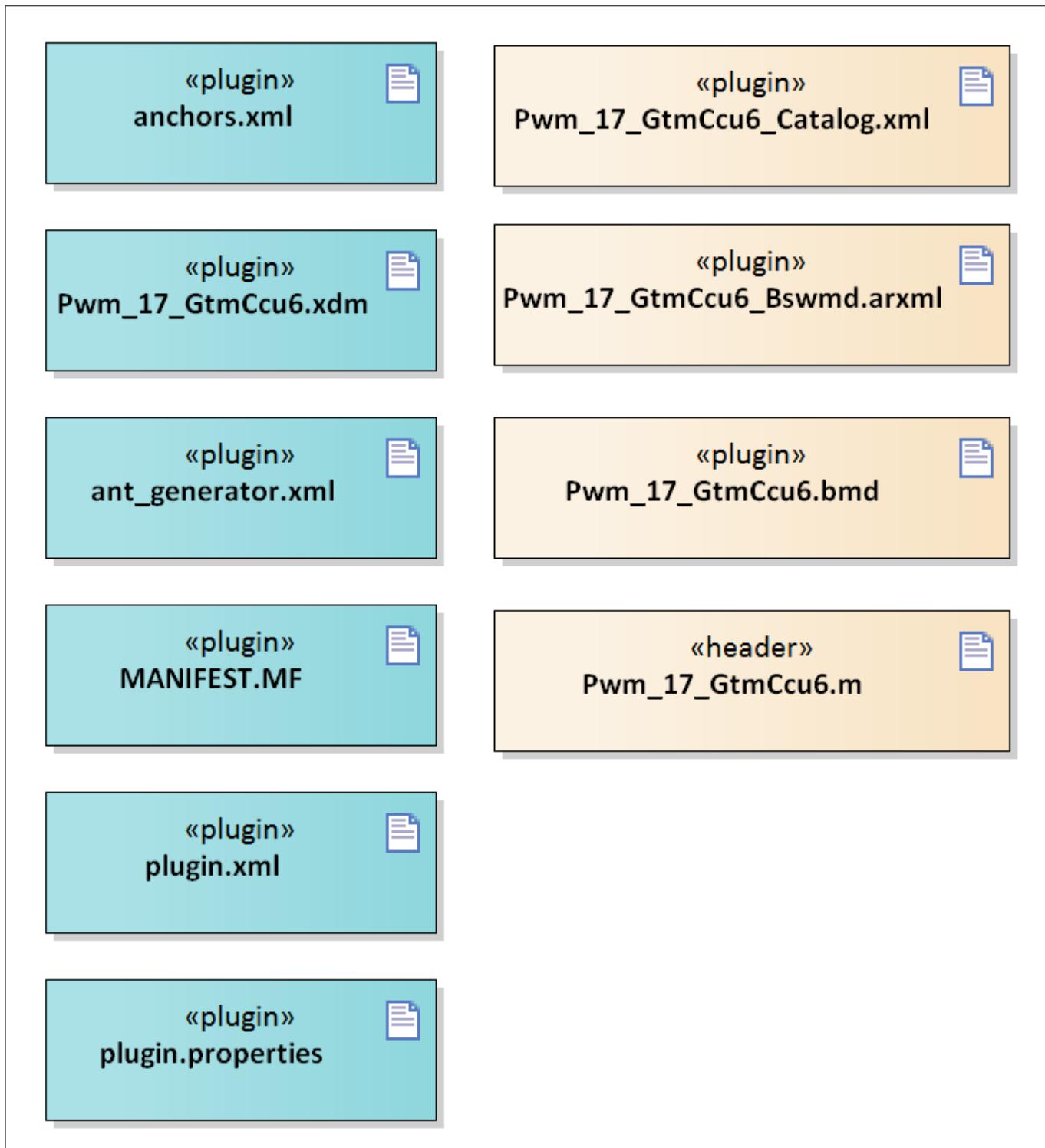


Figure 170 **Code generator plugin files**

Table 1667 **Code generator plugin files**

File name	Description
MANIFEST.MF	Tresos plugin support file containing the meta data for the PWM driver
Pwm_17_GtmCcu6.bmd	AUTOSAR format XML data model schema file (for each device)

Pwm_17_GtmCcu6 driver

Table 1667 Code generator plugin files (continued)

File name	Description
Pwm_17_GtmCcu6.m	File contains macros for the PWM code generation
Pwm_17_GtmCcu6.xdm	Tresos format XML data model schema file
Pwm_17_GtmCcu6_Bswmd.arxml	AUTOSAR format module description file
Pwm_17_GtmCcu6_Catalog.xml	AUTOSAR format catalog file
anchors.xml	Tresos anchors support file for the PWM driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point
plugin.properties	Tresos plugin support file for the PWM driver
plugin.xml	Tresos plugin support file for the PWM driver

20.1.4 Integration hints

This section lists the key points that an integrator or user of the PWM driver must consider.

20.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of MCAL, but are required to integrate the PWM driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of the ECU. Specifically, in the context of MCAL, the EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

Initialization of PWM:

The user of PWM driver may use APIs of EcuM to initialize the driver. The initialization of the PWM driver should be invoked from each CPU core, which intends to use the services of the PWM driver. All cores can execute initialization simultaneously.

De-initialization of PWM:

The user of PWM driver may use APIs of EcuM to de-initialize the driver. The de-initialization of the PWM driver should be invoked from each CPU core that uses the services of the PWM driver. All cores can execute de-initialization simultaneously.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the Pwm_17_GtmCcu6_MemMap.h file.

The Pwm_17_GtmCcu6_MemMap.h file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the

Pwm_17_GtmCcu6 driver

elements are relocated to the correct memory region. A sample implementation listing the memory-section macros is shown as follows:

```
***** GLOBAL RAM DATA -- NON-CACHED LMU *****

#if defined PWM_17_GTMCCU6_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
*****User pragmas here for Non-cached LMU*****
#undef PWM_17_GTMCCU6_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR
#elif defined PWM_17_GTMCCU6_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#ifndef _TASKING_C_TRICORE_
*****User pragmas here for Non-cached LMU*****
#endif
#undef PWM_17_GTMCCU6_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

***** CORE[x] CONFIG DATA -- PF[x] ****/* [x]=0..5*/
#if defined PWM_17_GTMCCU6_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
*****User pragmas here for PF[x]*****
#undef PWM_17_GTMCCU6_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#undef MEMMAP_ERROR
#elif defined PWM_17_GTMCCU6_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
*****User pragmas here for PF[x]*****
#undef PWM_17_GTMCCU6_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#undef MEMMAP_ERROR
***** CODE -- PF[x] ****/
#if defined PWM_17_GTMCCU6_START_SEC_CODE_ASIL_B_GLOBAL
*****User pragmas here for PF[x]*****
#undef PWM_17_GTMCCU6_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#elif defined PWM_17_GTMCCU6_STOP_SEC_CODE_ASIL_B_GLOBAL
*****User pragmas here for PF[x]*****
#undef PWM_17_GTMCCU6_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#endif
#if defined MEMMAP_ERROR
#error "Pwm_17_GtmCcu6_MemMap.h, wrong pragma command"
#endif

```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The PWM driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the PWM driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

Pwm_17_GtmCcu6 driver

The DEM module is not required for the integration of the PWM driver.

- **SchM**

The SchM module is a part of the RTE that manages the BSW scheduler. The PWM driver uses the exclusive areas defined in the `SchM_Pwm_17_GtmCcu6.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the PWM driver are:

HandleNotification critical section to protect update of notification related global variable and SFRs

PeriodAndDutyUpdate critical section to protect update of period and duty related shadow SFRs

The `SchM_Pwm_17_GtmCcu6.h` and `SchM_Pwm_17_GtmCcu6.c` files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the PWM driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows:

```
***** Sample implementation of SchM_Pwm_17_GtmCcu6.c ****/
#include "Os.h"
void SchM_Enter_Pwm_17_GtmCcu6_HandleNotification(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();/* Suspend CPU core interrupt */
}
void SchM_Exit_Pwm_17_GtmCcu6_HandleNotification(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts();/* Resume CPU core interrupt */
}
void SchM_Enter_Pwm_17_GtmCcu6_PeriodAndDutyUpdate(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts();/* Suspend CPU core interrupt */
}
void SchM_Exit_Pwm_17_GtmCcu6_PeriodAndDutyUpdate(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts();/* Resume CPU core interrupt */
}
```

- **Safety error**

The PWM driver reports all the detected safety errors through the `Mcal_ReportSafetyError()` API.

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The PWM driver does not implement any notifications. However, the PWM driver reports the rising/falling/both edges through notification functions. These notification functions can be configured by the user in the EB Tresos for each PWM channel separately.

The PWM driver does not expect any callbacks from application but it needs a callback from the MCU module for ISR handling.

Pwm_17_GtmCcu6 driver

- **Operating system (OS)**

The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application.

The OS files provided by the MCAL package is only an example code and must be updated by the integrator with the actual OS files for the desired function.

20.1.4.2 Multicore and Resource Manager

The PWM driver supports execution of its APIs simultaneously from all CPU cores. The user should allocate logical PWM channels to the CPU cores at pre-compile time using the Resource manager module. The following are the key points to be considered with respect to multicore in the driver:

- Logical PWM channels are allocated to CPU cores at pre-compile time. For example, Pwm_Channel0, Pwm_Channel1.
- It must be ensured that PWM channel id passed as parameter while invoking an API belongs to the same core.
- DETs are raised in case APIs are invoked with mismatch of core and channel id.
- Interrupts raised by a PWM channel must be serviced by the CPU core to which the channel has been allocated to.
- PWM channels using GTM-ATOM, channel GTM-ATOM[i]_CH[X] and ATOM[i]_CH[x+1] must be allocated to the same core as these two channels share the same interrupt line.
- PWM channels using GTM-TOM, channel GTM-TOM[i]_CH[X] and TOM[i]_CH[x+1] must be allocated to the same core as these two channels share the same interrupt line.
- Locating constants, variables and configuration data to the correct memory space should be done by the user. Memory sections are marked GLOBAL (common to all cores) and CORE[x](specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of the PWM driver is placed under single MemMap section. It can be relocated to any PFlash.

Data section:

The RAM variable memory sections marked as specific to core should be relocated to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region.

Configuration data and constants:

The configuration data section sections marked as specific to core should be relocated to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

Note: Relocating code, data and constants to a distant memory space would impact execution timings.

Note: If the driver operates from single (master) core, all the sections may be relocated to the PFlash/ DSPR/DLMU of the same CPU core.

20.1.4.3 MCU support

The PWM driver is dependent on the MCU driver for clock configuration and timer IP-related services. The initialization of the PWM driver must be started only after completing the MCU initialization. The following must be considered while configuring the MCU driver in tresos:

- The GTM/CCU6 hardware timers used by the PWM driver must be reserved in the MCU configuration for exclusive use by the PWM.

Pwm_17_GtmCcu6 driver

Access of shared GTM SFRs

If channels of same the TOM/ATOM module are shared between the application and the PWM driver, the user shall ensure that shared register of TOM/ATOM modules are accessed using the MCU timer IP library APIs.

20.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure the port pins used by the PWM driver through the PORT configuration and initialize the port pins prior to invoking the PWM initialization.

20.1.4.5 DMA support

The PWM driver does not use any services provided by the DMA driver.

20.1.4.6 Interrupt connections

The interrupt connections of the PWM driver are described in this section.

If the user wants notifications, the user should enable interrupts in the Interrupt configuration register. The interrupt configuration registers of different hardware used by PWM channels are as follows.

Table 1668 GTM interrupts

GTM hardware used	SRC register
GTM-TOM	SRC_GTMTO _w x (w= TOM module; x= TOM channel)
GTM-ATOM	SRC_GTMAT _w M (w= ATOM module; x= ATOM channel)

Table 1669 CCU6 interrupts

CCU6 kernel used	CCU6 comparator used	SRC register
CCU60	CCU60	SRC_CCU60SR0
CCU60	CCU61	SRC_CCU60SR1
CCU60	CCU62	SRC_CCU60SR2
CCU60	CCU63	SRC_CCU60SR3
CCU61	CCU60	SRC_CCU61SR0
CCU61	CCU61	SRC_CCU61SR1
CCU61	CCU62	SRC_CCU61SR2
CCU61	CCU63	SRC_CCU61SR3

The MCU driver through its timer-related services provides interrupt handler for GTM and CCU6 timers.

All the ISRs to the GTM must be routed to the `Mcu_17_Gtm_TomChannelIsr` or `Mcu_17_Gtm_AtomChannelIsr` API, which further invokes `Pwm_17_GtmCcu6_Timer_Isr`.

All the ISRs to the CCU6 must be routed to the `Mcu_17_Ccu6_ChannelIsr` API, which further invokes `Pwm_17_GtmCcu6_Timer_Isr`. An example ISR handling is shown as follows.

Pwm_17_GtmCcu6 driver

```
/* include MCU timer header file */
#include "Mcu_17_TimerIp.h"
*****SRC_ GTMTOM0SR0*****
ISR(GTMTOM0SR0_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();

    /* Parameter is Channel Number */
    Mcu_17_Gtm_TomChannelIsr (TOM_MODULE_0 , TOM_CHANNEL_0);
    /* TOM_MODULE_0 = 0, TOM_CHANNEL_0 = 0 */
}

*****SRC_ CCU60SR0*****
ISR(CCU60SR0_ISR){
    /* Enable Global Interrupts */
    ENABLE();

    /* Parameter is Channel Number */
    Mcu_17_Ccu6_ChannelIsr(CCU6_KERNEL_0,CCU6_CHANNEL_0);
}
```

Pwm_17_GtmCcu6 driver

20.1.4.7 Example usage

Driver initialization

The following code listing shows the steps involved in the initialization of the PWM driver.

```
#include "Mcu.h"
#include "Irq.h"
#include "Port.h"
#include "Pwm_17_GtmCcu6.h"

Mcu_Init(&Mcu_Config);
Mcu_InitClock(0);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();
/* Configure Interrupt priority */
IrqGtm_Init();
/* Port Initialization */
Port_Init(&Port_ConfigRoot);
/* Pwm Initialization */
Pwm_17_GtmCcu6_Init(&Pwm_17_GtmCcu6_Config);
```

After invoking the `Pwm_17_GtmCcu6_Init` API, the PWM output signals are generated for the configured duty and period.

Note:

- User must ensure that the `Pwm_17_GtmCcu6_Init` API is called before using any other runtime APIs provided by the PWM driver.

Driver de-initialization

The following code listing depicts the steps involved in the de-initialization of the PWM driver.

Note: De-initializing the PWM driver will put all the channels to the idle state.

```
/* De-initialize PWM driver */
Pwm_17_GtmCcu6_DeInit();
/* Pwm Channel output is set to Idle */
```

Set duty cycle

The following code snippet shows invoking of the `Pwm_17_GtmCcu6_SetDutyCycle` API.

```
/* Change Duty to 25% when PwmDutyShiftInTicks is OFF*/
Pwm_17_GtmCcu6_SetDutyCycle(0,0x2000);
/* Change Duty to 25% when PwmDutyShiftInTicks is ON*/
/* For channel 0, the period is 48, 25% would result in 12 ticks */
Pwm_17_GtmCcu6_SetDutyCycle(0,12);
```

Set period and duty cycle

Pwm_17_GtmCcu6 driver

The following code snippet shows invoking of Pwm_17_GtmCcu6_SetDutyCycle API.

```
/* Change Duty to 25% and period 60 ticks for channel 4 when
PwmDutyShiftInTicks is OFF */
Pwm_17_GtmCcu6_SetPeriodAndDuty(4, 60, 0x2000);
/* Change Duty to 25% and period 60 ticks for channel 4 when
PwmDutyShiftInTicks is ON */
/* For Period 60, 25% would result in 15 ticks */
Pwm_17_GtmCcu6_SetPeriodAndDuty(4, 60, 15);
```

Notification, set idle and get output state

The following code listing shows invoking of the notification APIs, Pwm_17_GtmCcu6_SetOutputToIdle and Pwm_17_GtmCcu6_GetOutputState API.

```
/* Pwm Initialization */
Pwm_17_GtmCcu6_Init(&Pwm_Config);
/* Enable Notification for channel 3 */
Pwm_17_GtmCcu6_EnableNotification(3, PWM_RISING_EDGE);
/* Disable Notification for channel 3 */
Pwm_17_GtmCcu6_DisableNotification(3);
/* Set Output of channel 3 to Idle */
Pwm_17_GtmCcu6_SetOutputToIdle(3);
/* Get Output State of channel 3 */
/* Here variable 'State' holds the output state of channel 3 */
State = Pwm_17_GtmCcu6_GetOutputState(3);
```

20.1.5 Key architectural considerations

20.1.5.1 User mode support

The PWM driver operates in both User-1 and Supervisor modes without the need of any configuration parameter to configure the behavior.

[cover parentID PWM={1A65EADD-AFD0-4845-B2D2-8257E086DD67}]

[cover parentID PWM={ED41EDC1-CB6C-4821-BED8-1735365FE93D}]

Pwm_17_GtmCcu6 driver

20.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Channel allocation to cores**

Integrator shall ensure the following points while allocating channels to cores:

- If the PwmHandleShiftByOffset configuration parameter is set to ON, all GTM channels of TGC/AGC should be allocated to the same core. This will enable synchronous start of all channels in same TGC/AGC.
- For PWM channels of type GTM, Channel x and Channel x+1 of same TGC/AGC shall be allocated to same core, as they share same interrupt node.
- To enable synchronous start of CCU6 channels, all CCU6 channels of the kernel shall be allocated to same core.

[cover parentID PWM={A1C61D00-C7D6-49cc-B0AE-CEF01CC900D4}]

- **Config pointer to initialization**

When the PWM driver is used in the multicore environment, user has to ensure that the same configuration pointer is passed to the initialization function from different cores.

[cover parentID PWM={170B536A-4745-4890-87FD-6155B7B45F0E}]

- **Notification when duty is 0% or 100%**

Decision: When notification for duty 0% or 100% is enabled by the user for fixed and variable period channels then the following notification combination shall be used.

- Duty is 0%:
 - Polarity - high; notify value - rising edge
 - Polarity - low; notify value - falling edge
- Duty is 100%:
 - Polarity - high; notify value - falling edge
 - Polarity - low; notify value - rising edge

The notify value is the input parameter for the Pwm_17_GtmCcu6_EnableNotification function.

Rational: If both edges are enabled for 0% /100 % duty, the differentiation of edges is not possible.

[cover parentID PWM={B1EE9B66-F309-4a94-BBCC-8487D15184C3}]

- **ShiftOffset = ON for shifted channels**

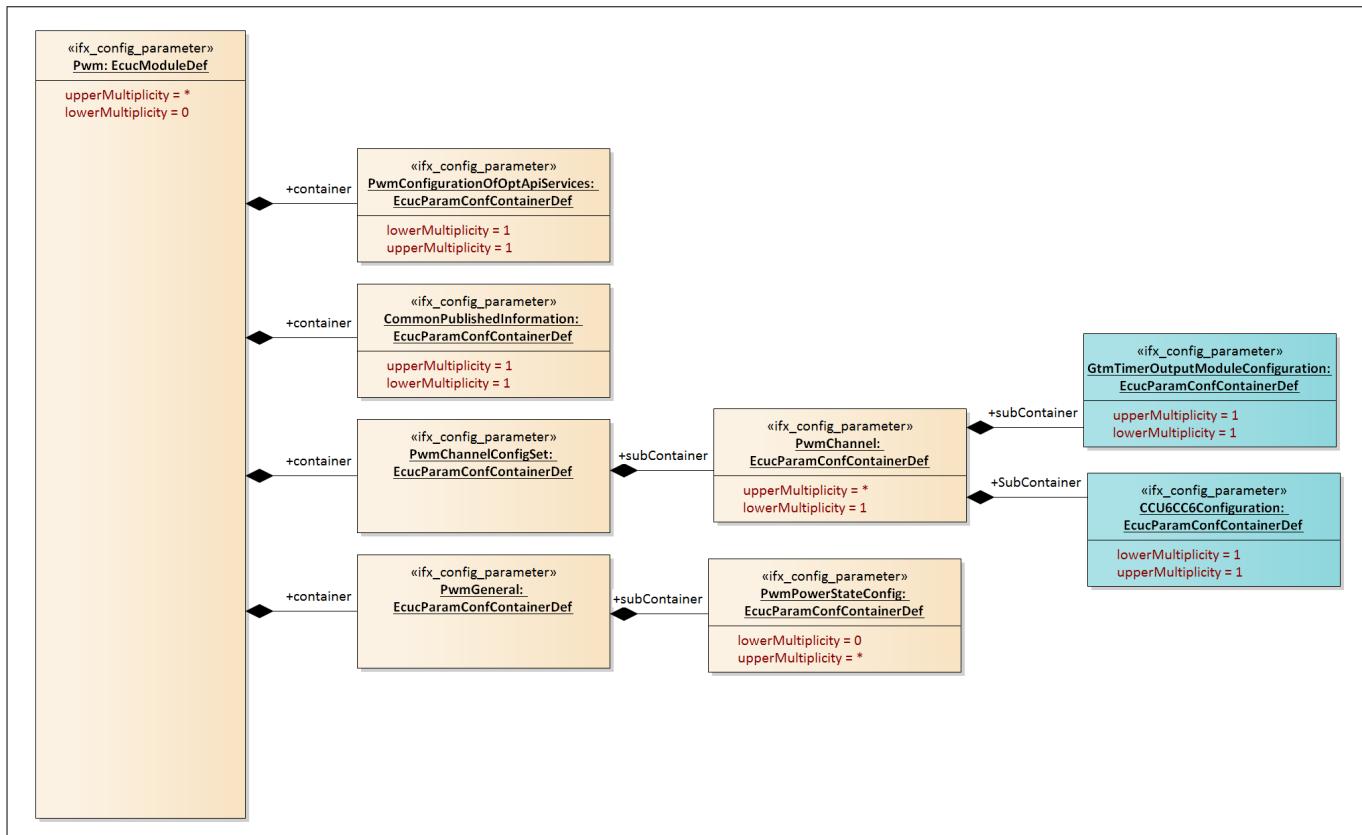
If the shifted PWM channels are used then it is recommended to set the PwmHandleShiftByOffset configuration parameter to ON. If PwmHandleShiftByOffset is set to OFF, the shifted period channels start late compared to the reference channel due to which the first period of shifted period channels may not match the period of the reference channel.

[cover parentID PWM={05182FFE-74CA-4bf4-AE87-BD22DE99E628}]

Pwm_17_GtmCcu6 driver

20.3 Reference information

20.3.1 Configuration interfaces


Figure 171 Container hierarchy along with their configuration parameters

20.3.1.1 Container: CCU6CC6Configuration

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

20.3.1.1.1 CCU6KernelUsed

Table 1670 Specification for CCU6KernelUsed

Name	CCU6KernelUsed		
Description	Lists all the CCU6 kernels Note: <ul style="list-style-type: none"> - CCU6 resource allocation is done at the kernel level for each module Once the kernel is allocated to the module, the T12 or T13 timer of the kernel can only be assigned to one of the channels within the module. - Default value is set to blank as user has to select the appropriate reference value from the MCU driver. 		
Multiplicity	1..1	Type	EcucReferenceDef

Pwm_17_GtmCcu6 driver
Table 1670 Specification for CCU6KernelUsed (continued)

Range	Reference to Node: McuCcu6ModuleAllocationConf		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.1.2 CCU6TimerClockSelect
Table 1671 Specification for CCU6TimerClockSelect

Name	CCU6TimerClockSelect		
Description	Selects the clock source for T12 or T13 timer. Note: The default value is CCU6_CONFIGURABLE_CLOCK_0 as it is the lowest configurable divider value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CCU6_CONFIGURABLE_CLOCK_0: Selected clock is fcc6 or fcc6/256 (pre scalar enabled). Note: The fcc6 is the module clock for the CCU6 kernel. For more information, refer to the Hardware Target Specification. CCU6_CONFIGURABLE_CLOCK_1: Selected clock is fcc6/2 or fcc6/512 (prescalar enabled) CCU6_CONFIGURABLE_CLOCK_2: Selected clock is fcc6/4 or fcc6/1024 (prescalar enabled) CCU6_CONFIGURABLE_CLOCK_3: Selected clock is fcc6/8 or fcc6/2048 (prescalar enabled) CCU6_CONFIGURABLE_CLOCK_4: Selected clock is fcc6/16 or fcc6/4096 (prescalar enabled) CCU6_CONFIGURABLE_CLOCK_5: Selected clock is fcc6/64 or fcc6/16384 (prescalar enabled) CCU6_CONFIGURABLE_CLOCK_6: Selected clock is fcc6/32 or fcc6/8192 (prescalar enabled) CCU6_CONFIGURABLE_CLOCK_7: Selected clock is fcc6/128 or fcc6/32768 (prescalar enabled)		
Default value	CCU6_CONFIGURABLE_CLOCK_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Pwm_17_GtmCcu6 driver
20.3.1.1.3 CCU6TimerPrescalarEnabled
Table 1672 Specification for CCU6TimerPrescalarEnabled

Name	CCU6TimerPrescalarEnabled		
Description	<p>If CCU6TimerPrescalarEnabled is FALSE then T12 or T13 clock = fcc6 divided by (2 power CCU6TimerClockSelect)</p> <p>If CCU6TimerPrescalarEnabled is TRUE then T12 or T13 clock = fcc6 divided by (2 power (CCU6TimerClockSelect+8))</p> <p>Note: The default value is set to FALSE as the hardware default value is FALSE.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.1.4 CCU6TimerUsed
Table 1673 Specification for CCU6TimerUsed

Name	CCU6TimerUsed		
Description	<p>Selects the T12 or T13 timer.</p> <p>Note: The default value is T12 as it is the lowest configurable timer for the kernel.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>T12: T12: selected T12 timer in the Ccu6x kernel</p> <p>T13: T13: selected T13 timer in the CCU6x kernel</p>		
Default value	T12		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	CCU6KernelUsed		

Pwm_17_GtmCcu6 driver
20.3.1.1.5 Cc6xChannel
Table 1674 Specification for Cc6xChannel

Name	Cc6xChannel		
Description	Selects a CC6x channel. Note: The default value is chosen as Cc60 as it is the CCU6 lowest comparator.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	Cc60: CC60 comparator selected Cc61: CC61 comparator selected Cc62: CC62 comparator selected		
Default value	Cc60		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.2 Container: CommonPublishedInformation

Contains published information about vendor and versions

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

20.3.1.2.1 ArMajorVersion
Table 1675 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	Provides the major version of the AUTOSAR specification		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Pwm_17_GtmCcu6 driver

20.3.1.2.2 ArMinorVersion

Table 1676 Specification for ArMinorVersion

Name	ArMinorVersion		
Description	Provides the minor version of the AUTOSAR specification		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.2.3 ArPatchVersion

Table 1677 Specification for ArPatchVersion

Name	ArPatchVersion		
Description	Provides the patch version of the AUTOSAR specification		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.2.4 ModuleId

Table 1678 Specification for ModuleId

Name	ModuleId		
Description	Provides the module Id		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		

Pwm_17_GtmCcu6 driver
Table 1678 Specification for ModuleId (continued)

Default value	121		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.2.5 Release
Table 1679 Specification for Release

Name	Release		
Description	Indicates the TC3xx device derivative used for the implementation		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per the hardware derivate		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.2.6 SwMajorVersion
Table 1680 Specification for SwMajorVersion

Name	SwMajorVersion		
Description	Module Major version		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 255		
Default value	As per Driver Version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

Pwm_17_GtmCcu6 driver
Table 1680 Specification for SwMajorVersion (continued)

Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.2.7 SwMinorVersion
Table 1681 Specification for SwMinorVersion

Name	SwMinorVersion		
Description	Provides the minor version of the software		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver Version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.2.8 SwPatchVersion
Table 1682 Specification for SwPatchVersion

Name	SwPatchVersion		
Description	Provides the patch version of the software		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver Version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Pwm_17_GtmCcu6 driver

20.3.1.2.9 VendorApiInfix

Table 1683 Specification for VendorApiInfix

Name	VendorApiInfix		
Description	Provides the vendor-specific name		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	GtmCcu6		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.2.10 VendorId

Table 1684 Specification for VendorId

Name	VendorId		
Description	Provides the Infineon vendor ID in the HIS software specification		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.3 Container: GtmTimerOutputModuleConfiguration

This container contains the elements for configuring the GTM timer hardware (TOM or ATOM). The settings in this container are used to configure the timing needs of the TOM or ATOM timer.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

Pwm_17_GtmCcu6 driver

20.3.1.3.1 GtmTimerClockSelect

Table 1685 Specification for GtmTimerClockSelect

Name	GtmTimerClockSelect		
Description	Selects the clock source for the TOM or ATOM timer. The default value is GTM_FIXED_CLOCK_0 as it is the lowest configurable divider value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_CONFIGURABLE_CLOCK_0: Configurable Clock 0 is selected for the ATOM module. GTM_CONFIGURABLE_CLOCK_1: Configurable Clock 1 is selected for the ATOM module. GTM_CONFIGURABLE_CLOCK_2: Configurable Clock 2 is selected for the ATOM module. GTM_CONFIGURABLE_CLOCK_3: Configurable Clock 3 is selected for the ATOM module. GTM_CONFIGURABLE_CLOCK_4: Configurable Clock 4 is selected for the ATOM module. GTM_CONFIGURABLE_CLOCK_5: Configurable Clock 5 is selected for the ATOM module. GTM_CONFIGURABLE_CLOCK_6: Configurable Clock 6 is selected for the ATOM module. GTM_CONFIGURABLE_CLOCK_7: Configurable Clock 7 is selected for the ATOM module. GTM_FIXED_CLOCK_0: Fixed Clock 0 is selected for the TOM module. GTM_FIXED_CLOCK_1: Fixed Clock 1 is selected for the TOM module. GTM_FIXED_CLOCK_2: Fixed Clock 2 is selected for the TOM module. GTM_FIXED_CLOCK_3: Fixed Clock 3 is selected for the TOM module. GTM_FIXED_CLOCK_4: Fixed Clock 4 is selected for the TOM module.		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PwmAssignedHwUnit		

20.3.1.3.2 GtmTimerPortPinSelect

Table 1686 Specification for GtmTimerPortPinSelect

Name	GtmTimerPortPinSelect		
Description	Selects the output port and pin for the TOM or ATOM timer. Range: The available port pins depends on target device and the selected timer unit. For more information, refer to the Hardware Target Specification Note: The default value is set to NONE as user has to select the appropriate port reference.		

Pwm_17_GtmCcu6 driver
Table 1686 Specification for GtmTimerPortPinSelect (continued)

Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NONE: No port pin is selected. TOUT<w>_SEL<x>_PORT<y>_PIN<z>; w: Tout Number x: Alternate selection (A-L) y: Port number z: Pin number		
Default value	NONE		
Post-build variant value	TRUE		
Value configuration class	Post-Build		
Origin	IFX		
Dependency	PwmAssignedHwUnit		

20.3.1.3.3 GtmTimerUsed
Table 1687 Specification for GtmTimerUsed

Name	GtmTimerUsed		
Description	The TOM or ATOM channel resource assigned to the PWM channel. This parameter lists all the GTM timer channels (TOM or ATOM). The referred timer channel in the MCU should have TomChannelUsage/AtomChannelUsage as USED_BY_PWM_DRIVER. Note: The default value is set to BLANK as user has to select the appropriate reference value from the MCU driver.		
Multiplicity	1..1	Type	EcucChoiceReferenceDef
Range	Reference to Node: McuGtmAtomChannelAllocationConf, McuGtmTomChannelAllocationConf		
Default value	NULL		
Post-build variant value	TRUE		
Value configuration class	Post-Build		
Origin	IFX		
Dependency	PwmAssignedHwUnit		

Pwm_17_GtmCcu6 driver

20.3.1.4 Container: Pwm

This container contains the configurations of the PWM driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

20.3.1.4.1 Config Variant

Table 1688 Specification for Config Variant

Name	Config Variant		
Description			
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VariantPostBuild: Post Build Support		
Default value	VariantPostBuild		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.5 Container: PwmChannel

This container contains the configuration of an individual PWM channel.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

20.3.1.5.1 PwmAssignedHwUnit

Table 1689 Specification for PwmAssignedHwUnit

Name	PwmAssignedHwUnit		
Description	Hardware resource used for PWM generation: GTM or CCU6. Note: This parameter will hold only CCU6 option for GTM-less devices. In this case default value will be CCU6.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	CCU6: CCU6 timer selected. GTM: GTM timer selected.		
Default value	GTM		

Pwm_17_GtmCcu6 driver
Table 1689 Specification for PwmAssignedHwUnit (continued)

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.5.2 PwmChannelClass
Table 1690 Specification for PwmChannelClass

Name	PwmChannelClass		
Description	Class of PWM channel		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	PWM_FIXED_PERIOD: The PWM channel has a fixed period. Only duty cycle can be changed. PWM_FIXED_PERIOD_CENTER_ALIGNED: This class of PWM channel refers to a PWM_FIXED_PERIOD channel for the period and the duty cycle will fall exactly on the middle of the period. Only duty cycle can be changed for this channel class. Note: PWM_FIXED_PERIOD_CENTER_ALIGNED channel is applicable when GTM hardware is available in the device. PWM_FIXED_PERIOD_SHIFTED: This class of PWM channel refers to a PWM_FIXED_PERIOD channel for the period and the duty cycle is shifted by a required fixed percentage. Only duty cycle can be changed. PWM_VARIABLE_PERIOD: This class of PWM channel has a variable period and duty cycle. Both can be changed during run time. This channel should not be referred by any other channel and does not refer to any channel either. Note: PWM_VARIABLE_PERIOD is not applicable if PwmSetPeriodAndDuty is set to false		
Default value	PWM_FIXED_PERIOD		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PwmSetPeriodAndDuty		

Pwm_17_GtmCcu6 driver

20.3.1.5.3 PwmChannelId

Table 1691 Specification for PwmChannelId

Name	PwmChannelId		
Description	<p>Logical channel Id of the PWM channel. This value is assigned to the symbolic name derived from the PwmChannel container short name. The Id value must be consecutive.</p> <p>Note: A consecutive value is calculated for each new PWM channel. The minimum channel ID is selected as the default value.</p> <p>Note: The value of PwmChannelId should be unique in a configuration set.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - Total number of channels configured - 1		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.5.4 PwmCoherentUpdate

Table 1692 Specification for PwmCoherentUpdate

Name	PwmCoherentUpdate		
Description	<p>Switch for enabling the update of the duty/period parameter at the end of the current period.</p> <p>TRUE: update of period/duty cycle is done at the end of period of the currently generated waveform (current waveform is finished).</p> <p>FALSE: update of period/duty cycle is done immediately (just after the service call, the current waveform is cut).</p> <p>Depending on the PwmChannelClass the update end period will either work for duty cycle or duty and period. Both duty and period are updated for the variable period channel and only duty cycle is updated for other PwmChannelClass channels.</p> <p>Note: This parameter is only applicable for GTM channels.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Pwm_17_GtmCcu6 driver
Table 1692 Specification for PwmCoherentUpdate (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PwmAssignedHwUnit, PwmChannelClass, PwmChannelCoherentSelection		

20.3.1.5.5 PwmDutycycleDefault
Table 1693 Specification for PwmDutycycleDefault

Name	PwmDutycycleDefault		
Description	<p>Default value of the duty cycle for a channel. It can be entered as absolute ticks or as relative percentage based on the PwmDutyShiftInTicks parameter.</p> <p>When the PwmDutyShiftInTicks is set to OFF, the value is relative to period.</p> <p>0 is 0%, 0x8000 is 100%</p> <p>When the PwmDutyShiftInTicks is set to ON, the value is in absolute ticks.</p> <p>0 is 0%, If the value same is greater than or equal to PwmPeriodDefault then duty cycle is 100%</p> <p>Range:</p> <p>0 to 0x8000, if PwmDutyShiftInTicks is STD_OFF for TOM/ATOM/CCU6</p> <p>0 to 0xFFFF, if PwmDutyShiftInTicks is STD_ON and the module is TOM,CCU6</p> <p>0 to 0xFFFFFFF, if PwmDutyShiftInTicks is STD_ON and the module is ATOM</p> <p>Note: The default is set to FALSE as the hardware reset is FALSE.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PwmDutyShiftInTicks		

20.3.1.5.6 PwmIdleState
Table 1694 Specification for PwmIdleState

Name	PwmIdleState
-------------	--------------

Pwm_17_GtmCcu6 driver

Table 1694 Specification for PwmIdleState (continued)

Description	Represents the output state of the PWM after the signal is stopped (that is call to Pwm_17_GtmCcu6_SetOutputToldle). Note: Default is hardware reset value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PWM_HIGH: The PWM channel output will be set to high (3 or 5 V) in idle state. PWM_LOW: The PWM channel output will be set to low (0 V) in idle state.		
Default value	PWM_LOW		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.5.7 PwmMcuClockReferencePoint

Table 1695 Specification for PwmMcuClockReferencePoint

Name	PwmMcuClockReferencePoint		
Description	Contains reference to the McuClockReferencePoint. Since this parameter is not used, it is, hence disabled.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

20.3.1.5.8 PwmNotification

Table 1696 Specification for PwmNotification

Name	PwmNotification
-------------	-----------------

Pwm_17_GtmCcu6 driver
Table 1696 Specification for PwmNotification (continued)

Description	<p>The PwmNotification is used by the PWM driver to invoke the user-defined function for edge generation of the respective channel. The parameter can be a name or the address (numeric value) of the notification function.</p> <p>Note1: Since the name of the function is configurable, the default value is kept as NULL.</p> <p>Note2: The PWM driver does not validate the configured function name or address for correctness and the responsibility falls on the user.</p> <p>If McuTomChannelEventHandledByDsadc/McuAtomChannelEventHandledByDsadc is TRUE for the channel, notification callback will be handled by DSADC and not by PWM, when an event occurs for the TOM/ATOM channel respectively. Hence, PwmNotification parameter will not be editable for that channel.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	McuAtomChannelEventHandledByDsadc, McuTomChannelEventHandledByDsadc, PwmNotificationSupported		

20.3.1.5.9 PwmPeriodDefault
Table 1697 Specification for PwmPeriodDefault

Name	PwmPeriodDefault		
Description	<p>Value of period used for initialization.</p> <p>Significant if PwmChannelClass is PWM_FIXED_PERIOD or PWM_VARIABLE_PERIOD</p> <p>Range:</p> <p>0 to 0xFFFF, when module is TOM or CCU6</p> <p>0 to 0xFFFFFFFF, when module is ATOM</p> <p>Note: The default value is set to match the hardware reset value</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Pwm_17_GtmCcu6 driver
Table 1697 Specification for PwmPeriodDefault (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PwmChannelClass		

20.3.1.5.10 PwmPolarity
Table 1698 Specification for PwmPolarity

Name	PwmPolarity		
Description	Defines the starting polarity of each PWM channel. Note: The default is set to PWM_LOW as the hardware reset value is FALSE.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	PWM_HIGH: PWM channel output is high at the beginning of the cycle and then goes low when the duty count is reached PWM_LOW: PWM channel output is low at the beginning of the cycle and then goes high when the duty count is reached		
Default value	PWM_LOW		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.5.11 PwmReferenceChannel
Table 1699 Specification for PwmReferenceChannel

Name	PwmReferenceChannel		
Description	Significant if PwmChannelClass is PWM_FIXED_PERIOD_SHIFTED or PWM_FIXED_PERIOD_CENTER_ALIGNED Reference channel must be provided if PwmChannelClass is PWM_FIXED_PERIOD_SHIFTED or PWM_FIXED_PERIOD_CENTER_ALIGNED. Reference channel can only be of PWM_FIXED_PERIOD. Note: The default value is blank as user has to select the appropriate reference value from the MCU driver.		
Multiplicity	1..1	Type	EcucReferenceDef

Pwm_17_GtmCcu6 driver
Table 1699 Specification for PwmReferenceChannel (continued)

Range	Reference to Node: PwmChannel		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PwmChannelClass		

20.3.1.5.12 PwmShiftValue
Table 1700 Specification for PwmShiftValue

Name	PwmShiftValue		
Description	Configures the initial shift value with respect to the PWM_FIXED_PERIOD class channel. It can be entered as absolute ticks or as relative percentage based on the PwmDutyShiftInTicks parameter. When PwmDutyShiftInTicks is set to OFF, the value is relative to period. 0 is 0%, 0x8000 is 100% When PwmDutyShiftInTicks is set to ON, the value is in absolute ticks. Value 0 is 0%, Value same as PwmPeriodDefault of reference channel is 100% Significant if PwmChannelClass is PWM_FIXED_PERIOD_SHIFTED Note: The default is set to match the hardware reset value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PwmChannelClass, PwmDutyShiftInTicks		

20.3.1.6 Container: PwmChannelConfigSet

This container contains the configuration parameters and sub containers of the AUTOSAR PWM driver

Pwm_17_GtmCcu6 driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

20.3.1.7 Container: PwmConfigurationOfOptApiServices

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

20.3.1.7.1 PwmDeInitApi

Table 1701 Specification for PwmDeInitApi

Name	PwmDeInitApi		
Description	Adds/removes the Pwm_17_GtmCcu6_DeInit() API from the code. TRUE Pwm_17_GtmCcu6_DeInit() API is available to the user. FALSE Pwm_17_GtmCcu6_DeInit() API is not available to the user. Note: The Pwm_17_GtmCcu6_DeInit() API is disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.7.2 PwmGetOutputState

Table 1702 Specification for PwmGetOutputState

Name	PwmGetOutputState		
Description	Adds/removes the Pwm_17_GtmCcu6_GetOutputState() API from the code. TRUE Pwm_17_GtmCcu6_GetOutputState() API is available to the user. FALSE Pwm_17_GtmCcu6_GetOutputState() API is not available to the user. Note: The Pwm_17_GtmCcu6_GetOutputState() API is disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef

Pwm_17_GtmCcu6 driver
Table 1702 Specification for PwmGetOutputState (continued)

Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.7.3 PwmSetDutyCycle
Table 1703 Specification for PwmSetDutyCycle

Name	PwmSetDutyCycle		
Description	Adds/removes the service Pwm_17_GtmCcu6_SetDutyCycle() API from the code. TRUE: Pwm_17_GtmCcu6_SetDutyCycle() API is available to the user FALSE: Pwm_17_GtmCcu6_SetDutyCycle() API is not available to the user Note: Pwm_17_GtmCcu6_SetDutyCycle() API is disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.7.4 PwmSetOutputToIdle
Table 1704 Specification for PwmSetOutputToIdle

Name	PwmSetOutputToIdle
Description	Adds/removes the service Pwm_17_GtmCcu6_SetOutputToIdle() from the code.

Pwm_17_GtmCcu6 driver
Table 1704 Specification for PwmSetOutputToldle (continued)

	TRUE: Pwm_17_GtmCcu6_SetOutputToldle() API is available to the user FALSE: Pwm_17_GtmCcu6_SetOutputToldle() API is not available to the user Note: Pwm_17_GtmCcu6_SetOutputToldle() API is disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.7.5 PwmSetPeriodAndDuty
Table 1705 Specification for PwmSetPeriodAndDuty

Name	PwmSetPeriodAndDuty		
Description	Adds/removes the service Pwm_17_GtmCcu6_SetPeriodAndDuty() from the code. TRUE: Pwm_17_GtmCcu6_SetPeriodAndDuty() API is available to the user FALSE: Pwm_17_GtmCcu6_SetPeriodAndDuty() API is not available to the user Note: Pwm_17_GtmCcu6_SetPeriodAndDuty() API is disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Pwm_17_GtmCcu6 driver

20.3.1.7.6 PwmVersionInfoApi

Table 1706 Specification for PwmVersionInfoApi

Name	PwmVersionInfoApi		
Description	Adds/removes the service Pwm_17_GtmCcu6_GetVersionInfo from the code. TRUE: Pwm_17_GtmCcu6_GetVersionInfo() API is available to the user. FALSE: Pwm_17_GtmCcu6_GetVersionInfo() API is not available to the user. Note: The Pwm_17_GtmCcu6_GetVersionInfo() API is disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.8 Container: PwmGeneral

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

20.3.1.8.1 PwmChannelCoherentSelection

Table 1707 Specification for PwmChannelCoherentSelection

Name	PwmChannelCoherentSelection		
Description	Switch for enabling the channel-wise update of the duty/period parameter at the end of the current period. TRUE: Update of period/duty cycle is done based on the PWM channel-specific configuration PwmCoherentUpdate configuration parameter. FALSE: Update of period/duty cycle is done based on the PwmDutycycleUpdatedEndPeriod and PwmPeriodUpdatedEndPeriod global parameters. Note: This parameter is only applicable for GTM channels.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		

Pwm_17_GtmCcu6 driver
Table 1707 Specification for PwmChannelCoherentSelection (continued)

Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PwmAssignedHwUnit		

20.3.1.8.2 PwmDevErrorDetect
Table 1708 Specification for PwmDevErrorDetect

Name	PwmDevErrorDetect		
Description	Switches the (DET) detection and notification to ON or OFF. TRUE: enabled FALSE: disabled		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.8.3 PwmDutyShiftInTicks
Table 1709 Specification for PwmDutyShiftInTicks

Name	PwmDutyShiftInTicks		
Description	Determines whether duty cycle and shift values are absolute or relative to period. TRUE: duty cycle and shift value to be entered as absolute ticks FALSE: duty cycle and shift values are entered relative to period 0 is 0%, 0x8000 is 100%. Note: The default value is set to FALSE as it is a non-AUTOSAR feature		

Pwm_17_GtmCcu6 driver
Table 1709 Specification for PwmDutyShiftInTicks (continued)

Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.8.4 PwmDutycycleUpdatedEndperiod
Table 1710 Specification for PwmDutycycleUpdatedEndperiod

Name	PwmDutycycleUpdatedEndperiod		
Description	Global switch for enabling the update of the duty cycle parameter at the end of the current period. TRUE: update of duty cycle is done at the end of period of the currently generated waveform (current waveform is finished). FALSE: update of duty cycle is done immediately (just after the service call, the current waveform is updated with the new duty cycle value). Default value is set to Hardware default value. Note: If the global switch is TRUE then for PWM_VARIABLE_PERIOD channel both duty and period will get updated at the end of period of the currently generated waveform. Note: This parameter is only applicable for GTM channels.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PwmAssignedHwUnit, PwmChannelCoherentSelection		

Pwm_17_GtmCcu6 driver

20.3.1.8.5 PwmEnable0Or100DutyNotification

Table 1711 Specification for PwmEnable0Or100DutyNotification

Name	PwmEnable0Or100DutyNotification		
Description	<p>If set to ON, notifications are supported for 0% and 100% for the fixed and variable period channels.</p> <p>If set to OFF, notifications are not supported for 0% and 100% for the fixed and variable period channels.</p> <p>Note: The default value is set to FALSE to minimize the executable code size.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PwmNotificationSupported		

20.3.1.8.6 PwmHandleShiftByOffset

Table 1712 Specification for PwmHandleShiftByOffset

Name	PwmHandleShiftByOffset		
Description	<p>When PwmHandleShiftByOffset is enabled then the shifted channels start with an offset and will work similar to the fixed period channel.</p> <p>When PwmHandleShiftByOffset is not selected the shifted channels are triggered from the referenced fixed period channel.</p> <p>Note: This parameter is only applicable for GTM channels.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

Pwm_17_GtmCcu6 driver
Table 1712 Specification for PwmHandleShiftByOffset (continued)

Origin	IFX	Scope	LOCAL
Dependency	PwmAssignedHwUnit		

20.3.1.8.7 PwmIndex
Table 1713 Specification for PwmIndex

Name	PwmIndex		
Description	Instance ID of the PWM module instance		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.8.8 PwmInitCheckApi
Table 1714 Specification for PwmInitCheckApi

Name	PwmInitCheckApi		
Description	Switch to enable safety Init check API. Note: The detection of safety-related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Pwm_17_GtmCcu6 driver**20.3.1.8.9 PwmLowPowerStatesSupport****Table 1715 Specification for PwmLowPowerStatesSupport**

Name	PwmLowPowerStatesSupport		
Description	<p>This Parameter is disabled as power modes are not supported.</p> <p>Adds/removes all power state management-related APIs (PWM_SetPowerState, PWM_GetCurrentPowerState, PWM_GetTargetPowerState, PWM_PreparePowerState, PWM_Main_PowerTransitionManager) indicating that the hardware offers low power state management.</p>		
Multiplicity	0..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.8.10 PwmMultiCoreErrorDetect**Table 1716 Specification for PwmMultiCoreErrorDetect**

Name	PwmMultiCoreErrorDetect		
Description	Enables or disables the multicore-related DET detection and reporting. It is applicable only when DET/Safety is enabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	PwmDevErrorDetect		

Pwm_17_GtmCcu6 driver**20.3.1.8.11 PwmNotificationSupported****Table 1717 Specification for PwmNotificationSupported**

Name	PwmNotificationSupported		
Description	Switch to indicate that the notifications are supported. TRUE: notifications are supported FALSE: notifications are not supported Note: The default value is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.8.12 PwmPeriodUpdatedEndperiod**Table 1718 Specification for PwmPeriodUpdatedEndperiod**

Name	PwmPeriodUpdatedEndperiod		
Description	Global switch for enabling the update of the period parameter at the end of the current period. TRUE: update of period and duty cycle for variable period channel is done at the end of period of currently generated waveform (current waveform is finished) FALSE: update of period and duty cycle for variable period channel is done immediately (just after the service call, the current waveform is updated with new period value) The PWM_FIXED_PERIOD, PWM_FIXED_PERIOD_SHIFTED and PWM_FIXED_PERIOD_CENTER_ALIGNED class channels do not get affected due to this parameter setting. Note: Default value is set to Hardware default value. Note: This parameter is only applicable for GTM channels.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

Pwm_17_GtmCcu6 driver
Table 1718 Specification for PwmPeriodUpdatedEndperiod (continued)

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	PwmAssignedHwUnit, PwmChannelCoherentSelection		

20.3.1.8.13 PwmPowerStateAsynchTransitionMode
Table 1719 Specification for PwmPowerStateAsynchTransitionMode

Name	PwmPowerStateAsynchTransitionMode		
Description	This parameter is disabled as power modes are not supported. Enables/disables support of the PWM driver to the asynchronous power state transition.		
Multiplicity	0..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.8.14 PwmSafetyEnable
Table 1720 Specification for PwmSafetyEnable

Name	PwmSafetyEnable		
Description	Switch to enable reporting of safety errors (range and plausibility checks). Note: When this switch is enabled, AUTOSAR DETs are enabled by default. The detection of safety-related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		

Pwm_17_GtmCcu6 driver
Table 1720 Specification for PwmSafetyEnable (continued)

Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

20.3.1.9 Container: PwmPowerStateConfig

This container is disabled as power modes are not supported. Each instance of this parameter defines a power state and the callback to be invoked when this power state is reached.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

20.3.1.9.1 PwmPowerState

Table 1721 Specification for PwmPowerState

Name	PwmPowerState		
Description	This Parameter is disabled as power modes are not supported. Each instance of this parameter describes a different power state supported by the PWM hardware. It should be defined by the hardware supplier and used by the PWM driver to reference specific hardware configurations which set the PWM hardware module in the referenced power state.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.1.9.2 PwmPowerStateReadyCbkRef

Table 1722 Specification for PwmPowerStateReadyCbkRef

Name	PwmPowerStateReadyCbkRef
-------------	--------------------------

Pwm_17_GtmCcu6 driver
Table 1722 Specification for PwmPowerStateReadyCbkRef (continued)

Description	This parameter is disabled as power modes are not supported. Each instance of this parameter contains a reference to a power mode callback defined in a CDD or loHwAbs component.		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

20.3.2 Functions - Type definitions

20.3.2.1 Pwm_17_GtmCcu6_ChannelType

Table 1723 Specification for Pwm_17_GtmCcu6_ChannelType

Syntax	Pwm_17_GtmCcu6_ChannelType	
Type	uint8	
File	Pwm_17_GtmCcu6.h	
Range	0-199	The range is for the number of TOM channels, ATOM channels and CCU6 channels for the device variant. The maximum number of channels will vary depending on the device variant. 200 is for the superset device variant.
Description	Specifies the identification (ID) for a channel.	
Source	AUTOSAR	

20.3.2.2 Pwm_17_GtmCcu6_NotifiPtrType

Table 1724 Specification for Pwm_17_GtmCcu6_NotifiPtrType

Syntax	Pwm_17_GtmCcu6_NotifiPtrType	
Type	Pointer to a function of type void Function_Name (void)	
File	Pwm_17_GtmCcu6.h	

Pwm_17_GtmCcu6 driver
Table 1724 Specification for Pwm_17_GtmCcu6_NotifiPtrType (continued)

Description	Channel notification function pointer
Source	AUTOSAR

20.3.2.3 Pwm_17_GtmCcu6_PeriodType
Table 1725 Specification for Pwm_17_GtmCcu6_PeriodType

Syntax	Pwm_17_GtmCcu6_PeriodType	
Type	uint32	
File	Pwm_17_GtmCcu6.h	
Range	0-16777215	Range: 0 to 0xFFFF for TOM/CCU6 0 to 0xFFFFFFFF for ATOM
Description	Definition of the period of a PWM channel	
Source	AUTOSAR	

20.3.2.4 Pwm_17_GtmCcu6_OutputStateType
Table 1726 Specification for Pwm_17_GtmCcu6_OutputStateType

Syntax	Pwm_17_GtmCcu6_OutputStateType	
Type	uint8	
File	Pwm_17_GtmCcu6.h	
Range	PWM_17_GTMCCU6_HIGH	The PWM channel is in the high state. The PWM channel output will be in the high state (3 or 5 V).
	PWM_17_GTMCCU6_LOW	The PWM channel is in low state. The PWM channel output will be in the low state (0 V).
Description	Output state of a PWM channel. Note that this will be read from the output state on the TOM or ATOM channel connected to the port pin. This will not be read directly from the port pin. However, both will be the same. This type is used to read PwmPolarity and PwmIdleState.	
Source	AUTOSAR	

20.3.2.5 Pwm_17_GtmCcu6_EdgeNotificationType
Table 1727 Specification for Pwm_17_GtmCcu6_EdgeNotificationType

Syntax	Pwm_17_GtmCcu6_EdgeNotificationType	
Type	uint8	
File	Pwm_17_GtmCcu6.h	

Pwm_17_GtmCcu6 driver
Table 1727 Specification for Pwm_17_GtmCcu6_EdgeNotificationType (continued)

Range	PWM_17_GTMCCU6_RISING_EDGE	Notification is called when the rising edge occurs on the PWM output signal.
	PWM_17_GTMCCU6_FALLING_EDGE	Notification is called when the falling edge occurs on the PWM output signal.
	PWM_17_GTMCCU6_BOTH_EDGES	Notification is called when both the rising and falling edge occur on the PWM output signal.
Description	Definition of the type of edge notification of a PWM channel. The edges are passed in the Pwm_17_GtmCcu6_EnableNotification() API.	
Source	AUTOSAR	

20.3.2.6 Pwm_17_GtmCcu6_ChannelClassType
Table 1728 Specification for Pwm_17_GtmCcu6_ChannelClassType

Syntax	Pwm_17_GtmCcu6_ChannelClassType	
Type	uint8	
File	Pwm_17_GtmCcu6.h	
Range	PWM_17_GTMCCU6_FIXED_PERIOD	
	PWM_17_GTMCCU6_FIXED_PERIOD_SHIFTED	
	PWM_17_GTMCCU6_FIXED_PERIOD_CENTRED_ALIGNED	
	PWM_17_GTMCCU6_VARIABLE_PERIOD	
Description	Defines the class of a PWM channel	
Source	AUTOSAR	

20.3.2.7 Pwm_17_GtmCcu6_ConfigType
Table 1729 Specification for Pwm_17_GtmCcu6_ConfigType

Syntax	Pwm_17_GtmCcu6_ConfigType	
Type	Structure	
File	Pwm_17_GtmCcu6.h	
Range	Hardware dependent structure[]	The contents of the initialization data structure are hardware specific.
Description	This type definition is used to configure the overall PWM configuration. The pointer to the object of this data type is used in the Pwm_17_GtmCcu6_Init() API to initialize the PWM driver.	
Source	AUTOSAR	

Pwm_17_GtmCcu6 driver**20.3.3 Functions - APIs****20.3.3.1 Pwm_17_GtmCcu6_DeInit****Table 1730 Specification for Pwm_17_GtmCcu6_DeInit API**

Syntax	void Pwm_17_GtmCcu6_DeInit (void)	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	De-initializes the PWM module and signals. This function de-initializes the PWM driver in the context of core from where this API is invoked.	
Source	AUTOSAR	
Error handling	DET: PWM_17_GTMCCU6_E_UNINIT: Error reported when the API service is used without module initialization Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	PwmDeInitApi	
User hints	-	

20.3.3.2 Pwm_17_GtmCcu6_DisableNotification**Table 1731 Specification for Pwm_17_GtmCcu6_DisableNotification API**

Syntax	void Pwm_17_GtmCcu6_DisableNotification (
---------------	--

Pwm_17_GtmCcu6 driver
Table 1731 Specification for Pwm_17_GtmCcu6_DisableNotification API (continued)

	const Pwm_17_GtmCcu6_ChannelType ChannelNumber)	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	ChannelNumber	Numeric identifier of the PWM
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to disable the PWM signal edge notification	
Source	AUTOSAR	
Error handling	DET: PWM_17_GTMCCU6_E_PARAM_CHANNEL: Error reported when the API service is used with an invalid channel identifier PWM_17_GTMCCU6_E_UNINIT: Error reported when the API service is used without module initialization PWM_17_GTMCCU6_E_CORE_CHANNEL_MISMATCH: Error reported when ChannelId is not allocated to the core from which the API is called Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	PwmNotificationSupported	
User hints	-	

20.3.3.3 Pwm_17_GtmCcu6_EnableNotification
Table 1732 Specification for Pwm_17_GtmCcu6_EnableNotification API

Syntax	void Pwm_17_GtmCcu6_EnableNotification (const Pwm_17_GtmCcu6_ChannelType ChannelNumber, const Pwm_17_GtmCcu6_EdgeNotificationType Notification)
Service ID	0x07

Pwm_17_GtmCcu6 driver
Table 1732 Specification for Pwm_17_GtmCcu6_EnableNotification API (continued)

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	ChannelNumber Notification	Numeric identifier of the PWM Type of the notification PWM_17_GTMCCU6_RISING_EDGE or PWM_17_GTMCCU6_FALLING_EDGE or PWM_17_GTMCCU6_BOTH_EDGES
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to enable the PWM signal edge notification according to the notification parameter	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>PWM_17_GTMCCU6_E_UNINIT: Error reported when the API service is used without module initialization</p> <p>PWM_17_GTMCCU6_E_PARAM_CHANNEL: Error reported when the API service is used with an invalid channel identifier</p> <p>PWM_17_GTMCCU6_E_CORE_CHANNEL_MISMATCH: Error reported when ChannelId is not allocated to the core from which the API is called</p> <p>PWM_17_GTMCCU6_E_PARAM_POINTER: For the Pwm_17_GtmCcu6_GetVersionInfo() API this error is reported when called with a NULL parameter.</p> <p>For the Pwm_17_GtmCcu6_EnableNotification() API this error is reported when no notification function is configured in the EB Tresos</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>PWM_17_GTMCCU6_E_PARAM_NOTIFICATION: The Pwm_17_GtmCcu6_EnableNotification() API service is called with invalid notification type. This is reported when safety is enabled</p> <p>PWM_17_GTMCCU6_E_NO_NOTIF_CONFIGURED: Error is reported by Pwm_17_GtmCcu6_EnableNotification() API, when invoked on a non-DSADC triggering PWM channel with no notification configured.</p> <p>PWM_17_GTMCCU6_E_INVALID_EDGE_NOTIF: The Pwm_17_GtmCcu6_EnableNotification() API service is called with invalid notification type for PWM channel whose interrupt is routed to DSADC.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	PwmNotificationSupported	
User hints	-	

Pwm_17_GtmCcu6 driver**20.3.3.4 Pwm_17_GtmCcu6_GetOutputState****Table 1733 Specification for Pwm_17_GtmCcu6_GetOutputState API**

Syntax	<pre>Pwm_17_GtmCcu6_OutputStateType Pwm_17_GtmCcu6_GetOutputState (const Pwm_17_GtmCcu6_ChannelType ChannelNumber)</pre>	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	ChannelNumber	Numeric identifier of the PWM
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Pwm_17_GtmCcu6_OutputStateType	-
Description	Service to read the internal state of the PWM output signal	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>PWM_17_GTMCCU6_E_UNINIT: Error reported when the API service is used without module initialization</p> <p>PWM_17_GTMCCU6_E_PARAM_CHANNEL: Error reported when the API service is used with an invalid channel identifier</p> <p>PWM_17_GTMCCU6_E_CORE_CHANNEL_MISMATCH: Error reported when ChannelId is not allocated to the core from which the API is called</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	PwmGetOutputState	
User hints	-	

20.3.3.5 Pwm_17_GtmCcu6_GetVersionInfo**Table 1734 Specification for Pwm_17_GtmCcu6_GetVersionInfo API**

Syntax	void Pwm_17_GtmCcu6_GetVersionInfo
	(

Pwm_17_GtmCcu6 driver
Table 1734 Specification for Pwm_17_GtmCcu6_GetVersionInfo API (continued)

	Std_VersionInfoType * const VersionInfoPtr)	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer to store the version information of this module
Parameters (in - out)	-	-
Return	void	-
Description	Service returns the version information of the PWM module	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>PWM_17_GTMCCU6_E_PARAM_POINTER: For the Pwm_17_GtmCcu6_GetVersionInfo() API this error is reported when called with a NULL parameter.</p> <p>For the Pwm_17_GtmCcu6_EnableNotification() API this error is reported when no notification function is configured in the EB Tresos</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	PwmVersionInfoApi	
User hints	-	

Pwm_17_GtmCcu6_Init
Table 1735 Specification for Pwm_17_GtmCcu6_Init API

Syntax	void Pwm_17_GtmCcu6_Init (const Pwm_17_GtmCcu6_ConfigType * const ConfigPtr)
Service ID	0x00
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Non Reentrant

Pwm_17_GtmCcu6 driver
Table 1735 Specification for Pwm_17_GtmCcu6_Init API (continued)

Parameters (in)	ConfigPtr	Pointer to configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This function initializes the PWM driver in the context of the core from where the Init is invoked.</p> <p>Init function initializes the resource allocated to current core.</p> <p>Note: After the MCU initialization, the PWM Init could be called from any of the cores.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>PWM_17_GTMCCU6_E_INIT_FAILED: Pwm_17_GtmCcu6_Init() API service is called with a wrong parameter</p> <p>PWM_17_GTMCCU6_E_ALREADY_INITIALIZED: Pwm_17_GtmCcu6_Init() API service is called while the PWM driver has already been initialized</p> <p>PWM_17_GTMCCU6_E_CORE_NOT_CONFIGURED: Error reported when PWM module is not configured for the core from which it was called</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

20.3.3.7 Pwm_17_GtmCcu6_InitCheck
Table 1736 Specification for Pwm_17_GtmCcu6_InitCheck API

Syntax	<pre>Std_ReturnType Pwm_17_GtmCcu6_InitCheck (const Pwm_17_GtmCcu6_ConfigType * const ConfigPtr)</pre>	
Service ID	0x10	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Valid address pointing to config

Pwm_17_GtmCcu6 driver
Table 1736 Specification for Pwm_17_GtmCcu6_InitCheck API (continued)

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : If initialization comparison is succeeds E_NOT_OK : If initialization comparison fails
Description	<p>This routine verifies the initialization the PWM driver.</p> <p>Note: Init check should be call before invoking any runtime APIs.</p> <p>Sequence:</p> <ol style="list-style-type: none"> 1. Invoke Pwm_17_GtmCcu6_Init from a core. 2. Invoke Pwm_17_GtmCcu6_InitCheck from the same core. 	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	PwmInitCheckApi	
User hints	-	

20.3.3.8 Pwm_17_GtmCcu6_SetDutyCycle
Table 1737 Specification for Pwm_17_GtmCcu6_SetDutyCycle API

Syntax	<pre>void Pwm_17_GtmCcu6_SetDutyCycle (const Pwm_17_GtmCcu6_ChannelType ChannelNumber, const uint32 DutyCycle)</pre>	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	ChannelNumber	Numeric identifier of the PWM
	DutyCycle	Duration of ON time of PWM
Parameters (out)	-	-
Parameters (in - out)	-	-

Pwm_17_GtmCcu6 driver
Table 1737 Specification for Pwm_17_GtmCcu6_SetDutyCycle API (continued)

Return	void	-
Description	<p>Service sets the duty cycle of the PWM channel.</p> <p>Since ATOM timer channel could support maximum of 24 bits, the DutyCycle parameter is uint32 when PWM_17_GTMCCU6_DUTY_SHIFT_IN_TICKS is set to ON.</p> <p>When PWM_17_GTMCCU6_DUTY_SHIFT_IN_TICKS is set to OFF, the DutyCycle parameter is uint16. Function prototype will change accordingly based on the PWM_17_GTMCCU6_DUTY_SHIFT_IN_TICKS switch.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>PWM_17_GTMCCU6_E_PARAM_CHANNEL: Error reported when the API service is used with an invalid channel identifier</p> <p>PWM_17_GTMCCU6_E_UNINIT: Error reported when the API service is used without module initialization</p> <p>PWM_17_GTMCCU6_E_CORE_CHANNEL_MISMATCH: Error reported when ChannelId is not allocated to the core from which the API is called</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>PWM_17_GTMCCU6_E_PARAM_DUTY: Error reported when the Pwm_17_GtmCcu6_SetDutyCycle() or Pwm_17_GtmCcu6_SetPeriodAndDuty() API is called with incorrect duty</p> <p>When PwmDutyShiftInTicks =OFF Valid values are 0 and 0x8000</p> <p>When PwmDutyShiftInTicks =ON Valid values are 0 - Period (16 bit / 24 bit)</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	PwmSetDutyCycle	
User hints	-	

20.3.3.9 Pwm_17_GtmCcu6_SetOutputToIdle
Table 1738 Specification for Pwm_17_GtmCcu6_SetOutputToIdle API

Syntax	<pre>void Pwm_17_GtmCcu6_SetOutputToIdle (const Pwm_17_GtmCcu6_ChannelType ChannelNumber)</pre>
Service ID	0x04
Sync/Async	Synchronous

Pwm_17_GtmCcu6 driver
Table 1738 Specification for Pwm_17_GtmCcu6_SetOutputToIdle API (continued)

ASIL Level	B	
Re-entrancy	Reentrant for different channels	
Parameters (in)	ChannelNumber	Input Channel ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service sets the PWM output to the configured Idle state	
Source	AUTOSAR	
Error handling	DET: PWM_17_GTMCCU6_E_UNINIT: Error reported when the API service is used without module initialization PWM_17_GTMCCU6_E_PARAM_CHANNEL: Error reported when the API service is used with an invalid channel identifier PWM_17_GTMCCU6_E_CORE_CHANNEL_MISMATCH: Error reported when ChannelId is not allocated to the core from which the API is called Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	PwmSetOutputToldle	
User hints	After the call of the function Pwm_17_GtmCcu6_SetOutputToldle, variable period type channels should be reactivated using the Api Pwm_17_GtmCcu6_SetPeriodAndDuty() to activate the PWM channel with the newly passed period and duty.	

20.3.3.10 Pwm_17_GtmCcu6_SetPeriodAndDuty
Table 1739 Specification for Pwm_17_GtmCcu6_SetPeriodAndDuty API

Syntax	<pre>void Pwm_17_GtmCcu6_SetPeriodAndDuty (const Pwm_17_GtmCcu6_ChannelType ChannelNumber, const Pwm_17_GtmCcu6_PeriodType Period, const uint16 DutyCycle)</pre>
Service ID	0x03
Sync/Async	Synchronous
ASIL Level	B

Pwm_17_GtmCcu6 driver
Table 1739 Specification for Pwm_17_GtmCcu6_SetPeriodAndDuty API (continued)

Re-entrancy	Reentrant for different channels	
Parameters (in)	ChannelNumber Period DutyCycle	Numeric identifier of the PWM New Period of PWM signal Duration of ON time of PWM
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Service sets the Duty and Period of the PWM signal.</p> <p>Since ATOM timer channel could support maximum of 24 bits, the DutyCycle parameter is uint32 when PWM_17_GTMCCU6_DUTY_SHIFT_IN_TICKS is set to ON.</p> <p>If PWM_17_GTMCCU6_DUTY_SHIFT_IN_TICKS is set to OFF, the DutyCycle parameter is uint16. Function prototype will change accordingly based on the PWM_17_GTMCCU6_DUTY_SHIFT_IN_TICKS switch.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>PWM_17_GTMCCU6_E_PERIOD_UNCHANGEABLE: Usage of unauthorized PWM service on PWM channel configured a fixed period</p> <p>PWM_17_GTMCCU6_E_PARAM_CHANNEL: Error reported when the API service is used with an invalid channel identifier</p> <p>PWM_17_GTMCCU6_E_UNINIT: Error reported when the API service is used without module initialization</p> <p>PWM_17_GTMCCU6_E_CORE_CHANNEL_MISMATCH: Error reported when ChannelId is not allocated to the core from which the API is called</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>PWM_17_GTMCCU6_E_PARAM_PERIOD: Error reported when the Pwm_17_GtmCcu6_SetPeriodAndDuty() APIs are called with incorrect period</p> <p>TOM/CCU6: valid values are 0-0xFFFF</p> <p>ATOM: valid values are 0-0xFFFFFFFF</p> <p>PWM_17_GTMCCU6_E_PARAM_DUTY: Error reported when the Pwm_17_GtmCcu6_SetDutyCycle() or Pwm_17_GtmCcu6_SetPeriodAndDuty() API is called with incorrect duty</p> <p>When PwmDutyShiftInTicks =OFF Valid values are 0 and 0x8000</p> <p>When PwmDutyShiftInTicks =ON Valid values are 0 - Period (16 bit / 24 bit)</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

Pwm_17_GtmCcu6 driver
Table 1739 Specification for Pwm_17_GtmCcu6_SetPeriodAndDuty API (continued)

Configuration dependencies	PwmSetPeriodAndDuty
User hints	-

20.3.4 Notifications and Callbacks

20.3.4.1 Pwm_17_GtmCcu6_Isr

Table 1740 Specification for Pwm_17_GtmCcu6_Isr API

Syntax	<pre>void Pwm_17_GtmCcu6_Isr (const Pwm_17_GtmCcu6_ChannelType ChannelNumber, const uint32 IsrStatus)</pre>	
Service ID		
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ChannelNumber IsrStatus	PWM channel number which caused the interrupt. This parameter gives the information about the comparator which caused the interrupt.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Handler to call the configured notification function	
Source	IFX	
Error handling	<p>DET:</p> <p>PWM_17_GTMCCU6_E_UNINIT: Error reported when the API service is used without module initialization</p> <p>PWM_17_GTMCCU6_E_PARAM_CHANNEL: Error reported when the API service is used with an invalid channel identifier</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>PWM_17_GTMCCU6_E_INVALID_ISR: Error reported when PWM ISR is called with incorrect compare match interrupt</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

Pwm_17_GtmCcu6 driver

Table 1740 Specification for Pwm_17_GtmCcu6_Isr API (continued)

Configuration dependencies	-
User hints	-

20.3.5 Scheduled functions

The PWM driver does not have any scheduled functions.

20.3.6 Interrupt service routines

The PWM driver does not require any interrupt handler.

20.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

20.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Note: The following error IDs are also reported as safety errors.

Table 1741 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
Pwm_17_GtmCcu6_Init() API service is called while the PWM driver has already been initialized.	AUTOSAR	PWM_17_GTMCCU6_E_ALREADY_INIT IALIZED=0x14	Pwm_17_GtmCcu6_Init
Error reported when ChannelId is not allocated to the core from which the API is called.	IFX	PWM_17_GTMCCU6_E_CORE_CHANNEL_MISMATCH=0x65	Pwm_17_GtmCcu6_SetPeriodAndDuty, Pwm_17_GtmCcu6_SetOutputToIdle, Pwm_17_GtmCcu6_GetOutputState, Pwm_17_GtmCcu6_EnableNotification, Pwm_17_GtmCcu6_DisableNotification, Pwm_17_GtmCcu6_SetDutyCycle
Error reported when PWM module is not configured for the core from which it was called.	IFX	PWM_17_GTMCCU6_E_CORE_NOT_CONFIGURED=0x64	Pwm_17_GtmCcu6_Init

Pwm_17_GtmCcu6 driver
Table 1741 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
Pwm_17_GtmCcu6_Init() API service is called with a wrong parameter.	AUTOSAR	PWM_17_GTMCCU6_E_INIT_FAILED=0x10	Pwm_17_GtmCcu6_Init
Error reported when the API service is used with an invalid channel identifier.	AUTOSAR	PWM_17_GTMCCU6_E_PARAM_CHAN_NEL=0x12	Pwm_17_GtmCcu6_Isr, Pwm_17_GtmCcu6_SetOutputToIdle, Pwm_17_GtmCcu6_EnableNotification, Pwm_17_GtmCcu6_DisableNotification, Pwm_17_GtmCcu6_GetOutputState, Pwm_17_GtmCcu6_SetPeriodAndDuty, Pwm_17_GtmCcu6_SetDutyCycle
For the Pwm_17_GtmCcu6_GetVersionInfo() API this error is reported when called with a NULL parameter. For the Pwm_17_GtmCcu6_EnableNotification() API this error is reported when no notification function is configured in the EB Tresos.	AUTOSAR	PWM_17_GTMCCU6_E_PARAM_POINTER=0x15	Pwm_17_GtmCcu6_EnableNotification, Pwm_17_GtmCcu6_GetVersionInfo
Usage of unauthorized PWM service on PWM channel configured a fixed period.	AUTOSAR	PWM_17_GTMCCU6_E_PERIOD_UNCHANGEABLE=0x13	Pwm_17_GtmCcu6_SetPeriodAndDuty
Error reported when the API service is used without module initialization.	AUTOSAR	PWM_17_GTMCCU6_E_UNINIT=0x11	Pwm_17_GtmCcu6_DeInit, Pwm_17_GtmCcu6_Isr, Pwm_17_GtmCcu6_SetOutputToIdle, Pwm_17_GtmCcu6_EnableNotification, Pwm_17_GtmCcu6_DisableNotification, Pwm_17_GtmCcu6_GetOutputState, Pwm_17_GtmCcu6_SetPeriodAndDuty, Pwm_17_GtmCcu6_SetDutyCycle

Pwm_17_GtmCcu6 driver

20.3.7.2 Production errors

The driver does not report any production errors.

20.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 1742 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
The Pwm_17_GtmCcu6_EnableNotification() API service is called with invalid notification type for PWM channel whose interrupt is routed to DSADC.	IFX	PWM_17_GTMCCU6_E_INVALID_EDGE_NOTIF=0xCD	Pwm_17_GtmCcu6_EnableNotification
Error reported when PWM ISR is called with incorrect compare match interrupt.	IFX	PWM_17_GTMCCU6_E_INVALID_ISR=0xC8	Pwm_17_GtmCcu6_Isr
Error is reported by Pwm_17_GtmCcu6_EnableNotification() API, when invoked on a non-DSADC triggering PWM channel with no notification configured.	IFX	PWM_17_GTMCCU6_E_NO_NOTIF_CONFIGURED=0xCC	Pwm_17_GtmCcu6_EnableNotification
Error reported when the Pwm_17_GtmCcu6_SetDutyCycle() or Pwm_17_GtmCcu6_SetPeriodAndDuty() API is called with incorrect duty When PwmDutyShiftInTicks =OFF Valid values are 0 and 0x8000 When PwmDutyShiftInTicks =ON Valid values are 0 - Period (16 bit / 24 bit).	IFX	PWM_17_GTMCCU6_E_PARAM_DUTY=0xC9	Pwm_17_GtmCcu6_SetPeriodAndDuty, Pwm_17_GtmCcu6_SetDutyCycle
The Pwm_17_GtmCcu6_EnableNotification() API service is called with invalid notification type. This is reported when safety is enabled.	IFX	PWM_17_GTMCCU6_E_PARAM_NOTIFICATION=0xCA	Pwm_17_GtmCcu6_EnableNotification
Error reported when the Pwm_17_GtmCcu6_SetPeriodAndDuty() APIs are	IFX	PWM_17_GTMCCU6_E_PARAM_PERIOD=0xCB	Pwm_17_GtmCcu6_SetPeriodAndDuty

Pwm_17_GtmCcu6 driver

Table 1742 Description of safety errors reported (continued)

Description	Source	Error code and value	Applicable APIs
called with incorrect period TOM/CCU6: valid values are 0-0xFFFF ATOM: valid values are 0-0xFFFFFFF.			

20.3.7.4 Runtime errors

The driver does not report any runtime errors.

20.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

20.3.8.1 Deviations

The section describes the deviations from software specification.

Table 1743 Known deviations

Reference	Deviation
ECUC_Pwm_00124 :PwmPeriodDefault	According to the AUTOSAR specification, the period value is entered in seconds. However, the PWM driver expects the value to be entered in ticks.
PWM_FIXED_PERIOD_CENTER_ALIGNED	PWM_FIXED_PERIOD_CENTER_ALIGNED range is added along with the other AUTOSAR ranges as microcontroller-specific feature.
Safety error for unintended service request	Refer to Reporting of unintended service requests .

20.3.8.2 Limitations

The section describes the limitations from software specification.

PWM driver limitation:

- Fixed period shifted channel and fixed period center-aligned channels should be present in the same TGC/AGC of the same TOM/ATOM module where the referenced fixed period channel is present.
- The transition from 100% to other percentages with the shift value will not be the same as the ideal case: The transition from 100% to other percentages with a shift will not happen as per expected signal in case of coherent update. This is because three signal level changes are needed to produce the expected waveform. The registers CM0 and CM1 are updated from shadow registers at the end of the cycle. The signal will remain in the same level till the CM0 value is reached, as shown in the following diagram.

Note: This limitation is also applicable for cases where call is made from the

Pwm_17_GtmCcu6_SetOutputToIdle API to some other duty.

Note:

CM0 - Period Match register

CM1- Duty Match register

Pwm_17_GtmCcu6 driver

SR0 - Period Match Shadow register

SR1 - Duty Match Shadow register

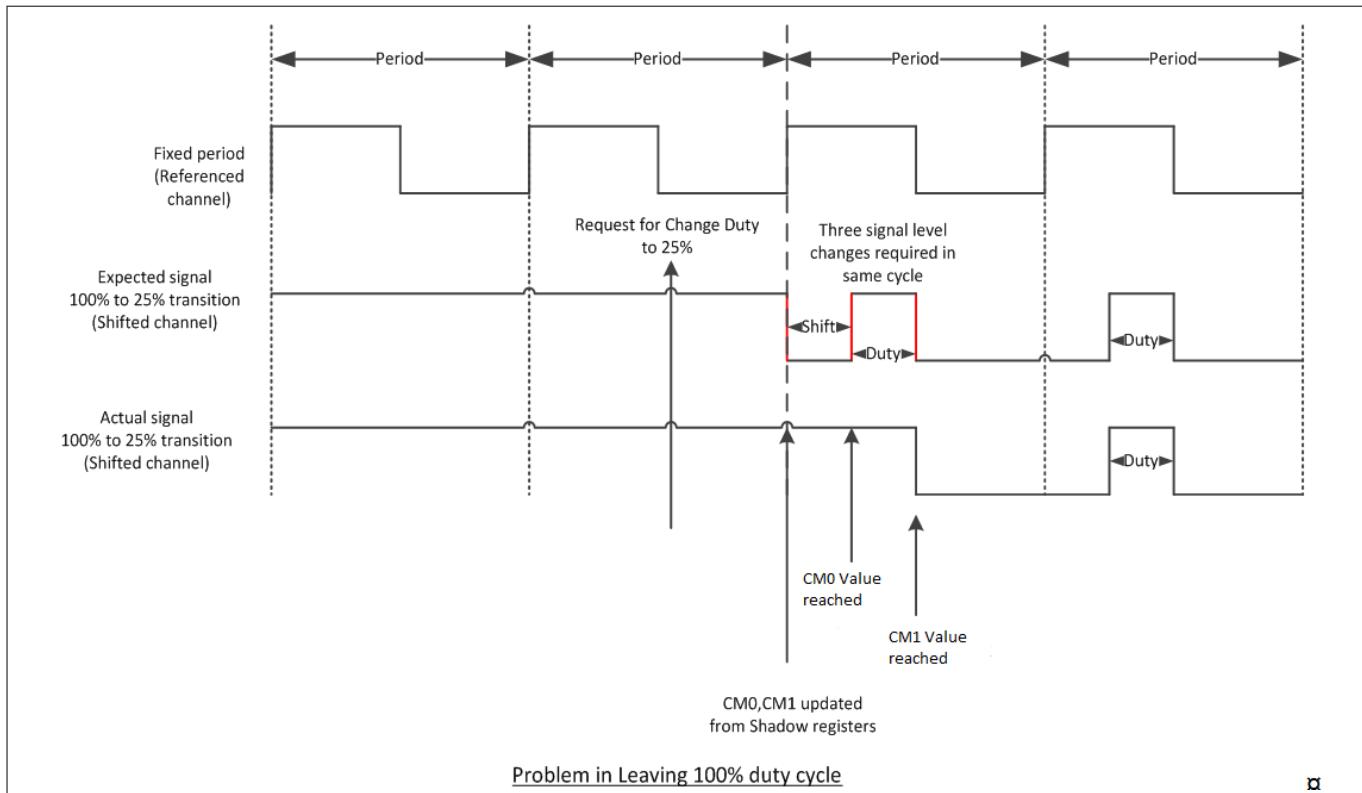
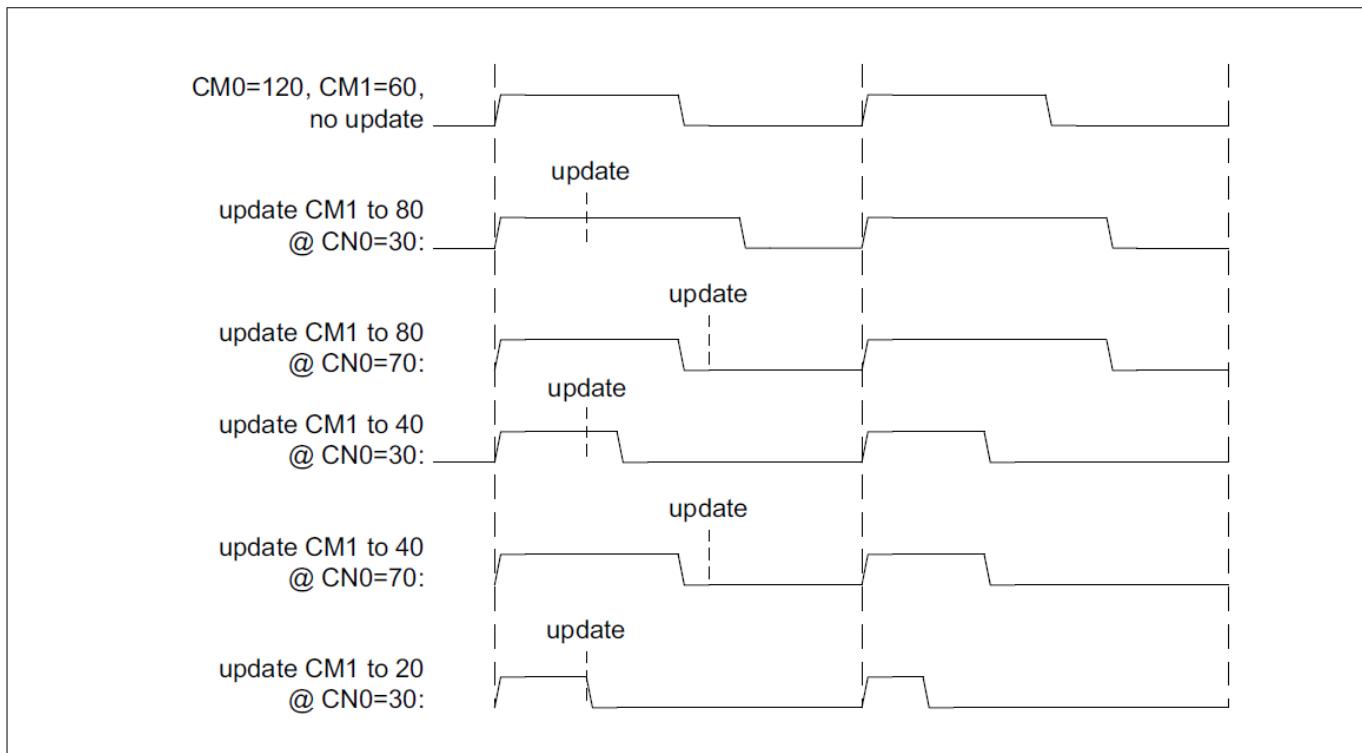
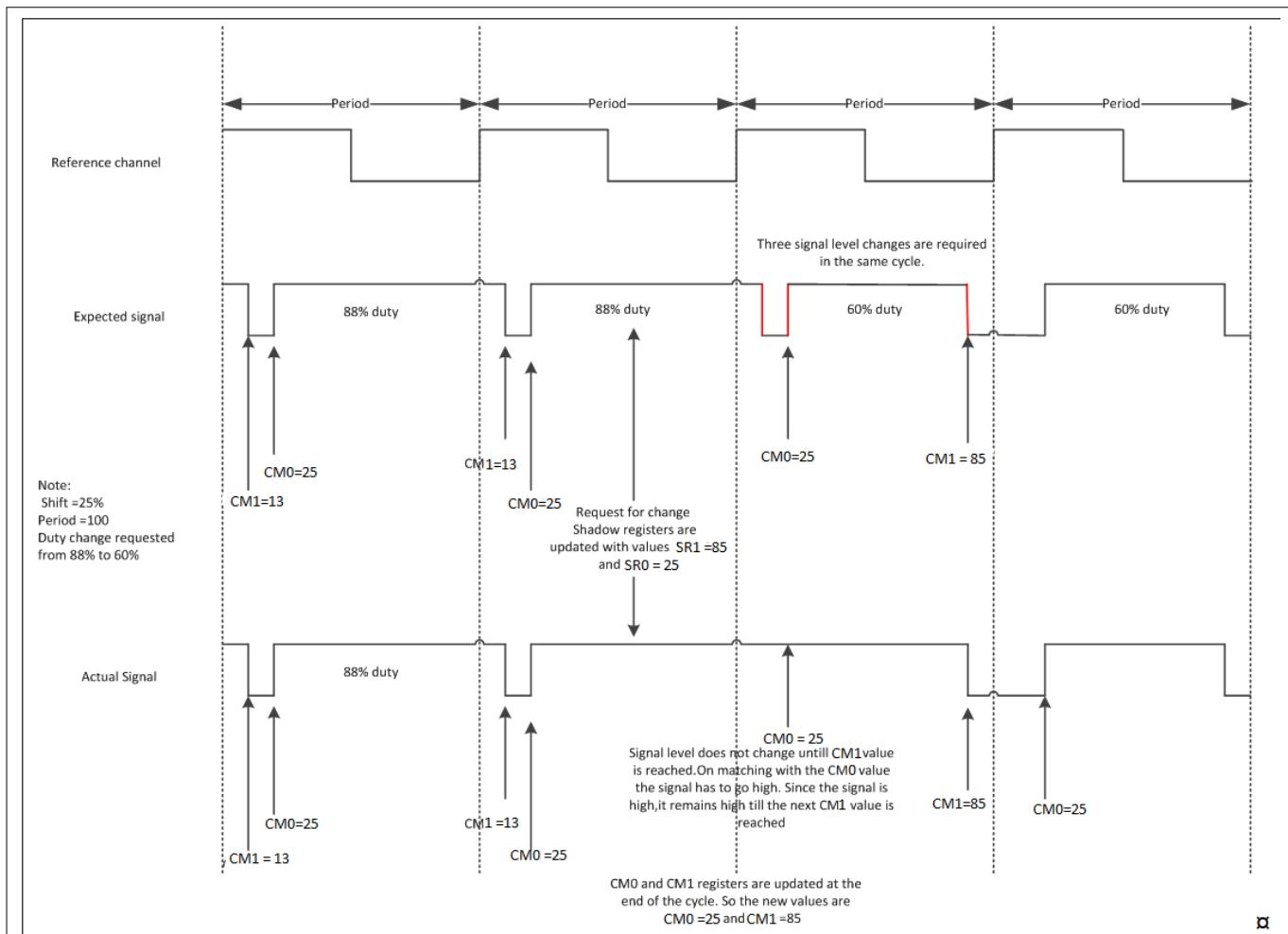


Figure 172 Transition from 100% to other percentages with a shift value

- In case of non-coherent update the register CMx (CM0 period and CM1 duty) are updated directly (applicable for Pwm_17_GtmCcu6_SetDutyCycle, Pwm_17_GtmCcu6_SetPeriodAndDuty and Pwm_17_GtmCcu6_SetOutputToIdle APIs). Depending on the point of time of the update of CMx registers in relation to the actual values of CN0 and CMx, the new duty cycle is applied in the current period or the following period. The new duty cycle may jitter from update to update by a maximum of one period. Refer the following diagram. For shifted and center aligned channels update will be done with respect to period of reference fixed period channel.

Pwm_17_GtmCcu6 driver

Figure 173 Non-coherent update of duty-cycle

- In case of coherent updated shifted channels where shift is not equal to zero, when duty change is requested from a condition where shift + duty is greater than period to shift + duty less than period then the signal will remain on duty for more than a period value. Refer figure below. If a request is made to change the duty from the condition shift and duty-cycle value together is greater than period ($\text{Shift}+\text{Duty} > \text{Period}$) to less than period ($\text{Shift}+\text{Duty} < \text{Period}$) in the case for a coherent updated shifted channel where shift value is not equal to zero, the following behavior will be observed.

Pwm_17_GtmCcu6 driver

Figure 174 Duty update with a shift value behavior

In this diagram, on compare match with CM0 value the signal will go high and on compare match with CM1 value the signal will go low. The values are calculated as $CM0/SR0 = \text{Shift}$ and $CM1/SR1 = (\text{Shift} + \text{Duty \%}) / \text{period}$. Here in the figure initially 88% duty was requested and shift was 25%. So if the period is 100 ticks then, $CM0 = 25$ and $CM1 = 13 ((25+88)\%100)$. Later when the duty of 60% is requested the values calculated are $SR0 = 25$ (Shift) and $SR1 = 85$ using $((25+60)\%100)$.

The CM1 and CM0 values are updated from Shadow registers SR1 and SR0 respectively (in the case of coherent update). When the duty change is requested then the function `Pwm_17_Gtm_SetDutyCycle` will update the corresponding shadow registers and return immediately after completion of the function. The updating of shadow registers to CM0 and CM1 registers will happen only after the end of the period.

Since the new values are updated to CM1 register from shadow register the signal will remain in the same state till the CM1 value is reached as shown in the figure. The signal will remain high for a time greater than period as shown in the figure above because there are three signal level changes required in the same cycle which could not be handled with two values CM0 and CM1 available at our disposal.

- For the shifted channels (`PwmHandleShiftByOffset = false`) in the conditions shift + duty exactly equal to period and if a transition is asked for `Pwm_17_GtmCcu6_SetOutputToldle` function or to a duty of 100% then an incorrect duty for one cycle could be observed.
- In case shifted channels (`PwmHandleShiftByOffset = true`) the changes will be happened by the end of the period of the same channel. It will not depend on the reference channel. The limitations in table below are also applicable. Background: A global configuration parameter has been added `PwmHandleShiftByOffset` in the `PwmGeneral` container and once this parameter is selected then the shifted channels are configured similar to fixed period channels and but they are started by an offset. The offset is calculated by offset =

Pwm_17_GtmCcu6 driver

period – shift. This offset acts as the required shift. All the channels in the TGC where these kind of shifted channels are present are triggered by a global host trigger. This host trigger will help in enabling all the channels in the TGC at the same time, and since all the channels are started the same time the corresponding shift is achieved accordingly. Please refer to respective API's to know how these kind of shifted channels are handled.

Table 1744 Difference between the two types of handling for shifted channels

Subject	FPS(PwmHandleShiftByOffset=false)	FPS(PwmHandleShiftByOffset=true)
Handling of shift	Compare register is used	Counter register is preloaded with period - shift
Handling of period	Fixed period reference channel's compare register is used	No reference channel, same channels compare register is used, however in configuration reference channel is provided for AUTOSAR compatibility
Host trigger	Host trigger is not used	Host trigger is used.
TGC restriction due to host trigger	Channels in the same TGC can be shared across other drivers.	Channels in the same TGC cannot be shared across other drivers.
Handling of shift + duty crossing period	For more than one cycle the output line is either high or low during change of duty	No limitation in this case.
Leaving 100% duty cycle	The signal will remain in the 100% duty till the next compare value is reached; The shift will not be visible.	The signal will remain in the 100% duty till the next compare value is reached; The shift will not be visible.
Update of new duty cycle	In case of coherent update the new duty cycle always happens at the end of reference channel period	In case of coherent update the new duty cycle happens at the end of its own period. This means in corner cases the new duty cycle might happen in the same cycle of the configured reference channel.
Pwm_17_GtmCcu6_SetOutputToldle function	Always happens in the next cycle.	Always happens in the next cycle.
Update of 0% and 100% duty cycle	Happens at the start of the next cycle of reference channel	Happens at the start of the next cycle after the shift value is elapsed.

- In case when handling of shifted channels by offset is enabled (PwmHandleShiftByOffset = true) other drivers (GPT, WDG) should not use the channels in the same TGC/ AGC (A host trigger is given and it can impact the other drivers behavior). For TOM, channels 0-7 are in same TGC and 8-15 in another TGC. For ATOM 0-7 are in a single AGC.
- It is preferred that all the shifted channels using reference channel in case of PwmHandleShiftByOffset = true should be present in the same TGC. Because all the channels in the TGC are started by a host trigger to enable all channels at the same time, Then the immediate next TGC is enabled, So in case if the shifted channel is present in next TGC it could have an impact.

Pwm_17_GtmCcu6 driver

- When notification for 0% or 100% is enabled by the user for fixed and variable period channels the notification for falling edge does not work for 0% duty for polarity high, similarly rising edge notification will not work 0% duty for polarity low, however the user can configure notification as both edges to get notification without worrying about the polarity.
- For fixed and variable period channels when notification is asked for falling edge for a channel whose polarity is HIGH and idle state LOW after calling SetOutputToIdle function, notifications are not generated. Similarly if a channel with polarity LOW and idle state HIGH and rising edge notification is asked, notification is not generated. This is because internally SetOutputToIdle function moves to 0% or 100% duty based on IDLE state and polarity.
- In case of non-coherent PWM channel with polarity PWM_LOW and idle state PWM_HIGH, when transition from idle state to non-zero duty cycle state, with rising edge notification enabled, a notification is provided even when there is no rising edge in the first cycle only, this due to CM1 compare match.
- In case of coherent PWM channel with polarity PWM_LOW and idle state PWM_LOW, when transition from idle state to non-zero duty cycle state, with rising edge notification enabled, a notification is provided even when there is no rising edge in the first cycle only, this due to CM0 compare match.
- For shifted (PwmHandleShiftByOffset = OFF) and center aligned channels incorrect period match notification is expected.
- When PwmShiftByOffset is ON, signal output generation will be sequential, if maximum channels are configured last set of channels may see incorrect signal levels in first cycle.
- Set output to idle for CCU6 channels

Applicable APIs: `Pwm_17_GtmCcu6_SetOutputToIdle` and `Pwm_17_GtmCcu6_DeInit`.

Behavior1:

Precondition: Polarity is HIGH and idle state is LOW

If set to idle is called when current state is HIGH, then switch to idle occurs after a delay due to hardware limitation.

Behavior2:

Precondition: Polarity is LOW and Idle State is HIGH.

If set to idle is called when current state is LOW, then switch to idle occurs after a delay due to hardware limitation.

Hardware Limitation: Duty and period values cannot be updated directly to running register.

Workaround in design is to stop the timer, update new values and perform shadow transfer, which leads to delay.

[cover parentID PWM={A0182969-A76A-4eee-AEF8-0FEE0170C4CE}]

- CCU6 Configuration:

Non-coherent update of duty cycle and period is not supported for PWM channels of type CCU6.

Centre aligned channels is not supported for PWM channel of type CCU6.

- For PWM channels of type ccu6, the first cycle will have period value as period+1 instead of period.
- Callback notification for a TOM or ATOM channel will be invoked by DSADC and not by PWM, when for the channel McuTomChannelEventHandledByDsadc = ON or McuAtomChannelEventHandledByDsadc = ON, respectively.
- Un-intended signal level changes (glitch) at the port pins are observed during the initialization phase of the PWM driver under certain scenarios. The scenarios are explained in the following sections:

Scenario 1:

A PWM channel with the following configuration will exhibit a glitch during its initialization:

`PwmAssignedHwUnit = GTM`

`PwmPolarity = PWM_LOW`

Pwm_17_GtmCcu6 driver

GtmTimerPortPinSelect = TOUT<n>_SEL_A_PORT<x>_PIN<y> (implies, TOUT configured with SELA line)

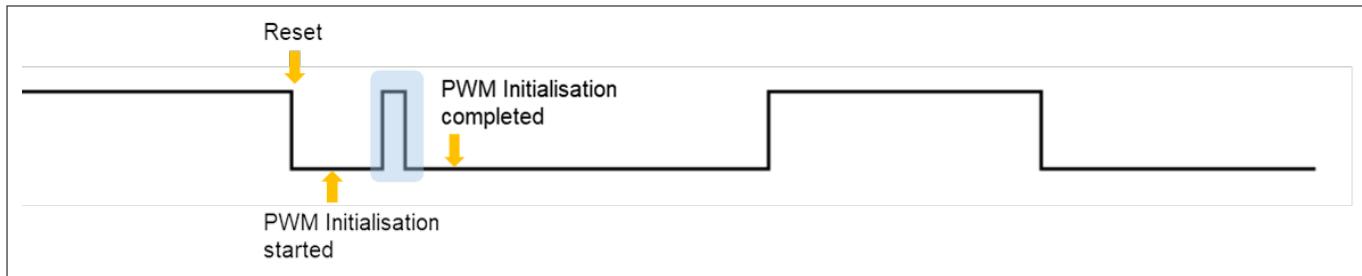


Figure 175 Glitch in Scenario 1

Scenario 2:

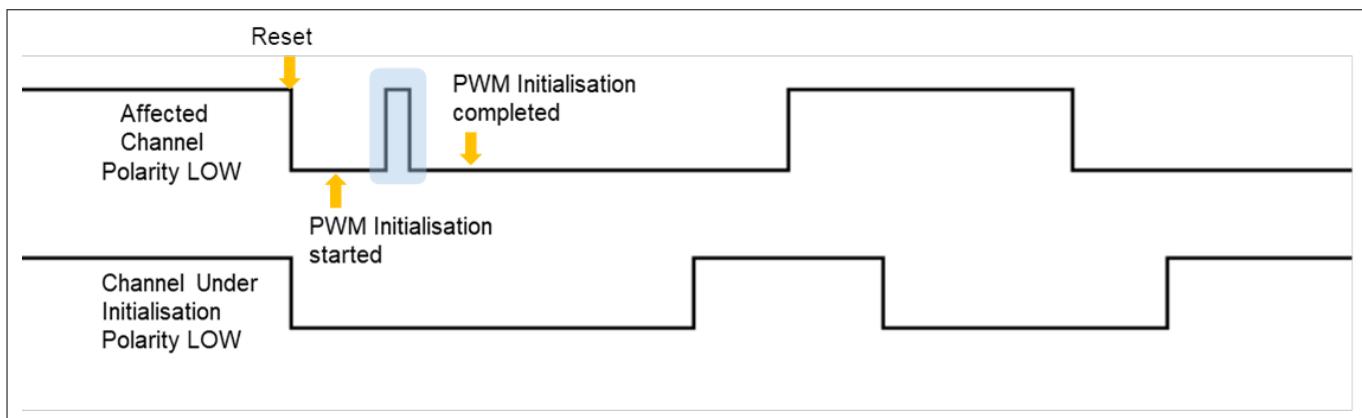
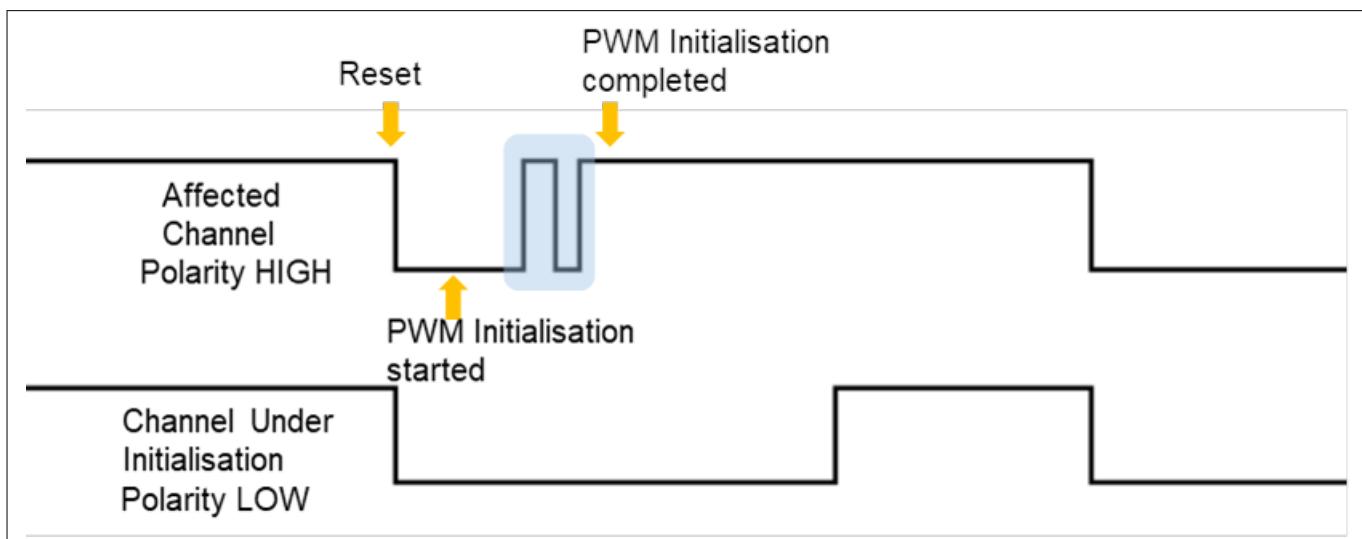
Initialization of one of the GTM PWM channels may trigger a glitch on the output pins of other TOUTSEL/S. The scenario occurs when the affected TOUTSEL/s holds a default value (HW reset value) such that the channel under initialization is driving the affected TOUTSEL (in turn the port pin also). Hence, during the initialization phase a glitch is observed on such channels.

The following table summarizes the scenario with an example:

Table 1745 Example Channel Configuration

Channel Under Initialization	Affected Channels
PwmAssignedHwUnit = GTM	PwmAssignedHwUnit = GTM
GtmTimerUsed = TOM2_7	GtmTimerUsed = ATOM5_7
GtmTimerPortPinSelect = TOUT101_SEL_B_PORT11_PIN12	<p>GtmTimerPortPinSelect =TOUT126_SELC_PORT11_PIN14</p> <p>Here,</p> <ol style="list-style-type: none"> 1. TOUT126_SELA_PORT11_PIN14 is connected to TOM2_7. 2. TOUT126 is connected to TOM2_7(SELA) after HW reset. 3. PortPinInitialMode of PORT11_PIN14 = ALT1 (GTM muxed output) and Port_Init() is completed.

Note: *Glitches are not observed if the PwmPolarity of "Channel Under Initialization" and "Affected Channels" is PWM_HIGH.*

Pwm_17_GtmCcu6 driver

Figure 176 Glitch in Scenario 2(a)

Figure 177 Glitch in Scenario 2(b)

Note: The above scenarios are applicable only during the initialization phase of the PWM driver (`Pwm_17_GtmCcu6_Init()`). Post-initialization such a glitch is not observed.

Work Around:

In order to avoid such a glitch on the port pins. The user may follow one of the following workarounds:

- To set the port pins used by the PWM channels as input during the initialization phase of the PWM driver. After completion of the initialization, the port pin's ALT mode can be restored to PWM.
- To set the port pins used for PWM channels as "Output Low" or "Output High" during the initialization phase of the PWM driver. After completion of the initialization, the port pin's ALT mode can be restored to PWM.

20.3.9 Unsupported hardware features

The following hardware features of the CCU6 timer are not supported by the PWM driver:

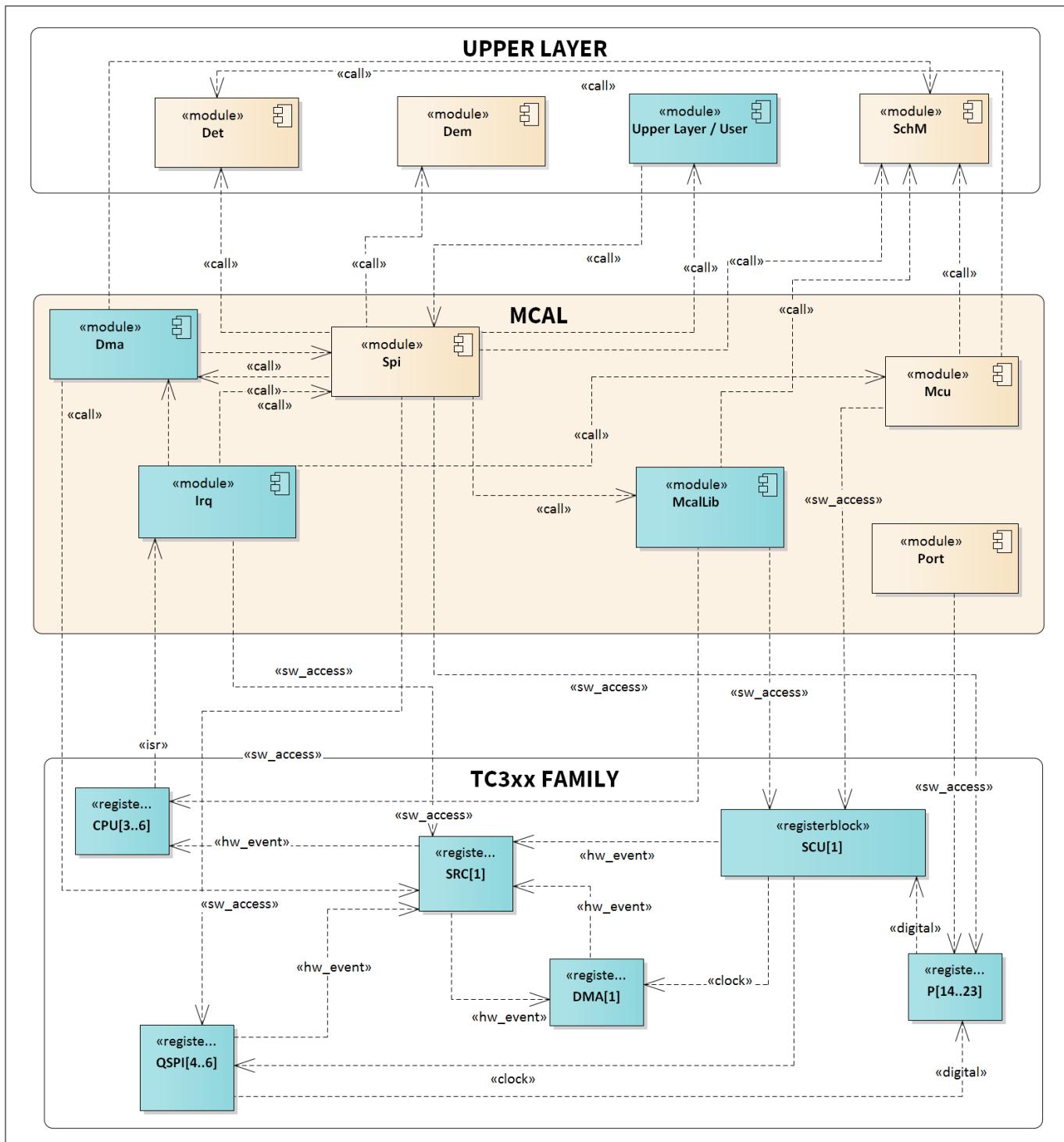
- Capture mode of CCU6 T12 and T13 timer units
- Trap handling of CCU6
- Multi-channel mode
- Hall sensor mode

SPI driver**21 SPI driver****21.1 User information****21.1.1 Description**

The SPI driver operates in the master and full duplex communication modes only. The driver supports synchronous and asynchronous communication supporting Level-0, Level-1 and Level-2 type configurations.

21.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

SPI driver

Figure 178 Mapping of hardware-software interfaces

21.1.2.1 QSPI: primary hardware peripheral
Hardware functional features

The SPI driver uses the QSPI for Synchronous and Asynchronous data transfer. The key hardware functional features used by the driver are:

- QSPI FIFOs (Tx and Rx) are configured to work in the continuous data mode

SPI driver

- QSPI FIFOs (Tx and Rx) interrupts are configured to work in the single move mode
- SPI driver uses the QSPI move counter mode during asynchronous data transfer

The unsupported features of the QSPI are:

- High speed input capture
- Slave mode
- Long data block transfer
- ASCLIN
- MIX entry

Users of the hardware

The SPI driver exclusively utilizes the QSPI module.

Hardware diagnostic features

The SMU alarms configured for the QSPI are not monitored by the SPI driver.

Hardware events

The SPI driver uses the following hardware events from the QSPI IP:

- On a transmitter FIFO event - TXF
- On a receiver FIFO event - RXF
- On an error condition (TxFIFO underflow / overflow, RxFIFO underflow / overflow, Expect timeout, parity error) - ERRORFLAGS
- On phase transition (end of frame) - PT2

21.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The SPI driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the FSPB, fQSPI clock signals for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the SPI driver.

Hardware events

Hardware events from the SCU are not used by the SPI driver.

21.1.2.3 SRC: dependent hardware peripheral

Hardware functional features

The SPI driver depends on the interrupt router for raising an interrupt to the CPU or DMA based on the transmit FIFO event, receive FIFO event, error conditions and Phase transition, which indicates the status of data transmission and reception.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software.

Hardware diagnostic features

The SMU alarms configured for the interrupt router are not monitored by the SPI driver.

Hardware events

SPI driver

The interrupt events raised by the interrupt router are serviced by the CPU or DMA. The SPI driver provides interrupt handlers as software interfaces, which must be invoked from the ISR.

21.1.2.4 Port: dependent hardware peripheral

Hardware functional features

The MOSI, MISO, SCLK and SLSO signals are routed to the QSPI through the port pads. MOSI, MISO, SCLK and SLSO configured and enabled through the PORT driver. For CS_VIA_GPIO, the PORT registers are directly accessed by the SPI driver for asserting/de asserting the chip select (SLSO).

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from port pads are not used by the SPI driver.

21.1.2.5 DMA: dependent hardware peripheral

Hardware functional features

The SPI driver uses the DMA in the Linked list mode for the transmission and reception of data in the Asynchronous mode (Level-1, 2) of transfer. The SPI driver uses the interface APIs provided by the DMA driver to use the DMA functionality.

Users of the hardware

The DMA module is exclusively owned by the DMA driver, but the functionality is shared by many MCAL drivers. The DMA module is triggered for every element transmitted or received on the QSPI interface.

Hardware diagnostic features

The move engine (ME) error is enabled during the data transmission.

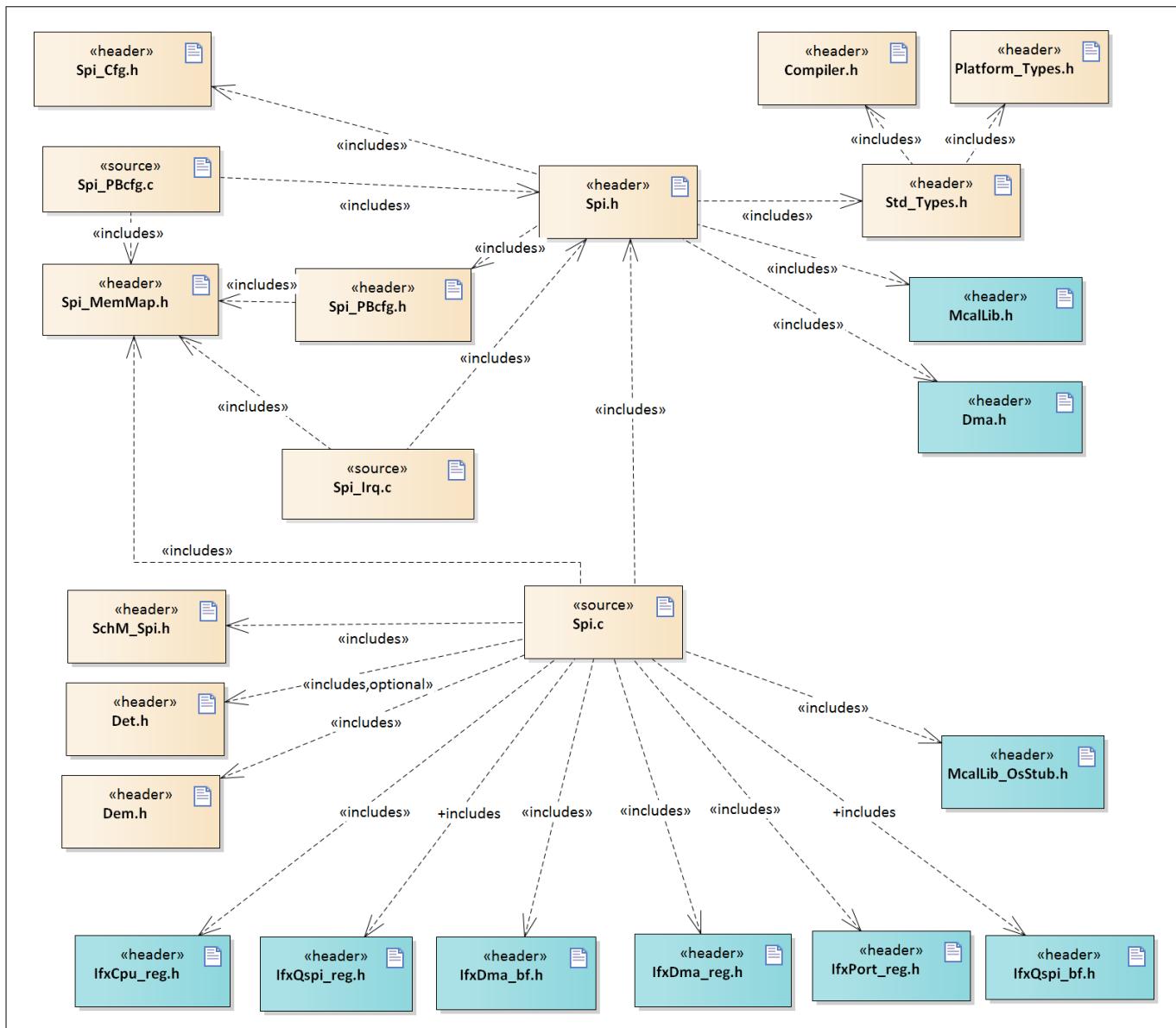
Hardware events

If any ME error is encountered during the data transfer then the DMA raises an error which is handled by the DMA driver.

21.1.3 File structure

21.1.3.1 C file structure

This section provides details of the C files of the SPI driver.

SPI driver

Figure 179 **C file structure**
Table 1746 **C file structure**

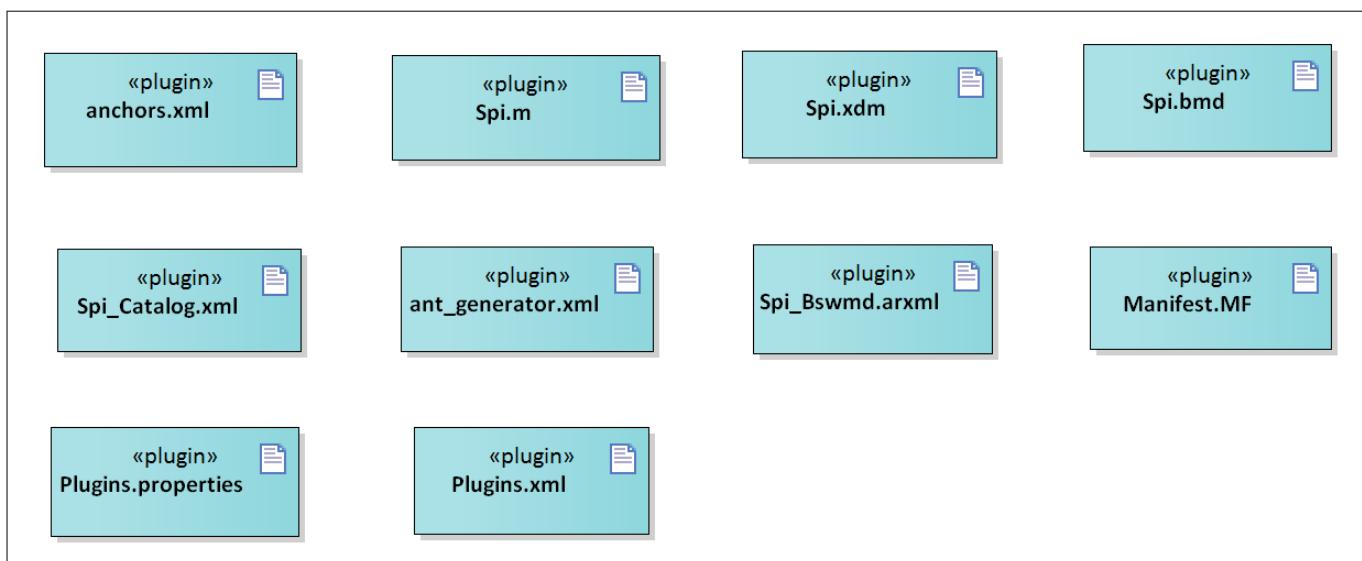
File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Dem.h	Provides the exported interfaces of Diagnostic Event Manager
Det.h	Provides the exported interfaces of Development Error Tracer
Dma.h	Header file (static) defining prototypes of data structures and APIs
IfxCpu_reg.h	SFR header file for CPU
IfxDma_bf.h	SFR header file for DMA
IfxDma_reg.h	SFR header file for DMA
IfxPort_reg.h	SFR header file for Port
IfxQspi_bf.h	SFR header file for QSPI

SPI driver
Table 1746 C file structure (continued)

File name	Description
IfxQspi_Reg.h	SFR header file for QSPI
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore™. This shall be included by other drivers to call OS APIs.
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Spi.h	Export Header for Schm functions of SPI driver. Functions to protect the critical sections.
Spi.c	File (Static) containing implementation of APIs
Spi.h	Header file (Static) defining prototypes of data structures and APIs
Spi_Cfg.h	Header file (Generated) containing constants and pre-processor macros
Spi_Irq.c	IRQ file for handling all QSPI interrupts.
Spi_MemMap.h	Memmap file is used to define the section of memory to which variables or constants will be placed
Spi_PBcfg.c	File (Generated) containing objects to data structures
Spi_PBcfg.h	File (Generated) containing declaration of the post-build configuration data structures
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

21.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the SPI driver.


Figure 180 Code generator plugin files

SPI driver

Table 1747 Code generator plugin files

File name	Description
Manifest.MF	Tresos plugin support file containing the metadata for SPI driver
Plugins.properties	Tresos plugin support file for the SPI driver
Plugins.xml	Tresos plugin support file for the SPI driver
Spi.bmd	AUTOSAR format XML data model schema
Spi.m	Macros for XDM logic verification
Spi.xdm	Tresos format XML data model schema file
Spi_Bswmd.arxml	AUTOSAR format module description file
Spi_Catalog.xml	AUTOSAR format catalogue file
anchors.xml	Tresos anchors support file for the SPI driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point

21.1.4 Integration hints

This section describes the key points that an integrator or user must consider to use the features of SPI driver.

21.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the SPI driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the Spi_MemMap.h file.

The Spi_MemMap.h file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

SPI driver

```

#if defined SPI_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
/*User pragmas here for Non-cached LMU*****/
#endif

#define SPI_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR

#elif defined SPI_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#ifndef _TASKING_C_TRICORE_
/*User pragmas here for Non-cached LMU*****/
#endif

#define SPI_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR

/* CORE[x] CONFIG DATA -- PF[x] ****/ /*[x]=0..5*/
#ifndef SPI_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
/*User pragmas here for PF[x]*****/
#endif

#define SPI_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#define MEMMAP_ERROR

#ifndef SPI_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
/*User pragmas here for PF[x]*****/
#endif

#define SPI_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#define MEMMAP_ERROR

/* CODE -- PF[x] ****/
#ifndef SPI_START_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here for PF[x]*****/
#endif

#define SPI_START_SEC_CODE_ASIL_B_GLOBAL
#define MEMMAP_ERROR

#ifndef SPI_STOP_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here for PF[x]*****/
#endif

#define SPI_STOP_SEC_CODE_ASIL_B_GLOBAL
#define MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "SPI_MemMap.h, wrong pragma command"
#endif

```

• DET

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The SPI driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the SPI driver must process all the errors reported to the DET module through the `Det_ReportError()` API. The files `Det.h` and `Det.c` are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

• DEM

The DEM module is a part of the AUTOSAR stack that handles all the production errors reported by the BSW modules. The SPI driver reports all the production errors to the DEM modules through the `Dem_ReportErrorStatus()` API. The user of the SPI driver must process all the production errors (fail / pass) reported to the DEM module through the `Dem_ReportErrorStatus()` API. The files `Dem.h` and `Dem.c` are provided in the MCAL package as a stub code and needs to be replaced with a complete DEM module during the integration phase.

Note: Reentrancy of the `Spi_SyncTransmit` API is dependent on the reentrancy of `Dem_ReportErrorStatus()` API. As per their design, the modules APIs are reentrant for

SPI driver

different hardware units. However, in case `Dem_ReportErrorStatus()` API is implemented as non-reentrant, the APIs inherit the property of the same.

- **SchM**

The SchM module is a part of the RTE that manages the BSW Scheduler. The SPI driver uses the exclusive areas defined in `SchM_Spi.h` file to protect the SFRs and variables from concurrent accesses from different threads. the SchMs identified for the SPI driver are:

- Queue_Update
- ChannelLock
- SyncLock

The `SchM_Spi.h` and `SchM_Spi.c` files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the SPI driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows.

SPI driver

```
***** Sample implementation of SchM_Spi.c *****

void SchM_Enter_Spi_Queue_Update(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Spi_Queue_Update(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Spi_ChannelLock(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Spi_ChannelLock(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Spi_SyncLock(void)
{
    /* Start of Critical Section */
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Spi_SyncLock(void)
{
    /* End of Critical Section */
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

The SPI driver reports all the detected safety errors through the `Mcal_ReportSafetyError()` API. The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the files `Mcal_SafetyError.c` and `Mcal_SafetyError.h` as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

SPI driver

The SPI driver implements notification functions `Spi_JobEndNotification` and `Spi_SeqEndNotification` for job and sequence completion respectively. These notification functions can be configured by the user in the EB tresos tool for each job and sequence separately.

Note: *Job and Sequence end notifications are only available for asynchronous communication.*

In Asynchronous communication, user should configure `Spi_QspiDmaCallout` function as the DMA call-back for RX channel in the respective DMA channel configuration. The configured call-back function `Spi_QspiDmaCallout` is triggered by DMA driver after completion of each channel transmission for updating BACON and start the transfer for successive channel.

- **Operating system (OS)**

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application. The OS files provided by the MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

21.1.4.2 Multicore and Resource Manager

The SPI driver supports execution of its APIs simultaneously from all CPU cores. The user should allocate resources of the SPI to the CPU cores at pre-compile time using the Resource Manager module. The following are the key points to be considered with respect to multicore in the driver:

- A kernel can be assigned to only one core and cannot be shared between cores. Multiple kernels can be assigned to a core.
- Channels can be re-used within cores, however, protection of data must be taken care by the application code.
- Application must ensure that the channel, job and sequence numbers passed to API belong to same core, else respective DET is triggered from the driver.
- Interrupts raised by the hardware must be serviced by the CPU core to which the kernel is allocated to.
- Locating of constants, variables and configuration data to the correct memory space should be done by the user. Memory sections are marked GLOBAL (common to all cores) and CORE[x](specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of the SPI driver is placed under single MemMap section. It can be relocated to any PFlash region.

Data section:

The RAM variable memory sections marked as specific to a core should be re-located to the DPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region.

Configuration data and constants:

The configuration data sections marked as specific to a core should be re-located to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

Note: *Relocating code, data or constants to a distant memory region would impact execution timings.*

Note: *If the driver operates from single (master) core, all the sections may be relocated to the PFlash/ DPR/DLMU of the same CPU core.*

SPI driver

21.1.4.3 MCU support

The SPI driver is dependent on the MCU driver for clock configuration. The initialization of the SPI driver must be started only after completing the MCU initialization. The following must be considered while configuring the MCU driver in tresos:

In the MCU configuration, following fields are to be configured in McuClockSettingConfig:

- McuQspiClockSourceSelection
- McuQspiFrequency

An example of selecting the clock source and suitable frequency is shown in the following screenshot.

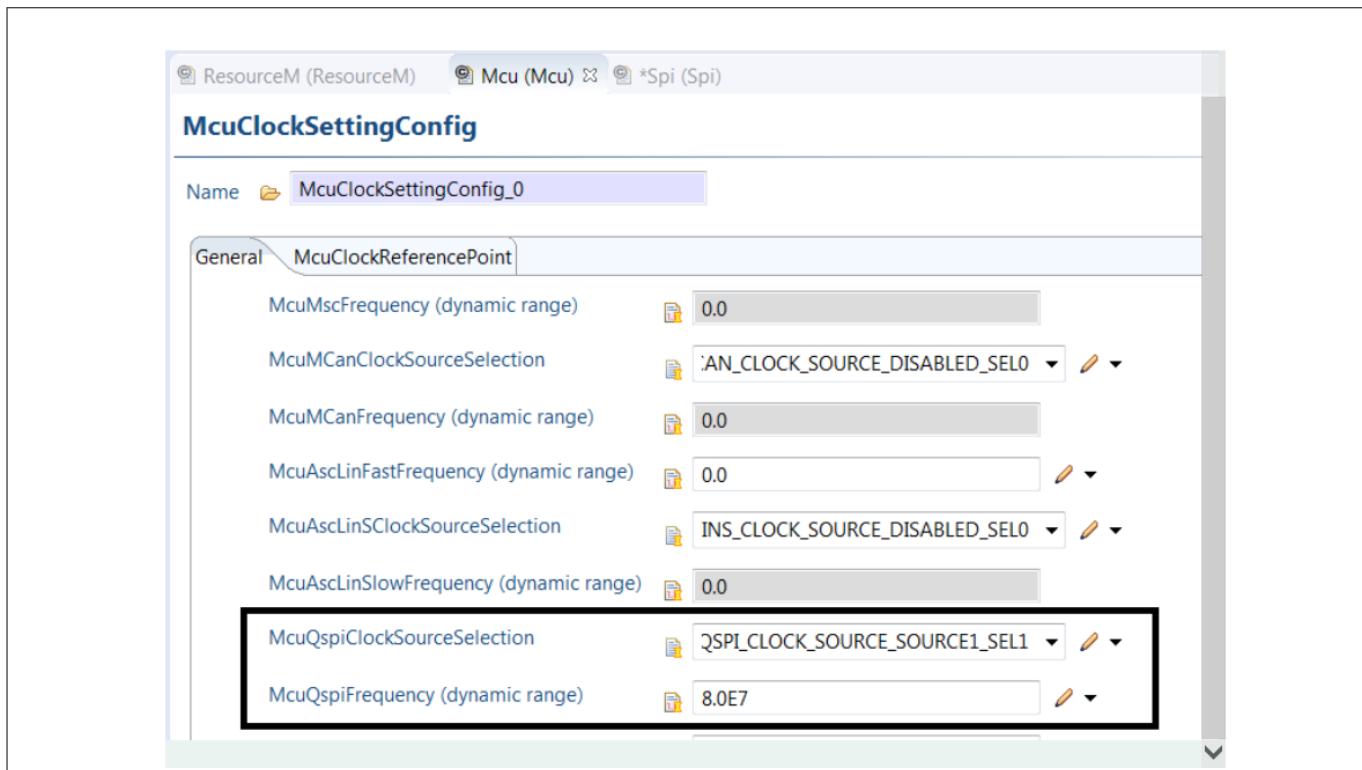


Figure 181 **QSPI Clock / Frequency selection**

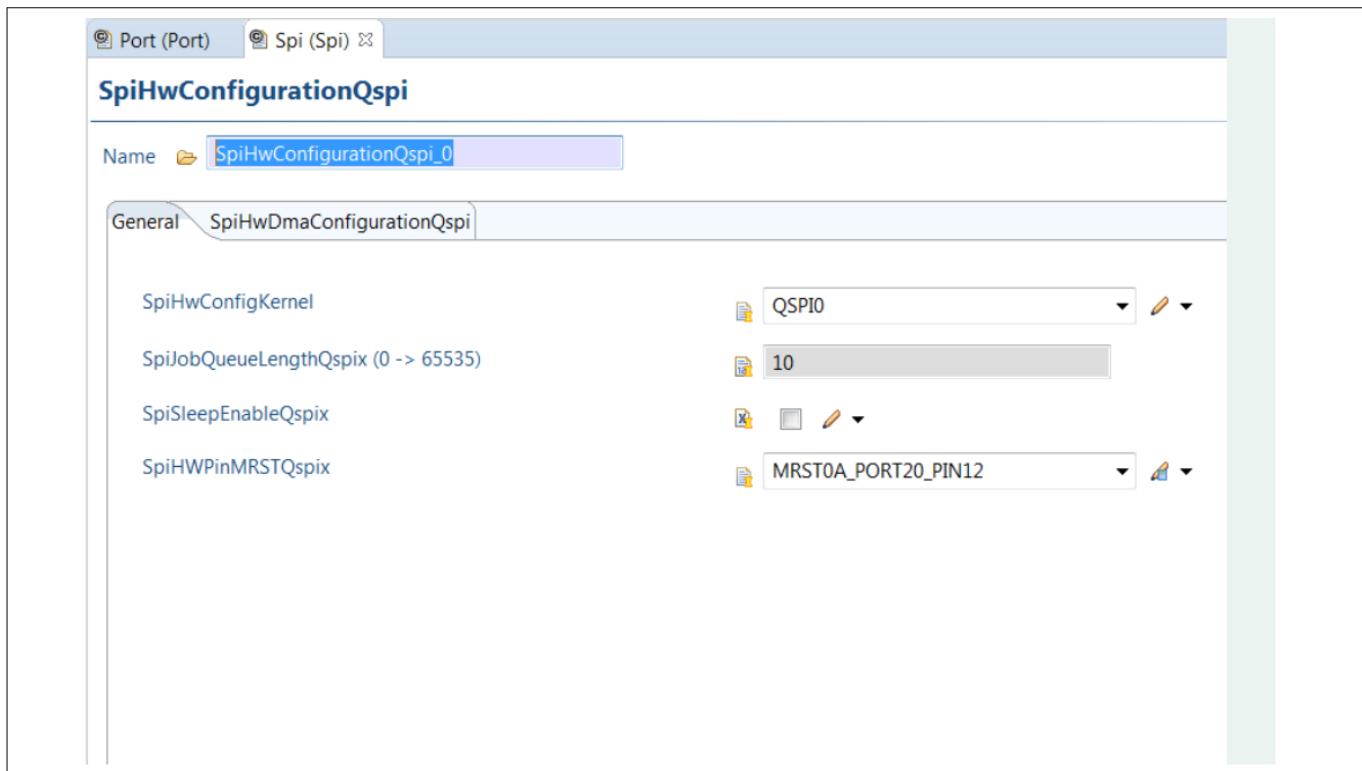
21.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the SPI driver through the PORT configuration and initialize the port pins prior to invoking the SPI initialization.

Following port pins for QSPI are to be configured as per the configuration:

- MRST - master receive slave transmit
- MTSR - master Transmit slave receive
- CLOCK - clock pin
- SLSO - hardware driven chip select OR GPIO to be operated as chip select

An example configuration for QSPI-0 is shown in the following diagram. Note that the MRST should be configured in the respective hardware window (SpiHwConfigurationQspi_0):

SPI driver

Figure 182 **QSPI-0 MRST configuration**

PortContainer											
PortPin											
Index	Name	PortPinId	PortPinSymbolicName	PortPinDirection	P...	PortPinI...	PortPinLe...	PortPin...	PortPin...	PortPin...	PortPin...
0	PortPin_0	320	PORT_20_PIN_0	PORT_PIN_IN	X	GPIO	PORT_PIN...	X	X	X	X
1	PortPin_1	321	PORT_20_PIN_1	PORT_PIN_IN	X	GPIO	PORT_PIN...	X	X	X	X
2	PortPin_10	332	PORT_20_PIN_12	PORT_PIN_IN	X	GPIO	PORT_PIN...	X	X	X	X
3	PortPin_11	333	PORT_20_PIN_13	PORT_PIN_OUT	X	ALT5	PORT_PIN...	X	X	X	X
4	PortPin_12	334	PORT_20_PIN_14	PORT_PIN_IN	X	GPIO	PORT_PIN...	X	X	X	X
5	PortPin_2	322	PORT_20_PIN_2	PORT_PIN_IN	X	GPIO	PORT_PIN...	X	X	X	X
6	PortPin_3	323	PORT_20_PIN_3	PORT_PIN_IN	X	GPIO	PORT_PIN...	X	X	X	X
7	PortPin_4	326	PORT_20_PIN_6	PORT_PIN_IN	X	GPIO	PORT_PIN...	X	X	X	X
8	PortPin_5	327	PORT_20_PIN_7	PORT_PIN_IN	X	GPIO	PORT_PIN...	X	X	X	X
9	PortPin_6	328	PORT_20_PIN_8	PORT_PIN_OUT	X	ALT3	PORT_PIN...	X	X	X	X
10	PortPin_7	329	PORT_20_PIN_9	PORT_PIN_OUT	X	ALT3	PORT_PIN...	X	X	X	X
11	PortPin_8	330	PORT_20_PIN_10	PORT_PIN_IN	X	GPIO	PORT_PIN...	X	X	X	X

Figure 183 **Configure MTSR, CLK and SLSO**

SPI driver

21.1.4.5 DMA support

DMA channels should be configured when the QSPI is operated in the Level-1 or Level-2 asynchronous mode. QSPI uses two DMA channels one for RX and another for TX of QSPI. These DMA channels must be reserved for the QSPI communication only and cannot be reused.

For internal buffers (Spi_TxIBufferCorex, Spi_RxIBufferCorex) and External buffers, Address space 0xD and 0xC shall not be used for DMA related usage. MemMap sections allocating memory in scratch pad RAM should always generate global addresses instead of local addresses.

The following configurations for DMA are applicable:

1. In the DMA, in the General configuration section, enable DmaTriggerApiConfiguration as minimum configuration. Enable other configuration items as required by application.
2. Add the respective DMA channel and configure the notification for the DMA Receive channel only.

No other configurations are required in DMA. Transaction control set configurations for DMA are handled in SPI module and does not need any configuration in DMA module.

Configure the QSPI priorities and DMA channels priorities as shown in the following figure.

Note: *QSPI TX and RX interrupt priority - respective DMA channel numbers are to be allocated as shown in the sample.*

Index	Name	DmaCh...	DmaCh...	DmaCh...	DmaTc...	DmaChannelNotificati...
0	DmaChannelConfig_0	0	0	1	1	Spi_QspiDmaCallout
1	DmaChannelConfig_1	1	0	1	1	NULL_PTR
2	DmaChannelConfig_2	2	0	1	1	NULL_PTR
3	DmaChannelConfig_3	3	0	1	1	NULL_PTR

Figure 184 DMA channel configuration

SPI driver

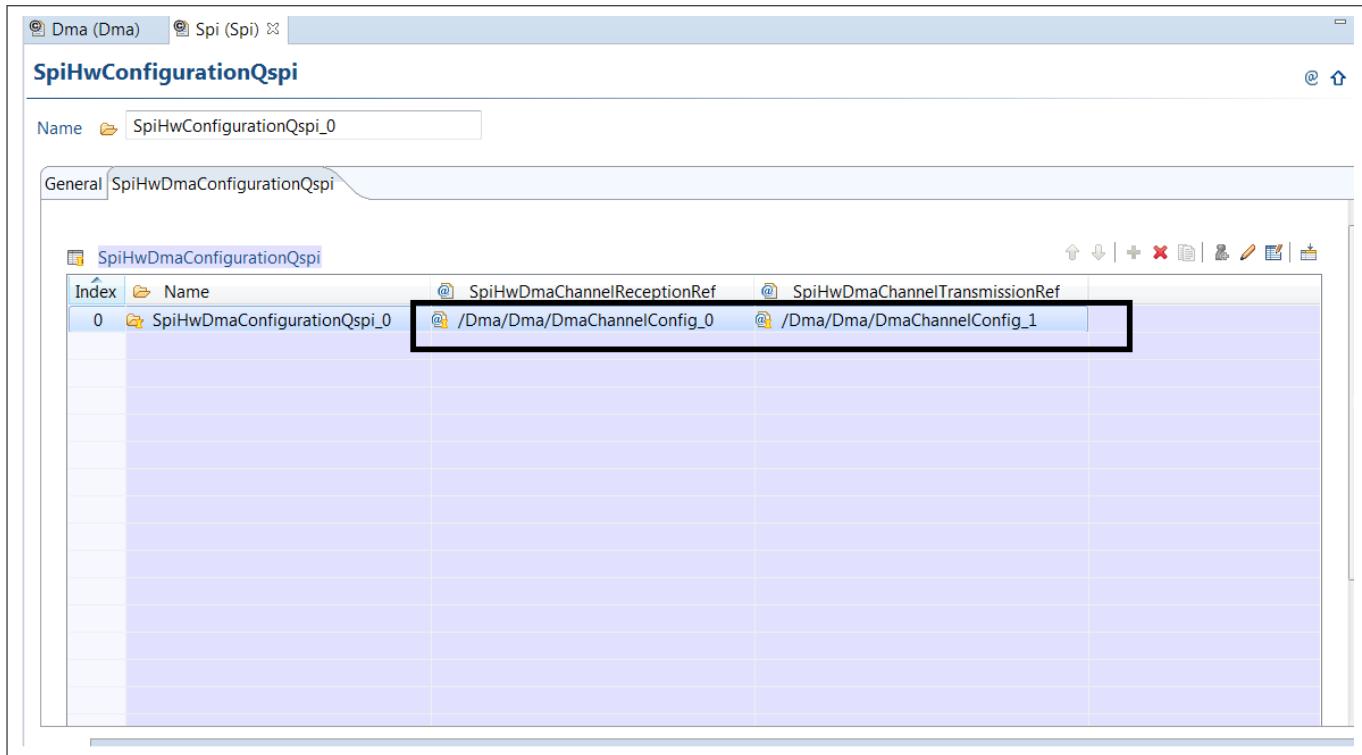


Figure 185 DMA Channel Assignment - SpiHwConfigurationQspi container

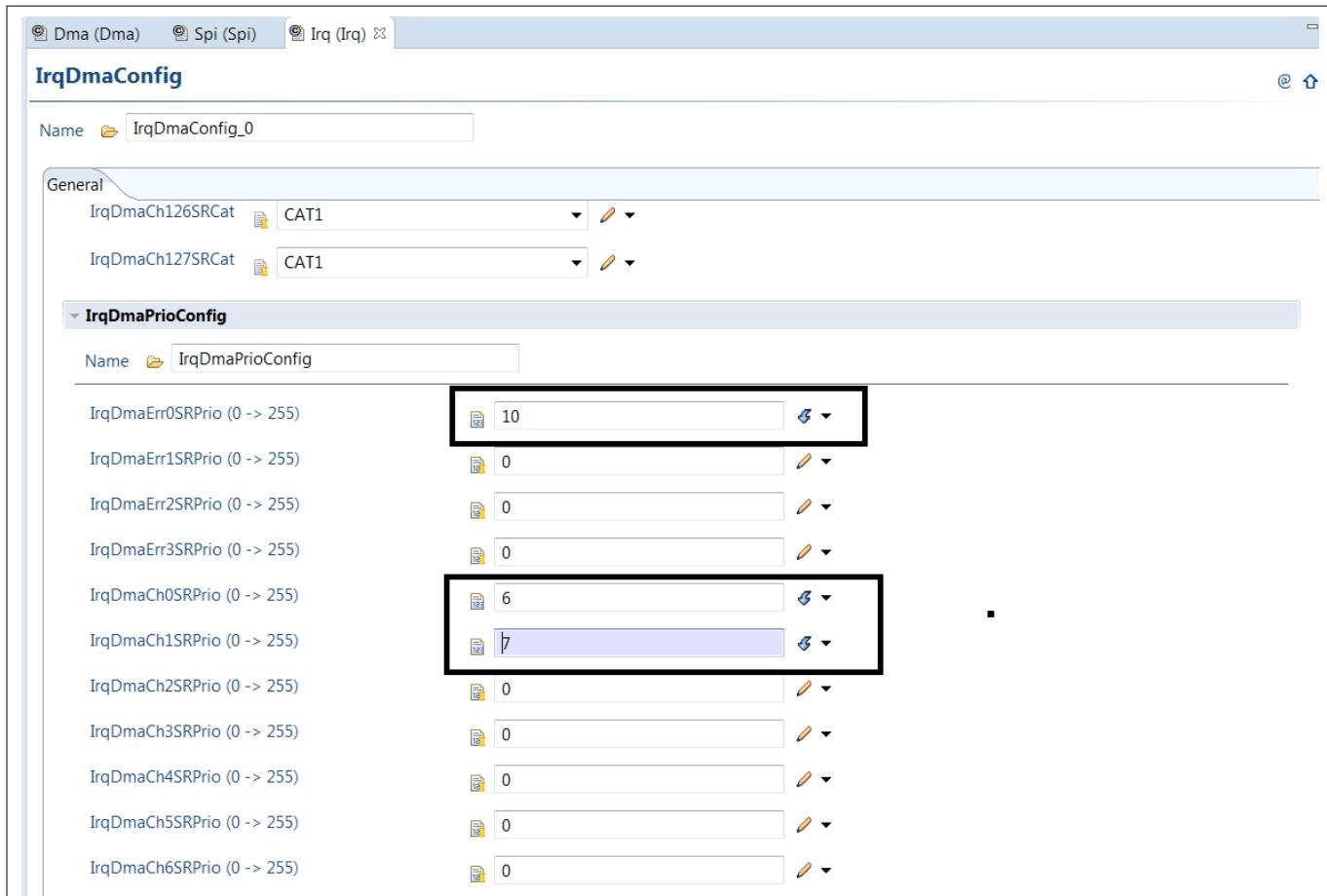
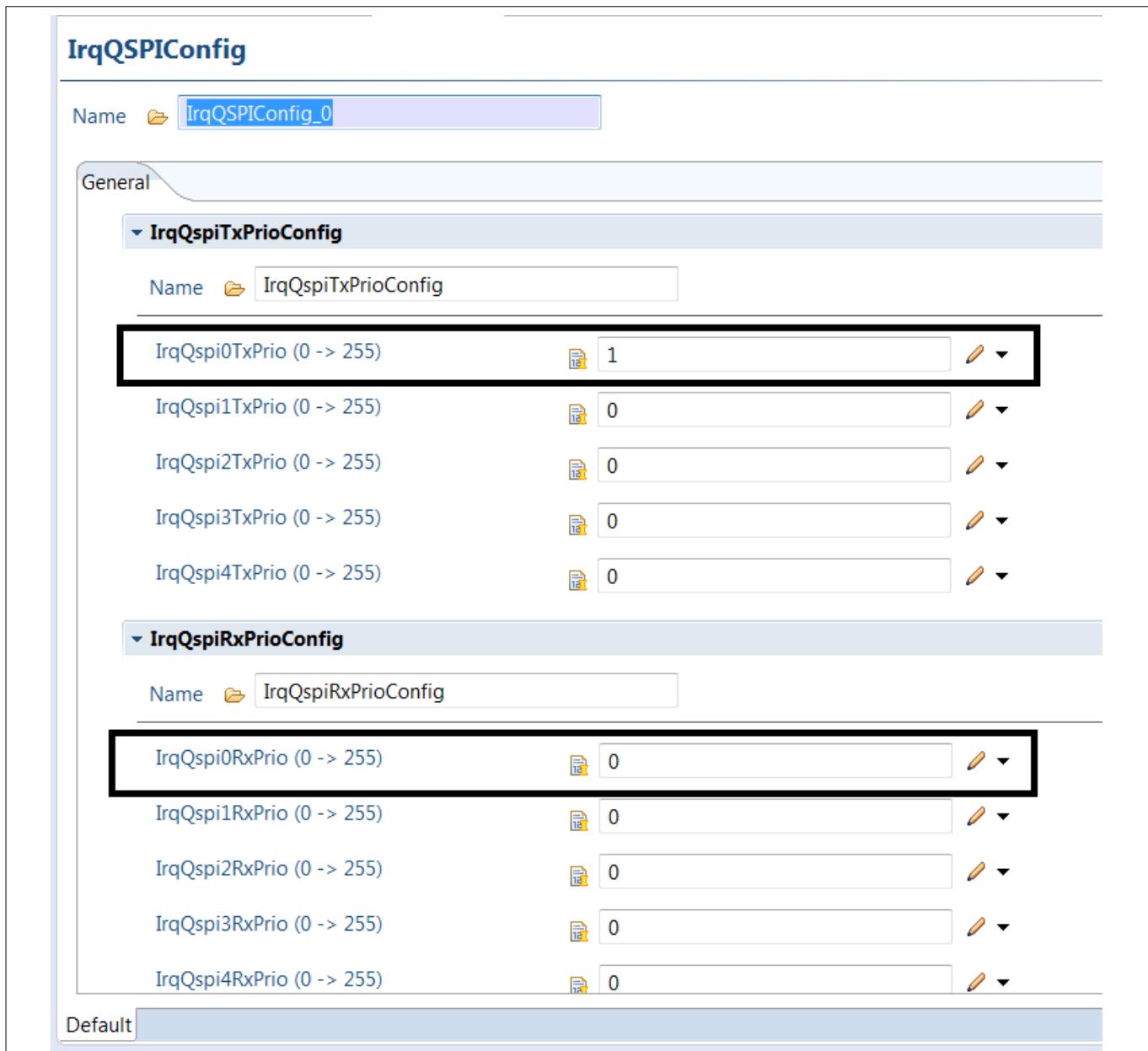


Figure 186 Assign priorities for DMA channels - IrdmConfig container

SPI driver
**Figure 187**

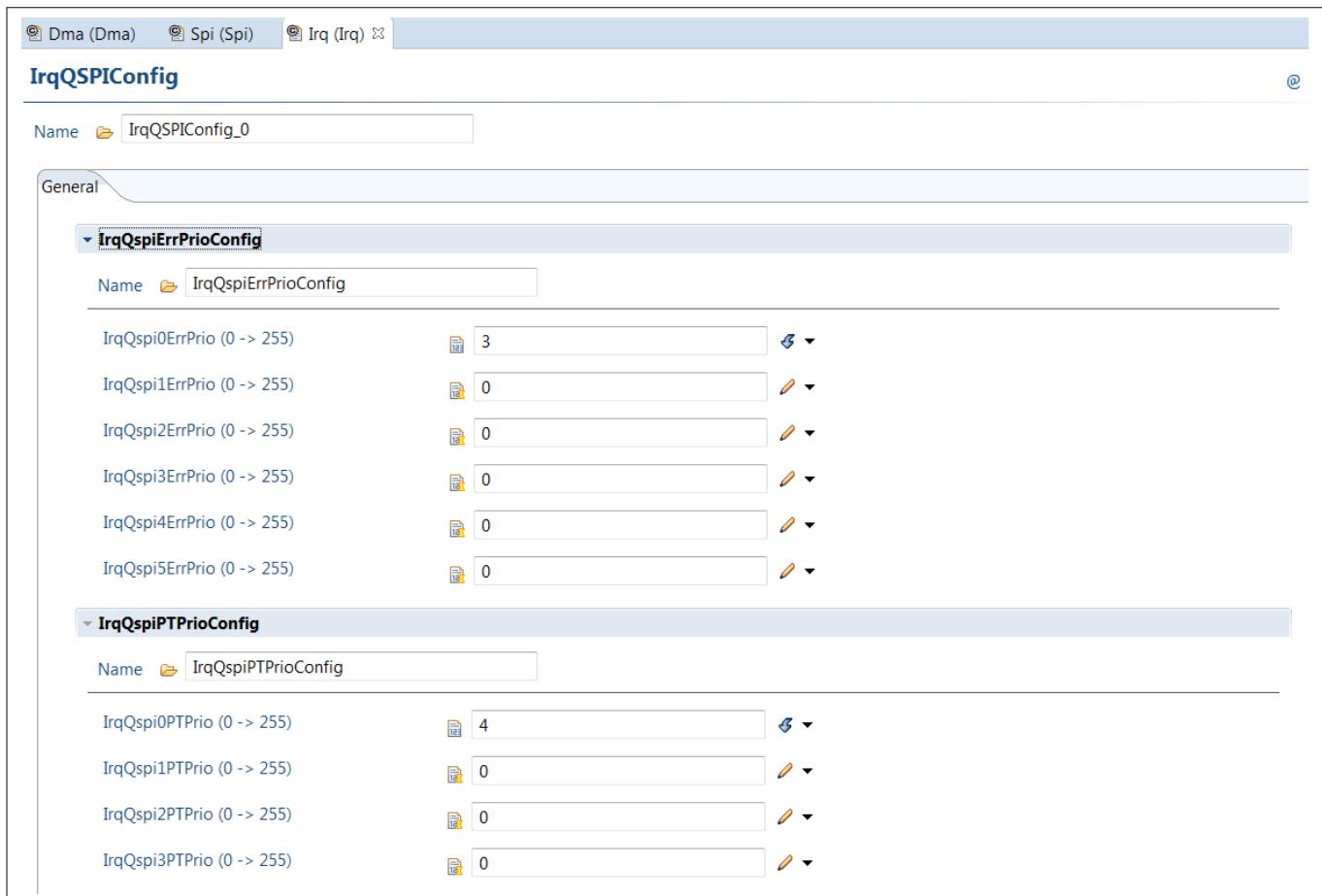
21.1.4.6 Interrupt connections

The interrupt connections of the SPI driver are described in this section.

- TX - QSPI Transmit - Triggers the DMA channel
- RX - QSPI Receive - Triggers the DMA channel
- PT2 - QSPI job frame complete - Triggers at the end of frame transmission (Job transmission complete)
- DMA would trigger a callout at the end of channel transmission and BACON will be updated for next successive channel transmission.

Note: Priority order: TX > RX > PT2.

Note that all the above interrupts are configured for asynchronous communication only.

SPI driver**Figure 188 Configure error and PT interrupt of QSPI**

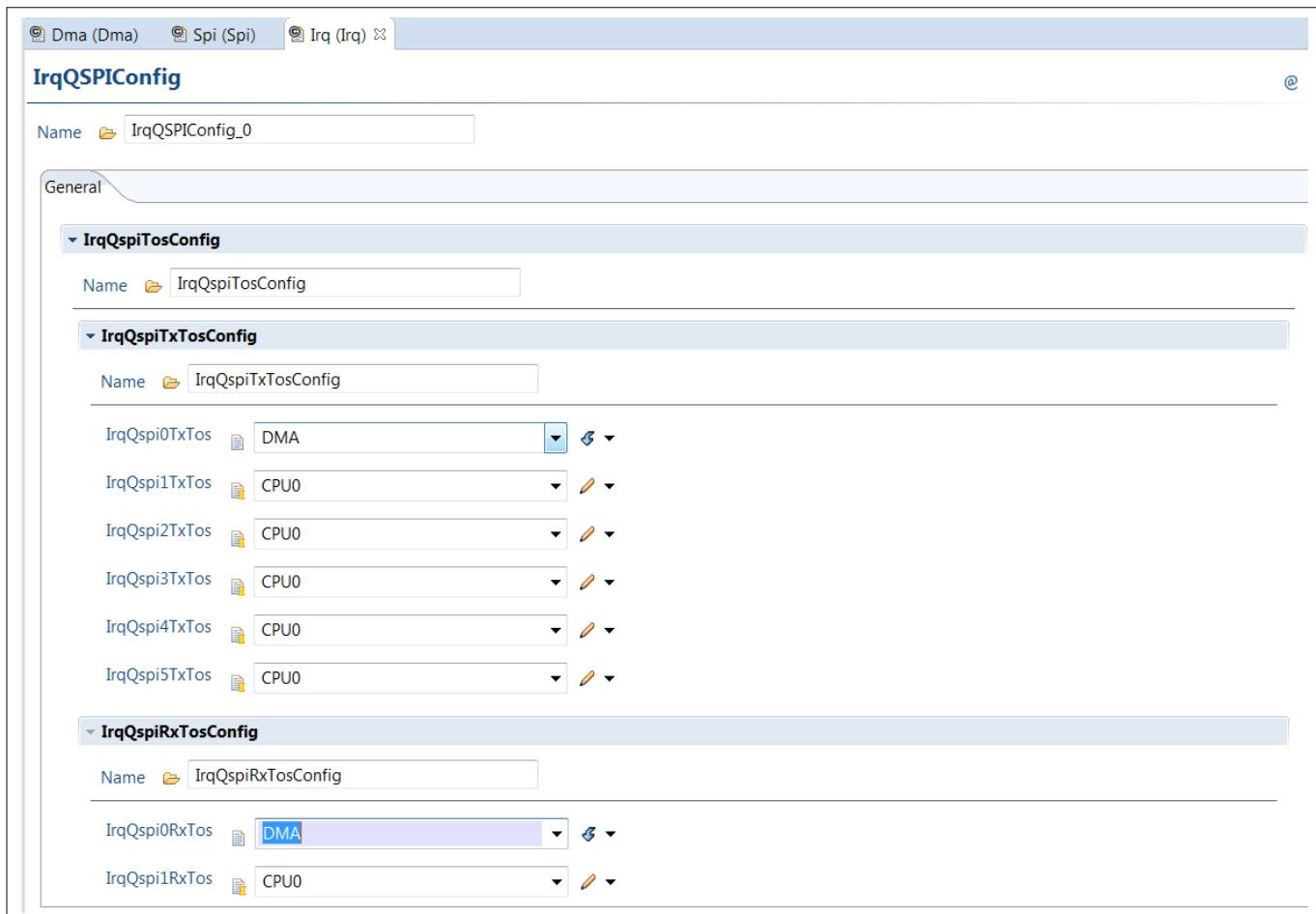
SPI driver


Figure 189 Configure type of service (TOS) - DMA / CPU

Note: *IrqQspi0ErrTos and IrqQspi0PTTos configured for CPUx.*

Note: *All interrupts must be configured for Asynchronous transmission (Level-1 and Level-2)*

A sample interrupt handler for QSPI0 kernel is depicted in the following code snippet:

```

/* Module header file inclusion */
#include "Spi.h"

ISR(QSPI0ERR_ISR)
{
    /* Call QSPI0 Error Interrupt handler */
    Spi_IsrQspiError(SPI_QSPI0_INDEX);
}

ISR(QSPI0PT_ISR)
{
    /* Call QSPI0 PT2 interrupt handler for frame completion */
    Spi_IsrQspiPT2(SPI_QSPI0_INDEX);
}

```

SPI driver

Note: A sample invocation of interrupts for DMA is depicted as follows (applicable for Level-1 and Level-2):

```
ISR(DMAERR0SR_ISR)
{
    /* Handle error through respective DMA ME */
    Dma_MEInterruptDispatcher();
}

ISR(DMACH0SR_ISR)
{
    /* DMA RX interrupt handler, SPI callback will be called through this
    interrupt */
    Dma_ChInterruptHandler(0U);
}
```

Note: *Note:*

The following API calls are allowed to use within the SPI callback notifications:

- Spi_ReadIB
- Spi_WriteIB
- Spi_SetupEB
- Spi_GetJobResult
- Spi_GetSequenceResult
- Spi_GetHWUnitStatus
- Spi_Cancel

All other SPI handler/driver APIs must not be called.

SPI driver**21.1.4.7 Example usage**

The following are the pre-requisite for SPI initialization:

Note: Global that needs to be defined in application code (Refer to the respective ***_PBCfg.c files of dependent modules for more details):

- Mcu_ConfigType Mcu_Config
- Port_ConfigType Port_Config
- Dma_ConfigType Dma_Config (Applicable for Level1 and Level2)
- Spi_ConfigType Spi_Config

Refer to the Integration hints of SPI driver and add all the dependent modules.

Follow the below sequence in the application code:

1. Initialize the MCU and clock using the Mcu_Init API.
2. Initialize the PORT driver using the Port_Init API.
3. Initialize the DMA driver using the Dma_init API, if the level configured is 1 or 2. This step is not applicable for Level-0 configuration.
4. Initialize the IRQ for dependent modules using the IrqDma_Init API and IrqSpi_Init API. Note that the IRQ driver is not a productive module and code provided is only a sample code. This step is not applicable for Level-0 configurations.
5. Initialize the SPI driver using the Spi_Init API.

SPI driver**Sample code for initialization**

```
extern const Mcu_ConfigType Mcu_Config;
extern const Port_ConfigType Port_Config;
extern const Spi_ConfigType Spi_Config;

#if (SPI_LEVEL_DELIVERED != 0)
extern const Dma_ConfigType Dma_Config;
#endif

int core0_main (void)
{
/* Initialize all dependent modules */
/* MCU Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Port Initialization */
Port_Init(&Port_Config);

/* Configure IRQ module */
#if (SPI_LEVEL_DELIVERED != 0)
IrqDma_Init();
IrqSpi_Init();
Dma_Init(&Dma_Config);

/* Enable service request for all the configured interrupts */
SRC_DMACH0.U |= 0x400U;
SRC_DMACH1.U |= 0x400U;
SRC_QSPI0RX.U |= 0x400;
SRC_QSPI0ERR.U |= 0x400;
SRC_QSPI0TX.U |= 0x400;
SRC_QSPI0PT.U |= 0x400;
#endif

/* Initialize SPI module */
Spi_Init(&Spi_Config);
}
```

Sample configuration for Level-0

SPI driver

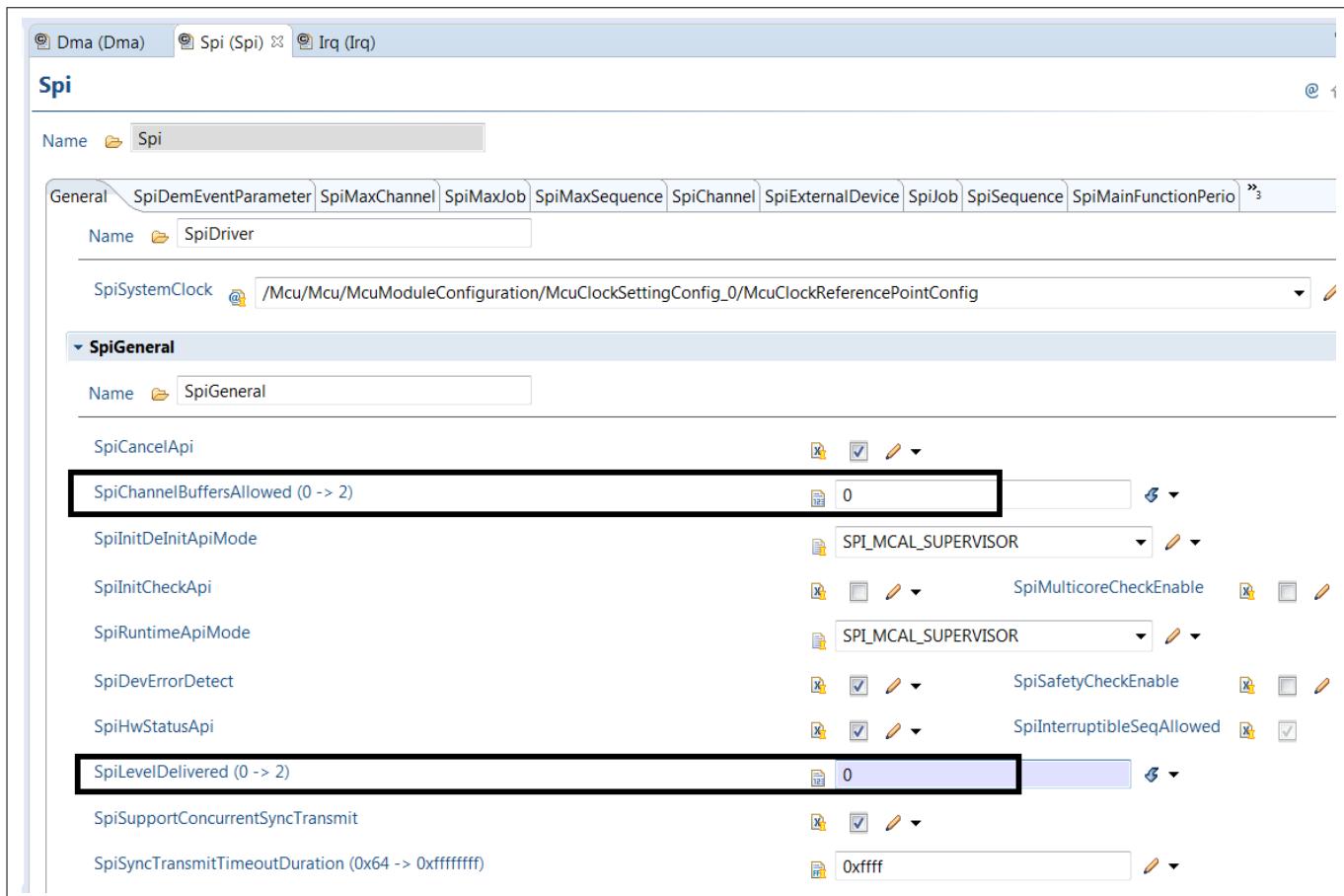


Figure 190 Create a configuration for Level - 0 with buffers as IB

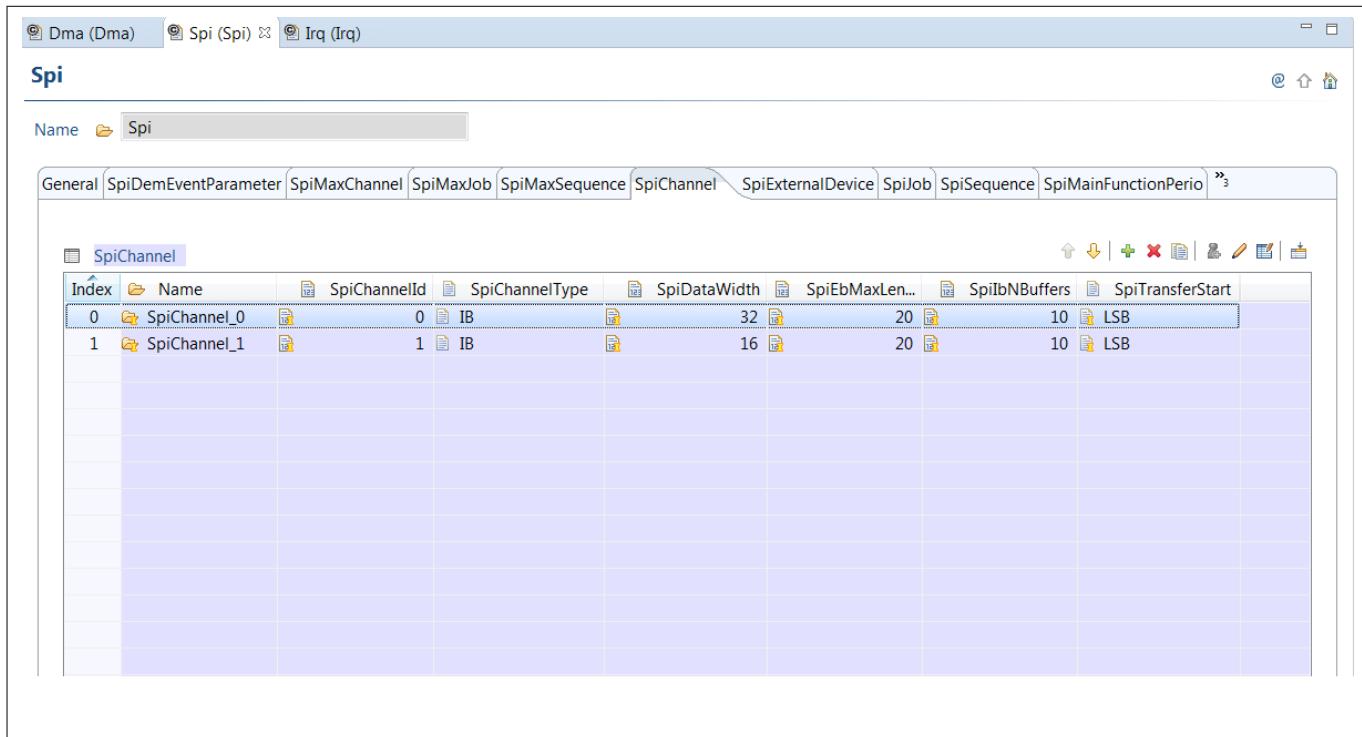


Figure 191 Create a channels of IB type and define datawidth and length

SPI driver

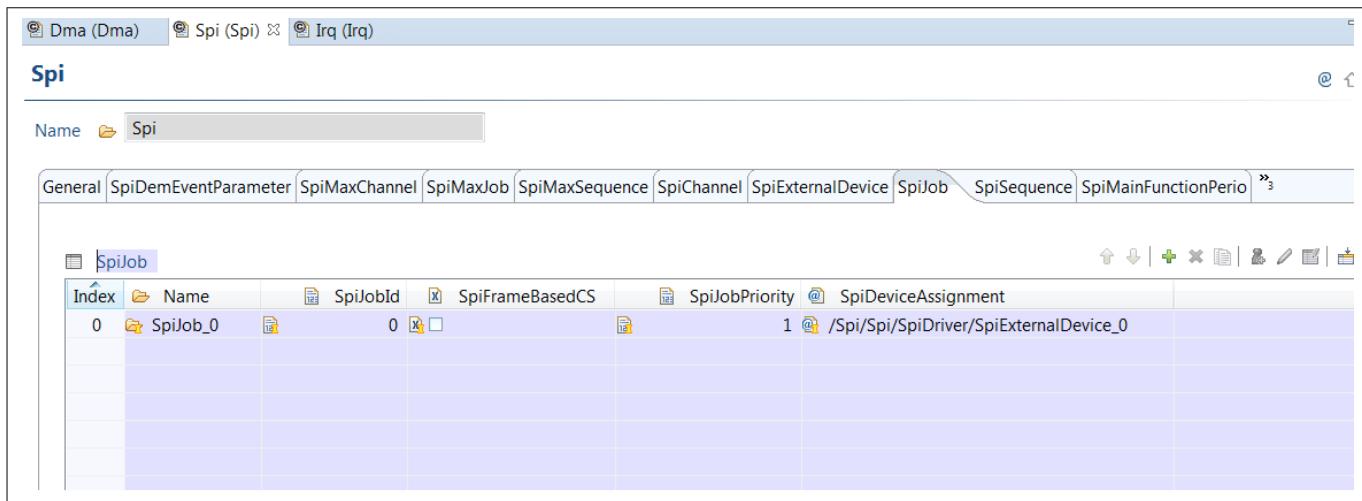


Figure 192 Choose the external device added in the configuration

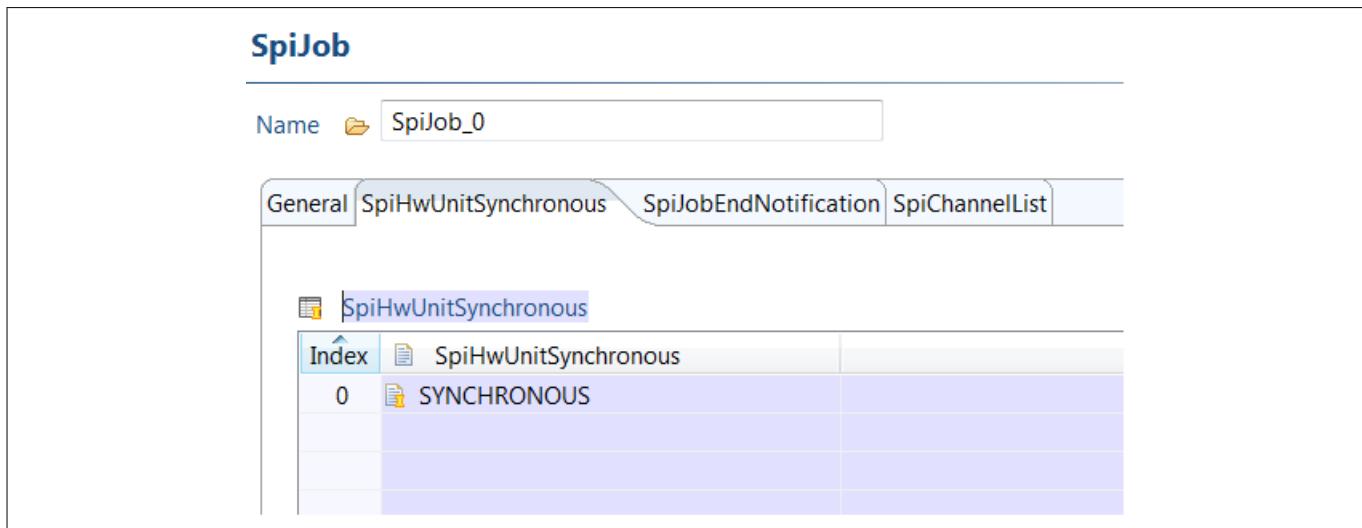


Figure 193 Choose job to be synchronous

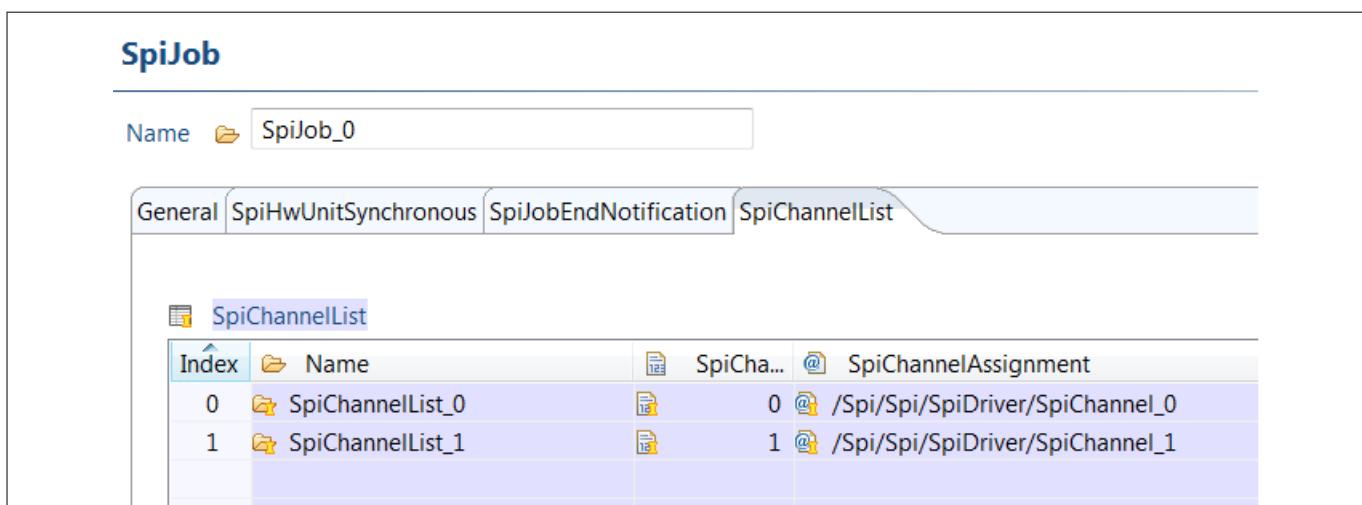
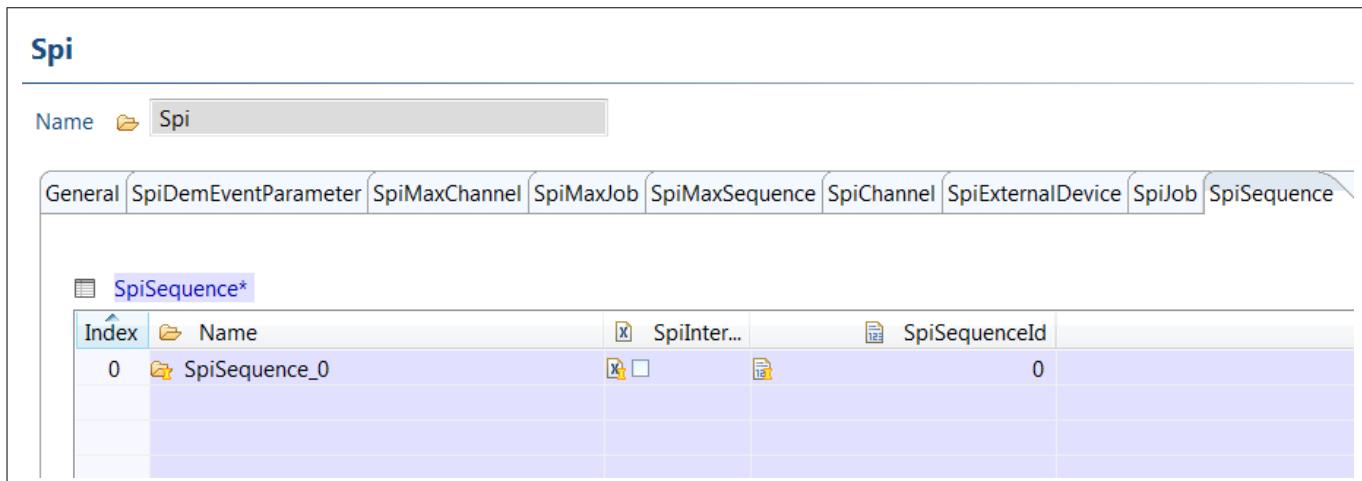
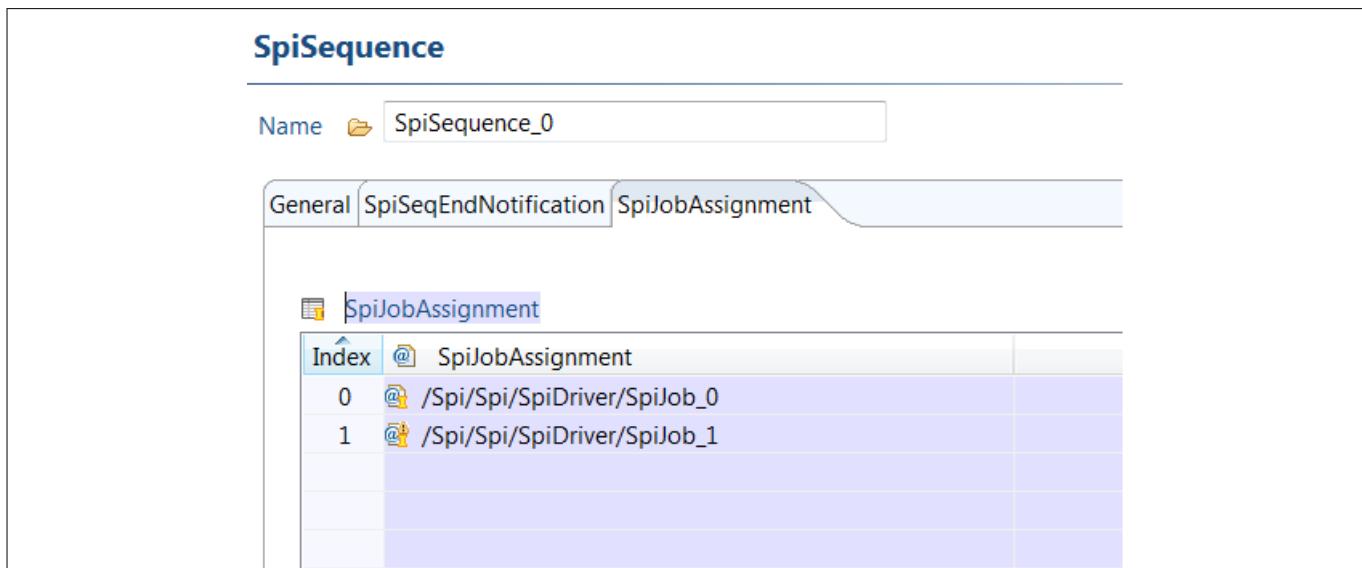
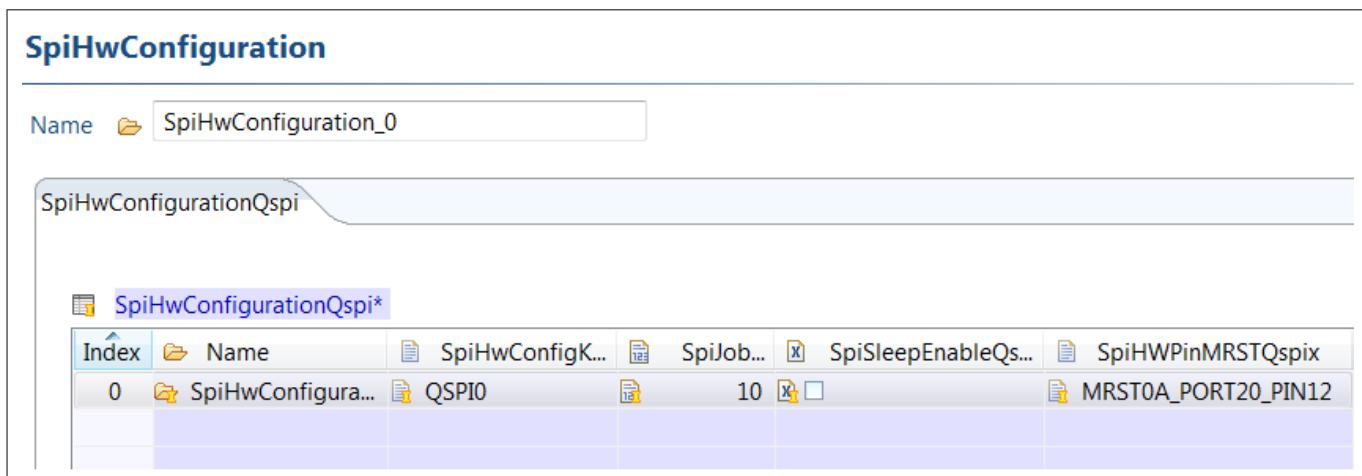


Figure 194 Choose the channels to be added for job

SPI driver

Figure 195 Add a sequence with jobs assigned

Figure 196 Configure jobs in Sequence

Figure 197 Configure QSPI Hardware
Sample code snippets

SPI driver

Setting up internal buffer for IB channels - synchronous transmission

The steps for setting up the IB are as follows:

1. Configure the source buffer to be transmitted via the API Spi_WriteIB.
2. After the buffers are setup for IB channel, transmit API should be invoked.
3. After transmission is completed, the received data should be read back from the internal buffer via the API Spi_ReadIB

```

/* Source buffers */
uint32 Spi_SrcBuf0[BUFFER_LENGTH] = {
    0x11111111,
    0x22222222,
    0xAAAAAAA,
    0x55555555};

uint32 Spi_SrcBuf1[BUFFER_LENGTH] = {
    0x22222222,
    0x11111111,
    0x77777777,
    0xAAAAAAA};

/* Destination buffers */
uint32 Spi_DestBuf0[BUFFER_LENGTH];
uint32 Spi_DestBuf1[BUFFER_LENGTH];

/* Initialize source buffers */
Spi_WriteIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_0,
(Spi_DataBufferType*)Spi_SrcBuf0);
Spi_WriteIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_1,
(Spi_DataBufferType*)Spi_SrcBuf1);

/* Transmit data */
u8 returnValue = Spi_SyncTransmit(SpiConf_SpiSequence_SpiSequence_0);

/* Read the received data from IB */
Spi_ReadIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_0,
(Spi_DataBufferType*)Spi_DestBuf0);
Spi_ReadIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_1,
(Spi_DataBufferType*)Spi_DestBuf1);

```

SPI driver**Setting up Internal Buffer for IB Channels - Asynchronous transmission**

```
/* Source buffers */
uint32 Spi_SrcBuf0[BUFFER_LENGTH] = {
    0x11111111,
    0x22222222,
    0xAAAAAAA,
    0x55555555};

uint32 Spi_SrcBuf1[BUFFER_LENGTH] = {
    0x22222222,
    0x11111111,
    0x77777777,
    0xAAAAAAA};

/* Destination buffer */
uint32 Spi_DestBuf0[BUFFER_LENGTH];
uint32 Spi_DestBuf1[BUFFER_LENGTH];

/* Write data to IB buffer */
Spi_WriteIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_0,
(Spi_DataBufferType*)Spi_SrcBuf0);
Spi_WriteIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_1,
(Spi_DataBufferType*)Spi_SrcBuf1);

/* Start data transmission */
u8 returnValue = Spi_AsyncTransmit(SpiConf_SpiSequence_SpiSequence_0);

/* Wait till the transmission is complete */
while(Spi_GetStatus() == SPI_BUSY);

/* Read the received data from IB buffer */
Spi_ReadIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_0,
(Spi_DataBufferType*)Spi_DestBuf0);
Spi_ReadIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_1,
(Spi_DataBufferType*)Spi_DestBuf1);
```

SPI driver

Polling transmission for asynchronous transmission

```

/* In Level-2, set the asynchronous transmission mode to interrupt (1) /
polling (0) */
Spi_SetAsyncMode(0);

/* Write data to IB buffer */
Spi_WriteIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_0,
(Spi_DataBufferType*)Spi_SrcBuf0);
Spi_WriteIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_1,
(Spi_DataBufferType*)Spi_SrcBuf1);

/* start data transmission */
u8returnValue = Spi_AsyncTransmit(SpiConf_SpiSequence_SpiSequence_0);

/* poll till the transmission completes */
while(Spi_GetStatus() == SPI_BUSY)
{
    Spi_MainFunction_Handling();
}

/* Read data from IB buffer */
Spi_ReadIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_0,
(Spi_DataBufferType*)Spi_DestBuf0);
Spi_ReadIB((Spi_ChannelType)SpiConf_SpiChannel_SpiChannel_1,
(Spi_DataBufferType*)Spi_DestBuf1);

```

21.1.5 Key architectural considerations

21.1.5.1 Transmission modes supported

The following transmission modes are supported.

Level 0: Supports only synchronous transmission

In this mode, data to be transmitted is directly copied to TX FIFO. Data is transmitted in the order defined in the configuration. Note that this functionality is a blocking call, that is, until the transmission completes or error occurs, the API will not return the status.

Level 1: Supports only asynchronous transmission

In this mode, the respective interrupts are configured and DMA is configured in Linklist mode for data transfer. The following interrupts are configured in Level 1 mode: TXF, RXF, PT2 and Error. The DMA driver invokes the callback function registered by the QSPI module on completion of channel transmission. Note that all the jobs of sequences that can be interrupted will be placed in priority order if interruptible sequence feature is enabled. Note that for asynchronous transmission each kernel is allocated with an independent queue to handle jobs as per priority.

Level 2: Supports both synchronous and asynchronous transmission. Asynchronous transmission is further supported with either of Interrupt and polling mode. In the polling mode, DMA is still used for TX and RX transfers in interrupt mode, however the RX complete event is polled to check if channel transmission is complete to trigger the start of next channel transmission. Error and PT2 flags are polled to indicate if the frame is complete. Interrupt mode in Level 2 is same as Level 1 implementation. Note that In level-2 mode either synchronous or asynchronous sequences can be configured for transmission.

SPI driver

21.1.5.2 General configuration

Decision on configuration of hardware

FIFO configuration

- TXFIFO / RXFIFO are configured for the Single move mode
- Move counter mode enabled with contentious move mode for transmission of data

Asynchronous communication (L1 and L2 mode)

By using the Move counter, SLSO de-assertion is handled by the hardware when the MCCOUNT reaches to "0".

The following interrupts/callbacks are configured for QSPI for Asynchronous mode

- TXF - Handled by DMA - Transmit FIFO interrupt - Request for feeding FIFO
- RXF - Handled by DMA - Receive FIFO interrupt - Request for emptying the FIFO
- ERROR - Handled by CPU - On QSPI Hardware error
- DMA Callback - Handled by CPU - Occurs on completion of channel data transmission
- DMA Error Callback - Handled by CPU - On DMA transfer errors (Move engine errors).
- PT2 interrupt - Handled by CPU - Occurs on complete transmission of job. Note that PT2 interrupt is needed to know the end of frame and counter can be loaded only when the IP is in the IDLE state

DMA usage / configuration

- Do note that two dedicated DMA channels should be assigned /QSPI each for TX and RX
- DMA is configured for asynchronous communication only and is configured to one DMA move (that is, one DMA transfer has one DMA move). DMA is configured to work in DMA linklist mode and two list are maintained for transmission and reception accordingly
- TCS memory for DMA linklist is allocated in the QSPI module and shall be passed to DMA by configuring suitably based on channel configuration by SPI module
- Any errors during the DMA transfer from move engine will be handled by the DMA module and should callback the error handler in the respective core to which kernel is assigned
- In order to achieve asynchronous communication priority Queue is implemented which is applicable for asynchronous communication only. Note that the jobs in Queue are maintained in priority order and each kernel is assigned with a Queue to maintain job id and its properties

Synchronous communication (L0 mode)

- TX FIFO is directly fed with data without using the DMA.
- In Synchronous communication, transfer of data is done in blocking call, and the transmit function waits for data to be received or till the timeout occurs.
- No Queuing mechanism used for synchronous communication.

21.1.5.3 Multicore decision

TC3xx is designed to have maximum six instances of QSPI and this varies based on the device variant. Each instance is defined to be a kernel and same is used through the document to represent single instance of QSPI IP. A kernel can be assigned to one of the cores and cannot be shared between cores. Multiple kernels can be assigned to a core.

All cores will work independent of each other, so configuration of master core is not applicable for this driver. However, if kernel is configured and not assigned to any core then configuration would be generated for master core. Spi_Init and Spi_Deinit can be called by any core and same will not affect the operation on other cores except for the one being called.

All APIs will be able to access the information configured for the local core only. For example, Spi_GetJobResult and Spi_GetSequenceResult APIs can return status for the jobs and sequence assigned

SPI driver

to same core only, APIs cannot return results for the sequence assigned to different cores respective DETs will be raised if cross-core information is requested.

The `Spi_GetStatus` API will return the status local core. For example, if two kernels are assigned to core 1, if communication is in progress for core 2, and if `Spi_GetStatus` is called on core 1, IDLE will be returned if no communication on core 1. If the `Spi_GetStatus` is called on core 2, BUSY will be returned.

21.1.5.4 Sequence, jobs and channels

Jobs cannot be shared between different sequences across different kernels within a sequence all the jobs should belong to the same hardware kernel.

Channels can be shared between two jobs within the same core. However protecting the content of the channel is the responsibility of the application code.

Maximum of 8190 elements can be transmitted for a job. This is the limitation of the counter that has been used by QSPI. However this limitation does not apply for Frame-based CS logic. For frame based CS, maximum elements of 16383 elements can be transmitted in a channel. For synchronous transfer, maximum of 65534 elements can be transmitted.

21.1.5.5 Lookup tables

Lookup tables are added to expedite the access in configuration structures. Lookup tables are added for Sequence Ids, Job ids and Channel ids. Ids map the physical index to application Ids (Index in Lookup) in core configuration.

In the multicore environment since the configuration is spread across cores, accessing specific information for sequence, job and channel is time consuming since the applications ids are different from the physical location in core configuration. These tables are generated during code generation and these tables are placed in a core Flash accessible by all the cores.

21.1.5.6 Interruptible sequence behavior

When SPI_INTERRUPTIBLE_SEQ_ALLOWED is ON

If the incoming sequence is interruptible, the individual jobs of a sequence are arranged in queue as per the priority of the job.

Note: *The order of placing the jobs in queue cannot guarantee that all the jobs of a sequence are placed in consecutive locations.*

If the incoming sequence is non-interruptible, Instead of individual jobs, the entire sequence is placed in queue as per the priority of the first job in sequence.

Note: *The order of placing the jobs in queue guarantees that all the jobs of a sequence are placed in consecutive locations.*

When SPI_INTERRUPTIBLE_SEQ_ALLOWED is OFF

All the incoming sequences are considered as non-interruptible and entire sequence is placed in queue as per the priority of first job in sequence.

Note: *The order of placing the jobs in queue guarantees that all the jobs of a sequence are placed in consecutive locations.*

SPI driver

21.1.5.7 External demultiplexer feature

If a QSPI is configured to operate in external demultiplexer mode, the SLSO1 to SLSO4 are driven with the value configured for configuration parameter `SpiCsIdentifier` of external device. To ensure glitch free selection, a strobe signal SLSO0 is provided and the value of the strobe delay is configured using configuration parameter `SpiSLSO0StrobeDelay`. The polarity for all these SLSO lines(SLSO0..SLSO4) are configured with the same value given for configuration parameter `SpiCsPolarity` of external device.

At any given point of time, a QSPI can be operated either in external demultiplexer mode or normal mode.

SPI driver

21.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **SPI API sequence**

Integrator shall make sure the following sequence of APIs are followed before calling the Spi_Init API: L0 Mode:
 Mcu_Init(&Mcu_Config); Mcu_InitClock(0U); Mcu_DistributePllClock(); Port_Init(&Port_Config);
 Spi_Init(&Spi_Config); L1, L2 mode: Mcu_Init(&Mcu_Config); Mcu_InitClock(0U);
 Mcu_DistributePllClock(); IrqDma_Init(); IrqSpi_Init(); Dma_Init(&Dma_Config);
 Port_Init(&Port_Config); Spi_Init(&Spi_Config);
 [cover parentID SPI={7C102304-D585-410a-9D19-2A54F6076E91}]

- **Spi_InitCheck**

Integrator shall make sure that all APIs of the SPI driver except for Spi_init, Spi_GetversionInfo API shall be called only after successful execution of the Spi_InitCheck API. SFRs related to other modules getting modified cannot be verified. This should be checked by dependent modules InitCheck APIs. Rationale: Checkinit API shall ensure all the pointers used for cores and kernels are properly initialized and state of driver is in known state before calling any API.

[cover parentID SPI={6073DDFE-0B7F-43e9-95C4-197677F5BC53}]

- **Global configuration pointer**

Global configuration pointer passed for the Spi_init API should be same across all the cores. Rationale: If other pointer is passed configuration is corrupted and behavior is unpredictable.

[cover parentID SPI={4603B79C-742E-464e-9824-B8C498554055}]

- **SchM implementation**

All Schm_x function calls are non-productive functions and are to be implemented from the application developer. Core-related interrupts are expected to be disabled between the entry and exit of the Schm calls. Following listed Schm functions are implemented in the SPI module: a. SchM_Enter_Spi_Queue_Update() / SchM_Exit_Spi_Queue_Update() b. SchM_Enter_Spi_SyncLock() / SchM_Exit_Spi_SyncLock() c. SchM_Enter_Spi_ChannelLock / SchM_Exit_Spi_ChannelLock() Rationale: Schm calls are made to protect the global variables shared across interrupt / different API context.

[cover parentID SPI={C087EEC6-8339-403c-A251-FB28C8CB6F9B}]

- **DMA Rx notification**

The SPI is dependent on DMA to do transfer of data. The SPI driver does not perform interrupt source check, it is the responsibility of the dependent module like DMA shall perform the interrupt source check before calling the respective channel notification / handlers. Rationale: Registers of dependent module are accessed only by module driver only.

[cover parentID SPI={84215E23-909B-46ae-A499-07FA622AE7FA}]

- **DMA resource allocation to SPI module**

The integrator shall ensure that two dedicated DMA channels are to be allocated for TX and RX of QSPI and these channels cannot be shared with any peripherals or changed dynamically. Rationale: DMA channels cannot be shared once allocated to QSPI.

[cover parentID SPI={43520DE0-4A16-427b-84AF-13A616D8B977}]

- **Trap handler**

The application developer should implement the trap handlers to handle the trap accordingly. Rationale: If CLC is not enabled and registers are updated, trap will be invoked. Note that this is a generic AOU and is applicable for all MCAL drivers.

[cover parentID SPI={3EE40B50-5F01-4fb7-9221-D75F1120E86C}]

- **Protection of global variables**

The integrator shall ensure that there is no interference to MCAL from other modules. Rationale : Variables can be corrupted by the QM software. Note that this is a generic AOU and is applicable for all MCAL drivers.

SPI driver

[cover parentID SPI={2417A341-9FDE-4a17-86ED-068992DAE09A}]

- **Watchdog triggering**

The integration should make sure that watchdog is enabled and is triggered in case of no response or driver is stuck in a busy state. Rationale : Due to hardware faults, interrupts may not get triggered and driver may be in BUSY state.

[cover parentID SPI={041495D5-3E7A-4708-8608-D817B5F51EC0}]

SPI driver

21.3 Reference information

21.3.1 Configuration interfaces

SPI driver

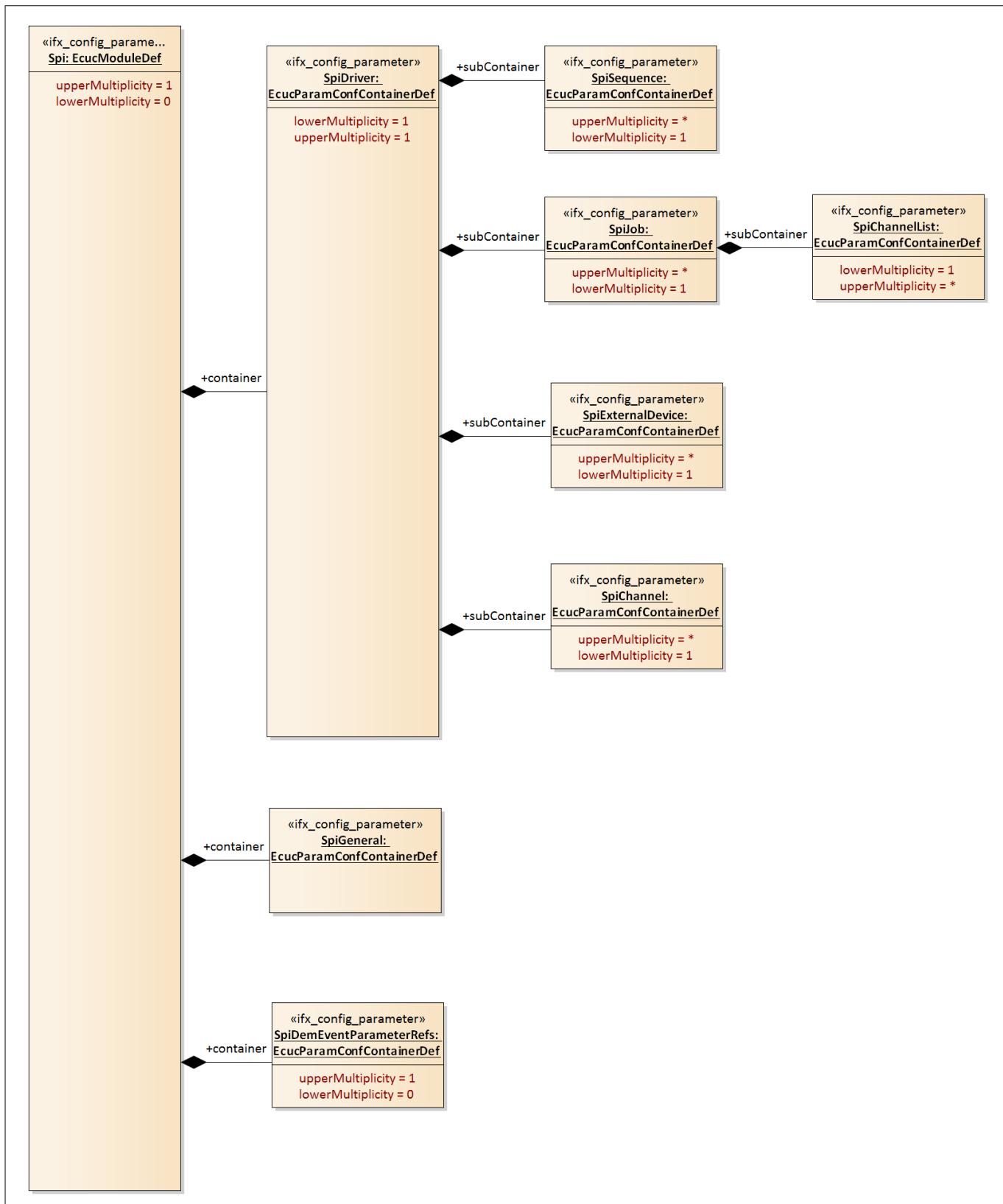


Figure 198 Container hierarchy along with their configuration parameters

21.3.1.1 Container: CommonPublishedInformation

Container holding all SPI specific published information parameters

SPI driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

21.3.1.1.1 ArMajorVersion**Table 1748 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Module Major version - 4		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.1.2 ArMinorVersion**Table 1749 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	Module Minor version - 2		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.1.3 ArPatchVersion**Table 1750 Specification for ArPatchVersion**

Name	ArPatchVersion
Description	AUTOSAR Revision version

SPI driver**Table 1750 Specification for ArPatchVersion (continued)**

Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.1.4 Module ID**Table 1751 Specification for Module ID**

Name	Module ID		
Description	Module id of SPI - 83		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	83		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.1.5 Release**Table 1752 Specification for Release**

Name	Release		
Description	This parameter indicates the TC3xx device derivative used for the implementation.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per Hardware unit configured		
Post-build variant value	FALSE	Post-build variant multiplicity	-

SPI driver**Table 1752 Specification for Release (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.1.6 SwMajorVersion**Table 1753 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	Module Majorversion		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 255		
Default value	As per driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.1.7 SwMinorVersion**Table 1754 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	Module Minor version		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

SPI driver**21.3.1.1.8 SwPatchVersion****Table 1755 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Module Patch version		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.1.9 Vendor ID**Table 1756 Specification for Vendor ID**

Name	Vendor ID		
Description	IFX Vendor ID - 17		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.2 Container: Spi

Configuration of the Spi (Serial Peripheral Interface) module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

SPI driver**21.3.1.2.1 Config Variant****Table 1757 Specification for Config Variant**

Name	Config Variant		
Description			
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VariantPostBuild:		
Default value	VariantPostBuild		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.3 Container: SpiBaudrateParams

Container to hold the configuration elements required for configuring the right baudrate.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

21.3.1.3.1 SpiBaudParamA**Table 1758 Specification for SpiBaudParamA**

Name	SpiBaudParamA		
Description	Bit Segment 1 Length expressed in quanta of Q b00 - 1 b01 - 2 b10 - 3 b11 - 4 Applicable only when the parameter SpiAutoCalcBaudParams is set to false. Note that this is the Hardware parameter for configuring the ECON register of QSPI and is applicable for configuring the right baudrate for QSPI communication. Default value 1, is chosen to select the wide range of baudrate. Note: Refer to TS document (AURIXTC3XX_ts), chapter QSPI, Details of baudrate and phase duration control		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 3		

SPI driver**Table 1758 Specification for SpiBaudParamA (continued)**

Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcBaudParams		

21.3.1.3.2 SpiBaudParamB**Table 1759 Specification for SpiBaudParamB**

Name	SpiBaudParamB		
Description	Bit Segment 2 Length expressed in quanta of Q b00 - 0 b01 - 1 b10 - 2 b11 - 3 Applicable only when the parameter SpiAutoCalcBaudParams is set to false. Note that this is the Hardware parameter for configuring the ECON register of QSPI and is applicable for configuring the right baudrate for QSPI communication. Default value 0, is chosen to select the wide range of baudrate. Note: Refer to TS document (AURIXTC3XX_ts), chapter QSPI; Details of baudrate and phase duration control		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 3		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcBaudParams		

SPI driver**21.3.1.3.3 SpiBaudParamC****Table 1760 Specification for SpiBaudParamC**

Name	SpiBaudParamC		
Description	Bit Segment 3 Length expressed in quanta of Q b00 - 0 (if B=0, than C is minimum 1 per hardware) b01 - 1 b10 - 2 b11 - 3 Note: If SpiBaudParamB = 0, then SpiBaudParamC should have minimum of value 1 Applicable only when the parameter SpiAutoCalcBaudParams is set to false. Note that this is the Hardware parameter for configuring the ECON register of QSPI and is applicable for configuring the right baudrate for QSPI communication. Default value 1, is chosen to select the wide range of baudrate. Note: Refer to TS document (AURIXTC3XX_ts), chapter QSPI; Details of baudrate and phase duration control		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 3		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcBaudParams		

21.3.1.3.4 SpiBaudParamQ**Table 1761 Specification for SpiBaudParamQ**

Name	SpiBaudParamQ		
Description	This defines the time quantum length used by A, B, and C to define the baud rate and duty cycle b000000 - 1 b000001 - 2 b111111 - 64 Applicable only when the parameter SpiAutoCalcBaudParams is set to false.		

SPI driver**Table 1761 Specification for SpiBaudParamQ (continued)**

	<p>Note that this is the Hardware parameter for configuring the ECON register of QSPI and is applicable for configuring the right baudrate for QSPI communication. Default value 10, is chosen to select the wide range of baudrate.</p> <p>Note: Refer to TS document (AURIXTC3XX_ts), chapter QSPI; Details of baudrate and phase duration control</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 63		
Default value	10		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcBaudParams		

21.3.1.3.5 SpiBaudParamTQ**Table 1762 Specification for SpiBaudParamTQ**

Name	SpiBaudParamTQ		
Description	<p>Global Time Quantum Length</p> <p>Common n-divider scaling the baud rates of all channels in direction of higher or lower baud rates.</p> <p>0 - division by 1 1 - division by 2 ... 255 - division by 256</p> <p>Applicable only when the parameter SpiAutoCalcBaudParams is set to false.</p> <p>Note that this is the Hardware parameter for configuring the ECON register of QSPI and is applicable for configuring the right baudrate for QSPI communication.</p> <p>Note: Refer to TS document (AURIXTC3XX_ts), chapter QSPI; Details of baudrate and phase duration control</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	TRUE	Post-build variant multiplicity	-

SPI driver**Table 1762 Specification for SpiBaudParamTQ (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcBaudParams		

21.3.1.4 Container: SpiChannel

This container contains the configuration parameters to describe a channel.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

21.3.1.4.1 SpiChannelId**Table 1763 Specification for SpiChannelId**

Name	SpiChannelId		
Description	<p>This is ID number assigned to SPI channel.</p> <p>By default SpiChannelId is set to 0, however this number is auto incremented on adding successive channels.</p> <p>Note:</p> <ol style="list-style-type: none"> 1) Due to Multi-core implementation physical and logical channel ids are to be generated. Logical channel ids are numbered sequentially, however in the configuration, physical memory location can be different due to assignment of channels to jobs of different cores. 2) Application should always use the same number generated as per the pbcfg.h file for accessing channel buffers. ex: SpiConf_SpiChannel_(x). x is derived from the name provided by the user in the configuration. 3) ID - 0xFF (255) value is used as delimiter to indicate the end of channel-id list. 		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

SPI driver**21.3.1.4.2 SpiChannelType****Table 1764 Specification for SpiChannelType**

Name	SpiChannelType		
Description	<p>This parameter specifies the buffer type (External Buffer / Internal Buffer) used by the channel.</p> <p>IB - Channel of internal buffer type EB - Channel of external buffer type</p> <p>By Default buffer type is set to EB since most applications prefer using EB over IB.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>EB: External Buffer IB: Internal Buffer</p>		
Default value	EB		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiChannelBuffersAllowed		

21.3.1.4.3 SpiDataWidth**Table 1765 Specification for SpiDataWidth**

Name	SpiDataWidth		
Description	<p>This parameter specifies the width of the data to be transmitted in terms of bits.</p> <p>Note: The QSPI supports the datawidth from 2 to 32 bits</p> <p>Default value for SpiDataWidth is set to 8 since the frames provided by any application would be defined in bytes (8-bits).</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	2 - 32		
Default value	8		
Post-build variant value		Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

SPI driver**21.3.1.4.4 SpiDefaultData****Table 1766 Specification for SpiDefaultData**

Name	SpiDefaultData		
Description	<p>Data to be transmitted if source buffer is defined to be NULL for Spi_SetupEB or Spi_WriteIB APIs.</p> <p>Defaultdata is set to '0' since this is application specific and needs to be defined as supported by external devices.</p>		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiDataWidth		

21.3.1.4.5 SpiEbMaxLength**Table 1767 Specification for SpiEbMaxLength**

Name	SpiEbMaxLength		
Description	<p>This parameter defines number of data to be transmitted for EB channel.</p> <p>Available only if (SpiChannelBuffersAllowed is 1 or 2) and (SpiChannelType is EB)</p> <p>By default SpiEBMaxLength is set to 8190 elements so that all communication modes are supported, L0, L1 and L2.</p> <p>Note:</p> <ol style="list-style-type: none"> 1) Since Move counter mode is used for transmission of the QSPI data, maximum of 8190 elements can be transmitted. i.e. for a job containing multiple channels sum of all elements in a job cannot be greater than 8190 elements. Application developer / integrator has to take care of checking if sum of all elements of EB does not exceed 8190 elements. 2) For jobs with FrameBasedCS enabled, maximum of 16383 elements can be transmitted per channel. 3) For Sync transfer maximum of 65535 elements can be transmitted (No DMA usage). 		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		
Default value	8191		
Post-build variant value	TRUE	Post-build variant multiplicity	-

SPI driver**Table 1767 Specification for SpiEbMaxLength (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiChannelType, SpiChannelBuffersAllowed		

21.3.1.4.6 SpilbN_buffers**Table 1768 Specification for SpilbN_buffers**

Name	SpiIbN_buffers		
Description	<p>Defines the size of channel buffer for IB channels.</p> <p>By Default size of IB buffer is chosen to be 10 elements as many data packets to SPI external devices can fit with-in this data length.</p> <p>Note on Channel width:</p> <ol style="list-style-type: none"> 1) less than 8 bit is considered as 8-bit channel, 8-bit to 15-bit (inclusive) is considered as 16-bit channel and above 16-bit is considered as 32-bit channel. So depending on data width, size of the buffer will be generated. 2) Available only if (SpiChannelBuffersAllowed is 0 or 2) and (SpiChannelType is IB) 3) Since Move counter mode is used for asynchronous transmission of the QSPI data, maximum of 8190 elements can be transmitted. i.e. for a job containing multiple channels, sum of all elements in a job cannot be greater than 8190 elements. These checks are performed on the IB type buffer at code generation time and checks are not done for EB buffer since EB buffer is allocated by application. Application developer / integrator has to take care of checking if sum of all elements of EB does not exceed 8190 elements. 4) For jobs with FrameBasedCS enabled, maximum of 16383 elements can be transmitted per channel. 5) For Sync transfer maximum of 65535 elements can be transmitted (No DMA usage). 		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 65535		
Default value	10		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiChannelType, SpiChannelBuffersAllowed		

SPI driver**21.3.1.4.7 SpiTransferStart****Table 1769 Specification for SpiTransferStart**

Name	SpiTransferStart		
Description	<p>This parameter specifies whether, Least Significant Bit (LSB) is transmitted first or Most Significant bit (MSB) transmitted first.</p> <p>By Default TransferStart is set to LSB, since most devices communicate over LSB frames.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>LSB: Transmission starts with the Least Significant Bit first</p> <p>MSB: Transmission starts with the Most Significant Bit first</p>		
Default value	LSB		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.5 Container: SpiChannelList

References to SPI channels and their order within the Job.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

21.3.1.5.1 SpiChannelAssignment**Table 1770 Specification for SpiChannelAssignment**

Name	SpiChannelAssignment		
Description	This parameter specifies the channel linked to this Job container.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: SpiChannel		
Default value	None		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

SPI driver

21.3.1.5.2 SpiChannelIndex

Table 1771 Specification for SpiChannelIndex

Name	SpiChannelIndex		
Description	This parameter specifies the order of Channels within the Job. Note that value 255 is used as delimiter to indicate that all the jobs assigned are completed.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.6 Container: SpiCsGpio

This container specifies the port pin which is used for Chip Select assertion/De-assertion. It is applicable if SpiCsSelection is CS_VIA_GPIO and has multiplicity of 0...1.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

21.3.1.6.1 SpiCsGpioPinSelection

Table 1772 Specification for SpiCsGpioPinSelection

Name	SpiCsGpioPinSelection		
Description	This parameter specifies the pin number of the port specified by SpiCsGpioPortSelection, used as Chip select, which is activated/de-activated by the SPI driver. Enabled only if SpiEnableCs is true and SpiCsSelection is CS_VIA_GPIO Please note that the range of port pins would vary across variants, refer to respective DS to check for the exact number of port-pins. A port can contain maximum of 15 pins. By Default the port-pin and port is set to 1, this has to be modified by application as per the Hardware mapping.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 15		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-

SPI driver**Table 1772 Specification for SpiCsGpioPinSelection (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiEnableCs, SpiCsSelection		

21.3.1.6.2 SpiCsGpioPortSelection**Table 1773 Specification for SpiCsGpioPortSelection**

Name	SpiCsGpioPortSelection		
Description	<p>This parameter specifies the port number whose one of the pin is used as Chip select, which is activated/de-activated by the SPI driver.</p> <p>Enabled only if SpiEnableCs is true and SpiCsSelection is CS_VIA_GPIO</p> <p>Note that the port number can vary from 0 to 41 depending on the device variant. Refer to respective DS to know the exact number of port available.</p> <p>By Default the port-pin and port is set to 1, this has to be modified by application as per the Hardware mapping.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 41		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiCsSelection, SpiEnableCs		

21.3.1.7 Container: SpiDelayParams

This container is Applicable only when the parameter SpiAutoCalcDelayParams is set to FALSE Defines the basic configuration for the current slave select. The parameters are vendor specific for use by SPI Hardware. Default values for this container is set to 1uSec delay.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Pre-Compile

21.3.1.7.1 SpiDelayParamIdleLength**Table 1774 Specification for SpiDelayParamIdleLength**

Name	SpiDelayParamIdleLength
-------------	-------------------------

SPI driver**Table 1774 Specification for SpiDelayParamIdleLength (continued)**

Description	Idle Delay Length Defines the length of both idle delays, IDLEA and IDLEB, in Tqspi units pre scaled with IPRE 0 represents 1 units 1 represents 2 unit ... 7 represents 8 units Applicable only when the parameter SpiAutoCalcDelayParams is set to false. Default value is set to obtain delay of 1uSec.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 7		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcDelayParams		

21.3.1.7.2 SpiDelayParamIdlePre**Table 1775 Specification for SpiDelayParamIdlePre**

Name	SpiDelayParamIdlePre		
Description	Pre-scalar for the Idle Delay Length in Tqspi units b000 represents 1 b001 represents 4 b010 represents 16 ... b111 represents 16384 Applicable only when the parameter SpiAutoCalcDelayParams is set to false. Default value is set to obtain delay of 1uSec.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

SPI driver**Table 1775 Specification for SpiDelayParamIdlePre (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcDelayParams		

21.3.1.7.3 SpiDelayParamLeadLength**Table 1776 Specification for SpiDelayParamLeadLength**

Name	SpiDelayParamLeadLength		
Description	Leading Delay Length Defines the length of the leading delay, in Tqspi units pre scaled with LPRE 0 - 1 units 1 - 2 unit ... 7 - 8 units Applicable only when the parameter SpiAutoCalcDelayParams is set to false. Default value is set to obtain delay of 1uSec.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 7		
Default value	7		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcDelayParams		

21.3.1.7.4 SpiDelayParamLeadPre**Table 1777 Specification for SpiDelayParamLeadPre**

Name	SpiDelayParamLeadPre
Description	Prescaler for the Leading Delay Length in Tqspi units b000 represents 1 b001 represents 4 b010 represents 16

SPI driver**Table 1777 Specification for SpiDelayParamLeadPre (continued)**

	... b111 represents 16384 Applicable only when the parameter SpiAutoCalcDelayParams is set to false. Default value is set to obtain delay of 1uSec.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcDelayParams		

21.3.1.7.5 SpiDelayParamTrailLength**Table 1778 Specification for SpiDelayParamTrailLength**

Name	SpiDelayParamTrailLength		
Description	Trailing Delay Length Defines the length of the trailing delay, in Tqspi units pre scaled with TPRE 0 - 1 units 1 - 2 unit ... 7 - 8 units Applicable only when the parameter SpiAutoCalcDelayParams is set to false. Default value is set to obtain delay of 1uSec.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 5		
Default value	5		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcDelayParams		

SPI driver**21.3.1.7.6 SpiDelayParamTrailPre****Table 1779 Specification for SpiDelayParamTrailPre**

Name	SpiDelayParamTrailPre		
Description	Prescaler for the Trailing Delay Length in Tqspi units b000 represents 1 b001 represents 4 b010 represents 16 ... b111 represents 16384 Applicable only when the parameter SpiAutoCalcDelayParams is set to false. Default value is set to obtain delay of 1uSec.		
Multiplicity	1..1		
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcDelayParams		

21.3.1.8 Container: SpiDemEventParameterRefs

Container for the references to DemEventParameter elements will be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameters DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

21.3.1.8.1 SPI_E_HARDWARE_ERROR**Table 1780 Specification for SPI_E_HARDWARE_ERROR**

Name	SPI_E_HARDWARE_ERROR		
Description	Reference to configured DEM event to report Hardware failure. If the reference is not configured the error will not be reported.		
Multiplicity	1..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node:		

SPI driver**Table 1780 Specification for SPI_E_HARDWARE_ERROR (continued)**

Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.9 Container: SpiDriver

This container contains the configuration parameters and sub containers of the AUTOSAR Spi module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

21.3.1.9.1 SpiMaxChannel**Table 1781 Specification for SpiMaxChannel**

Name	SpiMaxChannel		
Description	<p>This parameter represents number of channels allocated to all the cores. It will be gathered by tools during the configuration stage and hence not editable.</p> <p>This parameter is visible in Tresos only if the field is added after adding all channels, however code is generated for application use. By Default this value is set to '0' when no channels are added.</p>		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	1 - 255		
Default value	Depends on Max channels added		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.9.2 SpiMaxJob**Table 1782 Specification for SpiMaxJob**

Name	SpiMaxJob
-------------	-----------

SPI driver**Table 1782 Specification for SpiMaxJob (continued)**

Description	This parameter represents the total number of jobs allocated across cores. It will be gathered by tool during code generation stage and hence is not editable. This field is visible only if added after all jobs are added in system, however in code generation value is generated properly and available to application use. By default this value is set to '0' when no jobs are added.		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	1 - 5000		
Default value	Depends on total jobs added		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.9.3 SpiMaxSequence**Table 1783 Specification for SpiMaxSequence**

Name	SpiMaxSequence		
Description	This parameter represents the total number of sequence allocated across cores. It will be gathered by tool during configuration stage and hence is not editable. This parameter is visible in Tresos only if the field is added after adding all sequences in list, however value is generated in code generation for application use. By Default the value is set to '0' when no sequences are added.		
Multiplicity	0..1	Type	EcuIntegerParamDef
Range	1 - 255		
Default value	Depends on total sequences added		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

SPI driver**21.3.1.9.4 SpiSystemclock****Table 1784 Specification for SpiSystemclock**

Name	SpiSystemclock		
Description	This parameter refers to the system clock configured by MCU driver. It could refer to Fspb or Fqspi . This reference is used for BaudRate computation Reference to parameter of type McuClockSettingConf		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	None		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockSettingConfig		

21.3.1.10 Container: SpiExternalDevice

This container contains the configuration parameters to describe the external device (slave) properties. This container is attached/referenced to a job.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

21.3.1.10.1 SpiAutoCalcBaudParams**Table 1785 Specification for SpiAutoCalcBaudParams**

Name	SpiAutoCalcBaudParams		
Description	If the parameter is set to TRUE, then the configuration tool will automatically generate the Baudrate parameters (TQ, Q, A, B, C) based on the parameter SpiBaudrate. true: Automatically calculate the Baudrate parameters. false: Manually enter the Baudrate parameters by default value is set to true to calculate the register value for ECON register of QSPI to get the right baudrate as configured by the application.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		

SPI driver**Table 1785 Specification for SpiAutoCalcBaudParams (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.10.2 SpiAutoCalcDelayParams**Table 1786 Specification for SpiAutoCalcDelayParams**

Name	SpiAutoCalcDelayParams		
Description	<p>If the parameter is set to TRUE, then the configuration tool will automatically generate the delay parameters (IPRE, IDLE, LPRE, LEAD, TPRE, TRAIL) based on the parameters SpidleTime, SpiTimeClk2CS, SpiTrailingTime.</p> <p>true: Automatically calculate the delay parameters.</p> <p>false: Manually enter the Delay parameters</p> <p>By default value is set to true to calculate the delay parameters automatically during code generation for activation of chipselect.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.10.3 SpiBaudrate**Table 1787 Specification for SpiBaudrate**

Name	SpiBaudrate
Description	<p>This parameter defines the QSPI communication baudrate. This parameter allows using a range of values up to 50MHz.</p> <p>Enabled only when the parameter SpiAutoCalcBaudParams is set to TRUE.</p>

SPI driver**Table 1787 Specification for SpiBaudrate (continued)**

	<p>By default baudrate is set to 640k to support wide spread of devices, however this values are to be modified as per the application need and speed supported.</p> <p>Error:</p> <p>Provide configuration error if the baudrate value is incorrect for the given frequency(Fqspi)</p> <p>Note that the minimum baudrate is starting from 9600 Baud and maximum is 50MHz (chosen as per the Hardware limits) this is a deviation from AUTOSAR since the high limit is set to Infinity and low limit is set to 1Hz as per AUTOSAR.</p> <p>Note 1:</p> <p>(Max value is limited by the hardware 33MHz for Full-Duplex communication 50MHz for Simplex communication)</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	9600 - 50 MHz		
Default value	640000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiAutoCalcBaudParams		

21.3.1.10.4 SpiCsIdentifier**Table 1788 Specification for SpiCsIdentifier**

Name	SpiCsIdentifier		
Description	<p>This parameter specifies the Chip Select (CS) for the hardware specified by SpiHwUnit. Can range from Channel0 (0) to Channel15 (15).</p> <p>By Default SpiCsIdentifier is set to CHANNEL0.</p> <p>Note:</p> <p>This parameter is deviated from AUTOSAR in wherein, its type is Enumeration instead of String.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	CHANNEL0 - CHANNEL15		
Default value	CHANNEL0		
Post-build variant value	FALSE	Post-build variant multiplicity	-

SPI driver**Table 1788 Specification for SpiCsIdentifier (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiBaudrate		

21.3.1.10.5 SpiCsPolarity**Table 1789 Specification for SpiCsPolarity**

Name	SpiCsPolarity		
Description	This parameter defines the active polarity of chip select. By Default SpiCsPolarity is set to LOW to support wide variety of devices.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	HIGH: LOW:		
Default value	LOW		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiEnableCs		

21.3.1.10.6 SpiCsSelection**Table 1790 Specification for SpiCsSelection**

Name	SpiCsSelection		
Description	Indicates if the chip select is either through GPIO (Driven from the SPI driver) or SLSO (Driven by the QSPI Hardware). By Default SpiCsSelection is set to CS_VIA_PERIPHERAL_ENGINE so that using SLSO is the recommended mechanism to have better performance.		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	CS_VIA_GPIO: Chip select handled via GPIO by SPI driver. Note that if this option is selected for CS, timing between CS and data transfer cannot be guaranteed since the transfer is through DMA only or directly through Hardware FIFO.		

SPI driver**Table 1790 Specification for SpiCsSelection (continued)**

	CS_VIA_PERIPHERAL_ENGINE: Chip select is handled Peripheral Hardware engine.		
Default value	CS_VIA_PERIPHERAL_ENGINE		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiEnableCs		

21.3.1.10.7 SpiDataShiftEdge**Table 1791 Specification for SpiDataShiftEdge**

Name	SpiDataShiftEdge		
Description	<p>The data can be shifted on either leading edge or on trailing edge of the shift clock. This parameter defines the data shift with leading or trailing edge.</p> <p>Note that the default value is not defined for this element, since LEADING edge is used quite extensive in most applications, same has been configured as default value.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>LEADING: First data shift edge is a leading edge</p> <p>TRAILING: First data shift edge is a trailing edge</p>		
Default value	LEADING		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.10.8 SpiEnableCs**Table 1792 Specification for SpiEnableCs**

Name	SpiEnableCs		
Description	<p>This parameter specifies the chip select handling functions.</p> <p>If this parameter is enabled then parameter SpiCsSelection further details the type of chip selection.</p> <p>false - No Chip select is enabled</p>		

SPI driver**Table 1792 Specification for SpiEnableCs (continued)**

	true - Chip select is enabled By Default SpiEnableCs is set to true since chip select is used widely to communicate to SPI external device.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.10.9 SpiHwUnit**Table 1793 Specification for SpiHwUnit**

Name	SpiHwUnit		
Description	This parameter specifies the SPI hardware microcontroller peripheral allocated for transmission. QSPI0, QSPI1, QSPI2, QSPI3 and QSPI4 kernels can be configured. By Default value of this parameter is set to QSPI0. Note that as per AUTOSAR this parameter is defines as enum with names CSIBx, however to match the AURIX Hardware capability QSPIx is defined to replace CSIBx.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	QSPI0: QSPI Kernel 0 QSPI1: QSPI Kernel 1 QSPI2: QSPI Kernel 2 QSPI3: QSPI Kernel 3 QSPI4: QSPI Kernel 4 QSPI5: QSPI Kernel 5		
Default value	QSPI0		
Post-build variant value	FALSE	Post-build variant multiplicity	-

SPI driver**Table 1793 Specification for SpiHwUnit (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.10.10 SpindleTime**Table 1794 Specification for SpindleTime**

Name	SpiIdleTime		
Description	<p>This parameter IDLEA/IDLEB time is the QSPI hardware delay after which SLSO will be activated by Hardware.</p> <p>Default value for SpiTimeClk2CS is set to 1uSec which has been widely used from history.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.00000004 - 0.098304		
Default value	0.000001		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcDelayParams, SpiAutoCalcBaudParams		

21.3.1.10.11 SpiInternalLoopBackSupport**Table 1795 Specification for SpiInternalLoopBackSupport**

Name	SpiInternalLoopBackSupport		
Description	<p>Internal loop back between Tx and Rx. Shorts TX and RX internally to silicon without external connection to pins.</p> <p>By default loopbackmode is set to false since this mode is not widely used and is used only for debug purpose.</p> <p>Note: This parameter is used to configure the loopback mode as part of initialization. To control the loopback mode runtime Spi_ControlLoopBack can be used.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		

SPI driver**Table 1795 Specification for SpiInternalLoopBackSupport (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	
Dependency	-		

21.3.1.10.12 SpiParitySupport**Table 1796 Specification for SpiParitySupport**

Name	SpiParitySupport		
Description	This parameter indicates whether the parity feature to be enabled in Hardware or not. Default value for this parameter is set to UNUSED to support wide varieties of external devices.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	EVEN: Parity Bit added during Transmission to make the data as Even Parity ODD: Parity Bit is added during Transmission to make the data as Odd Parity UNUSED: Parity not configured		
Default value	UNUSED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.10.13 SpiShiftClockIdleLevel**Table 1797 Specification for SpiShiftClockIdleLevel**

Name	SpiShiftClockIdleLevel		
Description	This parameter defines the idle level of shift clock. The idle level of the shift clock can be configured to be idle level low or idle level high. By default SpiShiftClockIdlelevel is set to LOW to support wide range of application.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

SPI driver**Table 1797 Specification for SpiShiftClockIdleLevel (continued)**

Range	HIGH: Shift clock idle level is a high voltage level OR Slave device (SLSO / CS) is active if pin is held High LOW: Shift Clock idle level is low OR Slave device (SLSO / CS) is active if pin is held low		
Default value	LOW		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.10.14 SpiTimeClk2Cs**Table 1798 Specification for SpiTimeClk2Cs**

Name	SpiTimeClk2Cs		
Description	<p>This parameter is the minimum time (in seconds) between clock and chip select.</p> <p>This parameter is used to calculate the QSPI Hardware lead delay parameters LPRE and LEAD.</p> <p>Enabled only when the parameter SpiAutoCalcDelayParams is set to TRUE.</p> <p>Default value for SpiTimeClk2CS is set to 1uSec which has been widely used from history.</p> <p>Note that this value is set to default when the CS_VIA_GPIO is selected as the option since the timing cannot be guaranteed when GPIO is used as chip select.</p> <p>Min value of this parameter is derived through the capability of AURIX Hardware and derived based on the clock that can be supplied to peripheral, note that parameter values are deviated from AUTOSAR defined values.</p> <p>Error:</p> <p>Provide configuration error if the delay value is incorrect for the given frequency(Fqspi)</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.00000004 - 0.0001		
Default value	0.000001		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL

SPI driver**Table 1798 Specification for SpiTimeClk2Cs (continued)**

Dependency	SpiAutoCalcDelayParams, SpiAutoCalcBaudParams
-------------------	---

21.3.1.10.15 SpiTrailingTime**Table 1799 Specification for SpiTrailingTime**

Name	SpiTrailingTime		
Description	<p>Delay expected at the trailing phase of data transmission. This field would introduce delay at end of frame.</p> <p>Applicable only when the parameter SpiAutoCalcDelayParams is set to true. This parameter is IFX specific to make use of Hardware capability.</p> <p>Default value for SpiTimeClk2CS is set to 1uSec which has been widely used from history.</p> <p>Error:</p> <p>Provide configuration error if the delay value is incorrect for the given frequency(Fqspi)</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.00000004 - 0.098304		
Default value	0.000001		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiAutoCalcDelayParams, SpiAutoCalcBaudParams		

21.3.1.11 Container: SpiGeneral

General configuration settings for SPI-Handler

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

21.3.1.11.1 SpiCancelApi**Table 1800 Specification for SpiCancelApi**

Name	SpiCancelApi
Description	<p>This parameter specifies the availability of the API Spi_Cancel.</p> <p>This parameter is realized as,</p> <pre>#define SPI_CANCEL_API (STD_ON / STD_OFF)</pre> <p>TRUE - Spi_Cancel API is available</p> <p>FALSE - Spi_Cancel API is not available</p>

SPI driver**Table 1800 Specification for SpiCancelApi (continued)**

	By Default this feature is disabled, must be enabled if application demands to cancel a sequence at runtime.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.11.2 SpiChannelBuffersAllowed**Table 1801 Specification for SpiChannelBuffersAllowed**

Name	SpiChannelBuffersAllowed		
Description	<p>This parameter specifies the type of buffers available for the user</p> <p>This parameter is realized as,</p> <pre>#define SPI_CHANNEL_BUFFERS_ALLOWED (0U / 1U / 2U)</pre> <p>0 - Only internal buffers are selected in handler/driver, 1 - Only external buffers are selected in handler/driver, 2 - Both internal and external buffers are selected in handler/driver.</p> <p>By Default value of this parameter is set to 1 to support EB by default since widely used by application.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 2		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

SPI driver**21.3.1.11.3 SpiDevErrorDetect****Table 1802 Specification for SpiDevErrorDetect**

Name	SpiDevErrorDetect		
Description	<p>This parameter enables/disables development error detections.</p> <p>By Default this feature is enabled to support debug during integration with application code, must be disabled if successfully integrated to development environment.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.11.4 SpiHwStatusApi**Table 1803 Specification for SpiHwStatusApi**

Name	SpiHwStatusApi		
Description	<p>This parameter specifies whether API Spi_GetHWUnitStatus is available or not.</p> <p>This parameter is realized as,</p> <pre>#define SPI_HW_STATUS_API (STD_ON / STD_OFF)</pre> <p>By Default this feature is disabled, must be enabled if application demands to know individual status of hardware.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL

SPI driver**Table 1803 Specification for SpiHwStatusApi (continued)**

Dependency	-
-------------------	---

21.3.1.11.5 SpiInitCheckApi**Table 1804 Specification for SpiInitCheckApi**

Name	SpiInitCheckApi		
Description	Switches the Spi_InitCheck () API ON or OFF. By Default this feature is disabled, must be enabled if application demands safety features and needs to verify initialization sequence.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.11.6 SpiInitDeInitApiMode**Table 1805 Specification for SpiInitDeInitApiMode**

Name	SpiInitDeInitApiMode		
Description	This configuration parameter gives the mode in which Spi_Init and Spi_deinit API will be used. If this parameter is configured to SPI_MCAL_SUPERVISOR then Spi module directly writes to SFRs without using OS function. By Default mode is set to SPI_MCAL_SUPERVISOR since driver code executes in supervisor mode in most cases.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SPI_MCAL_SUPERVISOR: SPI_MCAL_USER1:		
Default value	SPI_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-

SPI driver**Table 1805 Specification for SpiInitDeInitApiMode (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.11.7 SpiInterruptibleSeqAllowed**Table 1806 Specification for SpiInterruptibleSeqAllowed**

Name	SpiInterruptibleSeqAllowed		
Description	<p>This parameter specifies whether interruptible sequences are allowed or not, if this field is STD_OFF and a sequence is defined to be interruptible, such sequence would be treated as non-interruptible.</p> <p>Significant only if SpiLevelDelivered is 1 or 2</p> <p>This parameter is realized as,</p> <pre>#define SPI_INTERRUPTIBLE_SEQ_ALLOWED (STD_ON / STD_OFF)</pre> <p>By Default this feature is disabled since having this feature will have slight performance impact due to sorting of queue involved at runtime and many applications do not need to have this feature enabled.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiLevelDelivered		

21.3.1.11.8 SpiLevelDelivered**Table 1807 Specification for SpiLevelDelivered**

Name	SpiLevelDelivered
Description	<p>This parameter is to select the type of communication driver has to support.</p> <p>L0 - Synchronous transmission (No DMA usage, blocking call).</p> <p>L1 - Asynchronous communication configuring related interrupts (DMA used, non-blocking).</p>

SPI driver**Table 1807 Specification for SpiLevelDelivered (continued)**

	L2 - Handles both Synchronous and Asynchronous communication. For asynchronous transmissions polling and Interrupt modes are supported. (DMA used for async sequences, non-blocking) This parameter is realized as, #define SPI_LEVEL_DELIVERED (0 / 1 / 2) By Default value of this parameter is set to 1 to Level-1 asynchronous communication, since widely used by application.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 2		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.11.9 SpiMainFunctionPeriod**Table 1808 Specification for SpiMainFunctionPeriod**

Name	SpiMainFunctionPeriod		
Description	This parameter defines the interval in which application has to call Spi_MainFunction_Handling. This function is used by the upper layer / application. Applicable only for Level-2 communication. Default value for polling is set to 10ms to support wide applications.		
Multiplicity	0..1	Type	EcucFloatParamDef
Range	0.0000001 Sec - 1 Sec		
Default value	0.01		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiLevelDelivered		

SPI driver**21.3.1.11.10 SpiMulticoreCheckEnable****Table 1809 Specification for SpiMulticoreCheckEnable**

Name	SpiMulticoreCheckEnable		
Description	<p>Switches ON / OFF the checks for multicore. By default multicore checks are enabled, if application is executing on single core this field can be disable leading to slight improvement in performance.</p> <p>By default this feature is enabled and must be disabled if drivers are successfully integrated to application environment.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.11.11 SpiRuntimeApiMode**Table 1810 Specification for SpiRuntimeApiMode**

Name	SpiRuntimeApiMode		
Description	<p>This configuration parameter gives the mode in which the Runtime API will be used. If this parameter is configured to SPI_MCAL_SUPERVISOR then Spi module directly writes to SFRs without using OS function.</p> <p>By Default mode is set to SPI_MCAL_SUPERVISOR since driver code executes in supervisor mode in most cases.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>SPI_MCAL_SUPERVISOR:</p> <p>SPI_MCAL_USER1:</p>		
Default value	SPI_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

SPI driver**Table 1810 Specification for SpiRuntimeApiMode (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.11.12 SpiSafetyCheckEnable**Table 1811 Specification for SpiSafetyCheckEnable**

Name	SpiSafetyCheckEnable		
Description	Enables / disables safety related checks. <code>#define SPI_SAFETY_CHECK_ENABLE (STD_ON / STD_OFF)</code> By default this feature is disabled and must be enabled in safety applications.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class		Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.11.13 SpiSupportConcurrentSyncTransmit**Table 1812 Specification for SpiSupportConcurrentSyncTransmit**

Name	SpiSupportConcurrentSyncTransmit		
Description	This parameter specifies whether concurrent synchronous transmission is allowed or not. i.e. if once of the instance of kernel is transmitting if request is for another kernel command should be accepted if this feature is enabled. Note that the value of this parameter is - False by default since the driver as a whole is a monolithic design as described by AUTOSAR. Available for SpiLevelDelivered 0 or 2. By Default this feature is disabled, must be enabled if application demands to transmit synchronous sequences on multiple QSPI at same time.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE		

SPI driver**Table 1812 Specification for SpiSupportConcurrentSyncTransmit (continued)**

	FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiLevelDelivered		

21.3.1.11.14 SpiSyncTransmittimeoutDuration**Table 1813 Specification for SpiSyncTransmittimeoutDuration**

Name	SpiSyncTransmittimeoutDuration		
Description	The parameter is used as timeout loop counter during synchronous transmission while waiting for data reception after transmission. Value is user configurable and can be changed as per the need of application. Timeout value is generated as part of root configuration accessible by all cores. Dependent on SpiLevelDelivered for L0 and L2.		
Multiplicity	1..1	Type	EcuclIntegerParamDef
Range	0x64 - 0xFFFFFFFF		
Default value	0xFFFF		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	SpiLevelDelivered		

21.3.1.11.15 SpiUserCallbackHeaderFile**Table 1814 Specification for SpiUserCallbackHeaderFile**

Name	SpiUserCallbackHeaderFile
Description	Header file name which will be included by the Spi. This parameter value must not represent a path. Note that the default value is only a representational value, this needs to be edited as per application need.

SPI driver**Table 1814 Specification for SpiUserCallbackHeaderFile (continued)**

Multiplicity	0..*	Type	EcucStringParamDef
Range	String		
Default value	Spi_UserDefined_Cbk.h		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiLevelDelivered		

21.3.1.11.16 SpiVersionInfoApi**Table 1815 Specification for SpiVersionInfoApi**

Name	SpiVersionInfoApi		
Description	Pre-processor switch to enable / disable the API to read out the driver version information The parameter is realized as, <code>#define SPI_VERSION_INFO_API (STD_ON / STD_OFF)</code> By Default this feature is disabled, must be enabled if application demands to know the version of driver at runtime.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.12 Container: SpiHwConfigurationQspix

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

SPI driver**21.3.1.12.1 SpiExternalDemux****Table 1816 Specification for SpiExternalDemux**

Name	SpiExternalDemux		
Description	<p>This parameter enables/disables the Spi External Demultiplexer.</p> <p>In the External Demultiplexer mode, SLSO1 to SLSO4 are used to drive the QSPI HW channel (0 to 15) and SLSO0 is used to as strobe signal in order to ensure glitch free selection.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiCsSelection, SpiEnableCs		

21.3.1.12.2 SpiHWPinMRSTQspix**Table 1817 Specification for SpiHWPinMRSTQspix**

Name	SpiHWPinMRSTQspix		
Description	<p>Port Pin selection for Master Receive Slave Transmit. This is used for the PISEL register MRIS field configuration.</p> <p>Refer DS for the list of pins applicable for specific QSPI, format of pin description is as below:</p> <p>QSPIx_MRSTy</p> <p>x - represents 0 to 6 based on the AURIX variant</p> <p>y - represents A, B, C, DN, DP, CN</p> <p>Respective Alt-x function to be selected from the configuration.</p> <p>This parameter is IFX specific to make use of Hardware provided capability for selecting the right MRST pins.</p>		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	Depends on Micro variant		
Post-build variant value	FALSE	Post-build variant multiplicity	-

SPI driver**Table 1817 Specification for SpiHWPinMRSTQspix (continued)**

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.12.3 SpiHwConfigKernel**Table 1818 Specification for SpiHwConfigKernel**

Name	SpiHwConfigKernel		
Description	This parameter is the symbolic name to identify the Kernel ID configuration This parameter is IFX specific to list all SPI kernels, by default set to QSPI0 needs to be modified as per the Hardware mapped.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	QSPI0: QSPI Kernel 0 QSPI1: QSPI Kernel 1 QSPI2: QSPI Kernel 2 QSPI3: QSPI Kernel 3 QSPI4: QSPI Kernel 4 QSPI5: QSPI Kernel 5		
Default value	QSPI0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.12.4 SpiJobQueueLengthQspix**Table 1819 Specification for SpiJobQueueLengthQspix**

Name	SpiJobQueueLengthQspix		
Description	This parameter specifies the maximum jobs that can be held in Queue for transmission at a time, once a job is transmitted location is freed-up for successive sequence. This Macro is generated for every QSPI kernel configured so that each kernel is independent and transmission happen parallel. Note:		

SPI driver**Table 1819 Specification for SpiJobQueueLengthQspix (continued)**

	1) Significant only for SpiLevelDelivered is 1 or 2 2) The SPI Job queue and the sequence queue are the circular queue. Ideally the queue size should be maximum jobs configured. The Syntax of the generated parameter for default value is as below: <pre>#define SPI_JOB_QUEUE_LENGTH_QSPIx (2U)</pre> 3) Note that if jobs are added for transmission and a cancel API is called on a sequence, if jobs are placed as per priority, jobs related to sequence is skipped and the location are not freed up in this special case.		
Multiplicity	0..1	Type	EcclIntegerParamDef
Range	0 - 65535		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	SpiLevelDelivered, SpiLevelDelivered		

21.3.1.12.5 SpiSLSO0StrobeDelay**Table 1820 Specification for SpiSLSO0StrobeDelay**

Name	SpiSLSO0StrobeDelay		
Description	This parameter is used to configure the strobe delay. In order to ensure glitch free selection, a strobe signal is provided, driven at SLSO0 pin. This signal is delayed relative to the SLSO1 to SLSO4 signals for LS (Lead_Strobe) and TS (Trail_Strobe) delays.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	2 - 31		
Default value	2		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiCsSelection, SpiEnableCs		

SPI driver**21.3.1.12.6 SpiSleepEnableQspix****Table 1821 Specification for SpiSleepEnableQspix**

Name	SpiSleepEnableQspix		
Description	Disable/enable entering into sleep mode upon sleep request from MCU. This parameter is used during spi initialization to configure the CLC register. true - QSPIx enters sleep mode upon sleep request from MCU. false - QSPIx does not enter sleep mode upon sleep request from MCU. This parameter is IFX specific to make use of Hardware provided capability. By default this feature is disabled and must be enabled if application demands.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SpiHwUnit		

21.3.1.13 Container: SpiHwDmaConfigurationQspix

Contains the references of DMA channels for QSPI Tx and Rx channels in configuration. This parameter is IFX specific to make use of Hardware provided capability to configure the associated DMA channels for SPI.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

21.3.1.13.1 SpiHwDmaChannelReceptionRef**Table 1822 Specification for SpiHwDmaChannelReceptionRef**

Name	SpiHwDmaChannelReceptionRef		
Description	This parameter refers to the DmaConfiguration. Channel is a reference field available through configured list of channels in DMA module. Available only if SpiLevelDelivered is 1 or 2		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: SpiHwDmaConfigurationQspix		
Default value	None		

SPI driver**Table 1822 Specification for SpiHwDmaChannelReceptionRef (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	SpiLevelDelivered		

21.3.1.13.2 SpiHwDmaChannelTransmissionRef**Table 1823 Specification for SpiHwDmaChannelTransmissionRef**

Name	SpiHwDmaChannelTransmissionRef		
Description	This parameter refers to the DmaConfiguration. Channel is a reference field available through configured list of channels in DMA module. Available only if SpiLevelDelivered is 1 or 2		
Multiplicity	0..1	Type	EcucReferenceDef
Range	Reference to Node: SpiHwDmaConfigurationQspix		
Default value	None		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	IFX	Scope	LOCAL
Dependency	SpiLevelDelivered		

21.3.1.14 Container: SpiJob

This container contains the configuration parameters to describe a job. A Job must contain at least one Channel. If a Job contains more than one Channel, all Channels contained have the same Job properties during transmission and will be linked together statically.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

21.3.1.14.1 SpiDeviceAssignment**Table 1824 Specification for SpiDeviceAssignment**

Name	SpiDeviceAssignment
Description	This parameter is a reference parameter to the external device container. Error:

SPI driver**Table 1824 Specification for SpiDeviceAssignment (continued)**

	Provide configuration error, if the same Device assigned for both Synchronous and Asynchronous Jobs.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: SpiExternalDevice		
Default value	None		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.14.2 SpiFrameBasedCS**Table 1825 Specification for SpiFrameBasedCS**

Name	SpiFrameBasedCS		
Description	<p>Frame based CS feature enabled / disables the assertion / de-assertion of SLSO for every element being transferred through SPI interface.</p> <p>This parameter is IFX specific to make use of Hardware provided capability. By Default this feature is disabled since this is application specific and not frequently used.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

21.3.1.14.3 SpiHwUnitSynchronous**Table 1826 Specification for SpiHwUnitSynchronous**

Name	SpiHwUnitSynchronous
-------------	----------------------

SPI driver**Table 1826 Specification for SpiHwUnitSynchronous (continued)**

Description	<p>If SpiHwUnitSynchronous is set to SYNCHRONOUS, then the job is used in a synchronous manner and vice versa.</p> <p>Since AUTOSAR requirement says to pre-assigned SPI buses required for sync transmission, this parameter is created to pre-assign the available SPI bus to Sync or Async.</p> <p>Applicable only if SpiLevelDelivered is 2</p> <p>By default value is set to ASYNCHRONOUS since most application use the L2 configuration and use asynchronous transfers.</p>		
Multiplicity	0..1	Type	EcucEnumerationParamDef
Range	ASYNCHRONOUS: SYNCHRONOUS:		
Default value	ASYNCHRONOUS		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiLevelDelivered		

21.3.1.14.4 SpiJobEndNotification**Table 1827 Specification for SpiJobEndNotification**

Name	SpiJobEndNotification		
Description	<p>This parameter defines the notification function name or function address. In case of function name, a function by this name must be defined in an application and will be called at end of job transmission.</p> <p>Significant if SpiLevelDelivered is 1 or 2.</p> <p>The SPI driver does not validate the configured function name or address for correctness and the responsibility falls on the user.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR	Post-build variant multiplicity	TRUE
Post-build variant value	TRUE	Multiplicity configuration class	Post-Build
Value configuration class	Post-Build		

SPI driver**Table 1827 Specification for SpiJobEndNotification (continued)**

Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiLevelDelivered		

21.3.1.14.5 SpiJobId**Table 1828 Specification for SpiJobId**

Name	SpiJobId		
Description	<p>This ID is assigned to job.</p> <p>Note:</p> <ol style="list-style-type: none"> 1) Due to Multi-core implementation physical and logical job ids are to be generated. Logical job ids are numbered sequentially, however in the configuration physical memory location can be different due to assignment of jobs to sequences of different cores. 2) Application should always use the same job id generated as per the pbcfg.h file for accessing job information. ex: SpiConf_SpiJobs_(x). x is derived from the name provided by the user in the configuration. <p>By Default value of ID is set to '0' however the value is auto-incremented if more than one job is added.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.14.6 SpiJobPriority**Table 1829 Specification for SpiJobPriority**

Name	SpiJobPriority		
Description	<p>This parameter defines the priority of a job.</p> <p>0 - lowest, 3 - highest priority</p> <p>By default priority of all jobs are set to '0' so that all jobs are scheduled as round-robin.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 3		
Default value	0		

SPI driver**Table 1829 Specification for SpiJobPriority (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.15 Container: SpiPublishedInformation

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

21.3.1.15.1 SpiMaxHwUnit**Table 1830 Specification for SpiMaxHwUnit**

Name	SpiMaxHwUnit		
Description	Total QSPI IP kernels available in the selected resource. This value would change based on the microcontroller variant.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 6		
Default value	Depends on the Hardware variant		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.16 Container: SpiSequence

This container contains the configuration parameters to describe a sequence. A sequence must contain at-least 1 job, if it contains more than one, all Jobs contained have the same Sequence properties during transmission and will be linked together statically.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

SPI driver**21.3.1.16.1 SpiInterruptibleSequence****Table 1831 Specification for SpiInterruptibleSequence**

Name	SpiInterruptibleSequence		
Description	<p>This parameter provides an option to suspend or not to suspend the sequence by another sequence.</p> <p>Significant only if SpiLevelDelivered is 1 or 2 and if SpiInterruptibleSequenceAllowed is true.</p> <p>true: Sequence could be suspended false: Sequence is not suspended</p> <p>By Default this feature is disabled so this field is set to false.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE FALSE</p>		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiInterruptibleSeqAllowed, SpiLevelDelivered		

21.3.1.16.2 SpiJobAssignment**Table 1832 Specification for SpiJobAssignment**

Name	SpiJobAssignment		
Description	<p>This parameter should reference to a list of jobs.</p> <p>Jobs have priorities assigned. Jobs linked in a Sequence must have decreasing priorities.</p> <p>That means the first Job must have the highest priority of all Jobs within the Sequence</p> <p>Error: A sequence cannot have Jobs on different QSPI Hardware modules otherwise configuration error is shown to rectify the same.</p> <p>Error: Provide configuration error if the jobs assigned are mapped to both Asynchronous and Synchronous bus.</p>		
Multiplicity	1..*	Type	EcucReferenceDef
Range	Reference to Node: SpiJob		
Default value	None		

SPI driver**Table 1832 Specification for SpiJobAssignment (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

21.3.1.16.3 SpiSeqEndNotification**Table 1833 Specification for SpiSeqEndNotification**

Name	SpiSeqEndNotification		
Description	<p>This parameter defines the notification function name or function address called after sequence transmission. A function by this type must be defined in an application and defined in Tresos.</p> <p>Significant only if SpiLevelDelivered is 1 OR 2</p> <p>The SPI driver does not validate the configured function name or address for correctness and the responsibility falls on the user.</p>		
Multiplicity	0..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	TRUE
Value configuration class	Post-Build	Multiplicity configuration class	Post-Build
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	SpiLevelDelivered		

21.3.1.16.4 SpiSequenceId**Table 1834 Specification for SpiSequenceId**

Name	SpiSequenceId
Description	<p>This ID is assigned to sequence.</p> <p>Default value is set to '0', however on adding multiple sequences id is automatically incremented.</p> <p>Note:</p>

SPI driver**Table 1834 Specification for SpiSequenceId (continued)**

	1) Due to Multi-core implementation physical and logical sequence ids are to be generated. Logical job ids are numbered sequentially, however in the configuration physical memory location can be different due to assignment of sequence to different cores / kernels. 2) Application should always use the same sequence id generated as per the pbcfg.h file for accessing sequence information. ex: SpiConf_SpiSequence_(x). x is derived from the name provided by the user in the configuration. 3) 0xFF (255) is used as delimiter, this value cannot be used for ID.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	ECU
Dependency	-		

21.3.2 Functions - Type definitions**21.3.2.1 Spi_AsyncModeType****Table 1835 Specification for Spi_AsyncModeType**

Syntax	Spi_AsyncModeType	
Type	Enumeration	
File	Spi.h	
Range	0 - SPI_POLLING_MODE	The asynchronous mechanism is ensured by polling, so interrupts related to the SPI busses handled asynchronously by CPU are disabled.
	1 - SPI_INTERRUPT_MODE	The asynchronous mechanism is ensured by interrupts, so interrupts related to the SPI busses handled asynchronously by CPU are enabled.
Description	This variables indicates if all the kernel's are executing in POLLING or INTERRUPT mode. This data type will be available or not according to the pre compile time parameter SPI_LEVEL_DELIVERED. This type is only relevant for LEVEL 2.	
Source	AUTOSAR	

SPI driver**21.3.2.2 Spi_ConfigType****Table 1836 Specification for Spi_ConfigType**

Syntax	Spi_ConfigType	
Type	Structure	
File	Spi.h	
Range	-[]	
Description	This structure holds the configuration of all the cores containing sequence, job, channel and Hardware unit information required to configure Hardware and transmit data over SPI interface.	
Source	AUTOSAR	

21.3.2.3 Spi_JobEndNotification**Table 1837 Specification for Spi_JobEndNotification**

Syntax	Spi_JobEndNotification	
Type	Pointer to a function of type void Function_Name (void)	
File	Spi.h	
Description	Callback routine type for each Job to notify the caller that a job has been finished.	
Source	AUTOSAR	

21.3.2.4 Spi_JobResultType**Table 1838 Specification for Spi_JobResultType**

Syntax	Spi_JobResultType	
Type	Enumeration	
File	Spi.h	
Range	0 - SPI_JOB_OK	The last transmission of the job has been finished successfully.
	1 - SPI_JOB_PENDING	The SPI Handler/Driver is performing (transmitting) a specific SPI job. The meaning of this status is equal to SPI_BUSY.
	2 - SPI_JOB_FAILED	The last transmission of the job has failed.
	3 - SPI_JOB_QUEUED	An asynchronous transmission of a Job has been accepted, while actual transmission for this Job has not started yet.

SPI driver**Table 1838 Specification for Spi_JobResultType (continued)**

Description	This type defines a range of specific Jobs status for SPI Handler/Driver. It informs about a SPI Handler/Driver job status and can be obtained calling the API service Spi_GetJobResult with the job ID.
Source	AUTOSAR

21.3.2.5 Spi_LoopBackType**Table 1839 Specification for Spi_LoopBackType**

Syntax	Spi_LoopBackType	
Type	Enumeration	
File	Spi.h	
Range	0 - SPI_LOOPBACK_DISABLE	Disables the Loopback mode
	1 - SPI_LOOPBACK_ENABLE	Enables the Loopback mode
Description	This type is used to enable/disable the loopback feature.	
Source	IFX	

21.3.2.6 Spi_SeqEndNotification**Table 1840 Specification for Spi_SeqEndNotification**

Syntax	Spi_SeqEndNotification	
Type	Pointer to a function of type void Function_Name (void)	
File	Spi.h	
Description	Callback routine type for each Sequence to notify the caller that a sequence has been finished.	
Source	AUTOSAR	

21.3.2.7 Spi_SeqResultType**Table 1841 Specification for Spi_SeqResultType**

Syntax	Spi_SeqResultType	
Type	Enumeration	
File	Spi.h	
Range	0 - SPI_SEQ_OK	The last transmission of the sequence has been finished successfully.
	1 - SPI_SEQ_PENDING	The SPI Handler/Driver is performing a specific SPI sequence. The meaning of this status is equal to SPI_BUSY.

SPI driver**Table 1841 Specification for Spi_SeqResultType (continued)**

	2 - SPI_SEQ_FAILED	The last transmission of the sequence has failed.
	3 - SPI_SEQ_CANCELLED	The last transmission of the sequence has been cancelled by user
Description	This type defines a range of specific sequences status for SPI Handler/Driver. It informs about a SPI Handler/Driver sequence status and can be obtained calling the API service Spi_GetSequenceResult with the sequence ID.	
Source	AUTOSAR	

21.3.2.8 Spi_StatusType**Table 1842 Specification for Spi_StatusType**

Syntax	Spi_StatusType	
Type	Enumeration	
File	Spi.h	
Range	0 - SPI_UNINIT 1 - SPI_IDLE 2 - SPI_BUSY	
Description	This type defines a range of specific status for SPI Handler/Driver. It informs about the SPI Handler/Driver status and can be obtained calling the API service Spi_GetStatus or the configurable Spi_GetHWUnitStatus.	
Source	IFX	

21.3.2.9 Spi_DataBufferType**Table 1843 Specification for Spi_DataBufferType**

Syntax	Spi_DataBufferType	
Type	uint8	
File	Spi.h	
Range	0-255	
Description	Type of application data buffer elements. Note: Channel width greater than 8, the SPI driver uses type uint16 to read or write the buffer. Similarly if channel width greater than 16, the SPI driver uses type uint32 to read or write the buffer. The SPI Driver will access the buffer as per little endian format (in accordance to architecture). Note: For 16-bit/32-bit transfer (Channel data width greater than 8) user data have to be word aligned.	

SPI driver**Table 1843 Specification for Spi_DataBufferType (continued)**

Source	AUTOSAR
---------------	---------

21.3.2.10 Spi_NumberOfDataType**Table 1844 Specification for Spi_NumberOfDataType**

Syntax	Spi_NumberOfDataType	
Type	uint16	
File	Spi.h	
Range	0-65535	
Description	Type for defining the number of data elements of the type Spi_DataBufferType to send and/or receive by channel. For Range details refer below parameters: SpiIBNBuffers SpiEBMaxLength	
Source	AUTOSAR	

21.3.2.11 Spi_ChannelType**Table 1845 Specification for Spi_ChannelType**

Syntax	Spi_ChannelType	
Type	Spi_ChannelType	
File	Spi.h	
Range	0-255	
Description	Specifies the identification (ID) for a Channel. Channel ID can be from 0-254. Value 255 is used as delimiter to identify if last channel is reached in the channel list. This is a deviation from AUTOSAR.	
Source	AUTOSAR	

21.3.2.12 Spi_JobType**Table 1846 Specification for Spi_JobType**

Syntax	Spi_JobType	
Type	uint16	
File	Spi.h	
Range	0-65535	
Description	Specifies the identification (ID) for a Job. Job ID can be from 0-65535. Value 65535 is used as delimiter to identify if last job in a job list. This is a deviation from AUTOSAR.	
Source	AUTOSAR	

SPI driver**21.3.2.13 Spi_SequenceType****Table 1847 Specification for Spi_SequenceType**

Syntax	Spi_SequenceType	
Type	Spi_SequenceType	
File	Spi.h	
Range	0-255	
Description	Specifies the identification (ID) for a sequence of jobs. Sequence ID can be from 0-254. Value 255 is used as delimiter to identify last sequence in sequence list. This is a deviation from AUTOSAR.	
Source	AUTOSAR	

21.3.2.14 Spi_HWUnitType**Table 1848 Specification for Spi_HWUnitType**

Syntax	Spi_HWUnitType	
Type	uint8	
File	Spi.h	
Range	0-5	
Description	Specifies the identification (ID) for a SPI Hardware microcontroller peripheral (unit). Range details: Module no (bit [0:3]) – QSPIx, where the range of x depends on the microcontroller derivative. Channel no (bit [4:7]) - Channel0-channel15	
Source	AUTOSAR	

21.3.3 Functions - APIs**21.3.3.1 Spi_AsyncTransmit****Table 1849 Specification for Spi_AsyncTransmit API**

Syntax	Std_ReturnType Spi_AsyncTransmit (const Spi_SequenceType Sequence)	
Service ID	0x03	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Sequence	Sequence ID.

SPI driver**Table 1849 Specification for Spi_AsyncTransmit API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Transmit request accepted E_NOT_OK: Transmit request not accepted
Description	<p>This API transmits the sequence asynchronously over the QSPI interface. This API is asynchronous, which means the application invoking the API is not blocked till the sequence is transmitted completely and completion of transmission would be notified (if configured).</p> <p>Is enabled only if SpiLevelDelivered is 1 or 2.</p> <p>Note:</p> <ol style="list-style-type: none"> 1) From multicore perspective, sequences assigned to core can only be transmitted else SPI_E_NOT_CONFIGURED DET will be returned. 2) Sequence ID to be used by application will be of format - SpiConf_SpiSequence_(x), x is user defined string for ex: SpiConf_SpiSequence_EEPROM_Write 	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>SPI_E_UNINIT: API service used without module initialization.</p> <p>SPI_E_PARAM_SEQ: API service called if the sequence ID is not in the range of the total sequence numbers allocated to all the cores.</p> <p>SPI_E_SEQ_PENDING: Indicates that sequence is in Pending state and requested action cannot be performed.</p> <p>SPI_E_NOT_CONFIGURED: Sequence / job / channel number passed is not configured to core on which request is made but with-in the max range of sequence.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SPI_E_HARDWARE_ERROR: On any error bit set in status register - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_FAILED).</p> <p>If no error is reported and successful transmission - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_PASSED)</p> <p>If any error is reported, application needs to monitor and take appropriate action.</p> <p>Note that SPI_E_HARDWARE_ERROR DEM is raised for QSPI Hardware errors or DMA ME errors.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	SpiLevelDelivered	
User hints	-	

SPI driver**21.3.3.2 Spi_Cancel****Table 1850 Specification for Spi_Cancel API**

Syntax	<pre>void Spi_Cancel (const Spi_SequenceType Sequence)</pre>	
Service ID	0x0C	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Conditional Reentrant - Reentrant for the core	
Parameters (in)	Sequence	Sequence ID.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This API cancels the on-going sequence transmission. Sets the sequence status to SPI_SEQ_CANCELLED and job status to SPI_JOB_OK. If any sequence notification is set, same will be called after completion of the ongoing job.</p> <p>In Multicore context, only the sequence assigned to core in which the API has been called can only be canceled else respective error will be returned.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>SPI_E_PARAM_SEQ: API service called if the sequence ID is not in the range of the total sequence numbers allocated to all the cores.</p> <p>SPI_E_UNINIT: API service used without module initialization.</p> <p>SPI_E_NOT_CONFIGURED: Sequence / job / channel number passed is not configured to core on which request is made but with-in the max range of sequence.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	SpiCancelApi	
User hints	-	

SPI driver**21.3.3.3 Spi_ControlLoopBack****Table 1851 Specification for Spi_ControlLoopBack API**

Syntax	<pre>Std_ReturnType Spi_ControlLoopBack (const Spi_HWUnitType HWUnit, const Spi_LoopBackType EnableOrDisable)</pre>	
Service ID	0x25	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant per QSPI HW	
Parameters (in)	HWUnit EnableOrDisable	Specifies the QSPI HW unit Specifies enable/disable the loopback mode
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK, if the loopback mode enable/disable is successful. E_NOT_OK, if the QSPI is not configured or is in a SPI_BUSY state or if the SPI driver is uninitialized.
Description	This API enables/disables the Loopback mode.	
Source	IFX	
Error handling	<p>DET:</p> <p>SPI_E_PARAM_UNIT: API service called with wrong parameter.</p> <p>SPI_E_UNINIT: API service used without module initialization.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>SPI_E_SAFETY_INVALID_PARAM: - If an invalid parameter is passed to Spi_SetAsyncMode API, the mode will be changed to the polling mode. As a safety check this DET is added. - If an invalid parameter is passed to ISR like wrong kernel which is not in range this safety DET will be triggered.</p> <p>- A safety check DET is reported when an invalid parameter is passed to the EnableOrDisable parameter of the Spi_ControlLoopBack API.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

SPI driver**21.3.3.4 Spi_DeInit****Table 1852 Specification for Spi_DeInit API**

Syntax	Std_ReturnType Spi_DeInit (void)	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : Switching to specified Mode is successful E_NOT_OK : Switching to specified mode was not successful
Description	<p>This API de-initializes the hardware and global variables related to SPI driver. The API must be called only after module initialization and is accepted to be processed only when the device is in IDLE state.</p> <p>In multicore context, this API can be called by any core, only the kernel information associated with the caller core will be de-initialized other core still continue to work.</p>	
Source	AUTOSAR	
Error handling	<p>DET: SPI_E_UNINIT: API service used without module initialization.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

21.3.3.5 Spi_GetHWUnitStatus**Table 1853 Specification for Spi_GetHWUnitStatus API**

Syntax	Spi_StatusType Spi_GetHWUnitStatus (
---------------	---

SPI driver**Table 1853 Specification for Spi_GetHWUnitStatus API (continued)**

	const Spi_HWUnitType HWUnit)	
Service ID	0x0B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	HWUnit	QSPI kernel Id
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Spi_StatusType	SPI_UNINIT - The QSPI Hardware is not initialized or not usable SPI_IDLE - The QSPI Hardware is not currently transmitting any Job SPI_BUSY - The QSPI Hardware is performing a SPI Job (transmit)
Description	<p>This API returns the status of the Hardware kernel requested for, If the QSPI kernel is busy transmitting a sequence SPI_BUSY is returned else SPI_IDLE is returned.</p> <p>In multicore context, API will be able to get the status of the kernels assigned to core on which the request is made, kernel assigned to different cores status cannot be obtained and returns un-predictable results.</p> <p>Note:</p> <ol style="list-style-type: none"> If Level - 0 Synchronous mode is configured - Irrespective of number of kernels assigned to core, a single status is maintained across multiple instance of kernels, so if any one kernel is busy transmitting the sequence status would be returned as BUSY. A monolithic approach is followed for synchronous kernels. If Level - 1 Asynchronous mode, each kernel assigned to core must have a status indicating the status of kernel. If any kernel is BUSY transmitting a sequence status would be returned BUSY. If level - 2 any kernel busy transmitting data would be returning busy status 	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>SPI_E_PARAM_UNIT: API service called with wrong parameter.</p> <p>SPI_E_UNINIT: API service used without module initialization.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	SpiHwStatusApi	

SPI driver**Table 1853 Specification for Spi_GetHWUnitStatus API (continued)**

User hints	-
-------------------	---

21.3.3.6 Spi_GetJobResult**Table 1854 Specification for Spi_GetJobResult API**

Syntax	<pre>Spi_JobResultType Spi_GetJobResult (const Spi_JobType Job)</pre>	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditional Reentrant - reentrant for the core	
Parameters (in)	Job	Job ID.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Spi_JobResultType	status of the job
Description	<p>This service returns the last transmission result of the specified Job. API returns the status of the job depending on whether job is queued, failed, pending or successful.</p> <p>In multicore context, API can only return the status of the jobs that are assigned to core in which API is called.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>SPI_E_PARAM_JOB: API service called with wrong parameter.</p> <p>SPI_E_UNINIT: API service used without module initialization.</p> <p>SPI_E_NOT_CONFIGURED: Sequence / job / channel number passed is not configured to core on which request is made but with-in the max range of sequence.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

SPI driver

21.3.3.7 Spi_GetSequenceResult

Table 1855 Specification for Spi_GetSequenceResult API

Syntax	<pre> Spi_SeqResultType Spi_GetSequenceResult (const Spi_SequenceType Sequence) </pre>	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditional Reentrant - Reentrant for the core	
Parameters (in)	Sequence	Sequence Id for which the status to be returned.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Spi_SeqResultType	Status of the Sequence
Description	<p>This service returns the last transmission result of the specified Sequence. API returns the status of the sequence depending on whether sequence is queued, failed, pending or successful</p> <p>In multicore context, API can only return the status of the sequence that are assigned to core in which API is called.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>SPI_E_PARAM_SEQ: API service called if the sequence ID is not in the range of the total sequence numbers allocated to all the cores.</p> <p>SPI_E_UNINIT: API service used without module initialization.</p> <p>SPI_E_NOT_CONFIGURED: Sequence / job / channel number passed is not configured to core on which request is made but with-in the max range of sequence.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

SPI driver**21.3.3.8 Spi_GetStatus****Table 1856 Specification for Spi_GetStatus API**

Syntax	<pre>Spi_StatusType Spi_GetStatus (void)</pre>	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Spi_StatusType	SPI_UNINIT The SPI Handler/Driver is not initialized or not usable SPI_IDLE The SPI Handler/Driver is not currently transmitting any Job SPI_BUSY The SPI Handler/Driver is performing a SPI Job (transmit)
Description	<p>This API returns the status of the driver as whole including synchronous and asynchronous transmissions (if configured).</p> <p>After reset and before Spi_Init() API is invoked, the status of the driver will be SPI_UNINIT</p> <p>In Multicore context, API will return the status of the driver for the kernels assigned to core only.</p>	
Source	AUTOSAR	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

SPI driver**21.3.3.9 Spi_GetVersionInfo****Table 1857 Specification for Spi_GetVersionInfo API**

Syntax	<pre>void Spi_GetVersionInfo (Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to the address where the driver information should be stored
Parameters (in - out)	-	-
Return	void	-
Description	This API updates the pointer address with the driver version information	
Source	AUTOSAR	
Error handling	<p>DET: SPI_E_PARAM_POINTER: APIs called with a null pointer.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	SpiVersionInfoApi	
User hints	-	

21.3.3.10 Spi_Init**Table 1858 Specification for Spi_Init API**

Syntax	<pre>void Spi_Init (const Spi_ConfigType * const ConfigPtr)</pre>
Service ID	0x00
Sync/Async	Synchronous
ASIL Level	B

SPI driver**Table 1858 Specification for Spi_Init API (continued)**

Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>The API Spi_Init initializes all the SFRs of QSPI kernels assigned to core, resets the global variables and sets the status to IDLE.</p> <p>In multicore context, Only the kernels assigned to core will be initialized.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>SPI_E_PARAM_POINTER: APIs called with a null pointer.</p> <p>SPI_E_ALREADY_INITIALIZED: API SPI_Init service called while the SPI driver has already been initialized.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

21.3.3.11 Spi_InitCheck**Table 1859 Specification for Spi_InitCheck API**

Syntax	<pre>Std_ReturnType Spi_InitCheck (const Spi_ConfigType * const ConfigPtr)</pre>	
Service ID	0x20	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to Configuration to be checked against
Parameters (out)	-	-

SPI driver**Table 1859 Specification for Spi_InitCheck API (continued)**

Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK on successfully checking the mentioned global variables / SFRs. E_NOT_OK if any of the global variables or SFR is not set as expected.
Description	<p>InitCheck API will check the registers and the critical variables are initialized as expected, it is mandated that the InitCheck API must be called after the Spi_Init API else the results will not be as expected.</p> <p>In Multicore context, only the globals and SFRs assigned to core will be verified in InitCheck API.</p> <p>Note that all API except Spi_Init and Spi_VersionCheck must be called only after successful return of initCheck when Safety is enabled.</p> <p>Note 1: Following pointers and SFR are verified in InitCheck API: ConfigPtr, Global core pointer, Global kernel pointer, Queue pointer for each kernel, buffer pointer in case of IB, Overall status of driver and kernel (should be IDLE), all QSPI related registers are set to reset state and all DMA RAM TCS elements source and destination address is updated during initialization.</p> <p>Note 2: Following global elements are not verified to be in reset state: content of IB buffer itself, content of the Queue variables and TCS memory itself, however critical variables of Queue like Queue index, start and end index are verified to be in reset state.</p> <p>Rationale: These variables will be updated at run-time and previous values will not affect any functionality since these will be updated for every transfers.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	SpiInitCheckApi	
User hints	-	

21.3.3.12 Spi_ReadIB**Table 1860 Specification for Spi_ReadIB API**

Syntax	Std_ReturnType Spi_ReadIB (const Spi_ChannelType Channel, Spi_DataBufferType * const DataBufferPointer)
---------------	---

SPI driver**Table 1860 Specification for Spi_ReadIB API (continued)**

Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - Reentrant for different channels for the core	
Parameters (in)	Channel	Channel ID.
Parameters (out)	DataBufferPointer	This is pointer to the destination buffer pointer to where the received data is copied
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Data from Internal buffer to the destination buffer is copied successfully E_NOT_OK: Data was not copied from internal buffer to the destination buffer
Description	<p>Service for reading synchronously the received channel data from Internal buffer to the destination buffer passed by application.</p> <p>In Multicore context, ReadIB will be successful for the channels assigned to core in which the request is made else the result is un-predictable.</p> <p>Note: Application should take care of protecting the buffer since driver do not use any protection mechanism to protect data. i.e. Application should sequence the call for writing to same channel only after reading the data else buffer corruption could occur.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>SPI_E_PARAM_POINTER: APIs called with a null pointer.</p> <p>SPI_E_UNINIT: API service used without module initialization.</p> <p>SPI_E_PARAM_CHANNEL: Incorrect parameter passed in API.</p> <p>SPI_E_NOT_CONFIGURED: Sequence / job / channel number passed is not configured to core on which request is made but with-in the max range of sequence.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	SpiChannelBuffersAllowed	
User hints	-	

SPI driver**21.3.3.13 Spi_SetAsyncMode****Table 1861 Specification for Spi_SetAsyncMode API**

Syntax	<pre>Std_ReturnType Spi_SetAsyncMode (const Spi_AsyncModeType Mode)</pre>	
Service ID	0x0d	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Mode	Specifies the asynchronous mechanism mode for SPI busses handled asynchronously in Level 2
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK Switching to new Mode is successful E_NOT_OK Switching to new mode was not successful
Description	<p>This API sets the asynchronous mode of handling transmission of sequences to either Polling mode or Interrupt mode.</p> <p>Available only in Level 2, mode cannot be updated / changed if any of the kernel assigned to core is in BUSY state.</p> <p>Note that each core can be configured to work in either polling or interrupt mode. API effects only the associated core variable indicating the mode in which it is working.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>SPI_E_UNINIT: API service used without module initialization.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>SPI_E_SAFETY_INVALID_PARAM: - If an invalid parameter is passed to Spi_SetAsyncMode API, the mode will be changed to the polling mode. As a safety check this DET is added. - If an invalid parameter is passed to ISR like wrong kernel which is not in range this safety DET will be triggered.</p> <p>- A safety check DET is reported when an invalid parameter is passed to the EnableOrDisable parameter of the Spi_ControlLoopBack API.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	SpiLevelDelivered	
User hints	-	

SPI driver**21.3.3.14 Spi_SetupEB****Table 1862 Specification for Spi_SetupEB API**

Syntax	<pre>Std_ReturnType Spi_SetupEB (const Spi_ChannelType Channel, const Spi_DataBufferType * const SrcDataBufferPtr, const Spi_DataBufferType ** const DesDataBufferPtr, const Spi_NumberOfDataType Length)</pre>	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Re-entrant - Reentrant for different channels for the core	
Parameters (in)	Channel SrcDataBufferPtr Length	Channel ID of the respective EB channel This is the pointer to source buffer for the EB channel Number of data elements to be transmitted. i.e., for 8-bit channel, if length is 2, then $2 * 8\text{-bit} = 16$ bits will be transferred, for 16-bit channel, if length is 2, then $2 * 16\text{-bit} = 32$ bits will be transferred and for 32-bit channel, if length is 2, then $2 * 32\text{-bit} = 64$ bits will be transferred
Parameters (out)	-	-
Parameters (in - out)	DesDataBufferPtr	This is the pointer to destination buffer to where the received data is copied
Return	Std_ReturnType	E_OK: Buffers has been setup for the EB channel E_NOT_OK: Buffer setup for the channel has not been accepted
Description	This API updates the source, destination and transfer length of channel. No DET is raised for pointers being NULL, if Source pointer is NULL indicates that default data to be used for transmission. If Destination address is NULL, then ignore the received data. In Multicore context, channel assigned to core can only be accessed.	
Source	AUTOSAR	
Error handling	DET: SPI_E_PARAM_LENGTH: Length parameter is greater than the defined limit. SPI_E_UNINIT: API service used without module initialization. SPI_E_PARAM_CHANNEL: Incorrect parameter passed in API.	

SPI driver**Table 1862 Specification for Spi_SetupEB API (continued)**

	<p>SPI_E_NOT_CONFIGURED: Sequence / job / channel number passed is not configured to core on which request is made but with-in the max range of sequence.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	SpiChannelBuffersAllowed
User hints	-

21.3.3.15 Spi_SyncTransmit**Table 1863 Specification for Spi_SyncTransmit API**

Syntax	<pre>Std_ReturnType Spi_SyncTransmit (const Spi_SequenceType Sequence)</pre>	
Service ID	0x0A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	Sequence	Sequence ID.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : Synchronous transmission request is accepted E_NOT_OK: Synchronous transmission request is rejected
Description	<p>This API transmits the sequence synchronously over the QSPI bus. Note that the API is a blocking API which means execution is blocked till the sequence is transmitted completely.</p> <p>Is enabled only if SpiLevelDelivered is 0 or 2.</p> <p>Note:</p> <ol style="list-style-type: none"> From multicore perspective, sequences assigned to core can only be transmitted else SPI_E_NOT_CONFIGURED DET will be returned. Sequence ID to be used by application will be of format - SpiConf_SpiSequence_(x), x is user defined string for ex: SpiConf_SpiSequence_EEPROM_Write 	
Source	AUTOSAR	
Error handling	DET:	

SPI driver**Table 1863 Specification for Spi_SyncTransmit API (continued)**

	<p>SPI_E_UNINIT: API service used without module initialization.</p> <p>SPI_E_PARAM_SEQ: API service called if the sequence ID is not in the range of the total sequence numbers allocated to all the cores.</p> <p>SPI_E_SEQ_IN_PROCESS: Synchronous transmission service called at wrong time.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SPI_E_HARDWARE_ERROR: On any error bit set in status register - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_FAILED).</p> <p>If no error is reported and successful transmission - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_PASSED)</p> <p>If any error is reported, application needs to monitor and take appropriate action.</p> <p>Note that SPI_E_HARDWARE_ERROR DEM is raised for QSPI Hardware errors or DMA ME errors.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	SpiLevelDelivered
User hints	-

21.3.3.16 Spi_WriteIB**Table 1864 Specification for Spi_WriteIB API**

Syntax	<pre>Std_ReturnType Spi_WriteIB (const Spi_ChannelType Channel, const Spi_DataBufferType * const DataBufferPtr)</pre>	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - Re-entrant for different channel for the core	
Parameters (in)	Channel DataBufferPtr	Channel ID. Pointer to source data buffer. If this pointer is null, it is assumed that the data to be transmitted is not relevant and the default transmit value of this channel will be used instead.
Parameters (out)	-	-
Parameters (in - out)	-	-

SPI driver**Table 1864 Specification for Spi_WriteIB API (continued)**

Return	Std_ReturnType	E_OK: specifies the data in the source buffer is copied into the local internal buffer E_NOT_OK: requested functionality is not done
Description	<p>This API copies the data to be transmitted in transmit buffer from the source pointer passed in the API by application.</p> <p>Dependency: If SpiChannelBufferAllowed is set to 0 OR 2 this API will be enabled during compilation.</p> <p>In multicore context, only the channels assigned to core can be written.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>SPI_E_PARAM_CHANNEL: Incorrect parameter passed in API.</p> <p>SPI_E_UNINIT: API service used without module initialization.</p> <p>SPI_E_NOT_CONFIGURED: Sequence / job / channel number passed is not configured to core on which request is made but with-in the max range of sequence.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	SpiChannelBuffersAllowed	
User hints	-	

21.3.4 Notifications and Callbacks**21.3.4.1 Spi_QspiDmaCallOut****Table 1865 Specification for Spi_QspiDmaCallOut API**

Syntax	<pre>void Spi_QspiDmaCallOut (const uint32 Channel, const Dma_EventsType Event)</pre>	
Service ID	0x21	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Channel Event	Channel number [0-127] an enumeration for all DMA events

SPI driver**Table 1865 Specification for Spi_QspiDmaCallOut API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	DMA callback is called at end of every channel transmission, during this callback channel is re-configured and respective DMA channels are re-triggered to start the next channel transfer. This callback cannot be avoided since each channels can have different data length and same has to be re-configured in the BACON register.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: SPI_E_SAFETY_INVALID_PARAM: - If an invalid parameter is passed to Spi_SetAsyncMode API, the mode will be changed to the polling mode. As a safety check this DET is added. - If an invalid parameter is passed to ISR like wrong kernel which is not in range this safety DET will be triggered. - A safety check DET is reported when an invalid parameter is passed to the EnableOrDisable parameter of the Spi_ControlLoopBack API. SPI_E_SAFETY_SPURIOUS_INTERRUPT: For every interrupt triggered, source of interrupt will be checked, if no source can be detected this safety error will be triggered. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

21.3.4.2 Spi_QspiDmaErrCallout**Table 1866 Specification for Spi_QspiDmaErrCallout API**

Syntax	<pre>void Spi_QspiDmaErrCallout (const uint32 Channel, const Dma_EventsType Event)</pre>	
Service ID	0x24	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Channel Event	Channel number [0-127] An enumeration for all DMA events.

SPI driver**Table 1866 Specification for Spi_QspiDmaErrCallout API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function is called from DMA module on detecting a move engine error during DMA transfer.	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SPI_E_HARDWARE_ERROR: On any error bit set in status register - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_FAILED).</p> <p>If no error is reported and successful transmission - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_PASSED)</p> <p>If any error is reported, application needs to monitor and take appropriate action.</p> <p>Note that SPI_E_HARDWARE_ERROR DEM is raised for QSPI Hardware errors or DMA ME errors.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

21.3.5 Scheduled functions**21.3.5.1 Spi_MainFunction_Handling****Table 1867 Specification for Spi_MainFunction_Handling API**

Syntax	void Spi_MainFunction_Handling (void)	
Service ID	0x10	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-

SPI driver**Table 1867 Specification for Spi_MainFunction_Handling API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function polls the SPI interrupt flags linked to QSPI Hardware units. This function will be called by application at regular interval as defined in the application.	
Source	AUTOSAR	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>SPI_E_SAFETY_INVALID_PARAM: - If an invalid parameter is passed to Spi_SetAsyncMode API, the mode will be changed to the polling mode. As a safety check this DET is added. - If an invalid parameter is passed to ISR like wrong kernel which is not in range this safety DET will be triggered.</p> <p>- A safety check DET is reported when an invalid parameter is passed to the EnableOrDisable parameter of the Spi_ControlLoopBack API.</p> <p>SPI_E_SAFETY_SPURIOUS_INTERRUPT: For every interrupt triggered, source of interrupt will be checked, if no source can be detected this safety error will be triggered.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	SpiLevelDelivered	
User hints	-	

21.3.6 Interrupt service routines**21.3.6.1 Spi_IsrQspiError****Table 1868 Specification for Spi_IsrQspiError API**

Syntax	void Spi_IsrQspiError (const uint8 Module)
Service ID	0x23
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Non Reentrant

SPI driver**Table 1868 Specification for Spi_IsrQspiError API (continued)**

Parameters (in)	Module	Kernel number 0-5
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This interrupt service routine handles the QSPI errors during asynchronous transmission.</p> <p>Sets the status of the Sequence to SPI_SEQ_FAILED and job status belonging to the sequence except which are completed to SPI_JOB_FAILED.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SPI_E_HARDWARE_ERROR: On any error bit set in status register - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_FAILED).</p> <p>If no error is reported and successful transmission - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_PASSED)</p> <p>If any error is reported, application needs to monitor and take appropriate action.</p> <p>Note that SPI_E_HARDWARE_ERROR DEM is raised for QSPI Hardware errors or DMA ME errors.</p> <p>Safety Errors:</p> <p>SPI_E_SAFETY_INVALID_PARAM: - If an invalid parameter is passed to Spi_SetAsyncMode API, the mode will be changed to the polling mode. As a safety check this DET is added. - If an invalid parameter is passed to ISR like wrong kernel which is not in range this safety DET will be triggered.</p> <p>- A safety check DET is reported when an invalid parameter is passed to the EnableOrDisable parameter of the Spi_ControlLoopBack API.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

21.3.6.2 Spi_IsrQspiPT2**Table 1869 Specification for Spi_IsrQspiPT2 API**

Syntax	void Spi_IsrQspiPT2 (const uint8 Module)
---------------	---

SPI driver**Table 1869 Specification for Spi_IsrQspiPT2 API (continued)**

Service ID	0x22	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Module	QSPI module index [0 – 5]
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	PT2 - Phase transition interrupt signals one out of all phases of Hardware state transition. This Interrupt Service routine marks the end of the frame transmission and is triggered only at the end of the job transmission. Total number of elements to be transmitted in a next job is updated in the MCCOUNT during this interrupt and respective DMA channels are re-triggered.	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: SPI_E_SAFETY_SPURIOUS_INTERRUPT: For every interrupt triggered, source of interrupt will be checked, if no source can be detected this safety error will be triggered. SPI_E_SAFETY_INVALID_PARAM: - If an invalid parameter is passed to Spi_SetAsyncMode API, the mode will be changed to the polling mode. As a safety check this DET is added. - If an invalid parameter is passed to ISR like wrong kernel which is not in range this safety DET will be triggered. - A safety check DET is reported when an invalid parameter is passed to the EnableOrDisable parameter of the Spi_ControlLoopBack API. <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

21.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

21.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

SPI driver

Note: The following error IDs are also reported as safety errors.

Table 1870 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
API SPI_Init service called while the SPI driver has already been initialized.	AUTOSAR	SPI_E_ALREADY_INITIALIZED=0x4A	Spi_Init
Sequence / job / channel number passed is not configured to core on which request is made but with-in the max range of sequence.	AUTOSAR	SPI_E_NOT_CONFIGURED=0x4B	Spi_SetupEB, Spi_ReadIB, Spi_GetJobResult, Spi_WritelB, Spi_GetSequenceResult, Spi_Cancel, Spi_AsyncTransmit
Incorrect parameter passed in API.	AUTOSAR	SPI_E_PARAM_CHANNEL=0x0A	Spi_ReadIB, Spi_SetupEB, Spi_WritelB
API service called with wrong parameter.	AUTOSAR	SPI_E_PARAM_JOB=0x0B	Spi_GetJobResult
Length parameter is greater than the defined limit.	AUTOSAR	SPI_E_PARAM_LENGTH=0x0D	Spi_SetupEB
APIs called with a null pointer.	AUTOSAR	SPI_E_PARAM_POINTER=0X10	Spi_GetVersionInfo, Spi_ReadIB, Spi_Init
API service called if the sequence ID is not in the range of the total sequence numbers allocated to all the cores.	AUTOSAR	SPI_E_PARAM_SEQ=0X0C	Spi_GetSequenceResult, Spi_SyncTransmit, Spi_Cancel, Spi_AsyncTransmit
API service called with wrong parameter.	AUTOSAR	SPI_E_PARAM_UNIT=0x0E	Spi_ControlLoopBack, Spi_GetHWUnitStatus
If a new sequence is requested to be transmitted and if slots are less than number of jobs in Queue this DET is raised. On this DET user can increase the Queue length in SpiJobQueueLengthQspix field.	AUTOSAR	SPI_E_QUEUE_FULL=0x4C	
Synchronous transmission service called at wrong time.	AUTOSAR	SPI_E_SEQ_IN_PROCESS=0X3A	Spi_SyncTransmit
Indicates that sequence is in Pending state and requested action cannot be performed.	AUTOSAR	SPI_E_SEQ_PENDING=0X2A	Spi_AsyncTransmit

SPI driver**Table 1870 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
API service used without module initialization.	AUTOSAR	SPI_E_UNINIT=0X1A	Spi_ControlLoopBack, Spi_GetHWUnitStatus, Spi_SetAsyncMode, Spi_GetJobResult, Spi_GetSequenceResult, Spi_ReadIB, Spi_SetupEB, Spi_SyncTransmit, Spi_Cancel, Spi_AsyncTransmit, Spi_WritelB, Spi_Delnit

21.3.7.2 Production errors

The following table lists all the production errors reported by the driver.

Table 1871 Description of production errors reported

Description	Source	Error code and value	Applicable APIs
On any error bit set in status register - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_FAILED). If no error is reported and successful transmission - Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_PASSED) If any error is reported, application needs to monitor and take appropriate action. Note that SPI_E_HARDWARE_ERROR DEM is raised for QSPI Hardware errors or DMA ME errors.	AUTOSAR	SPI_E_HARDWARE_ERROR=Assigned by DEM	Spi_QspiDmaErrCallout, Spi_IsrQspiError, Spi_SyncTransmit, Spi_AsyncTransmit

21.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

SPI driver
Table 1872 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
<ul style="list-style-type: none"> - If an invalid parameter is passed to Spi_SetAsyncMode API, the mode will be changed to the polling mode. As a safety check this DET is added. - If an invalid parameter is passed to ISR like wrong kernel which is not in range this safety DET will be triggered. - A safety check DET is reported when an invalid parameter is passed to the EnableOrDisable parameter of the Spi_ControlLoopBack API. 	IFX	SPI_E_SAFETY_INVALID_PARAM=0x65	Spi_MainFunction_Handling, Spi_QspiDmaCallOut, Spi_ControlLoopBack, Spi_IsrQspiError, Spi_IsrQspiPT2, Spi_SetAsyncMode
For every interrupt triggered, source of interrupt will be checked, if no source can be detected this safety error will be triggered.	IFX	SPI_E_SAFETY_SPURIOUS_INTERRUPT=0x66	Spi_MainFunction_Handling, Spi_QspiDmaCallOut, Spi_IsrQspiPT2

21.3.7.4 Runtime errors

The driver does not report any runtime errors.

21.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

21.3.8.1 Deviations

The section describes the deviations from software specification.

Table 1873 Known deviations

Reference	Deviation
SpiHwUnit	These parameters have the post-build variant value set to FALSE.
SpiHwUnitSynchronous	
SpiChannelType	
SpiHwUnitSynchronous	
SpiChannelAssignment	
SpiDataWidth	

SPI driver**Table 1873 Known deviations (continued)**

Reference	Deviation
SpilbNBuffers	
Address and Data CRC	The SPI driver does not implement the CRC mechanism provided by the DMA driver. It is the responsibility of the upper layer to perform the integrity check of the data by using additional CRC mechanisms.

21.3.8.2 Limitations

The section describes the limitations from software specification.

Table 1874 Known limitations

Reference	Limitation
DMA Transaction control set (TCS) Memory alignment	SPI driver uses the DMA to perform asynchronous data transfer which inturn uses TCS to store the configuration of DMA. Behaviour of the DMA / SPI driver is un-predictable if the TCS is not aligned to 32-byte boundary. Ex: Spi_DmaTxControlSetArrayQSPI0 Rationale: DMA Hardware mandates TCS to be 32-byte aligned.
Buffer alignment	All buffers that are used for transmission and reception of data shall be aligned to 4-byte boundary. Rationale: Odd address access would create a trap.
Sleep mode	Application should ensure that there is no ongoing SPI communication when CPU is requested for sleep. Rationale: If CPU enters sleep mode, clock to peripheral modules would be turned OFF.
SpidleTime, SpiTrailingTime, SpiTimeClk2Cs	These fields needs to be in default settings if chosen to be operated through CS_VIA_GPIO, since the timing cannot be guaranteed when operating through GPIO.
Level 2 configuration	When SPI is configured in Level 2, DMA module needs to be added as dependent module and DMA trigger APIs Dma_ChEnableHardwareTrigger, Dma_ChStartTransfer, Dma_ChStopTransfer, Dma_ChUpdate are to be enabled though the synchronous sequences are only configured.
Cache and non-cache sections of memory	The Global buffers Spi_TxIBBufferCorex, Spi_RxIBBufferCorex, Spi_DmaTxControlSetArrayQSPIx and Spi_DmaRxControlSetArrayQSPIx are extensively used by DMA for data transfer and for TCS, so memory sections for these global must not be cached. Note that EB buffers that are used by applications are to be allocated in non-cached section of memory.
Status request	In multi-threaded environment Spi_GetJobResult, Spi_GetSequenceResult may return the stale data of job and sequence if the job and sequence is not yet completed. In order to ensure that the results are for the job / sequence that has been

SPI driver**Table 1874 Known limitations (continued)**

Reference	Limitation
	executed application code must call the status API from the notification function.
SLSO on Power-on	On power-up, before the first frame is transmitted to any slave device CS lines are held in low state. Before the start of first frame all the SLSO pins are de-asserted and only the selected slave is asserted. For Successive frames SLSO levels will behave as expected and this is observed for hardware triggered SLSO only. No functional impact is expected from this behaviour.
TRL(Transaction Request Lost) Event Behaviour	If DMA Move Engine error notification is enabled, it is being observed that error handler of the DMA is triggered due to DMA hardware safety feature though the transfer is not erroneous. This TRL event is suppressed by the DMA driver to continue with normal operation. Note: User has to disable the configuration parameter <code>DmaTcsInterruptTransactionLoss</code> to avoid TRL interrupt.
Controlling Loopback mode feature	Application should ensure that the <code>Spi_ControlLoopBack</code> API is invoked only when <code>Spi_GetStatus</code> returns SPI_IDLE. This ensures that there is no ongoing transmission on any SPI kernel before enabling or disabling the loopback mode.
<code>SpiInitCheckApi</code>	<code>Spi_InitCheck</code> API is enabled when <code>SpiSafetyCheckEnable</code> is turned ON and will be disabled when <code>SpiSafetyCheckEnable</code> is OFF, exercising <code>SpiInitCheckApi</code> ON / OFF does not have an impact on enabling / disabling the <code>SpiInitCheckApi</code> .

21.3.9 Unsupported hardware features

The following hardware features of the SPI driver are not supported:

- High-speed input capture
- Slave mode
- Long data block transfer
- ASCLIN
- MIX entry

Wdg_17_Scu driver**22 Wdg_17_Scu driver****22.1 User information****22.1.1 Description**

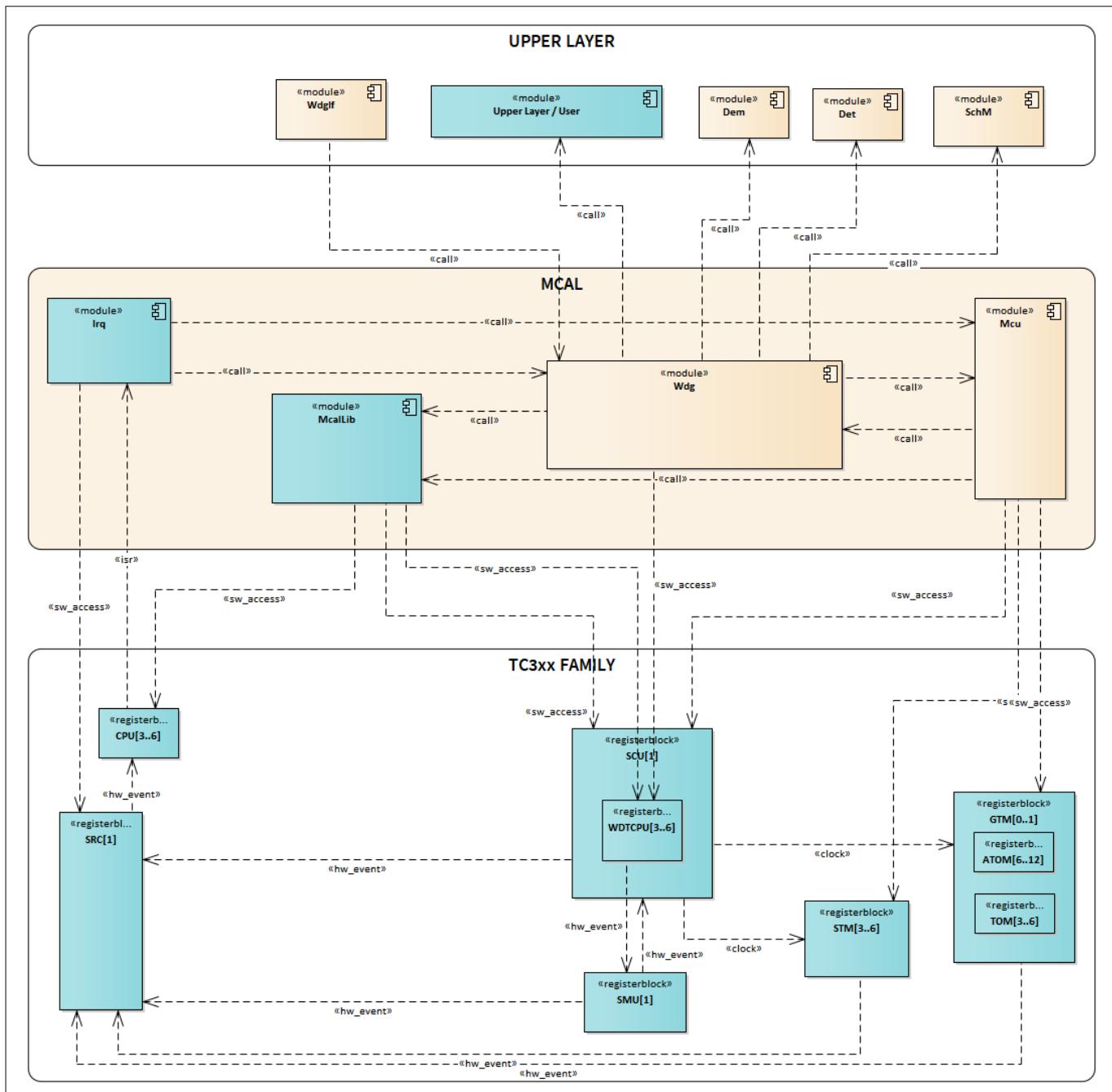
In AUTOSAR, the watchdog stack implements three monitoring mechanisms:

- Alive supervision: for supervision of timing of periodic software
- Deadline supervision: for aperiodic software
- Logical supervision: for supervision of the correctness of the execution sequence

The Watchdog (WDG) driver implements watchdog SWS as specified by AUTOSAR (4.2.2), which provides services for initialization, changing the operation mode and setting the trigger condition (timeout), for use by the user of watchdog hardware (WDT). The users of the WDG driver are: Mode management modules (for initialization), WdgM through WdgIf for setting trigger condition and driver operation mode setting. The WdgM implements the above mentioned supervision mechanisms. AUTOSAR extends triggering concept to support windowed watchdog mode. This calls for use of hardware timers to self-trigger WDT until the user-set trigger condition is met. For this purpose, the WDG driver provides options to select one of the two timers, either STM or GTM timers. Since code base would be common for all CPUs, to avoid presence of unused code, the timer selection is made static (pre-compile) for all WDTs (the driver uses either GTM or STM for all WDTs).

22.1.2 Hardware-software mapping

This section describes the system view of the driver and peripherals administered by it.

Wdg_17_Scu driver

Figure 199 **Mapping of hardware-software interfaces**

22.1.2.1 STM: dependent hardware peripheral

Hardware functional features

The WDG driver depends on the STM for the compare match event which is required for servicing the watchdog timer.

The STMxTIM0 (x = Core Id) register is used by watchdog timer for calculating mode-specific refresh time to service the watchdog timer. Each watchdog timer is mapped to a unique STM timer compare register.

Users of the hardware

The McalLib, MCU and WDG drivers are users of the STM. To avoid conflicts, the access to the STM registers takes place through APIs provided by the MCU driver.

Wdg_17_Scu driver

Hardware diagnostic features

Not applicable.

Hardware events

The WDG driver uses the following hardware events from the STM for compare match:

- STMIR0 is used when CMP0 is configured for the watchdog
- STMIR1 is used when CMP1 is configured for the watchdog

22.1.2.2 GTM: dependent hardware peripheral

Hardware functional features

The WDG driver depends on the GTM for compare match event, which is required for servicing the WDT. Each WDT is mapped to a unique TOM/ATOM channel.

The GTM TOM/ATOM functional block feature configured/accessed by the WDG driver is: ATOM/TOM compare match interrupt events feature is used to achieve WDG functionality. Rest of the GTM features are not used

Users of the hardware

The TOM/ATOM channels are used by the WDG, PWM, GPT and ADC drivers. To avoid conflicts, the access requests to the GTM channels is performed via the MCU driver.

Hardware diagnostic features

Not applicable.

Hardware events

The WDG driver uses the following hardware event from the GTM:

- Compare match event is used from the GTM to service the WDT

22.1.2.3 SCU: primary hardware peripheral

Hardware functional features

The WDG driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the FSPB clock signal for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, updates to all the ENDINIT protected registers are performed using the MCALLIB APIs.

Hardware events

Hardware events from the SCU are not used by the WDG driver.

WDG: primary hardware peripheral

The TC3xx contains the following WDTs:

- One system WDT
- One WDT per CPU

Hardware functional features

The WDG driver uses the WDTs for servicing at defined intervals as per application requirements. The key hardware functional features used by the driver are:

- 16-bit WDT counter is used to implement watchdog functionality (WDTCPUxSR)
- 16-bit user-definable reload value is used for normal WDT operation (WDTCPUxCON0). The reload value for normal mode shall be computed based on trigger condition at run time (A fixed reload value provided in the hardware for timeout mode shall only be used for updating CPU critical registers)

Wdg_17_Scu driver

- Supports disabling of WDT(WDTCPUxCON1)
- Selectable input frequency is fSPB/64, fSPB/256 or fSPB/16384 (WDTCPUxCON1). Frequency fSPB/64 is not used (fast mode frequency is fSPB/256 and slow mode frequency is fSPB/16384)

The unsupported features of the WDT are:

- System WDT (Support is provided for CPU WDT only)
- Incorporation of the corresponding ENDINIT bit and monitoring of its modifications. The WDG ENDINIT feature will only be used for CPU critical registers and will not be supported by WDG driver
- Automatic password sequencing mechanism is not supported. A fixed reload password will be used
- The UR bit in WDTCPU0CON1 register will not be configurable through WDG driver. Updating this bit depends on the state of the SMU which is not monitored by the WDG driver
- Time Check Password feature is not supported

Users of the hardware

The WDG and MCALLIB drivers exclusively utilize the WDT.

The MCALLIB driver uses the WDT to implement the ENDINIT protection feature in the timeout mode.

To avoid conflicts, the WDG and MCALLIB drivers use the critical section with the same exclusive area name while accessing the shared resources. User shall ensure that core-specific interrupts are disabled in the critical sections since CON0 register is updated inside these critical sections by both the drivers. User shall ensure that the software on each core accesses only the core-specific WDT registers.

Hardware diagnostic features

Following are the hardware error status flags monitored in the WDG driver:

- Overflow error detection
- Access error detection

The SMU alarms configured for the WDG are not monitored by the WDG driver.

Hardware events

Not applicable.

22.1.2.4 SMU: dependent hardware peripheral

Hardware functional features

The WDG driver does not depend on the SMU driver for any functionality.

However, the following features of SMU can be used in conjunction with the WDT:

- An overflow of the WDT can trigger an alarm request to the SMU
- SMU can be configured to reset the system in case of the WDT overflow

Users of the hardware

The SMU configuration is handled only by the SMU driver.

Hardware diagnostic features

The SMU alarms are not monitored by the WDG driver.

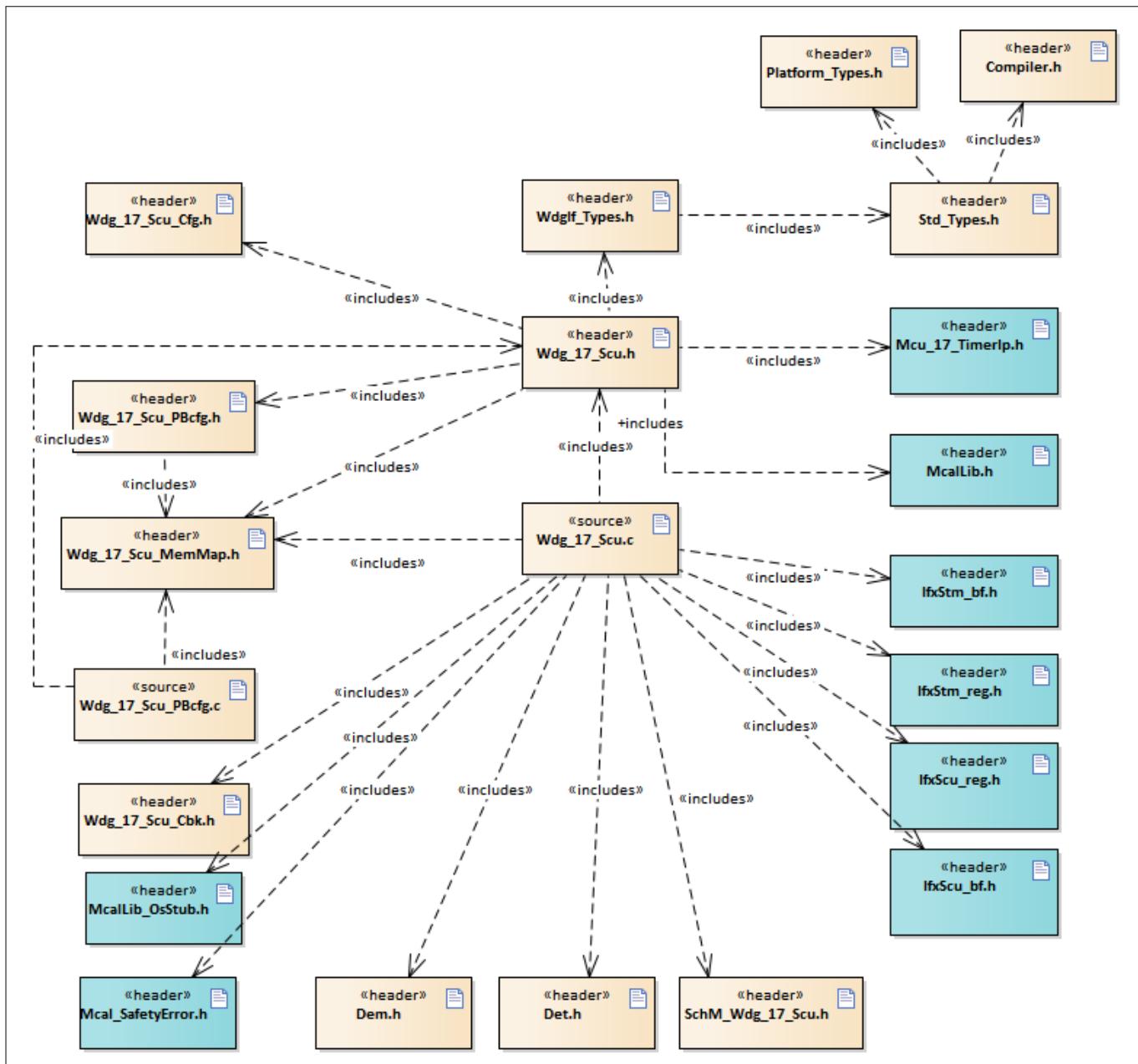
Hardware events

Not applicable.

22.1.3 File structure

22.1.3.1 C file structure

This section provides details of the C files of the WDG driver.

Wdg_17_Scu driver

Figure 200 **C file structure**
Table 1875 **C file structure**

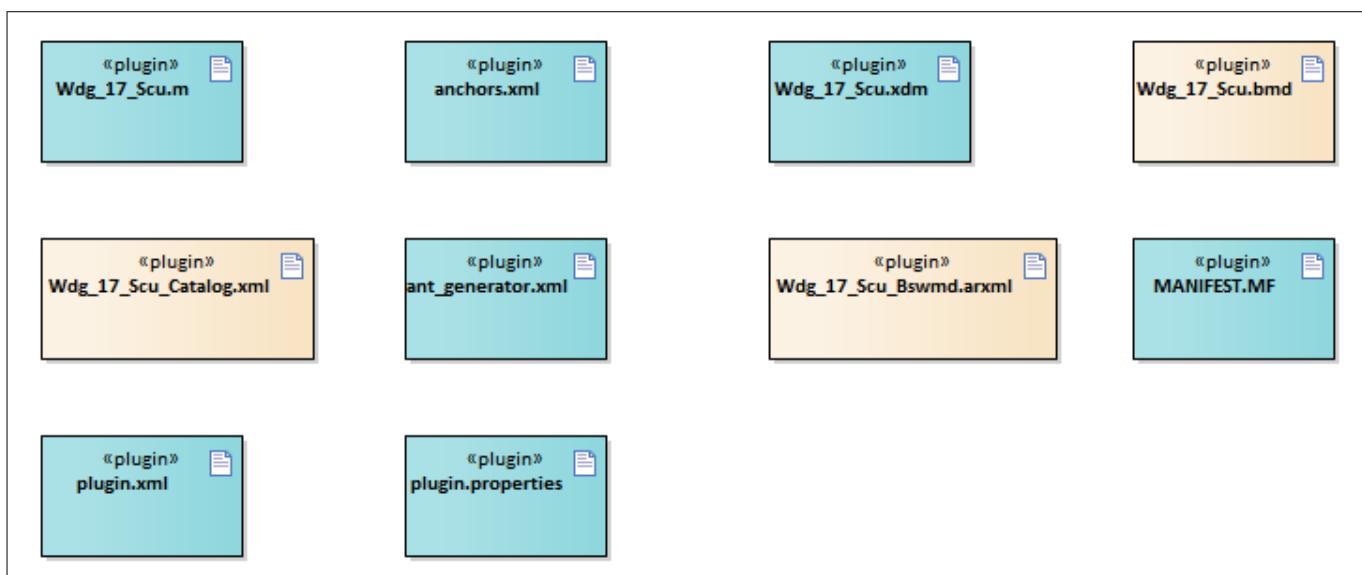
File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Dem.h	Provides the exported interfaces of Diagnostic Event Manager
Det.h	Provides the exported interfaces of Development Error Tracer
IfxScu_bf.h	SFR header file for SCU
IfxScu_reg.h	SFR header file for SCU
IfxStm_bf.h	SFR header file for STM
IfxStm_reg.h	SFR header file for STM

Wdg_17_Scu driver
Table 1875 C file structure (continued)

File name	Description
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore™. This shall be included by other drivers to call OS APIs.
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Wdg_17_Scu.h	Export header for SchM functions of WDG driver
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.
WdgIf_Types.h	Provides type definitions of the Watchdog Interface
Wdg_17_Scu.c	Source file providing implementation of APIs
Wdg_17_Scu.h	Header file providing prototypes of APIs, data types and interrupt handlers
Wdg_17_Scu_Cbk.h	Header file contains call back function of the WDG driver
Wdg_17_Scu_Cfg.h	Generated header file containing macros
Wdg_17_Scu_MemMap.h	File (Static) containing the memory section definitions used by the WDG driver
Wdg_17_Scu_PBcfg.c	Generated source file containing configuration data of the user
Wdg_17_Scu_PBcfg.h	Generated header file containing configuration data of the user

22.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the WDG driver.


Figure 201 Code generator plugin files

Wdg_17_Scu driver

Table 1876 Code generator plugin files

File name	Description
MANIFEST.MF	Tresos plugin support file containing the metadata for the WDG driver
Wdg_17_Scu.bmd	AUTOSAR format XML data model schema file (for each device)
Wdg_17_Scu.m	Code template macro file for WDG driver
Wdg_17_Scu.xdm	Tresos format XML data model schema file
Wdg_17_Scu_Bswmd.arxml	AUTOSAR format module description file
Wdg_17_Scu_Catalog.xml	AUTOSAR format catalog file
anchors.xml	Tresos anchors support file for the WDG driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configuration when using variation point
plugin.properties	Tresos plugin support file for the WDG driver
plugin.xml	Tresos plugin support file for the WDG driver

22.1.4 Integration hints

This section lists the key points that an integrator or user of the WDG driver must consider.

22.1.4.1 Integration with AUTOSAR stack

This section lists the modules which are not part of the MCAL, but are required to integrate the WDG driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of the ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Wdg_17_Scu_MemMap.h` file.

The `Wdg_17_Scu_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements

Wdg_17_Scu driver

are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows:

```
#define MEMMAP_ERROR
/*To be used for all global or static variables.*/
/*Variable to be cleared at startup or reset is placed here - .bss*/
#if defined WDG_17_SCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*User pragmas here*/
#define WDG_17_SCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR

#elif defined WDG_17_SCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
/*User pragmas here*/
#define WDG_17_SCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
#define MEMMAP_ERROR

/*Constants with attributes that show that they reside in one segment for
module
configuration.*/
#elif defined WDG_17_SCU_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
/*User pragmas here*/
#define WDG_17_SCU_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#define MEMMAP_ERROR

#elif defined WDG_17_SCU_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
/*User pragmas here*/
#define WDG_17_SCU_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#define MEMMAP_ERROR

/* Code Sections */
#elif defined WDG_17_SCU_START_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here*/
#define WDG_17_SCU_START_SEC_CODE_ASIL_B_GLOBAL
#define MEMMAP_ERROR

#elif defined WDG_17_SCU_STOP_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here*/
#define WDG_17_SCU_STOP_SEC_CODE_ASIL_B_GLOBAL
#define MEMMAP_ERROR

#endif
#if defined MEMMAP_ERROR
#error "Wdg_17_Scu_MemMap.h, wrong pragma command"
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The WDG driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the WDG driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

Wdg_17_Scu driver

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is a part of the AUTOSAR stack that handles all the production errors reported by the BSW modules. The WDG driver reports all the production errors to the DEM modules through the `Dem_ReportErrorStatus()` API. The user of the WDG driver must process all the production errors (fail / pass) reported to the DEM module through the `Dem_ReportErrorStatus()` API.

The `Dem.h` and `Dem.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DEM module during the integration phase.

- **SchM**

The SchM module is a part of the RTE that manages the BSW Scheduler. The WDG driver uses the exclusive areas defined in `SchM_Wdg_17_Scu.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the WDG driver are:

- TimerHandling
- ChangeMode
- CpuEndInit

The `SchM_Wdg_17_Scu.h` and `SchM_Wdg_17_Scu.c` files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions

Wdg_17_Scu driver

defined by the WDG driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows:

```
/*Sample implementation of SchM_Wdg_17_Scu.c*/

#include "IFX_Os.h"
#include "SchM_Wdg_17_Scu.h"

void SchM_Enter_Wdg_17_Scu_TimerHandling(void)
{
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Wdg_17_Scu_TimerHandling(void)
{
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Wdg_17_Scu_ChangeMode(void)
{
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Wdg_17_Scu_ChangeMode(void)
{
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Wdg_17_Scu_CpuEndInit(void)
{
    SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Wdg_17_Scu_CpuEndInit(void)
{
    ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

The WDG driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API. The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

The WDG driver does not expect any callbacks from application but it needs a call back from MCU driver for ISR handling.

- **Operating system (OS)**

Wdg_17_Scu driver

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or the application. The OS files provided by the MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

22.1.4.2 Multicore and Resource Manager

Multicore concept for WDG driver

Every CPU has one dedicated WDT. The global status variable is used to maintain the core-specific reload value, status (IDLE, BUSY, UNINIT), timeout counter value and mode (FAST, SLOW, OFF) of the WDT. Initialization is core specific (CPUx can update/modify registers and variables for WDTx only. x = 0,1,2,3,4,5). At least one CPU WDT must be configured if the WDG driver is used in an application. The key points to be considered with respect to multicore in the driver are as follows:

- Locating the constants, variables and configuration data to correct memory space should be carried out by the user. Memory sections are marked GLOBAL (common to all cores) and CORE[x] (specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

Code section:

The executable code of the WDG driver is placed under single MemMap section. It can be relocated to any PFlash region.

Data section:

The RAM variable memory sections marked as specific to a core should be relocated to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region.

Configuration data and constants:

The configuration data sections marked as specific to a core should be relocated to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

Note: Relocating code, data or constants to a distant memory region would impact execution timings.

Note: If the driver operates from a single (master) core, all the sections may be relocated to the PFlash/ DSPR/DLMU of the same CPU core.

Resource manager for WDG

The user should allocate resources of WDG to CPU cores at pre-compile time using the Resource manager module. WDG driver depends on resource manager if STM timer is configured for its timing needs. An STM timer should be allocated to the same core in the resource manager for which a CPU WDG is configured.

Wdg_17_Scu driver

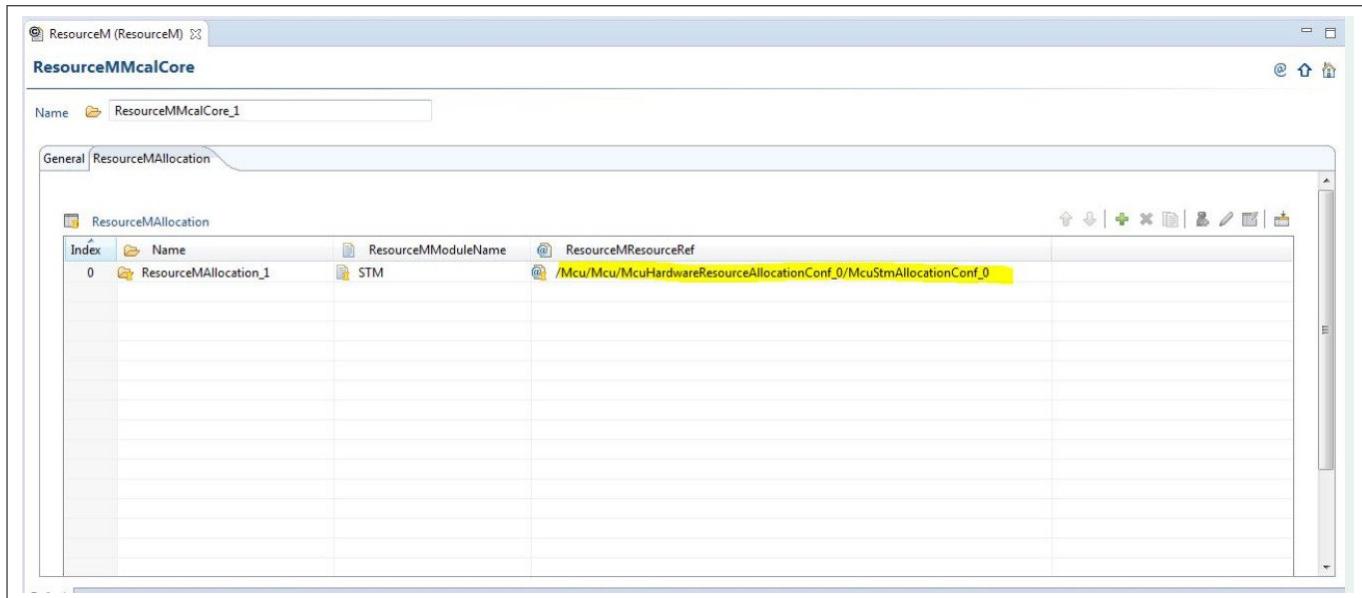


Figure 202 STM timer allocation in Resource Manager

22.1.4.3 MCU support

Timer support

The WDG driver is dependent on the MCU driver for clock configuration and GTM and STM timer services. The initialization of the WDG driver must be started only after completion of the MCU initialization.

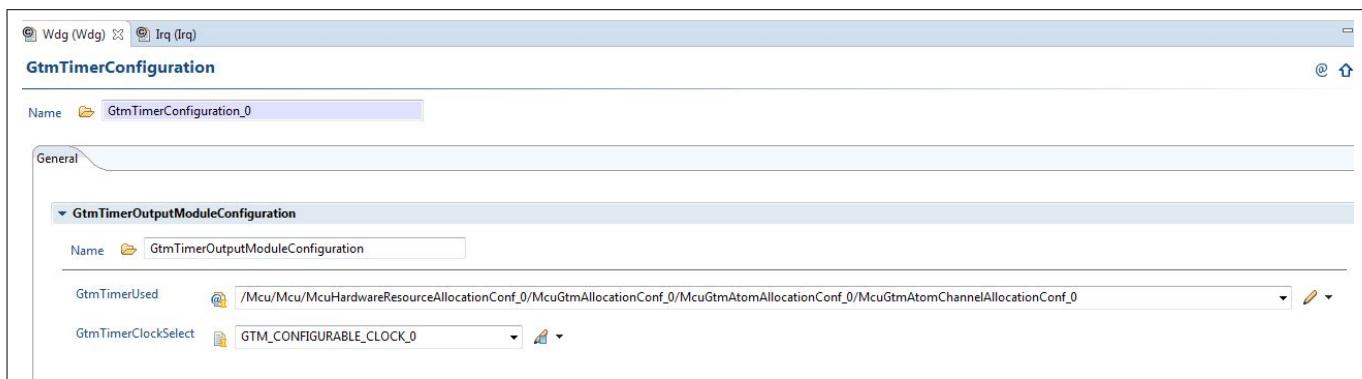
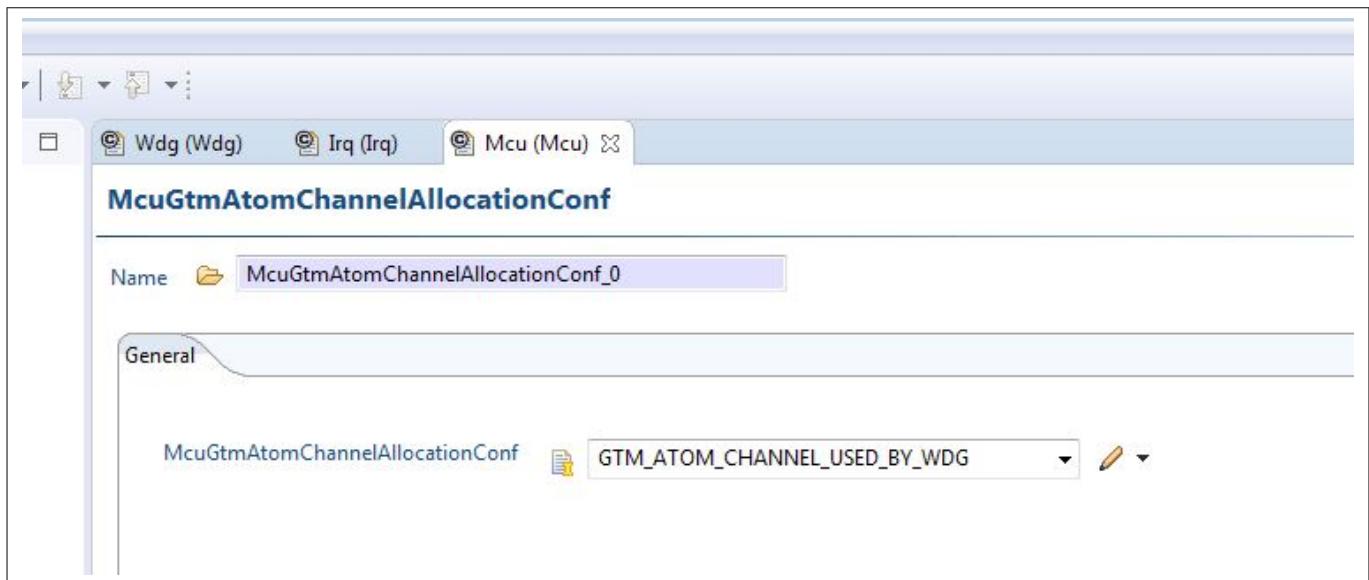
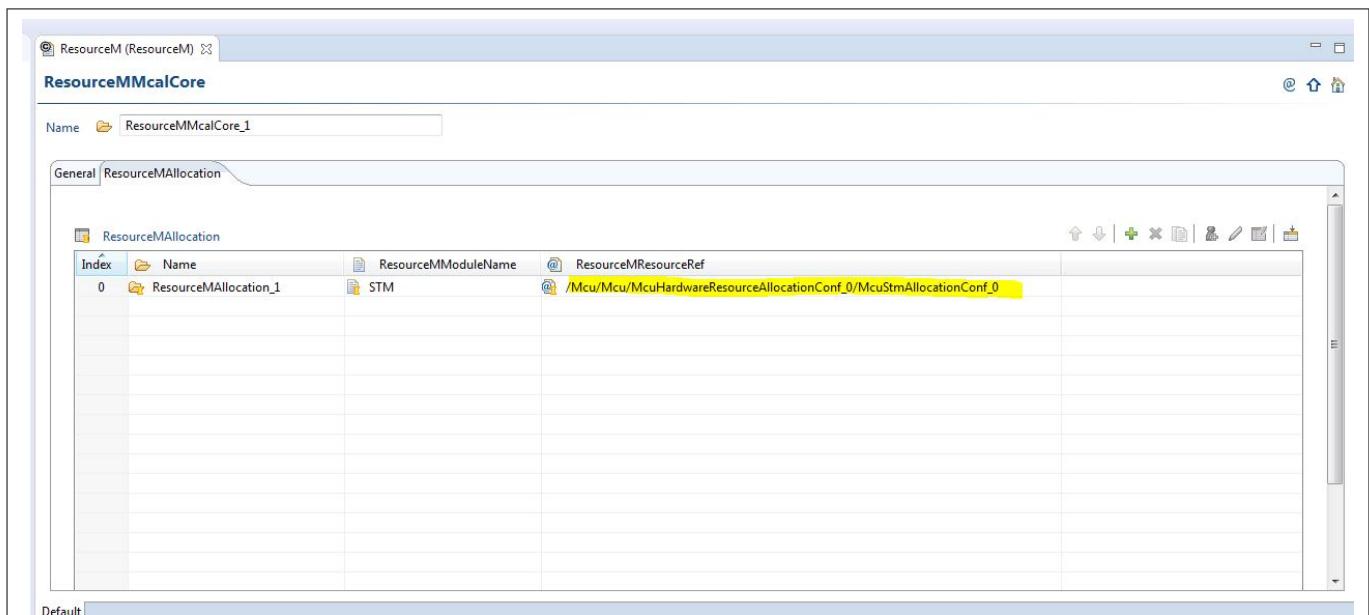


Figure 203 WDG configuration with a reference to GTM TOM/ATOM channel in MCU

Wdg_17_Scu driver**Figure 204** MCU configuration for GTM TOM/ATOM channel**Figure 205** STM timer reference in Resource Manager to MCU

Wdg_17_Scu driver

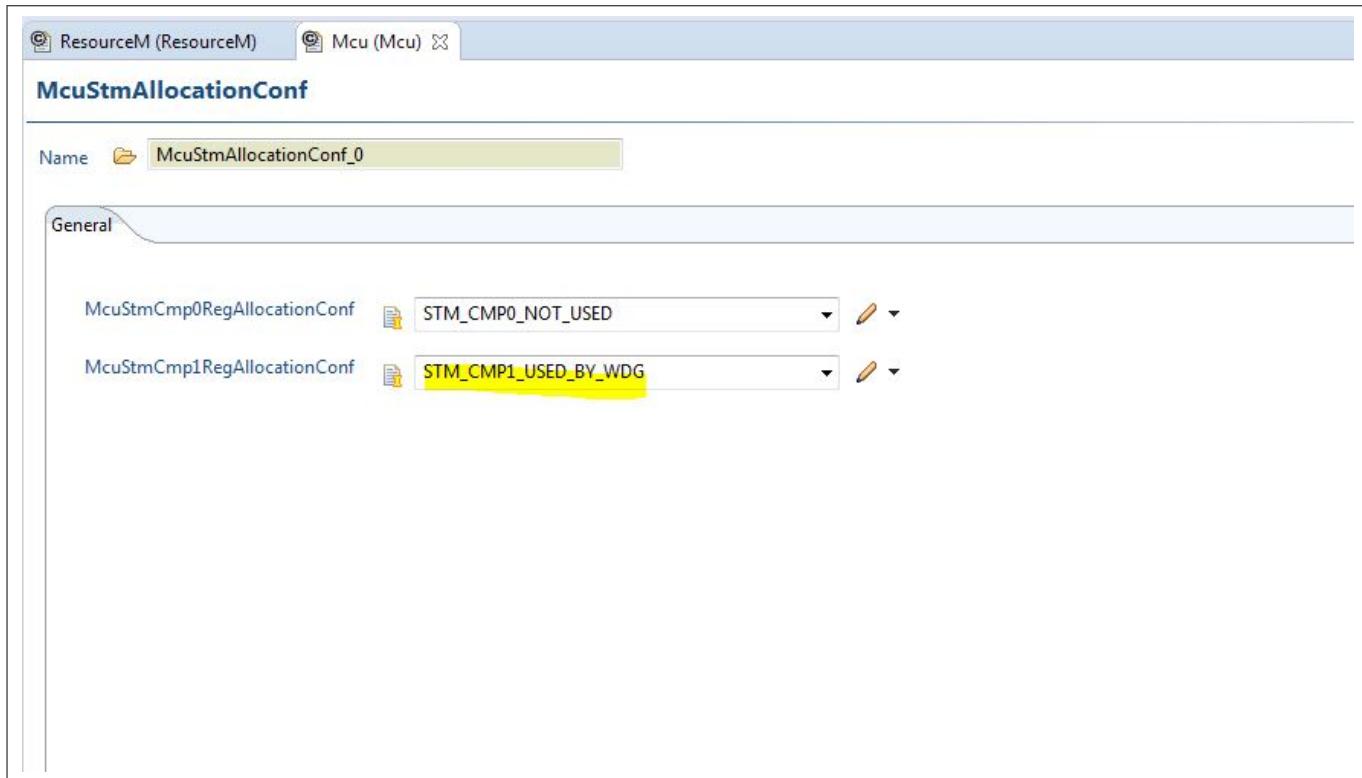


Figure 206 STM Compare Register allocation for WDG in MCU

22.1.4.4 Port support

The WDG driver does not use any services provided by the PORT driver.

22.1.4.5 DMA support

The WDG driver does not use any services provided by the DMA driver.

22.1.4.6 Interrupt connections

The interrupt connections of the Wdg driver are described in this section.

GTM timer

If GTM timer is used, the priority number for the interrupt must be configured in the IRQ module based on the TOM or ATOM channel configured.

For example, if ATOM 0 and Channel 0 is used for WDT, the interrupts have to be configured as shown in the following image:

Wdg_17_Scu driver

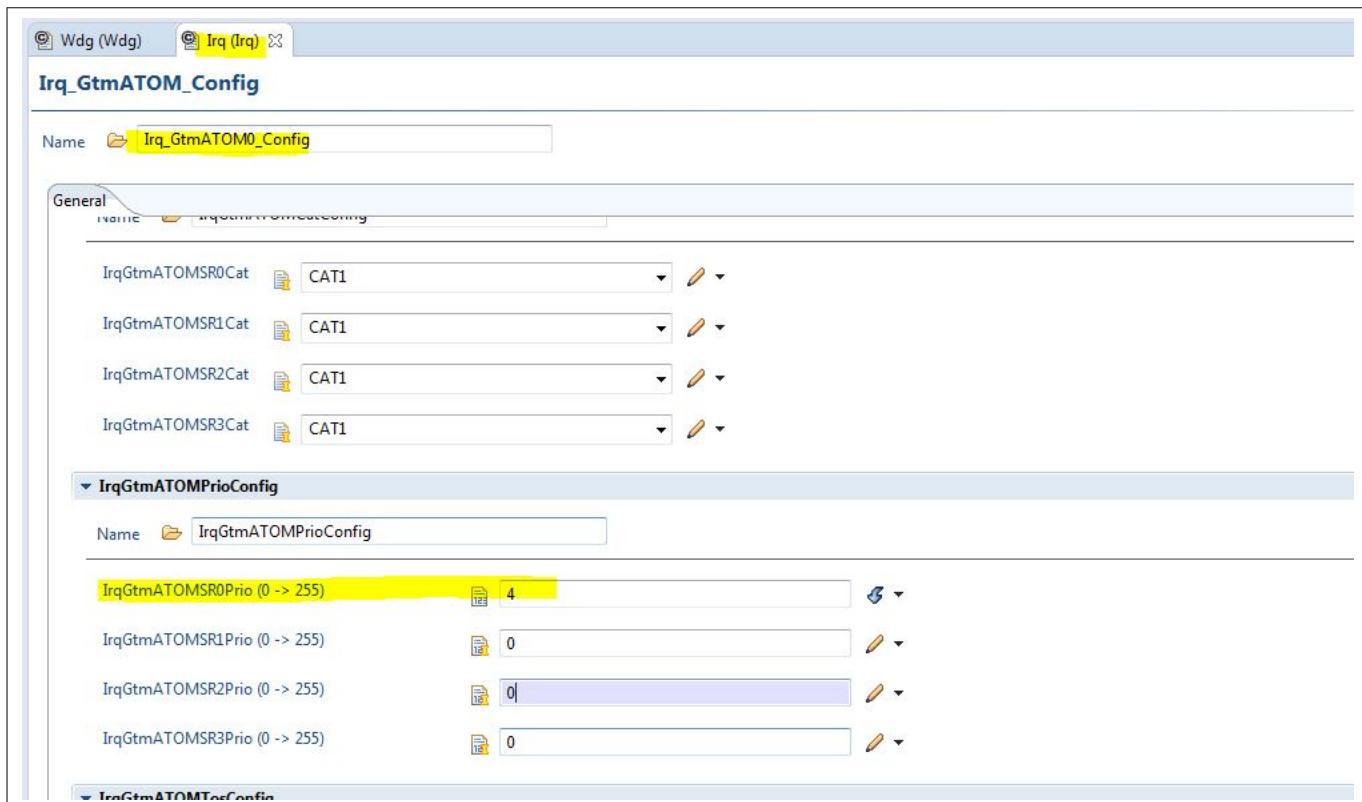


Figure 207 IRQ module configured for interrupt from ATOM 0 Channel 0 with priority number 4

Invoking the interrupt handlers provided by the driver must be done by the user. A sample invocation is shown as follows:

```

/* include MCU timer header file */
#include "Mcu_17_TimerIp.h"
*****SRC_ GTMTOM0SR0*****
ISR(GTMTOM0SR0_ISR)
{
/* Enable Global Interrupts */
ENABLE();
/* Parameter is Channel Number */
Mcu_17_Gtm_TomChannelIsr(0, 0);
}

*****SRC_ GTMATOM0SR0*****
ISR(GTMATOM0SR0_ISR)
{
/* Enable Global Interrupts */
ENABLE();
/* Parameter is Channel Number */
Mcu_17_Gtm_AtomChannelIsr(0, 0);
}

```

STM timer

If the STM timer is used, the priority number for the interrupt must be configured in the IRQ driver.

For example, if STM used for WDT, the interrupts have to be configured as shown in the following image:

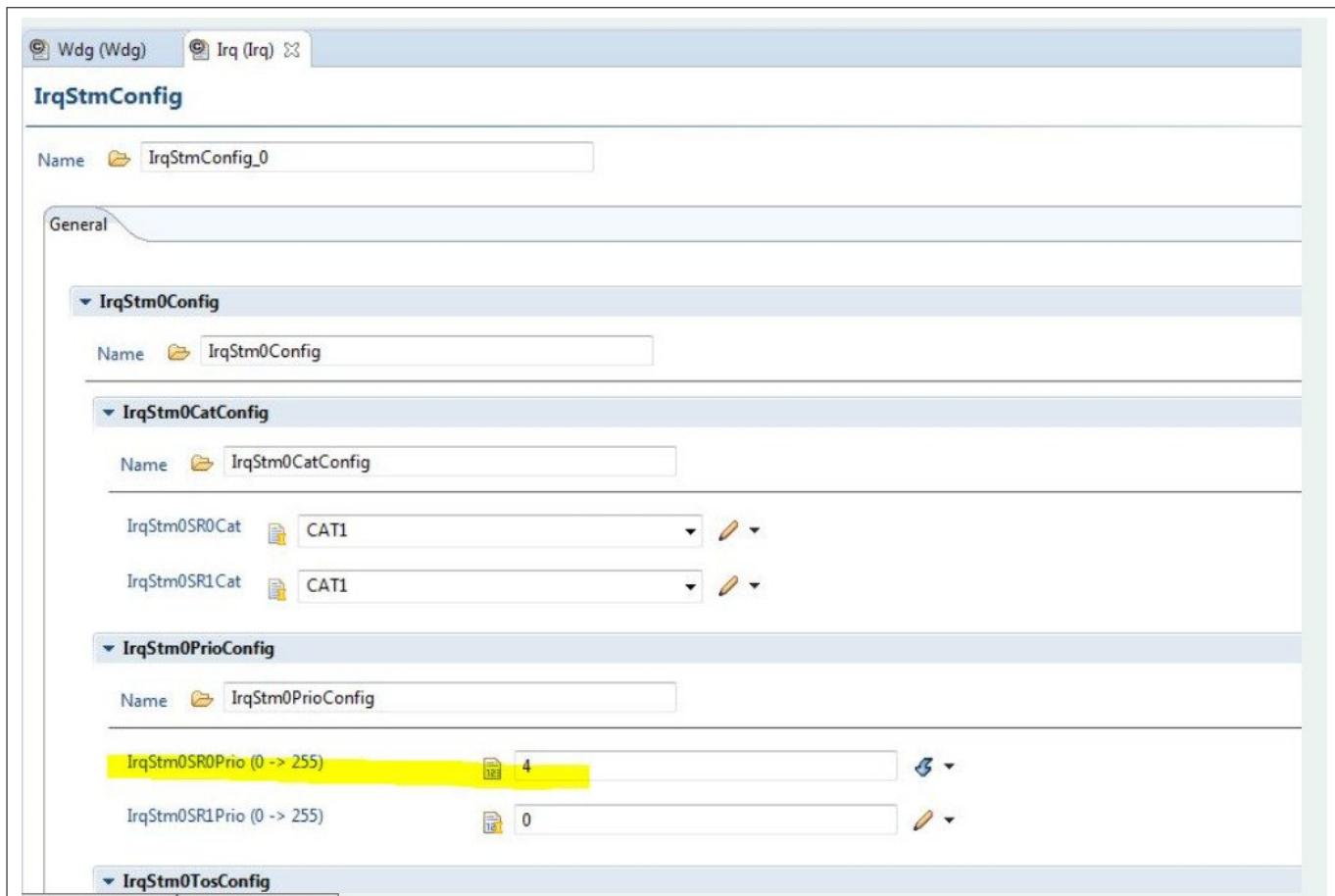
Wdg_17_Scu driver


Figure 208 IRQ module configured for interrupt from to STM0_CMP0 with priority number 4

Invoking the interrupt handlers provided by the driver must be done by the user. A sample invocation is shown as follows:

```

/* include MCU timer header file */
#include "Mcu_17_TimerIp.h"
*****SRC_ STM0SR0*****
ISR(STM0SR0_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    /* Call Interrupt function */
    Mcu_17_Stm_CompareMatchIsr(0U, 0U);
}
*****SRC_ STM1SR0*****
ISR(STM1SR0_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    /* Call Interrupt function */
    Mcu_17_Stm_CompareMatchIsr(1U, 0U);
}

```

Wdg_17_Scu driver

22.1.4.7 Example usage

Configuration

All the configuration parameters in the WDG driver mentioned in the Reference information section must be configured. If the GTM timer is used, the MCU must be configured with the TOM/ATOM channel and the corresponding IRQ must be configured with a priority number. If the STM timer is used, the MCU and Resource Manager must be configured with the corresponding STM compare register and the STM timer to the core, respectively. Also, the corresponding IRQ must be configured with a priority number.

Code must be generated without any errors.

Summary of WDG driver time related configuration parameters and their relationship		
Configuration Parameter	Range	Relationship (must be adhered to avoid code generation errors)
WdgCPUInitialTimeout	0.001s - 65.535s	Less Than WdgCPUMaxTimeout
WdgCPUMaxTimeout	0.001s - 65.535s	NA
WdgFastModeTimeoutValue	0.001s - 0.16777216s (fSPB = 100MHz, Clock divider = 256)	Less Than WdgSlowModeTimeoutValue. Greater Than WdgFastRefreshTime.
WdgSlowModeTimeoutValue	0.001s - 10.73741824s (fSPB = 100MHz, Clock divider = 16384)	Greater Than WdgSlowRefreshTime.
WdgFastRefreshTime	0.001s - WdgFastModeTimeoutValue	Less Than WdgFastModeTimeoutValue and WdgSlowRefreshTime
WdgSlowRefreshTime	0.001s - 10.73741824s	Greater Than WdgFastRefreshTime Less Than WdgSlowModeTimeoutValue

Figure 209 Summary of WDG driver time related configuration parameters and their relationship

Initialization of the WDG driver

Step 1. Include `Wdg_17_Scu.h` header file, to include extern definition of WDG driver configuration data structure.

Step 2. Include `Mcu.h` header file, To include extern definition of MCU driver configuration data structure.

Step 3. Initialize the MCU driver.

```

Mcu_Init(&Mcu_Config);

Mcu_InitClock(0); /* Initialize PLLs and Clocks by passing clock configuration
index */

while (Mcu_GetPllStatus() != MCU_PLL_LOCKED); /* wait until PLL is locked */

Mcu_DistributePllClock(); /* change clock source as PLL and ramp up/down to
configured clock frequencies */

```

Step 4. Initialize the IRQ driver and set the SRE bit.

Step 5. Initialize the WDG driver for the current core.

Note: SMU alarm group 8 can be configured to take an action based on the WDT alarms.

```
Wdg_17_Scu_Init(&Wdg_17_Scu_Config_0);
```

Normal operation of WDG driver

Step 1. After initializing MCU and WDG drivers, the mode of the WDG driver can be changed to FAST or SLOW.

Wdg_17_Scu driver

(Note: Mode of the WDG driver for a core can be modified to OFF mode only if WdgCPUDisableAllowed is enabled, else DET and DEM shall be raised)

```
Wdg_17_Scu_SetMode(WDGIF_FAST_MODE) ;
Wdg_17_Scu_SetMode(WDGIF_SLOW_MODE) ;
Wdg_17_Scu_SetMode(WDGIF_OFF_MODE); /*to turn OFF the WDG driver at run time
if WdgCPUDisableAllowed is enabled*/
```

Step 2. Call the `Wdg_17_Scu_SetTriggerCondition(timeout in ms)` API to service the WDG timer at every logical step.

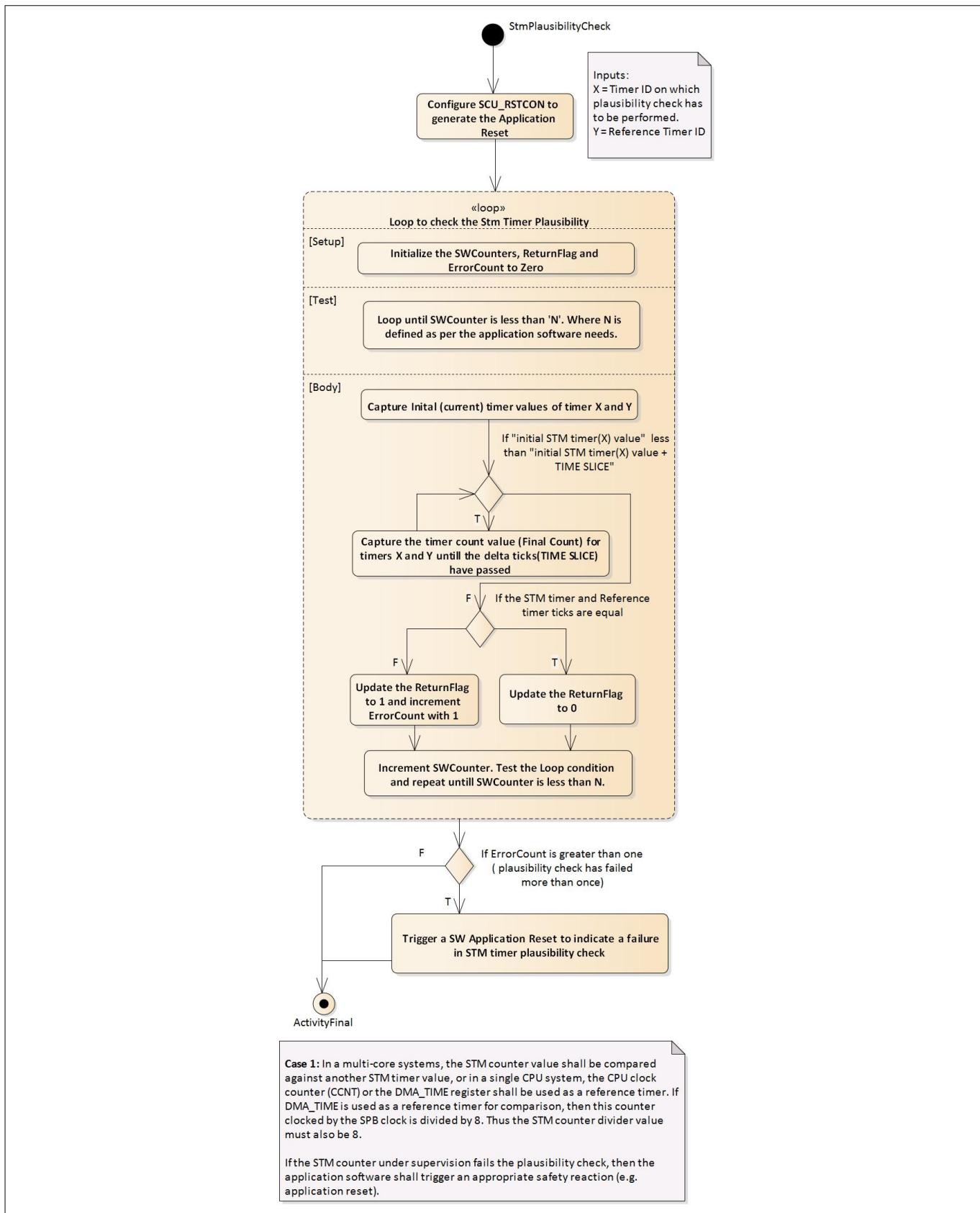
(Timeout passed as an input parameter to the API must be greater than the mode specific refresh time configured by the user. Otherwise, the trigger counter will be zero, WDT will not be serviced and this will lead to WDG alarm when the timer overflows.)

```
Wdg_17_Scu_SetTriggerCondition(1000); /*timeout in ms*/
```

Note : Once the WDG counter expires and if this API is not called then WDG timer will overflow. CPU will reset if SMU is configured to take the necessary action based on the WDG Alarm.

STM timer plausibility check

The application software shall perform a plausibility check of the STM counter value with a different timer/counter, before using it.

Wdg_17_Scu driver


Wdg_17_Scu driver**Figure 210 An example to perform STM timer plausibility check**

```

/* Casel: Example to implement STM plausibility check */

/*********************************************************************
** Example to implement STM plausibility check **
**   **
** Syntax : void Stm_PlausiblityCheck(const uint32 StmTimer, **
** const uint32 ReferenceTimer)  **
**   **
** Description : This function performs plausibility check of STM **
** counter by comparing with another STM timer.  **
**   **
** Parameters(in) : StmTimer : STM timer ID to on which plausibility **
** check has to be performed.  **
** ReferenceTimer :Reference STM Timer ID **
**   **
** Parameters (out) : none **
**   **
** Return value : 0: If the Plausibility Check is Successful **
** 1: If the Plausibility Check is Unsuccessful **
**   **
*****/uint8 Stm_PlausiblityCheck(const uint32 StmTimer, const uint32 ReferenceTimer)
{
    uint8 ReturnFlag = 0x1U;
    uint8 ErrorCount = 0x0U;
    SWCounter = 0x0U;

    /*Configure SCU RSTCON to generate Application reset*/
    Ifx_SCU_SWRSTCON SwResetReq;
    Ifx_SCU_RSTCON ResetConfig;
    ResetConfig.U = SCU_RSTCON.U;
    ResetConfig.B.SW = 2U;

    Mcal_WriteSafetyEndInitProtReg(&SCU_RSTCON.U, (uint32)ResetConfig.U);

    /*Loop N times. Until SWCounter is less than STM_N_CHECK_COUNT*/
    while( SWCounter < STM_N_CHECK_COUNT )
    {
        /*Capture the initial timer values*.
        X = Timer ID on which plausibility check has to be performed.
        Y = Reference Timer ID */
        X1 = StmTim0Info[StmTimer]->U;
        Y1 = StmTim0Info[ReferenceTimer]->U;
        /*Loop until the STM counter value has reached the final timer value*/

        while(StmTim0Info[StmTimer]->U < (X1 + STM_TIME_SLICE_COUNT))
        {
            /*Capture count till the final timer values is equal to
            STM_TIME_SLICE_COUNT + initial timer value */
            X2 = StmTim0Info[StmTimer]->U;

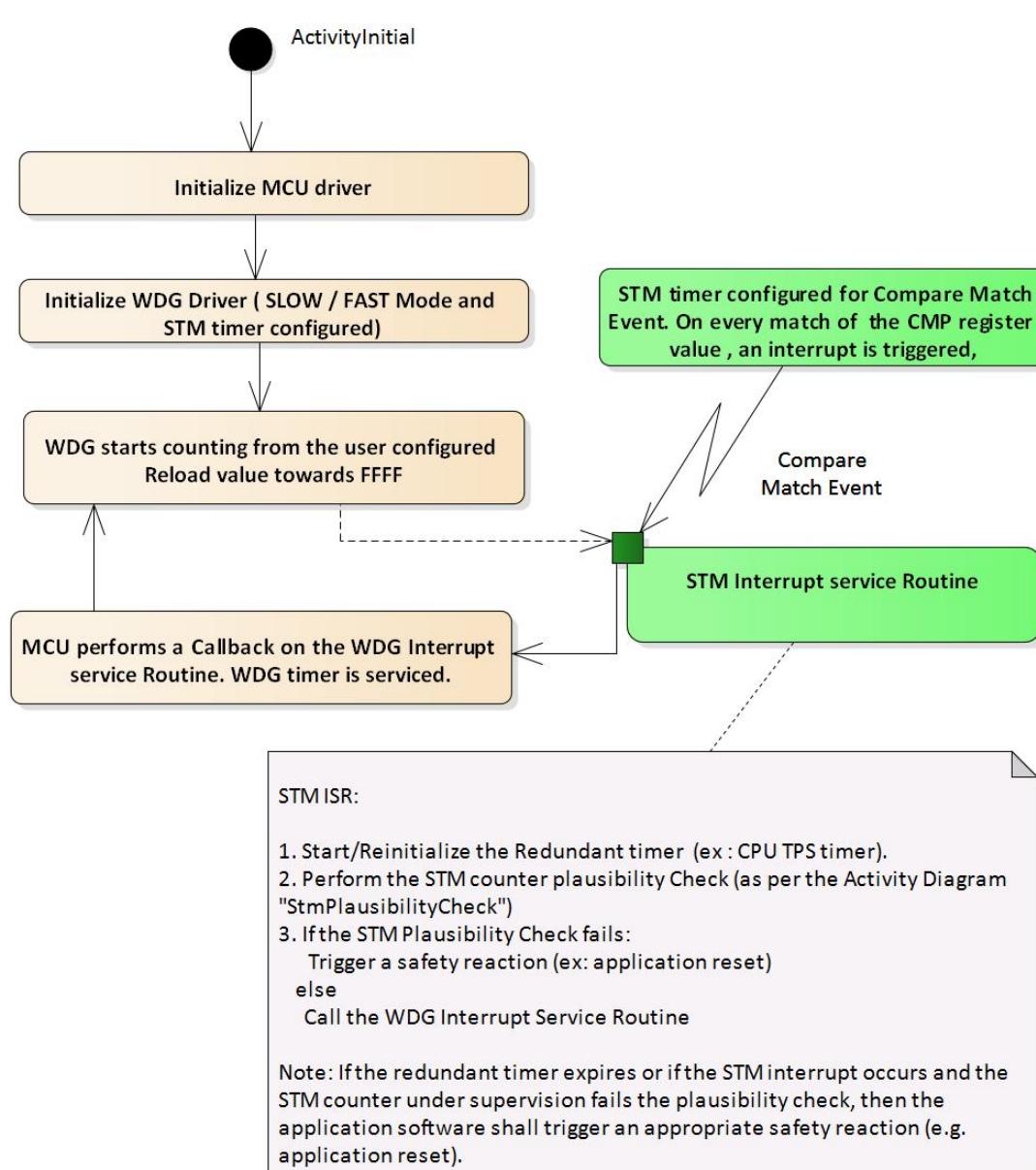
```

Wdg_17_Scu driver

```
Y2 = StmTim0Info[ReferenceTimer]->U;
}

/*If the STM timer and Reference timer ticks are equal*/
if((uint32)(X1-X2) == (uint32)(Y1-Y2))
{
ReturnFlag = 0x00U; /*Plausibility Check is successful*/
}
else
{
ReturnFlag = 0x01U; /*Plausibility Check is unsuccessful*/
ErrorCount+=0x01U; /* Increment the Error Count */
}
SWCounter++;
}

/*If plausibility check has failed more than once.Trigger an
application reset*/
if(ErrorCount > 0x1U)
{
/*Trigger a SW Application Reset to indicate a consistent
failure in STM timer value check*/
SwResetReq.U = SCU_SWRSTCON.U;
SwResetReq.B.SWRSTREQ = 0x1U;
Mcal_WritePeripEndInitProtReg(&SCU_SWRSTCON.U, (uint32)SwResetReq.U);
}
return ReturnFlag;
}
```

Wdg_17_Scu driver


Case 2: In case the STM is used as an ISR trigger, application software shall configure a redundant timer source (ex: the CPU TPS timer) and reinitialize it within the interrupt service routine of the STM interrupt.

In addition, within each STM interrupt service, the plausibility of the counter value shall be checked as described in Case 1 (Activity Diagram 'StmPlausibilityCheck').

Note: When WDG driver is configured to use STM timer for its' timing needs then, STM timer generates an interrupt to service the watchdog timer. Since the interrupt frame (ISR) is under users' responsibility, the user can check the plausibility of the STM timer and only then call the WDG interrupt handler.

Figure 211

Activity diagram depicting an example to perform STM timer plausibility check and ensure interrupt triggering time if in case STM timer is used as an ISR trigger

Wdg_17_Scu driver

22.1.5 Key architectural considerations

22.1.5.1 WDG driver states

The state of the WDG driver will be updated to WDG_UNINIT or WDG_BUSY or WDG_IDLE irrespective of whether the DET or safety error check is enabled or not. The driver state check in the APIs will prevent any unintended SFR or global update.

22.1.5.2 WDG driver critical section

The SCU_WDTCPUxCON0 (x is core Id) register is accessed by both MCALLIB and WDG (MCALLIB driver ensures that the REL bits of this register is not hampered and it writes back with the correct values once it exits the timeout mode). MCALLIB uses this register in the hardware timeout mode and WDG uses it in the hardware normal mode. Thus MCALLIB and WDG use the critical section with the same exclusive area (CpuEndinit).

22.1.5.3 WDG driver timer configuration and services

Every WDT configured for a core must be accompanied with an STM timer/GTM timer resource. All STM and GTM related registers will be configured by invoking MCU driver APIs.

If STM timer is used, the timer is allocated to the core using the Resource manager and the compare register is allocated using the MCU driver. The STM compare register used for WDT needs must be configured in the MCU driver. STM compare match interrupt from CMP0 will be routed through IR0 and CMP1 will be routed through IR1.

If GTM timer is used, the TOM/ATOM channel is allocated in the MCU driver and the respective clock is configured in the WDG driver. All the STM/GTM related registers required by the WDG driver are updated using services provided by the MCU driver. The source of the interrupt will be CMP0 (compare match interrupt) and not CMP1 (period match interrupt). The WDG driver mode specific reload values (SLOW/FAST) provided in the configuration is the timeout in seconds used by the WDG driver and the mode specific refresh (SLOW/FAST) values are the GTM/STM timer callback periods.

22.1.5.4 WDG driver timer dependency

Among the available timers, the GTM or STM timer is chosen to trigger the WDT. The GTM and STM have sufficient number of timers to support WDT on multiple cores. The GPT12 and CC6 timers are not considered due to the unavailability of the required number of timers and due to specific use cases of these timers. In an application, it could be possible that one STM timer is allocated per CPU. The same timer can be used by CPU WDT and is also accessed by the OS. Since the STM timer hardware provides two timer interrupts (CMP0, CMP1), one interrupt could be used by OS and the other by WDT.

22.1.5.5 WDG driver multicore error reporting

When the WDG initialization is done for a wrong core, the WDG_E_PARAM_CONFIG development or safety error is reported.

22.1.5.6 WDG driver password access

The internal WDG hardware shall be accessed in the hardware normal mode by the WDG driver with a static password, which is configured by the user. The entire password access sequence is handled by the WDG driver. The WDG hardware supports both static password and automatic password sequence mechanism. However, according to the AUTOSAR 4.2.2, the program flow monitoring shall be fulfilled by the upper layer (WDG manager) where automatic password sequencing mechanism could be used. Even though the password is configured by the user at runtime, the password to unlock the watchdog timer may be changed by the user or

Wdg_17_Scu driver

by another driver. Hence, the password is always recalculated before SCU_WDTCPUxCON0 (x = core Id) is accessed.

22.1.5.7 WDG driver trigger

The timeout passed to the `Wdg_SetTriggerCondition` API must be greater than the mode (SLOW/FAST) specific refresh time (configured by the user), to ensure that the WDT is serviced until the trigger counter value becomes zero. (If the mode is OFF, the WDT will not be serviced). If the timeout is less than the refresh time then the trigger counter will be zero and WDT will not be serviced and this will lead to WDG alarm when the timer overflows.

22.1.5.8 WDG driver code generator dependency

The WDG driver is dependent on MCU, MCALLIB and Resource Manager for successful code generation. In case these modules are not added to the configuration project, an error is raised.

22.1.5.9 WDG driver variation point

The Post-Build-Variant value of the parameters: `WdgInitialTimeout`, `WdgCPUInitialTimeout`, `WdgMaxTimeout` and `WdgCPUMaxTimeout` is set as TRUE since the values may vary across different variants configured across multiple cores.

22.1.5.10 STM timer plausibility check

If the WDG driver uses the STM timer for time-based triggers, the STM timer will generate interrupts to service the WDT as per the configured mode-specific refresh time (`WdgFastRefreshTime` and `WdgSlowRefreshTime`). The user shall check the plausibility of the STM timer and only if the plausibility check is successful, the `Wdg_17_Scu_Isr` API shall be invoked.

Wdg_17_Scu driver

22.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Critical section used while accessing CON0 register**

User shall ensure that the core-specific interrupts are disabled in the following critical sections: -
 SchM_Enter_Wdg_17_Scu_CpuEndInit and -SchM_Enter_McalLib_CpuEndInit.

[cover parentID WDG={51D8C43C-2314-4b4d-BF4C-263237F16EAB}]

- **GTM channel allocation to WDG timer**

The user shall ensure the following while allocating GTM channels to a particular core-specific WDG timer. The GTM channels sharing the same interrupt node shall be allocated to the same core in the IRQ driver. For example, if GTM channel x is allocated to the WDG driver and channel x+1 is allocated to the PWM driver, both the modules shall be in the same core since x and x+1 share interrupt node.

[cover parentID WDG={B578A86A-6687-4294-B6A7-A0B824FCE690}]

- **GTM channels sharing same interrupt nodes**

User shall ensure that timer channels of the GTM, which share the same interrupt nodes, shall not be configured for different CPU WDG timers.

[cover parentID WDG={49FDDBDF3-10F6-4ba2-8F8E-06D92DC37EA4}]

- **WDG driver initialization with the correct configuration pointer**

The user shall ensure that the WDG initialization is done with the correct configuration pointer. The Wdg_17_Scu_InitCheck() API shall be called after the WDG driver initialization and before starting any WDG driver functionality.

[cover parentID WDG={46AC4880-0030-4f96-85AA-465741024800}]

- **WDG overflow**

The user shall ensure that the watchdog overflow shall trigger an SMU alarm and enter into a safe state. However, the overflow error flag will be checked before servicing the WDT and a safety error will be reported to the upper layer.

[cover parentID WDG={CBD32342-5082-4356-9803-61385D57337E}]

Wdg_17_Scu driver

22.3 Reference information

22.3.1 Configuration interfaces

Wdg_17_Scu driver

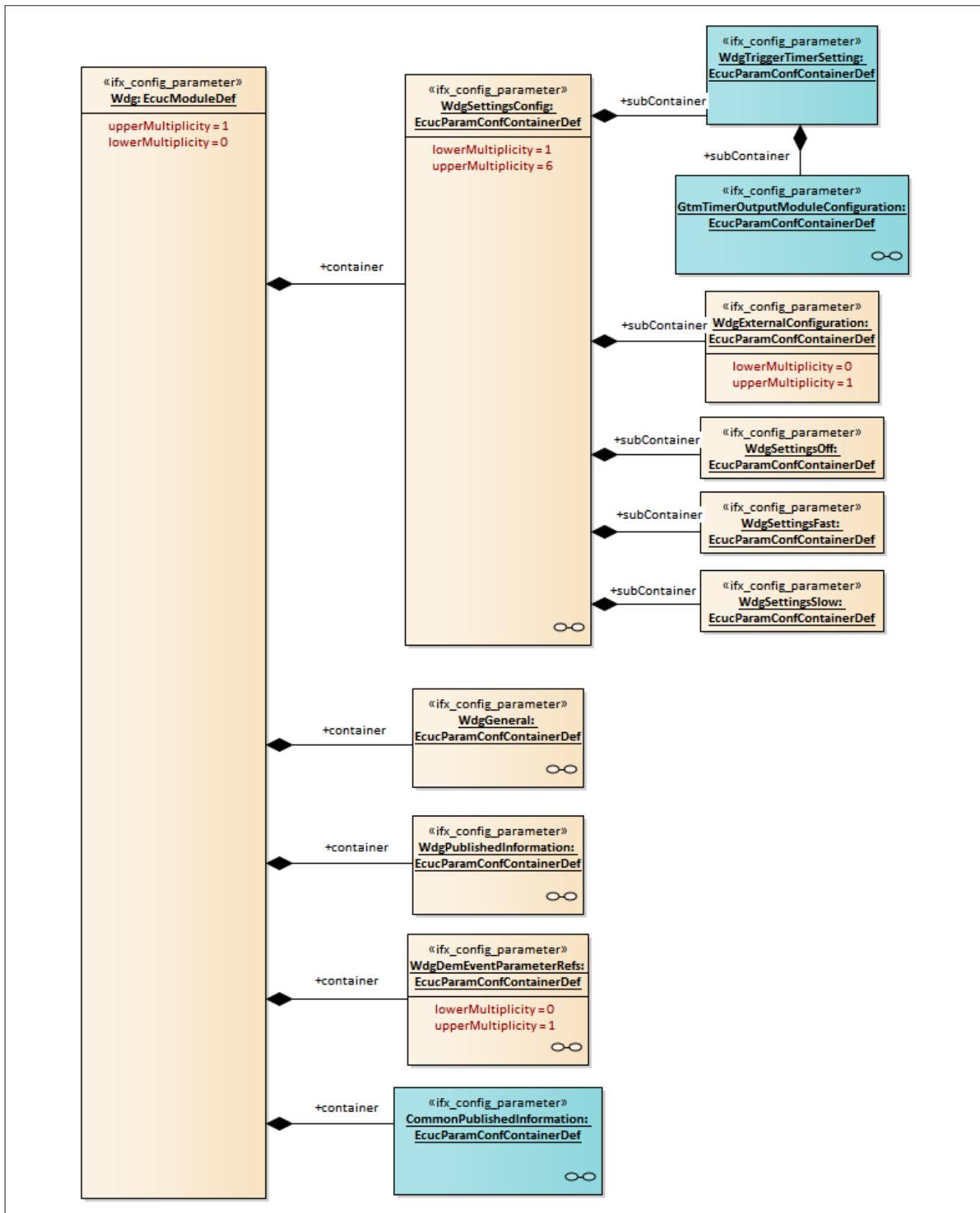


Figure 212 Container hierarchy along with their configuration parameters

Wdg_17_Scu driver

22.3.1.1 Container: GtmTimerConfiguration

This container contains the configuration elements for configuring GTM timer hardware.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Pre-Compile

22.3.1.2 Container: WdgSettingsConfig

This is a List of containers of Configuration items for the different watchdog settings. Note: All postbuild parameters are handled via this container.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

22.3.1.3 Container: CommonPublishedInformation

Container for common published information. (Based on BSW General)

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

22.3.1.3.1 ArMajorVersion

Table 1877 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	This parameter provides the major version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.3.2 ArMinorVersion

Table 1878 Specification for ArMinorVersion

Name	ArMinorVersion		
Description	This parameter provides the minor version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	2		

Wdg_17_Scu driver
Table 1878 Specification for ArMinorVersion (continued)

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.3.3 ArPatchVersion
Table 1879 Specification for ArPatchVersion

Name	ArPatchVersion		
Description	This parameter provides the patch version of the AUTOSAR Specification.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		
Default value	2		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.3.4 ModuleId
Table 1880 Specification for ModuleId

Name	ModuleId		
Description	This parameter provides the Module Id.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 65535		
Default value	102		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

Wdg_17_Scu driver**Table 1880 Specification for ModuleId (continued)**

Dependency	-
-------------------	---

22.3.1.3.5 Release**Table 1881 Specification for Release**

Name	Release		
Description	Aurix2G derivative used for the implementation Default value will depend on the derivative chosen and will be read from the derivative .properties file.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	Depends on the Derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.3.6 SwMajorVersion**Table 1882 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	This parameter provides the major version of the Software.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Wdg_17_Scu driver**22.3.1.3.7 SwMinorVersion****Table 1883 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	This parameter provides the minor version of the Software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.3.8 SwPatchVersion**Table 1884 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	This parameter provides the patch version of the Software.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.3.9 VendorApiInfix**Table 1885 Specification for VendorApiInfix**

Name	VendorApiInfix		
Description	In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the Vendordf and a vendor specific name.		

Wdg_17_Scu driver
Table 1885 Specification for VendorApiInfix (continued)

	<p>This parameter is used to specify the vendor specific name. For example, assuming that the VendorId of the implementer is 17 and the implementer chooses VendorApiInfix as Scu then the API name Wdg_SetMode defined in the SWS will translate to Wdg_17_Scu_SetMode.</p> <p>This parameter is mandatory for all modules with upper multiplicity greater than 1. It will not be used for modules with upper multiplicity equal to 1</p>		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	Scu		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.3.10 VendorId
Table 1886 Specification for VendorId

Name	VendorId		
Description	This parameter provides the Vendor Id		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.4 Container: GtmTimerOutputModuleConfiguration

This container contains the elements for configuring GTM timer hardware (TOM/ATOM). The settings here are used to configure the timing needs for GTM timer. Availability of this container depends on the underlying derivative.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

Wdg_17_Scu driver**22.3.1.4.1 GtmTimerClockSelect****Table 1887 Specification for GtmTimerClockSelect**

Name	GtmTimerClockSelect		
Description	<p>This parameter decides the Clock Source for TOM/ATOM timer.</p> <p>Values:</p> <p>If GtmTimerUsed is TOM module.</p> <ul style="list-style-type: none"> GTM_FIXED_CLOCK_0: Fixed Clock 0 is selected. GTM_FIXED_CLOCK_1: Fixed Clock 1 is selected. GTM_FIXED_CLOCK_2: Fixed Clock 2 is selected. GTM_FIXED_CLOCK_3: Fixed Clock 3 is selected. GTM_FIXED_CLOCK_4: Fixed Clock 4 is selected. <p>If GtmTimerUsed is ATOM module.</p> <ul style="list-style-type: none"> GTM_CONFIGURABLE_CLOCK_0: Configurable Clock0 is selected. GTM_CONFIGURABLE_CLOCK_1: Configurable Clock1 is selected. GTM_CONFIGURABLE_CLOCK_2: Configurable Clock2 is selected. GTM_CONFIGURABLE_CLOCK_3: Configurable Clock3 is selected. GTM_CONFIGURABLE_CLOCK_4: Configurable Clock4 is selected. GTM_CONFIGURABLE_CLOCK_5: Configurable Clock5 is selected. GTM_CONFIGURABLE_CLOCK_6: Configurable Clock6 is selected. GTM_CONFIGURABLE_CLOCK_7: Configurable Clock7 is selected. <p>Minimum CLOCK ID is selected as the default value.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>GTM_CONFIGURABLE_CLOCK_0: Configurable Clock 0 will be supplied to the Channel</p> <p>GTM_CONFIGURABLE_CLOCK_1: Configurable Clock 1 will be supplied to the Channel</p> <p>GTM_CONFIGURABLE_CLOCK_2: Configurable Clock 2 will be supplied to the Channel</p> <p>GTM_CONFIGURABLE_CLOCK_3: Configurable Clock 3 will be supplied to the Channel</p> <p>GTM_CONFIGURABLE_CLOCK_4: Configurable Clock 4 will be supplied to the Channel</p> <p>GTM_CONFIGURABLE_CLOCK_5: Configurable Clock 5 will be supplied to the Channel</p> <p>GTM_CONFIGURABLE_CLOCK_6: Configurable Clock 6 will be supplied to the Channel</p> <p>GTM_CONFIGURABLE_CLOCK_7: Configurable Clock 7 will be supplied to the Channel</p> <p>GTM_FIXED_CLOCK_0: Fixed Clock 0 will be supplied to the Channel</p> <p>GTM_FIXED_CLOCK_1: Fixed Clock 1 will be supplied to the Channel</p> <p>GTM_FIXED_CLOCK_2: Fixed Clock 2 will be supplied to the Channel</p> <p>GTM_FIXED_CLOCK_3: Fixed Clock 3 will be supplied to the Channel</p> <p>GTM_FIXED_CLOCK_4: Fixed Clock 4 will be supplied to the Channel</p>		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Wdg_17_Scu driver
Table 1887 Specification for GtmTimerClockSelect (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	GtmTimerUsed		

22.3.1.4.2 GtmTimerUsed
Table 1888 Specification for GtmTimerUsed

Name	GtmTimerUsed		
Description	<p>The TOM/ATOM Channel resource assigned to the watchdog timer.</p> <p>This parameter is list of all the GTM timer channels (TOM/ATOM) used by WDG Driver.</p> <p>Referred timer channel in MCU should have TomChannelUsage/ AtomChannelUsage as USED_BY_WDG_DRIVER.</p> <p>Note: If referred timer channel is not marked as "GTM_TOM_CHANNEL_USED_BY_WDG/ GTM_ATOM_CHANNEL_USED_BY_WDG" via the parameters McuGtmTomChannelAllocationConf / McuGtmAtomChannelAllocationConf in MCU module an error message will be raised.</p> <p>The TOM/ATOM channel should be unreserved in MCU, if WdgTriggerTimerSelection is changed to STM_TIMER or in MCU module an error message will be raised.</p> <p>Since the name of the dependent container is user configurable, the default value is kept as NULL.</p>		
Multiplicity	1..1	Type	EcuReferenceDef
Range	Reference to Node: McuGtmTomChannelAllocationConf, McuGtmAtomChannelAllocationConf		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.5 Container: Wdg

Configuration of the Wdg module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

Wdg_17_Scu driver
22.3.1.5.1 IMPLEMENTATION_CONFIG_VARIANT
Table 1889 Specification for IMPLEMENTATION_CONFIG_VARIANT

Name	IMPLEMENTATION_CONFIG_VARIANT		
Description	WDG driver will support the configuration variant VARIANT-POST-BUILD hence this is the default value. Note : Alias name for this parameter is Config Variant.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VariantPostBuild:		
Default value	VariantPostBuild		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.6 Container: WdgDemEventParameterRefs

This container lists down all the Production error event configuration parameters for WDG driver. If this container does not exist in the configuration, then DEM handling is not performed.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

22.3.1.6.1 WDG_E_DISABLE_REJECTED
Table 1890 Specification for WDG_E_DISABLE_REJECTED

Name	WDG_E_DISABLE_REJECTED		
Description	<p>The existence of this parameter decides if the WDG module would raise this particular dem.</p> <p>The user has to point to the right Dem event parameter for this fault.The short name of the pointed Dem parameter will be used as the symbol that will be passed as the first parameter to Dem_ReportEventStatus() function.</p> <p>The extended production error WDG_17_SCU_E_DISABLE_REJECTED will be reported with FAILED when disabling of the watchdog mode failed and will be reported with PASSED when disabling of the watchdog mode not failed.</p> <p>The minimum value of multiplicity depends on the WdgSafetyEnable. (if WdgSafetyEnable is enabled then multiplicity is 1..1 and if disabled then multiplicity is 0..1)</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef

Wdg_17_Scu driver
Table 1890 Specification for WDG_E_DISABLE_REJECTED (continued)

Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

22.3.1.6.2 WDG_E_MODE_FAILED
Table 1891 Specification for WDG_E_MODE_FAILED

Name	WDG_E_MODE_FAILED		
Description	<p>The existence of this parameter decides if the WDG module would raise this particular dem.</p> <p>The user has to point to the right Dem event parameter for this fault. The short name of the pointed Dem parameter will be used as the symbol that will be passed as the first parameter to Dem_ReportEventStatus() function.</p> <p>Reference to DemEventParameter.</p> <p>The extended production error WDG_17_SCU_E_MODE_FAILED will be reported with FAILED when setting of the watchdog mode failed and will be reported with PASSED when setting of the watchdog mode not failed.</p> <p>The minimum value of multiplicity depends on the WdgSafetyEnable. (if WdgSafetyEnable is enabled then multiplicity is 1..1 and if disabled then multiplicity is 0..1)</p>		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE
Value configuration class	Pre-Compile	Multiplicity configuration class	Pre-Compile
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Wdg_17_Scu driver

22.3.1.7 Container: WdgExternalConfiguration

This container contains the configuration items for external watchdog hardware. This container and its parameters are not used. The default container and parameter will be retained from AUTOSAR and disabled. (This configuration is not provided as the external watchdog is not supported by Infineon.)

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Pre-Compile

22.3.1.8 Container: WdgGeneral

All general parameters of the watchdog driver are collected here.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

22.3.1.8.1 WdgDevErrorDetect

Table 1892 Specification for WdgDevErrorDetect

Name	WdgDevErrorDetect		
Description	Switches the default error detection and notification ON or OFF. True : enabled (ON). False: disabled (OFF).		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

22.3.1.8.2 WdgDisableAllowed

Table 1893 Specification for WdgDisableAllowed

Name	WdgDisableAllowed		
Description	This parameter is disabled. Instead, WdgCPUDisableAllowed is provided to enable/disable CPU specific WDG for every CPU WDG configured. Compile switch to allow / forbid disabling the watchdog driver during runtime. True: Disabling the WDG driver at runtime is allowed.		

Wdg_17_Scu driver**Table 1893 Specification for WdgDisableAllowed (continued)**

	False: Disabling the WDG driver at runtime is not allowed.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

22.3.1.8.3 WdgIndex**Table 1894 Specification for WdgIndex**

Name	WdgIndex		
Description	Wdg driver ID which can be used by Wdg Interface as a reference. The default value can be modified by the user as per the needs of the application.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

22.3.1.8.4 WdgInitApiMode**Table 1895 Specification for WdgInitApiMode**

Name	WdgInitApiMode
Description	MCAL_SUPERVISOR: Wdg Init API will run in Supervisor mode. MCAL_USER1: Wdg Init API will run in USER1 mode.

Wdg_17_Scu driver
Table 1895 Specification for WdgInitApiMode (continued)

	Since WDG driver accesses the SFRs, it is more efficient to operate the WDG driver in supervisor mode. Hence, the default mode of operation is supervisor.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	WDG_MCAL_SUPERVISOR: APIs runs in Supervisor mode. WDG_MCAL_USER1: APIs runs in USER1 mode.		
Default value	WDG_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.8.5 WdgInitCheckApi
Table 1896 Specification for WdgInitCheckApi

Name	WdgInitCheckApi		
Description	Switches the InitCheck API ON or OFF. True : enabled (ON). False: disabled (OFF). The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

Wdg_17_Scu driver**22.3.1.8.6 WdgInitialTimeout****Table 1897 Specification for WdgInitialTimeout**

Name	WdgInitialTimeout		
Description	<p>This parameter is disabled.</p> <p>Instead, WdgCPUInitialTimeout is provided as initial timeout for every CPU WDG configured. This initial timeout (sec) for the trigger condition to be initialized during Init function. It will not be larger than WdgMaxTimeout.</p> <p>The default value is retained same as the WdgCPUInitialTimeout parameter. However, this parameter is disabled and not used by the WDG driver for any of its functionality.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.001s - 65.535s		
Default value	5s		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	WdgMaxTimeout		

22.3.1.8.7 WdgMaxTimeout**Table 1898 Specification for WdgMaxTimeout**

Name	WdgMaxTimeout		
Description	<p>This parameter is disabled.</p> <p>Instead, WdgCPUMaxTimeout is provided as maximum CPU specific WDG timeout for every CPU WDG configured. This is the maximum window period of the Watchdog timer.</p> <p>The default value is retained same as the WdgCPUMaxTimeout parameter. However, this parameter is disabled and not used by the WDG driver for any of its functionality.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.001s - 65.535s		
Default value	32s		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

Wdg_17_Scu driver**22.3.1.8.8 WdgRunArea****Table 1899 Specification for WdgRunArea**

Name	WdgRunArea		
Description	<p>This feature is not provided by Infineon and is disabled .</p> <p>The visibility is false for this parameter.</p> <p>Represents the WDG driver execution area is either from ROM(Flash) or RAM as required with the particular microcontroller.</p> <p>The default value is ROM. However, this parameter is disabled and not used by the WDG driver for any of its functionality.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>RAM: This feature is not provided by Infineon.</p> <p>This visibility is false. WDG driver to be executed out of RAM area</p> <p>ROM: This feature is not provided by Infineon.</p> <p>This parameter is disabled. WDG driver to be executed out of ROM area.</p>		
Default value	ROM		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

22.3.1.8.9 WdgRuntimeApiMode**Table 1900 Specification for WdgRuntimeApiMode**

Name	WdgRuntimeApiMode		
Description	<p>MCAL_SUPERVISOR: Wdg runtime APIs will run in Supervisor mode.</p> <p>MCAL_USER1: Wdg runtime APIs will run in USER1 mode.</p> <p>Since WDG driver accesses the SFRs, it is more efficient to operate the WDG driver in supervisor mode. Hence, the default mode of operation is supervisor.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>WDG_MCAL_SUPERVISOR: APIs runs in Supervisor mode.</p> <p>WDG_MCAL_USER1: APIs runs in USER1 mode.</p>		
Default value	WDG_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Wdg_17_Scu driver**Table 1900 Specification for WdgRuntimeApiMode (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.8.10 WdgSafetyEnable**Table 1901 Specification for WdgSafetyEnable**

Name	WdgSafetyEnable		
Description	Switch to enable reporting of safety DETs. True : enabled (ON). False: disabled (OFF). Note: when this switch is enabled AUTOSAR DETs are enabled by default. The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	WDG_E_MODE_FAILED, WDG_E_DISABLE_REJECTED, WdgDemEventParameterRefs		

22.3.1.8.11 WdgTriggerLocation**Table 1902 Specification for WdgTriggerLocation**

Name	WdgTriggerLocation
Description	This feature is not provided by Infineon and is disabled . The visibility is false for this parameter. Location (memory address) of the watchdog trigger routine. The default value is NULL_PTR. However, this parameter is disabled and not used by the WDG driver for any of its functionality.

Wdg_17_Scu driver**Table 1902 Specification for WdgTriggerLocation (continued)**

Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

22.3.1.8.12 WdgTriggerTimerSelection**Table 1903 Specification for WdgTriggerTimerSelection**

Name	WdgTriggerTimerSelection		
Description	Hardware timer used to service WDG during window period. It is common for all WDTs. GTM_TIMER (Used to service WDTs.) STM_TIMER (Used to service WDTs.) The default value can be modified by the user as per the needs of the application. Note: GTM_TIMER will not be available for selection in devices without GTM.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	GTM_TIMER: GTM timer is used to service WDTs, in devices with GTM. STM_TIMER: STM timer is used to service WDTs.		
Default value	STM_TIMER		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.8.13 WdgVersionInfoApi**Table 1904 Specification for WdgVersionInfoApi**

Name	WdgVersionInfoApi
-------------	-------------------

Wdg_17_Scu driver
Table 1904 Specification for WdgVersionInfoApi (continued)

Description	Compile switch to enable / disable the version information API. True : API enabled False: API disabled The optional APIs are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

22.3.1.9 Container: WdgPublishedInformation

Container holding all WDG specific published information parameters

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

22.3.1.9.1 WdgTriggerMode

Table 1905 Specification for WdgTriggerMode

Name	WdgTriggerMode		
Description	Watchdog trigger mode (toggle/window/both) The default value is WDG_WINDOW since this the trigger mode supported in the WDG driver. This value is not editable.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	WDG_BOTH: Watchdog trigger mode is both Window and Toggle. WDG_TOGGLE: Watchdog trigger mode is Toggle. WDG_WINDOW: Watchdog trigger mode is Window.		
Default value	WDG_WINDOW		
Post-build variant value	FALSE	Post-build variant multiplicity	-

Wdg_17_Scu driver
Table 1905 Specification for WdgTriggerMode (continued)

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	-		

22.3.1.10 Container: WdgSettingsConfig

This is available under WDG main Container. This container configures exactly one watchdog timer per CPU.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

22.3.1.10.1 WdgCPUDisableAllowed
Table 1906 Specification for WdgCPUDisableAllowed

Name	WdgCPUDisableAllowed		
Description	Compile switch to allow/forbid the Watchdog Driver change mode to WDGIF_OFF_MODE at runtime for the CPU WDT selected by container. This parameter is considered only if the dependent parameter WdgCPUDisableAllowed is True. This parameter is defined by Infineon. The optional features are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	WdgDefaultMode		

22.3.1.10.2 WdgCPUInitialPassowrd
Table 1907 Specification for WdgCPUInitialPassowrd

Name	WdgCPUInitialPassowrd
Description	Initial password for the password access of the CPU WDG. The default password after Application Reset is 00000000111100B .

Wdg_17_Scu driver
Table 1907 Specification for WdgCPUInitialPassowrd (continued)

	The default value is same as the application reset value but can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 16383		
Default value	60		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.10.3 WdgCPUInitialTimeout
Table 1908 Specification for WdgCPUInitialTimeout

Name	WdgCPUInitialTimeout		
Description	<p>This is the initial window period that is active as soon Wdg_17_Scu_Init is called for the core. It is used to calculate the value of the trigger counter which is used to service the WDT just after initialization.</p> <p>This parameter is defined by Infineon.</p> <p>The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.001s - 65.535s		
Default value	5s		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	WdgCPUMaxTimeout		

22.3.1.10.4 WdgCPUMaxTimeout
Table 1909 Specification for WdgCPUMaxTimeout

Name	WdgCPUMaxTimeout
-------------	------------------

Wdg_17_Scu driver
Table 1909 Specification for WdgCPUMaxTimeout (continued)

Description	This is the maximum window period for the core specific watchdog timer. It is considered as the maximum window period for this particular core. This parameter is defined by Infineon. The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.001s - 65.535s		
Default value	32s		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.10.5 WdgCoreId
Table 1910 Specification for WdgCoreId

Name	WdgCoreId		
Description	This parameter refers to CORE ID to which the watchdog settings belong. This parameter is defined by Infineon. CPU Watchdog available in each device [Range]: - TC39x = 0-5, - TC38x = 0-3, - TC37x, TC35x = 0-2, - TC36x, TC33x(ED\ADAS) = 0-1, - TC33x = 0. The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - depends on the device		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

Wdg_17_Scu driver
Table 1910 Specification for WdgCoreId (continued)

Dependency	-
-------------------	---

22.3.1.10.6 WdgDefaultMode
Table 1911 Specification for WdgDefaultMode

Name	WdgDefaultMode		
Description	<p>Default mode of WDG Driver initialization for WDTx. The Description on each Wdg Mode is given below.</p> <ul style="list-style-type: none"> - Off-Mode: The watchdog hardware is disabled / shut down. - Slow-Mode: Triggering the watchdog hardware can be done with a long timeout period. This mode for example can be used during system startup / initialization phase. - Fast-Mode: Triggering the watchdog hardware has to be done with a short timeout period. This mode for example can be used during normal operations of the ECU. <p>ImplementationType: WdgIf_ModeType</p> <p>The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>WDGIF_FAST_MODE: Default watchdog mode is "fast"</p> <p>WDGIF_OFF_MODE: Default watchdog mode is "Off"</p> <p>Dependency : This mode can be set if the WdgDisableAllowed is True and WdgCPUDisableAllowed is True</p> <p>WDGIF_SLOW_MODE: Default watchdog mode is "slow"</p>		
Default value	WDGIF_SLOW_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	AUTOSAR_ECUC	Scope	LOCAL
Dependency	WdgDisableAllowed		

22.3.1.10.7 WdgSystemClockRef
Table 1912 Specification for WdgSystemClockRef

Name	WdgSystemClockRef
Description	<p>This parameter refers to the system clock configured, It will refer to a valid McuClockSetting Configuration.</p> <p>Note: WDG module will work correctly only for this system clock setting. No changes should be done to the system clock if WDG is to be kept functioning.</p>

Wdg_17_Scu driver
Table 1912 Specification for WdgSystemClockRef (continued)

	Since the name of the dependent container is user configurable, the default value is kept as NULL.		
Multiplicity	1..1	Type	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	McuClockReferencePoint		

22.3.1.11 Container: WdgSettingsFast

Hardware dependent settings for the WDG driver fast mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

22.3.1.11.1 WdgFastModeTimeoutValue

Table 1913 Specification for WdgFastModeTimeoutValue

Name	WdgFastModeTimeoutValue		
Description	<p>This parameter is the fast mode timeout in seconds for WDTx.</p> <p>It is used to calculate the reload value of the WDT in fast mode.</p> <p>For TC3xx , fast timeout maximum value can be calculated as follows:</p> <ul style="list-style-type: none"> - fSPB = 100 MHz - Clock divider(fast) = 256 - WDT frequency = $100\ 000\ 000/256 = 390625$ Hz - 1 tick = $1/390625 = 0.00000256$s Minimum timeout = 0.00000256s Maximum timeout = $65536 * 0.00000256 = 0.16777216$s (approximately) Value should be less than WdgSlowModeTimeoutValue. Value should be greater than WdgFastRefreshTime. <p>The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	WdgFastRefreshTime (≥ 0.001 s) - WdgSlowModeTimeoutValue (≤ 0.16777216 s)		
Default value	0.01s		

Wdg_17_Scu driver
Table 1913 Specification for WdgFastModeTimeoutValue (continued)

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	WdgFastRefreshTime, WdgSlowModeTimeoutValue		

22.3.1.12 Container: WdgSettingsOff

Hardware dependent settings for the WDG driver off mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

22.3.1.13 Container: WdgSettingsSlow

Hardware dependent settings for the WDG driver slow mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

22.3.1.13.1 WdgSlowModeTimeoutValue
Table 1914 Specification for WdgSlowModeTimeoutValue

Name	WdgSlowModeTimeoutValue		
Description	<p>Slow mode timeout in seconds for WDTx. It is used to calculate the reload value of the WDT in slow mode.</p> <p>For TC3xx , Slow timeout maximum value can be calculated as follows:</p> <ul style="list-style-type: none"> - fSPB = 100 MHz - Clock divider(slow) = 16384 - WDT frequency = $100\ 000\ 000 / 16384 = 6103.515625$ Hz - 1 tick = $1 / 6103.515625 = 0.00016384$s <p>Minimum timeout = 0.00016384s</p> <p>Maximum timeout = $65536 * 0.00016384 = 10.73741824$s (approximately)</p> <p>Value should be greater than WdgSlowRefreshTime.</p> <p>The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.</p>		
Multiplicity	1..1	Type	EcuCFloatParamDef
Range	WdgSlowRefreshTime (≥ 0.001 s) - 10.73741824s		
Default value	0.02s		
Post-build variant value	TRUE	Post-build variant multiplicity	-

Wdg_17_Scu driver
Table 1914 Specification for WdgSlowModeTimeoutValue (continued)

Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	WdgSlowRefreshTime		

22.3.1.14 Container: WdgTriggerTimerSetting

This container contains the configuration elements for configuring GTM/STM hardware. The settings here are used to configure the timing needs for WDG module

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

22.3.1.14.1 WdgFastRefreshTime
Table 1915 Specification for WdgFastRefreshTime

Name	WdgFastRefreshTime		
Description	Fast mode GTM/STM callback period in seconds for WDTx. It is used to calculate the value of the trigger counter used to service the WDT for fast mode. Values greater than 1.0ms and less than WdgFastModeTimeoutValue, WdgSlowRefreshTime. The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.001s - WdgFastModeTimeoutValue (and WdgSlowRefreshTime)		
Default value	0.007s		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.1.14.2 WdgSlowRefreshTime
Table 1916 Specification for WdgSlowRefreshTime

Name	WdgSlowRefreshTime		
Description	Slow mode GTM/STM callback period in seconds for WDTx. It is used to calculate the value of the trigger counter used to service the WDT for slow mode. Values greater than 1.0ms, WdgFastRefreshTime and less than WdgSlowModeTimeoutValue.		

Wdg_17_Scu driver
Table 1916 Specification for WdgSlowRefreshTime (continued)

	The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	WdgFastRefreshTime (>0.001s) - WdgSlowModeTimeoutValue (<10.73741824s)		
Default value	0.015s		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

22.3.2 Functions - Type definitions

22.3.2.1 Wdg_17_Scu_ConfigType

Table 1917 Specification for Wdg_17_Scu_ConfigType

Syntax	Wdg_17_Scu_ConfigType	
Type	Structure	
Range	hardware dependent structure	The elements of the data structure are specific to the microcontroller.
Description	The elements of the data structure are specific to the microcontroller. This type definition will be detailed in Design based on data structure design. (Used for pointers to structures holding configuration data provided to the WDG module initialization routine for configuration of the module and watchdog hardware.)	
Source	AUTOSAR	

22.3.3 Functions - APIs

This section lists the APIs provided by the driver along with a short description of the functionality. The names of the APIs are extended with VendorId and VendorApiInfix since the upper multiplicity of the WDG driver is greater than 1.

22.3.3.1 Wdg_17_Scu_Init

Table 1918 Specification for Wdg_17_Scu_Init API

Syntax	void Wdg_17_Scu_Init (
---------------	---------------------------

Wdg_17_Scu driver
Table 1918 Specification for Wdg_17_Scu_Init API (continued)

	const Wdg_17_Scu_ConfigType * const ConfigPtr)	
Service ID	0x00	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This service is used to initialize each WDTx with configuration for the core from which this API invoked.</p> <p>To initialize WDTx, this API shall be called from CPUx (corresponding CPU core), with the pointer of corresponding core config data.</p>	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>WDG_17_SCU_E_PARAM_CONFIG: API service called with wrong / inconsistent parameter(s) .</p> <p>This DET will be reported if an API is called from a wrong core and if mode change was unsuccessful during initialization.</p> <p>WDG_17_SCU_E_INIT_FAILED: Initialization called with NULL pointer.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>WDG_17_SCU_E_DISABLE_REJECTED: Initialization or watchdog mode switch failed because it would disable the watchdog though this is not allowed in this configuration .</p> <ul style="list-style-type: none"> - FAILED: when disabling of the watchdog mode failed. - PASSED: when disabling of the watchdog mode not failed. <p>WDG_17_SCU_E_MODE_FAILED: Setting watchdog mode failed.</p> <ul style="list-style-type: none"> - FAILED : Setting Watchdog mode failed. - PASSED : Setting Watchdog mode passed. <p>Safety Errors:</p> <p>WDG_17_SCU_E_BUSY: If the state of the driver is BUSY when initialization is called asynchronously.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	

Wdg_17_Scu driver
Table 1918 Specification for Wdg_17_Scu_Init API (continued)

User hints	MCU should be in an uninitialized state before this function is called. Wdg_17_Scu_Init depends on MCU SPB clock.
-------------------	---

22.3.3.2 Wdg_17_Scu_InitCheck
Table 1919 Specification for Wdg_17_Scu_InitCheck API

Syntax	<pre>Std_ReturnType Wdg_17_Scu_InitCheck (const Wdg_17_Scu_ConfigType * const ConfigPtr)</pre>	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: if initialization was successful. E_NOT_OK: if initialization was unsuccessful.
Description	<p>This API returns the status of the module initialization in context to the core from where the API is invoked.</p> <p>The API is available when safety is enabled or initialization check is explicitly enabled.</p> <p>This API does not check any running counter values of GTM, STM and global variables.</p> <p>Note: Init check should be performed in the following sequence:</p> <ol style="list-style-type: none"> 1. Call Wdg_17_Scu_Init from a core. 2. Call Wdg_17_Scu_InitCheck from the same core. 	
Source	IFX	
Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	WdgInitCheckApi	
User hints	None.	

Wdg_17_Scu driver**22.3.3.3 Wdg_17_Scu_SetMode****Table 1920 Specification for Wdg_17_Scu_SetMode API**

Syntax	<pre>Std_ReturnType Wdg_17_Scu_SetMode (const WdgIf_ModeType Mode)</pre>	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Mode	One of the following statically configured modes: 1. WDGIF_OFF_MODE 2. WDGIF_SLOW_MODE 3. WDGIF_FAST_MODE
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: if mode change is successfull E_NOT_OK: if mode change is unsucessfull
Description	Switches the watchdog into the Mode Requested. To change mode for WDTx, the API shall be called from CPUx (corresponding CPU core).	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>WDG_17_SCU_E_PARAM_MODE: API service called with wrong / inconsistent parameter(s).</p> <p>WDG_17_SCU_E_DRIVER_STATE: API service used in wrong context (for example, module not initialized).</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>WDG_17_SCU_E_MODE_FAILED: Setting watchdog mode failed.</p> <ul style="list-style-type: none"> - FAILED : Setting Watchdog mode failed. - PASSED : Setting Watchdog mode passed. <p>WDG_17_SCU_E_DISABLE_REJECTED: Initialization or watchdog mode switch failed because it would disable the watchdog though this is not allowed in this configuration .</p> <ul style="list-style-type: none"> - FAILED: when disabling of the watchdog mode failed. - PASSED: when disabling of the watchdog mode not failed. <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	

Wdg_17_Scu driver
Table 1920 Specification for Wdg_17_Scu_SetMode API (continued)

User hints	WDG should be in an initialized state before this function is called. This function shall be called before watchdog timeout timer elapses.
-------------------	--

22.3.3.4 Wdg_17_Scu_SetTriggerCondition
Table 1921 Specification for Wdg_17_Scu_SetTriggerCondition API

Syntax	<pre>void Wdg_17_Scu_SetTriggerCondition (const uint16 timeout)</pre>	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	timeout	Timeout value (milliseconds) for setting the trigger counter.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Sets the timeout value for the trigger counter.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>WDG_17_SCU_E_PARAM_TIMEOUT: The passed timeout value is higher than the maximum timeout value.</p> <p>WDG_17_SCU_E_DRIVER_STATE: API service used in wrong context (for example, module not initialized).</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	WDG should be in an initialized state before this function is called. This function shall be called before watchdog timeout timer elapses. To set the timeout value for WDTx, the API must be called from CPUx (corresponding CPU core).	

Wdg_17_Scu driver**22.3.3.5 Wdg_17_Scu_GetVersionInfo****Table 1922 Specification for Wdg_17_Scu_GetVersionInfo API**

Syntax	<pre>void Wdg_17_Scu_GetVersionInfo (Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where to store the version information of this module.
Parameters (in - out)	-	-
Return	void	Pointer to where to store the version information of this module.
Description	Returns the version information of the module.	
Source	AUTOSAR	
Error handling	<p>DET:</p> <p>WDG_17_SCU_E_PARAM_POINTER: API is called with wrong pointer value (for example, NULL pointer) other than initialization</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	WdgVersionInfoApi	
User hints	None.	

22.3.4 Notifications and Callbacks**22.3.4.1 Wdg_17_Scu_Isr****Table 1923 Specification for Wdg_17_Scu_Isr API**

Syntax	<pre>void Wdg_17_Scu_Isr (const uint8 LogicalChId, const uint8 IsrStatus)</pre>
---------------	---

Wdg_17_Scu driver
Table 1923 Specification for Wdg_17_Scu_Isr API (continued)

Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	LogicalChild IsrStatus	channel ID This parameter gives information about which comparator cased the interrupt.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	ISR for GTM/STM interrupts	
Source	IFX	
Error handling	<p>DET:</p> <p>WDG_17_SCU_E_DRIVER_STATE: API service used in wrong context (for example, module not initialized).</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>WDG_17_SCU_E_MODE_DISABLED: If ISR is triggered when WDG is disabled.</p> <p>WDG_17_SCU_E_INVALID_INTERRUPT_SOURCE: WDG ISR is triggered if interrupt source is not CMP0 in case of GTM.</p> <p>Note : This error check is not done in case of STM since it is expected that the interrupt when CMP0 is configured will always arrive at STMIR0 and CMP1 is configured it will always arrive at STMIR1.</p> <p>WDG_17_SCU_E_ACCESS: If the Access Error Flag is Set while servicing an interrupt. (This bit is set when an illegal Password Access or Modify Access to register WDTxCON0 was attempted).</p> <p>WDG_17_SCU_E_OVERFLOW: If the Overflow Error Flag is Set while servicing an interrupt. (This bit is set when the WDT overflows from FFFFH to 0000H.).</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	<p>This function is called by functions in Mcu_17_Timerlp.c if the GTM/STM interrupt is configured to service watchdog driver.</p> <p>Timing constraint: This callback function shall be called before the watchdog timer overflows and rests the system.</p>	

Wdg_17_Scu driver

22.3.5 Scheduled functions

Not applicable for the driver.

22.3.6 Interrupt service routines

Not applicable for the driver. The WDT is serviced using callbacks from the MCU driver.

22.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

22.3.7.1 Development errors

The following table lists all the development errors reported by the driver.

Note: *The following error IDs are also reported as safety errors.*

Table 1924 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
Initialization called with NULL pointer.	AUTOSAR	WDG_17_SCU_E_INIT_FAILED=0x15	Wdg_17_Scu_Init
API service used in wrong context (for example, module not initialized).	AUTOSAR	WDG_17_SCU_E_DRIVER_STATE=0x10	Wdg_17_Scu_SetTriggerCondition, Wdg_17_Scu_Isr, Wdg_17_Scu_SetMode
API service called with wrong / inconsistent parameter(s) . This DET will be reported if an API is called from a wrong core and if mode change was unsuccessful during initialization.	AUTOSAR	WDG_17_SCU_E_PARAM_CONFIG=0x12	Wdg_17_Scu_Init
API service called with wrong / inconsistent parameter(s).	AUTOSAR	WDG_17_SCU_E_PARAM_MODE=0x11	Wdg_17_Scu_SetMode
API is called with wrong pointer value (for example, NULL pointer) other than initialization	AUTOSAR	WDG_17_SCU_E_PARAM_POINTER=0x14	Wdg_17_Scu_GetVersionInfo
The passed timeout value is higher than the maximum timeout value.	AUTOSAR	WDG_17_SCU_E_PARAM_TIMEOUT=0x13	Wdg_17_Scu_SetTriggerCondition

22.3.7.2 Production errors

The following table lists all the production errors reported by the driver.

Wdg_17_Scu driver
Table 1925 Description of production errors reported

Description	Source	Error code and value	Applicable APIs
Initialization or watchdog mode switch failed because it would disable the watchdog though this is not allowed in this configuration . - FAILED: when disabling of the watchdog mode failed. - PASSED: when disabling of the watchdog mode not failed.	AUTOSAR	WDG_17_SCU_E_DISABLE_REJECTE D=Assigned by DEM	Wdg_17_Scu_SetMode, Wdg_17_Scu_Init
Setting watchdog mode failed. - FAILED : Setting Watchdog mode failed. - PASSED : Setting Watchdog mode passed.	AUTOSAR	WDG_17_SCU_E_MODE_FAILED=Assi gned by DEM	Wdg_17_Scu_SetMode, Wdg_17_Scu_Init

22.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 1926 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
If the Access Error Flag is Set while servicing an interrupt. (This bit is set when an illegal Password Access or Modify Access to register WDTxCON0 was attempted).	IFX	WDG_17_SCU_E_ACCESS=0xCB	Wdg_17_Scu_Lsr
If the state of the driver is BUSY when initialization is called asynchronously.	IFX	WDG_17_SCU_E_BUSY=0xCC	Wdg_17_Scu_Init
WDG ISR is triggered if interrupt source is not CMP0 in case of GTM. Note : This error check is not done in case of STM since it is expected that the interrupt when CMP0 is configured will always arrive at STMIR0 and CMP1 is configured it will always arrive at STMIR1.	IFX	WDG_17_SCU_E_INVALID_INTERRUPT_SOURCE=0xC9	Wdg_17_Scu_Lsr

Wdg_17_Scu driver

Table 1926 Description of safety errors reported (continued)

Description	Source	Error code and value	Applicable APIs
If ISR is triggered when WDG is disabled.	IFX	WDG_17_SCU_E_MODE_DISABLED=0xC8	Wdg_17_Scu_Isr
If the Overflow Error Flag is Set while servicing an interrupt. (This bit is set when the WDT overflows from FFFFH to 0000H.).	IFX	WDG_17_SCU_E_OVERFLOW=0xCA	Wdg_17_Scu_Isr

22.3.7.4 Runtime errors

The driver does not report any runtime errors.

22.3.8 Deviations and limitations

This section describes the deviations and limitations from software specification.

22.3.8.1 Deviations

Not applicable for the WDG driver.

22.3.8.2 Limitations

Not applicable for the WDG driver.

22.3.9 Unsupported hardware features

The following features of the WDT peripheral are not supported by the WDG driver:

- System watchdog timer.
- Incorporation of the corresponding ENDINIT bit and monitoring its modifications. The WDG ENDINIT feature will only be used for CPU critical registers and will not be supported by the WDG driver.
- Automatic password sequencing mechanism.
- The UR bit in the WDTCPU0CON1 register will not be configurable through the WDG driver. Updating this bit depends on the state of the SMU, which is not monitored by the WDG driver.
- Time check password feature.

Revision history

Revision history

Major changes since the last version

Date	Version	Description
2019-10-10	1.30.0_9.0	<ul style="list-style-type: none"> • Mapping of MCAL to external safety mechanism section is updated. • Icu_17_TimerIp <ul style="list-style-type: none"> - Configuration parameters IcuReportWakeupSource, IcuSignalEdgeDetection, IcuSignalMeasurement, IcuTimestampMeasurement, IcuWakeups, IcuGetInputStateAPI, IcuGetDutyCycleValuesAPI, IcuGetTimeElapsedAPI, IcuEnableWakeupAPI and IcuEdgeDetectAPI is updated. • Fls_17_Dmu <ul style="list-style-type: none"> - Configuration dependency for Fls_17_Dmu_VerifyErase, Fls_17_Dmu_VerifySectorErase, Fls_17_Dmu_CtrlTimeoutDet, Fls_17_Dmu_SetMode and Fls_17_Dmu_MainFunction APIs is updated. - Limitations and deviations section is updated. • Mcu <ul style="list-style-type: none"> - Configuration parameter McuSysClkFrequency is added. Configuration parameter McuPllInputSrcSelection is updated. • McalLib <ul style="list-style-type: none"> - Hardware software mapping section is updated. • Can_17_McmCan <ul style="list-style-type: none"> - Sync/Async information for Can_17_McmCan_SetControllerMode API is updated • Pwm_17_GtmCcu6 <ul style="list-style-type: none"> - Limitations section is updated. • Spi <ul style="list-style-type: none"> - Limitations and deviations section is updated.
2019-08-05	8.0	Update the images and device information in the chapter Generic information.
2019-07-26	7.0	<ul style="list-style-type: none"> • Can_17_McmCan <ul style="list-style-type: none"> - Description for the Can_17_McmCan_Main_Function_BusOff API is updated - Description for the CanControllerId configuration parameter is updated
2019-07-23	6.0	<ul style="list-style-type: none"> • Hardware-software mapping and Integration hints are updated for all drivers. • Adc <ul style="list-style-type: none"> - Configuration parameters for BWD support is added. Configuration parameter for alias support is updated. • Can_17_McmCan <ul style="list-style-type: none"> - Description for the Can_17_McmCan_Main_Function_Read and Can_17_McmCan_IsrRxFIFOHandler APIs is updated. Description for the CanControllerTrcvDelayCompensationOffset configuration parameter is updated.

Revision history

Date	Version	Description
		<ul style="list-style-type: none"> - Limitations and deviations section is updated. - Key architectural considerations section is updated. • CanTrcv_17_W9255 <ul style="list-style-type: none"> - Integration hints section is updated. Critical section WakeFlagUpdate is removed. - Integration hints section for the MCU support is updated. • Dio <ul style="list-style-type: none"> - AoU for Dio_FlipChannel API is added. • Fee <ul style="list-style-type: none"> - Example usage section is updated. • Fls_17_Dmu <ul style="list-style-type: none"> - Safety error names are updated. - Limitations and deviations section is updated. • Gpt <ul style="list-style-type: none"> - Integration hints section is updated for SchM. • Icu_17_Timerlp <ul style="list-style-type: none"> - The IcuIncrementalInterfaceApi configuration parameter is made editable to enable the user to use incremental interface API. - Example usage for incremental interface mode is added. Configuration parameter IcuMaxChannel,GPT12DirPortSelection,GPT12DirPortSelection, GPT12InputPortSelection, TimChannelPortPinSelect, ErulInputPin,GPT12CounterType,GPT12BlockReference ,TimChannelInputSelect and CCChannelInputSelection are updated. Description for the APIs Icu_17_Timerlp_StartInclInterface, Icu_17_Timerlp_StopInclInterface, Icu_17_Timerlp_ReadEncCount and Icu_17_Timerlp_ReadEncCountDir is updated Description for the data type Icu_17_Timerlp_ChannelType is updated. • Mcallib <ul style="list-style-type: none"> - AoU for the usage of the Mcal_WriteSafetyEndInitProtReg API is added. • Mcu <ul style="list-style-type: none"> - User hints in the Mcu_InitRamSection and Mcu_SetMode APIs are updated. - AoUs for SMU alarm and sequence of Timerlp APIs is added. - The McuExtClock1Div configuration parameter is renamed to McuFoutClockDiv. Description for the McuClockReferencePointFrequency2 and McuAdasFrequency configuration parameters is updated. - Key architectural considerations section is updated. • Ocu <ul style="list-style-type: none"> - Key architectural considerations section is updated. - AoU section updated. - Limitations and deviations section updated. • Port <ul style="list-style-type: none"> - Limitations and deviations section updated. • Pwm_17_GtmCcu6 <ul style="list-style-type: none"> - AoU section is updated. - Datatype Pwm_17_GtmCcu6_ChannelType is updated.

Revision history

Date	Version	Description
		<ul style="list-style-type: none"> - DSADC notification-related information is added. - Error table is updated. • Spi <ul style="list-style-type: none"> - Limitations and deviations section is updated.
2019-04-22	5.0	Added support for the TC37xA and TC37xA_ED devices.
2019-04-11	4.0	<ul style="list-style-type: none"> • Added support for the TC35xA device. • Added the CanTrcv_17_V9251, CanTrcv_17_W9255 and OCU drivers.
2019-02-06	3.0	<p>In the <i>Generic information</i> chapter, added the <i>Post-build variant multiplicity and Multiplicity configuration class</i> section.</p> <p>Updated the <i>Configuration interfaces</i> and <i>Deviations</i> sections for the SPI driver.</p>
2019-02-04	2.0	<ul style="list-style-type: none"> • <i>Integration hints and Reference information</i> for all modules updated. • Module-specific AoUs updated. • Added the EB tesos: license handling section. • Added AoUs in the Common Assumptions of Use (AoUs) section. • Updated the ESM mapping in the Mapping of MCAL to external safety mechanism section.
2018-10-12	1.0	Initial version.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2019-10-10

Published by

**Infineon Technologies AG
81726 Munich, Germany**

**© 2019 Infineon Technologies AG
All Rights Reserved.**

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

**Document reference
IFX-vjw1559807504045**

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury