

MCAL User Manual for CD 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 CD 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, code snippets, file/folder names, directories, command line inputs
New	
Hyperlink	Provides quick and easy access to cross-referenced topics/sections
>	Indicates that a cascading sub-menu opens when you choose 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 DFlash, there are one or more banks. Banks support concurrent operations with some limitations due to common logic.
BL	Bit-line
BSW	Basic software
CC	Communication controller
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.
CMU	Clock management unit (functional block of GTM)

About this document
Table 2 Glossary (continued)

Term	Description
CPU	Central processing unit
DEM	Diagnostics event manager (MCAL module)
DET	Development error tracer
DF_EEPROM	Data Flash dedicated for EEPROM emulation
DFDBER	Double bit error
DFlash	Data Flash
DFSBER	Single bit error
DFTBER	Triple bit error
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
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 (triggering the watchdog should happen within certain minimum/maximum boundaries within the timeout period) and a timeout period of 5 ms.
FCE	Flexible CRC engine
FIFO	First in first out
GC	Garbage collection
GTM	Generic timer module (hardware)
I/O	Input/Output
IB	Driver-defined internal buffer / channel
IFX	Infineon Technologies
ISR	Interrupt service routine
LPdu	Datalink layer protocol data unit
LPM	Low power mode
MAC	Media access control
MC-ISAR	Microcontroller Infineon Software Architecture
MCU	Microcontroller unit
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.

About this document
Table 2 Glossary (continued)

Term	Description
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
NVRAM block	Management unit as seen by the NVRAM manager
Page	A page is an aligned group of data double words plus an ECC extension. It is the smallest unit that can be programmed. DFlash: 1 data double word (8 bytes) plus 22-bit ECC extension.
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.
Physical sector	It is a combination of 256 logical sectors (in DFlash), which comprises of 1 MB memory.
PLL	Phase lock loop
Pn_xxxxx	Port n register (xxxxx is register name)
QS	Quasi-static
QSPI	Queued serial peripheral interface
r	Read access
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
STM	System timer (hardware)

About this document
Table 2 Glossary (continued)

Term	Description
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.
w	Write access

Reference documents

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

- AURIX™ TC3xx MCAL User Manual for BASIC package, V1.40.0_13.0, 2020-05-22, Infineon Technologies Munich AG
- AURIX™ TC3xx User Manual, V1.4.0, 2019-12, Infineon Technologies AG
- AURIX™ TC39x-B Appendix to User Manual, V1.4.0, 2019-12, Infineon Technologies AG
- AURIX™ TC38x Appendix to User Manual, V1.4.0, 2019-12, Infineon Technologies AG
- AURIX™ TC37x Appendix to User Manual, V1.4.0, 2019-12, Infineon Technologies AG
- AURIX™ TC37xEXT Appendix to User Manual, V1.4.0, 2019-12, Infineon Technologies AG
- AURIX™ TC36x Appendix to User Manual, V1.4.0, 2019-12, Infineon Technologies AG
- AURIX™ TC35x Appendix to User Manual, V1.4.0, 2019-12, Infineon Technologies AG
- AURIX™ TC33xEXT Appendix to User Manual, V1.4.0, 2019-12, Infineon Technologies AG
- AURIX™ TC33x/TC32x Appendix to User Manual, V1.4.0, 2019-12, Infineon Technologies AG
- TC39x_AA_Errata_Sheet, Rel1.10, 2019-12-20
- TC39x_BA_Errata_Sheet, Rel1.6, 2019-12-02
- TC39x_BB_Errata_Sheet, Rel1.4, 2019-12-02
- TC39x_BC_Errata_Sheet, Rel1.4, 2020-03-31
- TC39x_BD_Errata_Sheet, Rel1.1, 2020-03-31
- TC38x_AA_Errata_Sheet, Rel1.6, 2019-12-02
- TC38x_AB_Errata_Sheet, Rel1.4, 2019-12-02
- TC38x_AC_Errata_Sheet, Rel1.3, 2019-12-02
- TC38x_AD_Errata_Sheet, Rel1.4, 2020-03-31
- TC38x_AE_Errata_Sheet, Rel1.1, 2020-03-31
- TC37x_AA_Errata_Sheet, Rel1.2, 2020-03-31
- TC37xEXT_AA_Errata_Sheet, Rel1.4, 2020-03-31
- TC37xEXT_AB_Errata_Sheet, Rel1.2, 2019-12-02
- TC36x_AA_Errata_Sheet, Rel1.2, 2020-03-31
- TC35x_AA_Errata_Sheet, Rel1.4, 2020-03-31

About this document

- TC35x_AB_Errata_Sheet, Rel1.3, 2020-03-31
- TC33x_AA_Errata_Sheet, Rel1.0, 2020-01-17
- TC33xEXT_AA_Errata_Sheet, Rel1.1, 2020-03-31
- TC33x_TC32x_AA_Errata_Sheet, Rel1.1, 2020-03-31
- Note: *The errata analysis (TC3xx_SW_MCAL_HWErrataAnalysis.xlsx, v1.40.0_15.0) for TC3xx devices is placed inside the BASIC package*

Table of contents**Table of contents**

About this document	1
Table of contents	6
1 DMA driver	24
1.1 User information	24
1.1.1 Description	24
1.1.2 Hardware-software mapping	24
1.1.2.1 DMA: primary hardware peripheral	25
1.1.2.2 SRC: dependent hardware peripheral	26
1.1.2.3 SCU: dependent hardware peripheral	26
1.1.3 File structure	27
1.1.3.1 C file structure	27
1.1.3.2 Code generator plugin files	28
1.1.4 Integration hints	29
1.1.4.1 Integration with AUTOSAR stack	29
1.1.4.2 Multicore and Resource Manager	33
1.1.4.3 MCU support	34
1.1.4.4 Port support	34
1.1.4.5 DMA support	34
1.1.4.6 Interrupt connections	34
1.1.4.7 Example usage	35
1.1.5 Key architectural considerations	40
1.1.5.1 Usage of General Purpose Software Request (GPSR)	40
1.2 Assumptions of Use (AoUs)	41
1.3 Reference information	44
1.3.1 Configuration interfaces	44
1.3.1.1 Container: CommonPublishedInformation	44
1.3.1.1.1 ArMajorVersion	45
1.3.1.1.2 ArMinorVersion	45
1.3.1.1.3 ArPatchVersion	45
1.3.1.1.4 ModuleId	46
1.3.1.1.5 Release	46
1.3.1.1.6 SwMajorVersion	47
1.3.1.1.7 SwMinorVersion	47
1.3.1.1.8 SwPatchVersion	48
1.3.1.1.9 VendorId	48
1.3.1.2 Container: Dma	48
1.3.1.3 Container: DmaChannelConfig	48
1.3.1.3.1 DmaChannelAssignedPartition	49
1.3.1.3.2 DmaChannelId	49

Table of contents

1.3.1.3.3	DmaChannelNotification	50
1.3.1.3.4	DmaChannelNumTransactionSet	50
1.3.1.3.5	DmaTcsInterruptTransactionLoss	51
1.3.1.4	Container: DmaChannelTransactionSet	51
1.3.1.4.1	DmaNextTcsIndex	51
1.3.1.4.2	DmaTcsAppendTimeStamp	52
1.3.1.4.3	DmaTcsAutoStartEnable	52
1.3.1.4.4	DmaTcsCircularBufferDestinationEnable	53
1.3.1.4.5	DmaTcsCircularBufferDestinationLength	53
1.3.1.4.6	DmaTcsCircularBufferSourceEnable	54
1.3.1.4.7	DmaTcsCircularBufferSourceLength	55
1.3.1.4.8	DmaTcsDaisyChaining	56
1.3.1.4.9	DmaTcsDataTransferInterrupt	56
1.3.1.4.10	DmaTcsDestinationAddress	57
1.3.1.4.11	DmaTcsDestinationAddressModificationFactor	58
1.3.1.4.12	DmaTcsDestinationAddressMovement	58
1.3.1.4.13	DmaTcsDoubleBuffer	59
1.3.1.4.14	DmaTcsHardwareTrigger	60
1.3.1.4.15	DmaTcsIndex	60
1.3.1.4.16	DmaTcsInterruptDataTransfer	61
1.3.1.4.17	DmaTcsInterruptDataTransferThreshold	61
1.3.1.4.18	DmaTcsInterruptDestinationAddressWrap	62
1.3.1.4.19	DmaTcsInterruptSourceAddressWrap	62
1.3.1.4.20	DmaTcsMoveLength	63
1.3.1.4.21	DmaTcsReferenceAddressCrc	63
1.3.1.4.22	DmaTcsReferenceDataCrc	64
1.3.1.4.23	DmaTcsShadowRegisterConfiguration	64
1.3.1.4.24	DmaTcsSourceAddress	65
1.3.1.4.25	DmaTcsSourceAddressModificationFactor	66
1.3.1.4.26	DmaTcsSourceAddressMovement	67
1.3.1.4.27	DmaTcsSwapDataCRCByteOrder	68
1.3.1.4.28	DmaTcsTransactionLength	68
1.3.1.4.29	DmaTcsTransferLength	69
1.3.1.4.30	DmaTcsTriggerFrequency	69
1.3.1.4.31	DmaUserHeaderFileWithExternDeclarations	70
1.3.1.5	Container: DmaGeneral	70
1.3.1.5.1	DmaBufferSwitchApiConfiguration	70
1.3.1.5.2	DmaChDeInitApiConfiguration	71
1.3.1.5.3	DmaDataPendingApiConfiguration	71
1.3.1.5.4	DmaDevErrorDetect	72
1.3.1.5.5	DmaGetVersionInfoApiConfiguration	73
1.3.1.5.6	DmaInitCheckApi	73

Table of contents

1.3.1.5.7	DmaInitDeInitApiMode	74
1.3.1.5.8	DmaMaxTransactionSetPerChannel	74
1.3.1.5.9	DmaMultiCoreErrorDetect	75
1.3.1.5.10	DmaRuntimeApiMode	75
1.3.1.5.11	DmaSafetyEnable	76
1.3.1.5.12	DmaSuspendApiConfiguration	76
1.3.1.5.13	DmaTriggerApiConfiguration	77
1.3.1.6	Container: DmaMoveEngineConfig	77
1.3.1.6.1	DmaMEDestinationErrorInterrupt	78
1.3.1.6.2	DmaMELinkedListErrorInterrupt	78
1.3.1.6.3	DmaMESourceErrorInterrupt	79
1.3.1.7	Container: DmaResourcePartition	79
1.3.1.7.1	DmaPermittedBusMaster	79
1.3.1.7.2	DmaResourcePartitionBusMode	80
1.3.1.7.3	DmaResourcePartitionErrorNotifRoutine	80
1.3.2	Functions - Type definitions	81
1.3.2.1	Dma_ChannelNotificationPtrType	81
1.3.2.2	Dma_ConfigType	81
1.3.2.3	Dma_ConfigUpdateType	81
1.3.2.4	Dma_CrcType	82
1.3.2.5	Dma_ErrorNotificationPtrType	83
1.3.2.6	Dma_EventsType	83
1.3.2.7	Dma_MoveEngineListType	84
1.3.3	Functions - APIs	84
1.3.3.1	Dma_Init	84
1.3.3.2	Dma_IsInitDone	85
1.3.3.3	Dma_ChInit	86
1.3.3.4	Dma_ChUpdate	87
1.3.3.5	Dma_ChDelInit	88
1.3.3.6	Dma_ChTransferFreeze	89
1.3.3.7	Dma_ChTransferResume	90
1.3.3.8	Dma_ChEnableHardwareTrigger	91
1.3.3.9	Dma_ChDisableHardwareTrigger	92
1.3.3.10	Dma_ChStartTransfer	93
1.3.3.11	Dma_ChStopTransfer	94
1.3.3.12	Dma_ChGetRemainingData	95
1.3.3.13	Dma_ChSwitchBuffer	96
1.3.3.14	Dma_GetEvents	97
1.3.3.15	Dma_ChStatusClear	99
1.3.3.16	Dma_ChInterruptEnable	100
1.3.3.17	Dma_ChInterruptDisable	101
1.3.3.18	Dma_GetVersionInfo	102

Table of contents

1.3.3.19	Dma_MEStatusClear	103
1.3.3.20	Dma_InitCheck	104
1.3.3.21	Dma_GetCrcValue	104
1.3.3.22	Dma_GetCurrentTimeStamp	106
1.3.3.23	Dma_IsChannelInitDone	107
1.3.4	Notifications and Callbacks	108
1.3.5	Scheduled functions	108
1.3.6	Interrupt service routines	108
1.3.6.1	Dma_ChInterruptHandler	108
1.3.6.2	Dma_MEInterruptDispatcher	109
1.3.7	Error codes classification	109
1.3.7.1	Development errors	110
1.3.7.2	Production errors	113
1.3.7.3	Safety errors	113
1.3.7.4	Runtime errors	113
1.3.8	Deviations and limitations	113
1.3.8.1	Deviations	113
1.3.8.2	Limitations	114
1.3.9	Unsupported hardware features	115
2	DSADC driver	117
2.1	User information	117
2.1.1	Description	117
2.1.2	Hardware-software mapping	117
2.1.2.1	EDSADC: primary hardware peripheral	118
2.1.2.2	GTM: dependent hardware peripheral	119
2.1.2.3	SCU: dependent hardware peripheral	119
2.1.2.4	EICR/IGCR: primary hardware peripheral	120
2.1.2.5	SRC: dependent hardware peripheral	120
2.1.2.6	DMA: dependent hardware peripheral	121
2.1.2.7	CONVERTER: dependent hardware peripheral	121
2.1.2.8	Port: dependent hardware peripheral	121
2.1.3	File structure	121
2.1.3.1	C File Structure	121
2.1.3.2	Code Generator Plugin Files	123
2.1.4	Integration hints	125
2.1.4.1	Integration with AUTOSAR stack	125
2.1.4.2	Multicore and Resource Manager	128
2.1.4.3	MCU support	128
2.1.4.4	Port support	128
2.1.4.5	DMA support	128
2.1.4.6	Interrupt connections	129

Table of contents

2.1.4.7	Example usage	131
2.1.5	Key architectural considerations	146
2.1.5.1	Mode of operation: result acquisition	146
2.1.5.2	Mode of operation: result handling	147
2.1.5.3	Accessing shared SFR	147
2.1.5.4	Settling time after the filter chain restart	147
2.1.5.5	Timestamp	148
2.2	Assumptions of Use (AoUs)	151
2.3	Reference information	153
2.3.1	Configuration interfaces	153
2.3.1.1	Container: Dsadc	155
2.3.1.2	Container: DsadcAuxFilterConfig	155
2.3.1.2.1	DsadcAuxCicFilterEnable	155
2.3.1.2.2	DsadcAuxFilterCicDecimationFactor	155
2.3.1.3	Container: DsadcCalibAlgoConfig	156
2.3.1.3.1	DsadcCICDecimationRate	156
2.3.1.3.2	DsadcCalibAlgoTargetValue	157
2.3.1.3.3	DsadcCalibCICFilterOutputShiftPos	157
2.3.1.3.4	DsadcCalibGainCorrMulFactor	158
2.3.1.3.5	DsadcGainCalibMulFactor	158
2.3.1.4	Container: DsadcCarrierGeneratorConfiguration	159
2.3.1.4.1	DsadcCarrierFrequencyClockDiv	159
2.3.1.4.2	DsadcCarrierSignalPolarity	160
2.3.1.4.3	DsadcCarrierSignalType	161
2.3.1.4.4	DsadcPwmGenerationMode	161
2.3.1.5	Container: DsadcChannelConfiguration	162
2.3.1.5.1	DsadcAccessMode	162
2.3.1.5.2	DsadcBufferFullNotification	163
2.3.1.5.3	DsadcChannelId	163
2.3.1.5.4	DsadcGateActiveLevel	164
2.3.1.5.5	DsadcHwChannelNum	164
2.3.1.5.6	DsadcNewResultNotification	165
2.3.1.5.7	DsadcTimestampFeature	166
2.3.1.5.8	DsadcTriggerMode	166
2.3.1.5.9	DsadcWindowCloseNotification	167
2.3.1.6	Container: DsadcCommonModeVoltConfig	168
2.3.1.6.1	DsadcComModeVoltNegAEnable	168
2.3.1.6.2	DsadcComModeVoltNegBEnable	168
2.3.1.6.3	DsadcComModeVoltNegCEnable	169
2.3.1.6.4	DsadcComModeVoltNegDEnable	169
2.3.1.6.5	DsadcComModeVoltPosAEnable	170
2.3.1.6.6	DsadcComModeVoltPosBEnable	171

Table of contents

2.3.1.6.7	DsadcComModeVoltPosCEnable	171
2.3.1.6.8	DsadcComModeVoltPosDEnable	172
2.3.1.6.9	DsadcCommonModeVoltageEnable	172
2.3.1.6.10	DsadcCommonModeVoltageSelect	173
2.3.1.7	Container: CommonPublishedInformation	173
2.3.1.7.1	ArMajorVersion	174
2.3.1.7.2	ArMinorVersion	174
2.3.1.7.3	ArPatchVersion	175
2.3.1.7.4	ModuleId	175
2.3.1.7.5	Release	175
2.3.1.7.6	SwMajorVersion	176
2.3.1.7.7	SwMinorVersion	176
2.3.1.7.8	SwPatchVersion	177
2.3.1.7.9	VendorId	177
2.3.1.8	Container: DsadcComparatorConfiguration	178
2.3.1.8.1	DsadcComparatorEventSelect	178
2.3.1.8.2	DsadcLowerBoundaryValue	178
2.3.1.8.3	DsadcUpperBoundaryValue	179
2.3.1.9	Container: DsadcConfigSet	179
2.3.1.10	Container: DsadcDemEventParameterRefs	179
2.3.1.10.1	DsadcClcFailureNotification	180
2.3.1.10.2	DsadcFifoFailureNotification	180
2.3.1.11	Container: DsadcDemodulatorConfiguration	180
2.3.1.11.1	DsadcIntegratorTriggerMode	181
2.3.1.11.2	DsadcResultDisplayMode	181
2.3.1.11.3	DsadcTriggerSelect	182
2.3.1.12	Container: DsadcErsEtlConfig	183
2.3.1.12.1	DsadcEruErsInputPin	183
2.3.1.12.2	DsadcEruErsRef	183
2.3.1.12.3	DsadcEruStatusFlagConfig	184
2.3.1.13	Container: DsadcFilterConfiguration	185
2.3.1.13.1	DsadcAlternateServiceReq	185
2.3.1.13.2	DsadcCICFilterDecimationFactor	185
2.3.1.13.3	DsadcCICFilterStartValue	186
2.3.1.13.4	DsadcFIR0FilterEnable	186
2.3.1.13.5	DsadcFIR1FilterDecimationEnable	187
2.3.1.13.6	DsadcFIR1FilterEnable	187
2.3.1.13.7	DsadcOffsetCompFilterEnable	188
2.3.1.13.8	DsadcOffsetCompValue	189
2.3.1.13.9	DsadcOffsetCompValueProtect	190
2.3.1.13.10	DsadcOvershootCompensationEn	190
2.3.1.13.11	DsadcPreFilterEnable	191

Table of contents

2.3.1.14	Container: DsadcGainCalibConfig	191
2.3.1.15	Container: DsadcGainCorrConfig	191
2.3.1.15.1	DsadcCICFilterOutputShiftPos	192
2.3.1.15.2	DsadcGainCorrMulFactor	192
2.3.1.16	Container: DsadcGeneral	193
2.3.1.16.1	DsadcDeInitApi	193
2.3.1.16.2	DsadcDevErrorDetect	193
2.3.1.16.3	DsadcInitCheckApi	194
2.3.1.16.4	DsadcInitDeInitApiMode	194
2.3.1.16.5	DsadcRuntimeApiMode	195
2.3.1.16.6	DsadcSafetyEnable	195
2.3.1.16.7	DsadcVersionInfoApi	196
2.3.1.17	Container: DsadcGlobalConfiguration	196
2.3.1.17.1	DsadcDitheringTrimValue	197
2.3.1.17.2	DsadcSleepMode	197
2.3.1.17.3	DsadcSupplyVoltageLevel	198
2.3.1.17.4	DsadcSyncClockGen	198
2.3.1.18	Container: DsadcIntegratorConfiguration	199
2.3.1.18.1	DsadcDiscardCount	199
2.3.1.18.2	DsadcIntegrationCount	199
2.3.1.19	Container: DsadcModulatorConfiguration	200
2.3.1.19.1	DsadcAnalogClockSyncDelay	200
2.3.1.19.2	DsadcClockDivider	200
2.3.1.19.3	DsadcDitheringEnable	201
2.3.1.19.4	DsadcInputGain	202
2.3.1.19.5	DsadcInputMuxActionMode	202
2.3.1.19.6	DsadcInputMuxControlMode	203
2.3.1.19.7	DsadcInputPinSelection	203
2.3.1.19.8	DsadcIntegratorResetEnable	204
2.3.1.19.9	DsadcNegativeInputLine	205
2.3.1.19.10	DsadcPositiveInputLine	205
2.3.1.20	Container: DsadcOguConfig	206
2.3.1.20.1	DsadcEruErsCh0PatternFlagEnable	206
2.3.1.20.2	DsadcEruErsCh1PatternFlagEnable	206
2.3.1.20.3	DsadcEruErsCh2PatternFlagEnable	207
2.3.1.20.4	DsadcEruErsCh3PatternFlagEnable	208
2.3.1.20.5	DsadcEruErsCh4PatternFlagEnable	208
2.3.1.20.6	DsadcEruErsCh5PatternFlagEnable	209
2.3.1.20.7	DsadcEruErsCh6PatternFlagEnable	209
2.3.1.20.8	DsadcEruErsCh7PatternFlagEnable	210
2.3.1.20.9	DsadcEruOguRef	210
2.3.1.21	Container: DsadcOvershootCompenConfig	211

Table of contents

2.3.1.21.1	DsadcSlewRateFilterRunTime	211
2.3.1.21.2	DsadcSlewRateFilterStrength	211
2.3.1.21.3	DsadcStepDetectionMode	212
2.3.1.21.4	DsadcStepDetectionThreshold	212
2.3.1.22	Container: DsadcRectificationConfiguration	213
2.3.1.22.1	DsadcNegSignDelayValue	213
2.3.1.22.2	DsadcPosSignDelayValue	214
2.3.1.22.3	DsadcRectificationEnable	214
2.3.1.22.4	DsadcSignSignalChannel	215
2.3.1.22.5	DsadcSignSignalSource	215
2.3.1.23	Container: DsadcTimestampConfiguration	216
2.3.1.23.1	DsadcInputMuxSetCopyEnable	216
2.3.1.23.2	DsadcTimestampCounterClockSel	217
2.3.2	Functions - Type definitions	217
2.3.2.1	Dsadc_CalibrationStatusType	217
2.3.2.2	Dsadc_ChannelStatusType	218
2.3.2.3	Dsadc_ChannelType	218
2.3.2.4	Dsadc_ConfigType	218
2.3.2.5	Dsadc_NotifyFnPtrType	219
2.3.2.6	Dsadc_ResultType	219
2.3.2.7	Dsadc_SizeType	219
2.3.2.8	Dsadc_TimeStampType	220
2.3.3	Functions - APIs	220
2.3.3.1	Dsadc_Init	220
2.3.3.2	Dsadc_Delnit	221
2.3.3.3	Dsadc_StartModulation	222
2.3.3.4	Dsadc_StopModulation	223
2.3.3.5	Dsadc_ReadStreamResults	224
2.3.3.6	Dsadc_ReadResult	226
2.3.3.7	Dsadc_GetStatus	227
2.3.3.8	Dsadc_SetupResultBuffer	228
2.3.3.9	Dsadc_StartCarrierSignal	230
2.3.3.10	Dsadc_StopCarrierSignal	231
2.3.3.11	Dsadc_EnableNotifications	231
2.3.3.12	Dsadc_DisableNotifications	232
2.3.3.13	Dsadc_GetTimestamp	233
2.3.3.14	Dsadc_StartCalibration	234
2.3.3.15	Dsadc_GetCalibrationStatus	235
2.3.3.16	Dsadc_InitCheck	237
2.3.3.17	Dsadc_GetVersionInfo	238
2.3.4	Notifications and Callbacks	238
2.3.4.1	Dsadc_Timerlsr	239

Table of contents

2.3.5	Scheduled functions	240
2.3.6	Interrupt service routines	240
2.3.6.1	Dsadc_Isr	240
2.3.7	Error codes classification	241
2.3.7.1	Development errors	241
2.3.7.2	Production errors	243
2.3.7.3	Safety errors	244
2.3.7.4	Runtime errors	244
2.3.8	Deviations and limitations	244
2.3.8.1	Deviations	244
2.3.8.2	Limitations	245
2.3.9	Unsupported hardware features	245
3	FLSLOADER driver	246
3.1	User information	246
3.1.1	Description	246
3.1.2	Hardware-software mapping	246
3.1.2.1	DMU: primary hardware peripheral	247
3.1.2.2	SCU: dependent hardware peripheral	248
3.1.3	File structure	249
3.1.3.1	C file structure	249
3.1.3.2	Code generator plugin files	250
3.1.4	Integration hints	251
3.1.4.1	Integration with AUTOSAR stack	252
3.1.4.2	Multicore and Resource Manager	255
3.1.4.3	MCU support	255
3.1.4.4	Port support	255
3.1.4.5	DMA support	256
3.1.4.6	Interrupt connections	256
3.1.4.7	Example usage	257
3.1.5	Key architectural considerations	272
3.1.5.1	User mode is not supported by the driver	272
3.2	Assumptions of Use (AoUs)	273
3.3	Reference information	274
3.3.1	Configuration interfaces	274
3.3.1.1	Container: CommonPublishedInformation	276
3.3.1.1.1	ArMajorVersion	277
3.3.1.1.2	ArMinorVersion	277
3.3.1.1.3	ArPatchVersion	277
3.3.1.1.4	ModuleId	278
3.3.1.1.5	Release	278
3.3.1.1.6	SwMajorVersion	279

Table of contents

3.3.1.1.7	SwMinorVersion	279
3.3.1.1.8	SwPatchVersion	280
3.3.1.1.9	VendorId	280
3.3.1.2	Container: FlsLoader	280
3.3.1.3	Container: FlsLoaderDFlash0ProtConfig	280
3.3.1.3.1	FlsLoaderDF0Prot	281
3.3.1.3.2	FlsLoaderDF0UcbPW0_0	281
3.3.1.3.3	FlsLoaderDF0UcbPW0_1	282
3.3.1.3.4	FlsLoaderDF0UcbPW1_0	282
3.3.1.3.5	FlsLoaderDF0UcbPW1_1	282
3.3.1.3.6	FlsLoaderDF0UcbPW2_0	283
3.3.1.3.7	FlsLoaderDF0UcbPW2_1	283
3.3.1.3.8	FlsLoaderDF0UcbPW3_0	284
3.3.1.3.9	FlsLoaderDF0UcbPW3_1	284
3.3.1.4	Container: FlsLoaderDFLASHConfig	285
3.3.1.5	Container: FlsLoaderGeneral	285
3.3.1.5.1	FlsLoaderCallOutFunction	285
3.3.1.5.2	FlsLoaderCallOutTime	286
3.3.1.5.3	FlsLoaderDevErrorDetect	286
3.3.1.5.4	FlsLoaderEnableLockCheck	287
3.3.1.6	Container: FlsLoaderOptionalApi	287
3.3.1.6.1	FlsLoaderDeInitApi	287
3.3.1.6.2	FlsLoaderLockUnlockApi	288
3.3.1.6.3	FlsLoaderVersionInfoApi	288
3.3.1.7	Container: FlsLoaderPF0Sector	289
3.3.1.7.1	FlsLoaderPFSectorWriteProtection	289
3.3.1.8	Container: FlsLoaderPF1Sector	290
3.3.1.8.1	FlsLoaderPFSectorWriteProtection	290
3.3.1.9	Container: FlsLoaderPF2Sector	290
3.3.1.9.1	FlsLoaderPFSectorWriteProtection	291
3.3.1.10	Container: FlsLoaderPF3Sector	291
3.3.1.10.1	FlsLoaderPFSectorWriteProtection	291
3.3.1.11	Container: FlsLoaderPF4Sector	292
3.3.1.11.1	FlsLoaderPFSectorWriteProtection	292
3.3.1.12	Container: FlsLoaderPF5Sector	292
3.3.1.12.1	FlsLoaderPFSectorWriteProtection	293
3.3.1.13	Container: FlsLoaderPFlash0ProtConfig	293
3.3.1.13.1	FlsLoaderPF0UcbPW0_0	293
3.3.1.13.2	FlsLoaderPF0UcbPW0_1	294
3.3.1.13.3	FlsLoaderPF0UcbPW1_0	294
3.3.1.13.4	FlsLoaderPF0UcbPW1_1	295
3.3.1.13.5	FlsLoaderPF0UcbPW2_0	295

Table of contents

3.3.1.13.6	FlsLoaderPF0UcbPW2_1	296
3.3.1.13.7	FlsLoaderPF0UcbPW3_0	296
3.3.1.13.8	FlsLoaderPF0UcbPW3_1	296
3.3.1.13.9	FlsLoaderPFlash0WriteProt	297
3.3.1.14	Container: FlsLoaderPFlash1ProtConfig	297
3.3.1.14.1	FlsLoaderPFlash1WriteProt	298
3.3.1.15	Container: FlsLoaderPFlash2ProtConfig	298
3.3.1.15.1	FlsLoaderPFlash2WriteProt	298
3.3.1.16	Container: FlsLoaderPFlash3ProtConfig	299
3.3.1.16.1	FlsLoaderPFlash3WriteProt	299
3.3.1.17	Container: FlsLoaderPFlash4ProtConfig	300
3.3.1.17.1	FlsLoaderPFlash4WriteProt	300
3.3.1.18	Container: FlsLoaderPFlash5ProtConfig	300
3.3.1.18.1	FlsLoaderPFlash5WriteProt	301
3.3.1.19	Container: FlsLoaderPFLASHConfig	301
3.3.2	Functions - Type definitions	301
3.3.2.1	FlsLoader_AddressType	301
3.3.2.2	FlsLoader_CallOutFunc	302
3.3.2.3	FlsLoader_ConfigType	302
3.3.2.4	FlsLoader_LengthType	302
3.3.2.5	FlsLoader_ReturnType	302
3.3.2.6	FlsLoader_ValueType	303
3.3.3	Functions - APIs	303
3.3.3.1	FlsLoader_DelInit	303
3.3.3.2	FlsLoader_Erase	304
3.3.3.3	FlsLoader_GetVersionInfo	305
3.3.3.4	FlsLoader_Init	306
3.3.3.5	FlsLoader_Lock	307
3.3.3.6	FlsLoader_UnLock	308
3.3.3.7	FlsLoader_Write	309
3.3.4	Notifications and Callbacks	310
3.3.5	Scheduled functions	311
3.3.6	Interrupt service routines	311
3.3.7	Error codes classification	311
3.3.7.1	Development errors	311
3.3.7.2	Production errors	311
3.3.7.3	Safety errors	311
3.3.7.4	Runtime errors	311
3.3.8	Deviations and limitations	312
3.3.8.1	Deviations	312
3.3.8.2	Limitations	312
3.3.9	Unsupported hardware features	312

Table of contents

4	SMU driver	313
4.1	User information	313
4.1.1	Description	313
4.1.2	Hardware-software mapping	313
4.1.2.1	PMS: primary hardware peripheral	314
4.1.2.2	SCU: dependent hardware peripheral	315
4.1.2.3	Port: dependent hardware peripheral	315
4.1.2.4	SMU: primary hardware peripheral	316
4.1.2.5	SRC: dependent hardware peripheral	316
4.1.3	File structure	317
4.1.3.1	C file structure	317
4.1.3.2	Code generator plugin files	319
4.1.4	Integration hints	320
4.1.4.1	Integration with AUTOSAR stack	320
4.1.4.2	Multicore and Resource Manager	323
4.1.4.3	MCU support	323
4.1.4.4	Port support	324
4.1.4.5	DMA support	324
4.1.4.6	Interrupt connections	324
4.1.4.7	Example usage	325
4.1.5	Key architectural considerations	335
4.1.5.1	Clearing alarm status during initialization	335
4.1.5.2	Initialization and de-initialization	335
4.1.5.3	SMU_core state machine transitions	335
4.1.5.4	Recovery timer handling	335
4.2	Assumptions of Use (AoUs)	336
4.3	Reference information	337
4.3.1	Configuration interfaces	337
4.3.1.1	Container: CommonPublishedInformation	337
4.3.1.1.1	ArMajorVersion	338
4.3.1.1.2	ArMinorVersion	338
4.3.1.1.3	ArPatchVersion	338
4.3.1.1.4	ModuleId	339
4.3.1.1.5	Release	339
4.3.1.1.6	SwMajorVersion	340
4.3.1.1.7	SwMinorVersion	340
4.3.1.1.8	SwPatchVersion	341
4.3.1.1.9	VendorId	341
4.3.1.2	Container: Smu	341
4.3.1.2.1	ConfigVariant	342
4.3.1.3	Container: SmuConfigSet	342

Table of contents

4.3.1.4	Container: SmuCoreAlarmBehavior	342
4.3.1.4.1	SmuCoreAlarmFSP	342
4.3.1.4.2	SmuCoreAlarmIntBeh	343
4.3.1.4.3	SmuCoreAlmBehaviourId	344
4.3.1.5	Container: SmuCoreAlarmGlobalConfig	344
4.3.1.5.1	SmuCoreCpu0ResetRequest	344
4.3.1.5.2	SmuCoreCpu1ResetRequest	345
4.3.1.5.3	SmuCoreCpu2ResetRequest	345
4.3.1.5.4	SmuCoreCpu3ResetRequest	346
4.3.1.5.5	SmuCoreCpu4ResetRequest	346
4.3.1.5.6	SmuCoreCpu5ResetRequest	347
4.3.1.5.7	SmuCoreCpuResetActivatePES	347
4.3.1.5.8	SmuCoreEnableFaultToRunState	348
4.3.1.5.9	SmuCoreIGCS0ActivatePES	349
4.3.1.5.10	SmuCoreIGCS1ActivatePES	349
4.3.1.5.11	SmuCoreIGCS2ActivatePES	350
4.3.1.5.12	SmuCoreInterruptSet0	350
4.3.1.5.13	SmuCoreInterruptSet1	351
4.3.1.5.14	SmuCoreInterruptSet2	352
4.3.1.5.15	SmuCoreNMIActivatePES	353
4.3.1.6	Container: SmuCoreAlarmGroup	354
4.3.1.6.1	SmuCoreAlmGrpId	354
4.3.1.7	Container: SmuCoreConfig	354
4.3.1.8	Container: SmuCoreFSPHandling	354
4.3.1.8.1	SmuCoreFSPFaultStateDuration	355
4.3.1.8.2	SmuCoreFSPPrescaler1	355
4.3.1.8.3	SmuCoreFSPPrescaler2	356
4.3.1.8.4	SmuCoreFSPSignalingMode	357
4.3.1.8.5	SmuCorePESOnFSP	358
4.3.1.9	Container: SmuCoreRecoveryTimer	358
4.3.1.9.1	SmuCoreEnableRT0	358
4.3.1.9.2	SmuCoreEnableRT1	359
4.3.1.9.3	SmuCoreRTDuration	359
4.3.1.10	Container: SmuCoreRT0Alarm	360
4.3.1.10.1	SmuCoreRT0AlarmGroupId	360
4.3.1.10.2	SmuCoreRT0AlarmId	361
4.3.1.11	Container: SmuCoreRT1Alarm	362
4.3.1.11.1	SmuCoreRT1AlarmGroupId	362
4.3.1.11.2	SmuCoreRT1AlarmId	363
4.3.1.12	Container: SmuDemEventParameterRefsConf	364
4.3.1.12.1	SmuActivateFSPFailureNotification	364
4.3.1.12.2	SmuActivatePESFailureNotification	364

Table of contents

4.3.1.12.3	SmuActivateRunStateFailureNotification	365
4.3.1.12.4	SmuClearAlarmStatusFailureNotification	365
4.3.1.12.5	SmuCoreAliveFailureNotification	366
4.3.1.12.6	SmuRTStopFailureNotification	366
4.3.1.12.7	SmuReleaseFSPFailureNotification	367
4.3.1.12.8	SmuSetAlarmStatusFailureNotification	367
4.3.1.12.9	SmuSffFailureNotification	368
4.3.1.13	Container: SmuGeneral	368
4.3.1.13.1	SmuAGStatusTimeout	368
4.3.1.13.2	SmuCoreFSP0OutputEnable	369
4.3.1.13.3	SmuCoreFSP0PortEnable	369
4.3.1.13.4	SmuCoreFSP1OutputEnable	370
4.3.1.13.5	SmuCoreFSP1PortEnable	370
4.3.1.13.6	SmuCoreGlitchFilterSCU	371
4.3.1.13.7	SmuCoreGlitchFilterSTS	371
4.3.1.13.8	SmuDevErrorDetect	372
4.3.1.13.9	SmuInitCheckApi	372
4.3.1.13.10	SmuInitDelInitApiMode	373
4.3.1.13.11	SmuRuntimeApiMode	373
4.3.1.13.12	SmuSafetyEnable	374
4.3.1.13.13	SmuStdbyEnable	374
4.3.1.13.14	SmuVersionInfoApi	375
4.3.1.14	Container: SmuStdbyAlarmBehavior	375
4.3.1.14.1	SmuStdbyAlarmFSP	376
4.3.1.14.2	SmuStdbyAlmBehaviourId	376
4.3.1.15	Container: SmuStdbyAlarmGlobalConfig	376
4.3.1.15.1	SmuStdbyEnableFSP0	377
4.3.1.15.2	SmuStdbyEnableFSP1	377
4.3.1.16	Container: SmuStdbyAlarmGroup	378
4.3.1.16.1	SmuStdbyAlmGrpId	378
4.3.1.17	Container: SmuStdbyConfig	378
4.3.2	Functions - Type definitions	378
4.3.2.1	Smu_AlarmGroupId	378
4.3.2.2	Smu_AlarmIdType	379
4.3.2.3	Smu_ConfigType	380
4.3.2.4	Smu_CoreAlarmActionType	380
4.3.2.5	Smu_CoreCommandType	381
4.3.2.6	Smu_CoreStateType	382
4.3.2.7	Smu_EnableRunStateType	383
4.3.2.8	Smu_FSPActionType	383
4.3.2.9	Smu_SffTestResType	384
4.3.3	Functions - APIs	384

Table of contents

4.3.3.1	Smu_Init	384
4.3.3.2	Smu_Delnit	385
4.3.3.3	Smu_GetAlarmAction	386
4.3.3.4	Smu_SetAlarmAction	388
4.3.3.5	Smu_ClearAlarmStatus	389
4.3.3.6	Smu_GetAlarmStatus	390
4.3.3.7	Smu_SetAlarmStatus	391
4.3.3.8	Smu_GetAlarmDebugStatus	392
4.3.3.9	Smu_LockConfigRegs	393
4.3.3.10	Smu_ReleaseFSP	395
4.3.3.11	Smu_ActivateFSP	396
4.3.3.12	Smu_SetupErrorPin	397
4.3.3.13	Smu_ReleaseErrorPin	398
4.3.3.14	Smu_RTStop	399
4.3.3.15	Smu_GetRTMissedEvent	400
4.3.3.16	Smu_ActivatePES	401
4.3.3.17	Smu_RegisterMonitor	402
4.3.3.18	Smu_GetSmuState	403
4.3.3.19	Smu_ActivateRunState	404
4.3.3.20	Smu_GetVersionInfo	405
4.3.3.21	Smu_CoreAliveTest	406
4.3.3.22	Smu_InitCheck	407
4.3.3.23	Smu_GetAlarmExecutionStatus	408
4.3.3.24	Smu_ClearAlarmExecutionStatus	409
4.3.4	Notifications and Callbacks	410
4.3.5	Scheduled functions	410
4.3.6	Interrupt service routines	410
4.3.7	Error codes classification	410
4.3.7.1	Development errors	411
4.3.7.2	Production errors	413
4.3.7.3	Safety errors	414
4.3.7.4	Runtime errors	414
4.3.8	Deviations and limitations	414
4.3.8.1	Deviations	414
4.3.8.2	Limitations	414
4.3.9	Unsupported hardware features	414
5	UART driver	415
5.1	User information	415
5.1.1	Description	415
5.1.2	Hardware-software mapping	415
5.1.2.1	ASCLIN: primary hardware peripheral	415

Table of contents

5.1.2.2	SCU: dependent hardware peripheral	416
5.1.2.3	Port: dependent hardware peripheral	417
5.1.2.4	SRC: dependent hardware peripheral	417
5.1.3	File structure	417
5.1.3.1	C file structure	417
5.1.3.2	Code generator plugin files	419
5.1.4	Integration hints	420
5.1.4.1	Integration with AUTOSAR stack	420
5.1.4.2	Multicore and Resource Manager	422
5.1.4.3	MCU support	422
5.1.4.4	Port support	425
5.1.4.5	DMA support	427
5.1.4.6	Interrupt connections	427
5.1.4.7	Example usage	431
5.1.5	Key architectural considerations	443
5.2	Assumptions of Use (AoUs)	444
5.3	Reference information	445
5.3.1	Configuration interfaces	445
5.3.1.1	Container: CommonPublishedInformation	445
5.3.1.1.1	ArMajorVersion	445
5.3.1.1.2	ArMinorVersion	446
5.3.1.1.3	ArPatchVersion	446
5.3.1.1.4	ModuleId	446
5.3.1.1.5	Release	447
5.3.1.1.6	SWMajorVersion	447
5.3.1.1.7	SWMinorVersion	448
5.3.1.1.8	SWPatchVersion	448
5.3.1.1.9	VendorId	449
5.3.1.2	Container: UartChannel	449
5.3.1.2.1	UartAutoCalcBaudParams	449
5.3.1.2.2	UartBaudRate	450
5.3.1.2.3	UartCTSEnable	450
5.3.1.2.4	UartCTSPinSelection	451
5.3.1.2.5	UartCTSPolarity	451
5.3.1.2.6	UartChanBaudDenominator	452
5.3.1.2.7	UartChanBaudNumerator	453
5.3.1.2.8	UartChanBaudOverSampling	453
5.3.1.2.9	UartChanBaudPrescalar	454
5.3.1.2.10	UartChannelId	454
5.3.1.2.11	UartDataLength	455
5.3.1.2.12	UartHwUnit	455
5.3.1.2.13	UartParityBit	456

Table of contents

5.3.1.2.14	UartRxChannelMode	456
5.3.1.2.15	UartRxPinSelection	457
5.3.1.2.16	UartStopBits	458
5.3.1.2.17	UartTxChannelMode	458
5.3.1.3	Container: UartConfigSet	458
5.3.1.4	Container: UartGeneral	459
5.3.1.4.1	UartAbortReadApi	459
5.3.1.4.2	UartAbortWriteApi	459
5.3.1.4.3	UartClockRef	460
5.3.1.4.4	UartCsrClksel	460
5.3.1.4.5	UartDeInitApi	461
5.3.1.4.6	UartDevErrorDetect	461
5.3.1.4.7	UartIndex	462
5.3.1.4.8	UartInitCheckApi	462
5.3.1.4.9	UartInitDeInitApiMode	463
5.3.1.4.10	UartMainFunctionReadPeriod	463
5.3.1.4.11	UartMainFunctionWritePeriod	464
5.3.1.4.12	UartRunTimeErrorDetect	464
5.3.1.4.13	UartSafetyEnable	465
5.3.1.4.14	UartSleepEnable	465
5.3.1.4.15	UartTimeoutCount	466
5.3.1.4.16	UartVersionInfoApi	466
5.3.1.5	Container: UartNotification	467
5.3.1.5.1	UartAbortReceiveNotifPtr	467
5.3.1.5.2	UartAbortTransmitNotifPtr	468
5.3.1.5.3	UartReceiveNotifPtr	468
5.3.1.5.4	UartTransmitNotifPtr	469
5.3.1.6	Container: Uart	469
5.3.2	Functions - Type definitions	469
5.3.2.1	Uart_ChannelIdType	469
5.3.2.2	Uart_ConfigType	470
5.3.2.3	Uart_ErrorIdType	470
5.3.2.4	Uart_MemType	470
5.3.2.5	Uart_NotificationPtrType	470
5.3.2.6	Uart_ReturnType	471
5.3.2.7	Uart_SizeType	471
5.3.2.8	Uart_StatusType	471
5.3.2.9	UartNotificationCallback	472
5.3.3	Functions - APIs	472
5.3.3.1	Uart_InitCheck	472
5.3.3.2	Uart_Init	473
5.3.3.3	Uart_Read	474

Table of contents

5.3.3.4	Uart_Write	475
5.3.3.5	Uart_AbortRead	476
5.3.3.6	Uart_AbortWrite	477
5.3.3.7	Uart_GetStatus	478
5.3.3.8	Uart_DelInit	479
5.3.3.9	Uart_GetVersionInfo	480
5.3.4	Notifications and Callbacks	481
5.3.5	Scheduled functions	481
5.3.5.1	Uart_MainFunction_Read	481
5.3.5.2	Uart_MainFunction_Write	482
5.3.6	Interrupt service routines	483
5.3.6.1	Uart_IsrError	483
5.3.6.2	Uart_IsrReceive	484
5.3.6.3	Uart_IsrTransmit	485
5.3.7	Error codes classification	486
5.3.7.1	Development errors	486
5.3.7.2	Production errors	487
5.3.7.3	Safety errors	487
5.3.7.4	Runtime errors	487
5.3.8	Deviations and limitations	488
5.3.8.1	Deviations	488
5.3.8.2	Limitations	488
5.3.9	Unsupported hardware features	488
	Revision history	489
	Disclaimer	492

DMA driver**1 DMA driver****1.1 User information****1.1.1 Description**

The DMA driver is responsible for providing the services and configuration options, for performing the DMA operations using the AURIX DMA hardware. Services for initialization, starting, stopping and updating the DMA channels are provided by the driver. The driver is designed as a post-build variant and it is possible to generate a hex file specifically for a desired configuration.

1.1.2 Hardware-software mapping

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

DMA driver

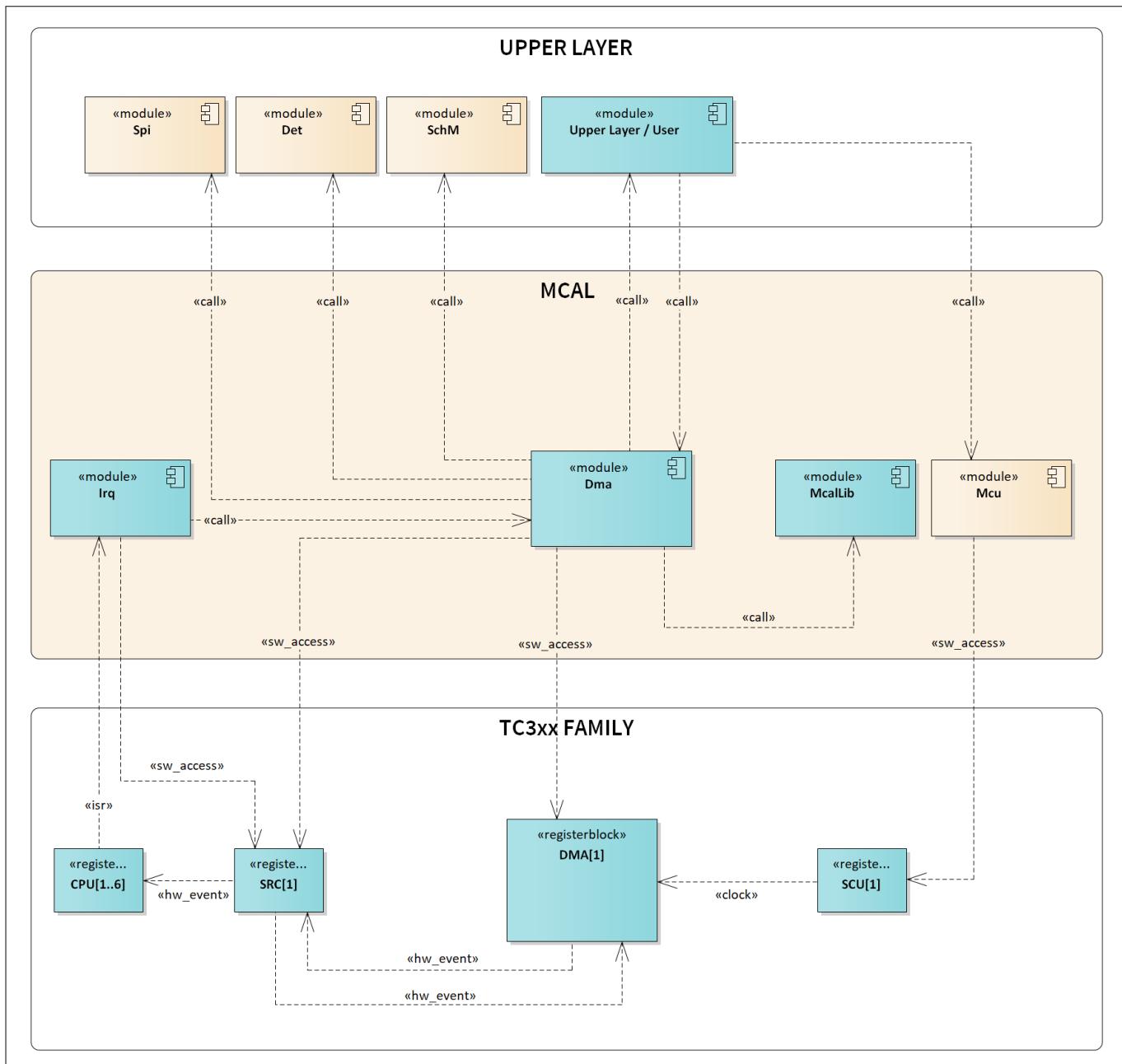


Figure 1 **Mapping of hardware-software interfaces**

1.1.2.1 DMA: primary hardware peripheral

Hardware functional features

The DMA driver uses the DMA channels and the DMA move engine of the DMA IP for DMA transactions. The DMA channel holds the transfer characteristics such as the source address, destination address, transfer length, triggering criteria and so on. The move engine is a bus master and performs the physical movement of data from the source memory (SRAM, peripherals) to the destination memory (SRAM, peripherals) based on the transfer characteristics. The transfer characteristics can be configured using the configuration tools. In addition, the transaction control set can be updated through the driver APIs.

The key hardware functional features used by the driver are:

- Double buffering operations - source and destination side
- Circular buffer - source and destination

DMA driver

- Linked list - accumulated linked list and safe linked list
- Channel trigger - by the software and the hardware
- Daisy chaining
- CRC calculations for address and data
- Time stamping of the transactions
- Interrupts from channel and resource partitions

The unsupported features of the DMA hardware are:

- Pattern detection feature
- Conditional linked list

Users of the hardware

The DMA driver exclusively utilizes the DMA IP. The peripherals which require data transfer from one memory location to another should use the DMA driver APIs to perform the transfers.

Hardware diagnostic features

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

Hardware events

The DMA driver uses the following hardware events from the DMA IP:

- DMA channel interrupt service requests
- DMA resource partition interrupt service requests

In addition, the DMA being a service provider, can take up the service requests from other interrupt sources as well.

1.1.2.2 SRC: dependent hardware peripheral

Hardware functional features

The DMA driver depends on the interrupt router for raising an interrupt to the CPU based on the channel interrupt events and resource partition error interrupt events. DMA also depends on the interrupt router for routing the events to DMA, which are flagged by peripherals as data transfer requests.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software. However, the SRC registers for the GPSR (General Purpose Service Request) are accessed by the DMA driver, to route the error interrupt requests to the appropriate core in a multicore environment.

Hardware diagnostic features

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

Hardware events

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

1.1.2.3 SCU: dependent hardware peripheral

Hardware functional features

The DMA driver depends on the SCU IP for the clock and ENDINIT functionalities. For functioning, the driver requires the fSPB clock signal.

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.

DMA driver

Hardware diagnostic features

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

Hardware events

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

1.1.3 File structure

1.1.3.1 C file structure

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

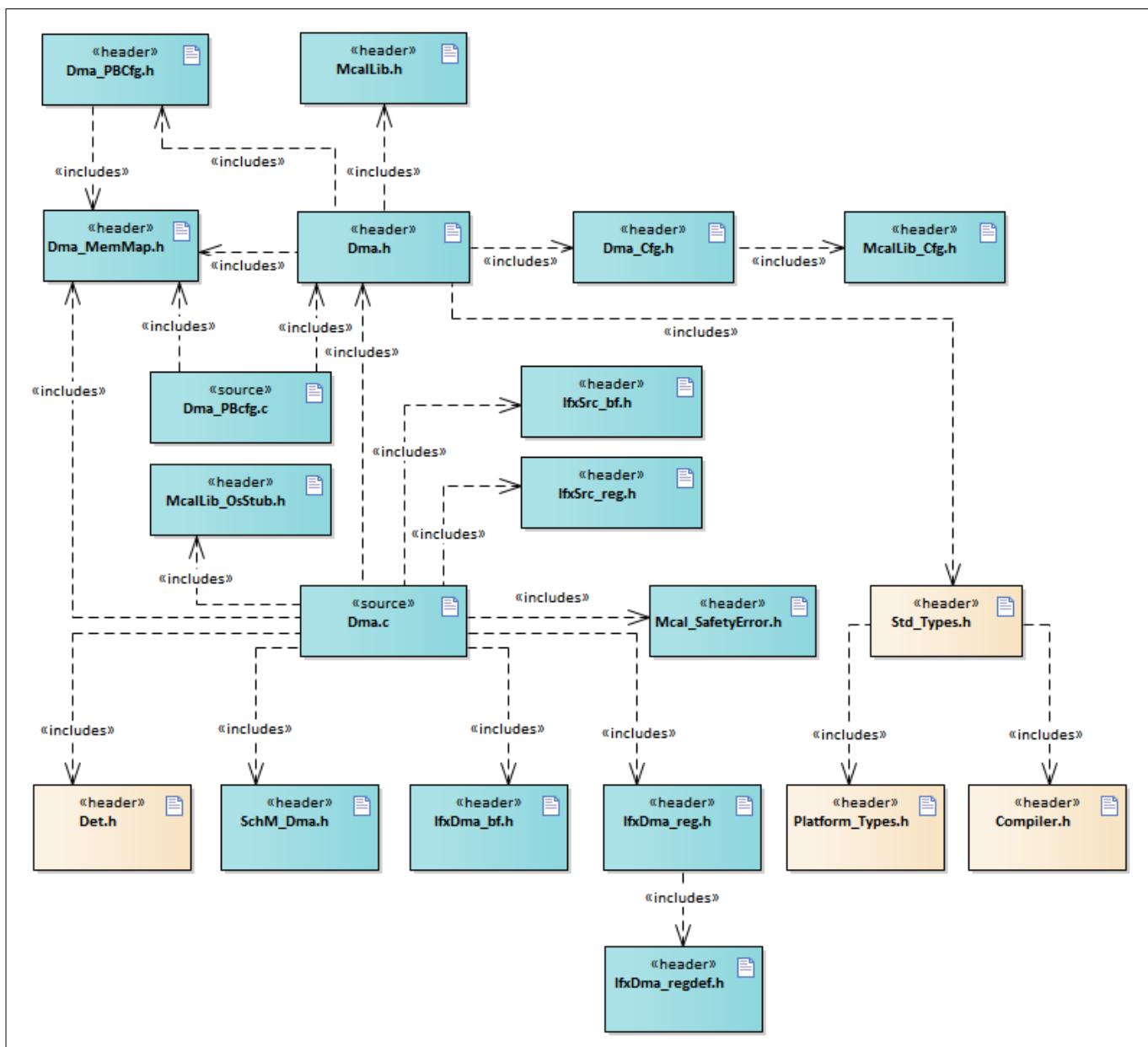


Figure 2 C file structure

DMA driver
Table 3 C file structure

File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Det.h	Provides the exported interfaces of Development Error Tracer
Dma.c	File (static) containing implementation of APIs
Dma.h	Header file (static) defining prototypes of data structures and APIs
Dma_Cfg.h	Header file (generated) containing constants and pre-processor macros
Dma_MemMap.h	File containing the memory section definitions used by the DMA driver
Dma_PBCfg.h	Post-Build header file for DMA driver
Dma_PBcfg.c	File (generated) containing objects to data structures
IfxDma_bf.h	SFR header file for DMA
IfxDma_reg.h	SFR header file for DMA
IfxDma_regdef.h	SFR header file for DMA
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_Cfg.h	Generated header file providing information on number of cores, DSPR, PSPR (start and end addresses) and system and backup clock information.
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
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Dma.h	Schedule manager header file for critical section management
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

1.1.3.2 Code generator plugin files

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

DMA driver



Figure 3 **Code generator plugin files**

Table 4 **Code generator plugin files**

File name	Description
Dma.bmd	AUTOSAR format XML data model schema file (for each device)
Dma.m	Code template macro file for DMA driver
Dma.xdm	Tresos format XML data model schema file
Dma_Bswmd.arxml	AUTOSAR format module description file
Dma_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the metadata for DMA driver
anchors.xml	Tresos anchors support file for the DMA 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 DMA driver
plugin.xml	Tresos plugin support file for the DMA driver

1.1.4 Integration hints

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

1.1.4.1 Integration with AUTOSAR stack

This section lists the modules that are not part of MCAL, but are required to integrate the DMA 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 the EcuM is used for initialization and de-initialization of the software

DMA driver

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 `Dma_MemMap.h` file.

The `Dma_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

DMA driver

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 DMA_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
    /***User pragmas here for Non-cached LMU***/
#define DMA_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined DMA_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
    #ifdef _TASKING_C_TRICORE_
        /***User pragmas here for Non-cached LMU***/
#define DMA_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
#define MEMMAP_ERROR

***** CORE[x] CONFIG DATA -- PF[x] ****/ /*[x]=0..5*/
#elif defined DMA_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
    /***User pragmas here for PF[x]***/
#define DMA_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined DMA_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
    /***User pragmas here for PF[x]***/
#define DMA_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
#define MEMMAP_ERROR

***** CORE[x] CONFIG DATA WITH 256bit ALIGNMENT -- PF[x] ****/ /*[x]=0..5*/
#elif defined DMA_START_SEC_CONFIG_DATA_ASIL_B_CORE0_256
    /***User pragmas here for PF[x], with 256 bit alignment***/
#define DMA_START_SEC_CONFIG_DATA_ASIL_B_CORE0_256
#define MEMMAP_ERROR
#elif defined DMA_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_256
    /***User pragmas here for PF[x], with 256 bit alignment***/
#define DMA_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_256
#define MEMMAP_ERROR

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

#endif

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

- **DET**

DMA driver

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The DMA driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the DMA 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 are required to be replaced with a complete DET module during the integration phase.

- **DEM**

The DEM module is not required for the integration of the DMA driver.

- **SchM**

The SchM module is a part of the RTE that manages the BSW scheduler. The DMA driver uses the exclusive areas defined in `SchM_Dma.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchM identified for the DMA driver is: `ChannelConfigUpdate`

The `SchM_Dma.h` and `SchM_Dma.c` files are provided in the MCAL package as an example code and are required to be updated by the integrator. The user must implement the SchM functions defined by the DMA 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_Dma.c *****/
#include "Os.h"

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

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

- **Safety error**

The DMA 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 DMA driver does not implement any notifications. However, it does report the interrupts raised by the DMA channels and error interrupts from the DMA Resource Partitions through notification functions. These notification functions can be configured by the user in the EB Tresos tool for each DMA channel and DMA Resource Partitions separately.

- **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 the application.

DMA driver

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

Note: *The DMA driver updates the SRC registers of GPSR (MODULE_SRC.GPSR.GPSR0.SR0) to clear the pending interrupt requests.*

1.1.4.2 Multicore and Resource Manager

The DMA driver supports execution of its APIs in simultaneously from all the CPU cores. The user has to allocate resources of the DMA 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 the multicore functionality in the driver:

- Initialization of the DMA driver should be invoked from each CPU core, which intends to use the services of the DMA driver. Initialization from logical master core must be completed prior to invoking the initialization services from other slave cores. The slave cores can execute initialization in parallel. When all the DMA resources are allocated to the master core, invoking the initialization from the master core alone is sufficient.
- DMA channels can be allocated to the any of the CPU cores. For example, DMAchannel-0 and DMAchannel-55 can be allocated to core-1; DMAchannel-21 and DMAchannel-35 can be allocated to core-2; DMAchannel-10 and DMAchannel-101 can be allocated to core-0 and so on.
- Any configured DMA channel, if unallocated in the Resource Manager, will be directly allocated to the master core.
- It must be ensured that the DMA channel number passed as a parameter while invoking a DMA API, belong to the same core on which the API was invoked.
- DETs and/or safety errors are raised in case APIs are invoked when there is a mismatch of core and DMA channel assignments.
- Channel interrupts raised by a DMA channel must be serviced by the CPU core to which the DMA channel has been allocated to. However, the Resource Partition interrupts must be serviced by the master CPU core.
- 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 DMA 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 particular core should be re-located to the DSPR/DLMU of the same core. Those marked as global should be relocated to the non-cached LMU region.

Configuration data and constants :

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

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

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

DMA driver**1.1.4.3 MCU support**

The DMA driver is dependent on the MCU driver for the clock configuration. The initialization of the DMA driver must be started only after completing the MCU initialization.

1.1.4.4 Port support

The DMA driver does not use any services provided by the PORT driver.

1.1.4.5 DMA support

Not applicable for the driver.

1.1.4.6 Interrupt connections

The interrupt connections of the DMA driver are described in this section:

DMA channel ISR

Each DMA channel has a dedicated interrupt. These interrupts are raised signifying a channel transfer completion, source wrap or a destination wrap. The channel interrupts may be enabled in the driver configuration. The integrator must ensure that the SR registers of these interrupts are configured appropriately.

DMA RP error ISR

Each Resource Partition has a dedicated interrupt line. These are the error interrupts possible during the DMA operations. These interrupts may be enabled in the driver configuration. In a multicore environment, the error interrupts are routed to the appropriate cores using a GPSR0 interrupt. The integrator must ensure that the SR registers of all these interrupts are configured appropriately.

Note: The DMA interrupts shall be assigned the interrupt priority higher than the interrupts from the users of DMA.

DMA driver

1.1.4.7 Example usage

The following are some of the key use cases of the DMA driver:

Note: For additional information, refer to the comments in the code snippets.

Initialization of DMA driver

The driver can be initialized by invoking the `Dma_Init()` API. During the invocation, the configuration structure pointer is passed as an input parameter.

Note: The driver gets initialized for the core (along with the channels allocated to the core) in which the API was invoked. The initialization should be carried out for each core where the driver is going to be used.

```
/* Include the header files */
#include "Dma.h"
#include "Mcu.h"

/* MCU Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(OU);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Initialize the DMA driver*/
Dma_Init(&Dma_Config);
/* Driver is initialized for the current core */
```

De-initializing DMA channels

A DMA channel can be selectively de-initialized if needed. The `Dma_ChDeInit()` API is used for the deinitialization.

```
/*
 * Configuration values mandatory for below code snippet:-
 * DmaDeinitApiConfiguration = TRUE
 * -> This is needed to invoke the channel de-init API.
 */

/* De-initialize the DMA channel 12 */
Dma_ChDeInit(12);
```

Starting a DMA channel

After the initialization is complete, the desired DMA channel can be started using the `Dma_ChStartTransfer()` API. Provide the channel number as the parameter for this API.

```
/* Start the DMA channel 5 */
Dma_ChStartTransfer(5);
```

Enabling the hardware trigger for the DMA channel

DMA driver

To enable the DMA channels to receive the interrupts from different sources, the `Dma_ChEnableHardwareTrigger()` API is used.

```
/*
 * Configuration values mandatory for below code snippet:-
 * DmaTriggerApiConfiguration = TRUE
 * -> This is needed to invoke the hardware trigger APIs.
 */

/* Enable the hardware triggers to the DMA channel 10 */
Dma_ChEnableHardwareTrigger(10);
```

Updating the channel settings during runtime

The driver provides the `Dma_ChUpdate()` API to update the following configuration items at the runtime – Source Address, Destination Address, Shadow Address, Address CRC, Data CRC and DMA channel configuration registers.

```
/* Perform an update of the source and destination addresses
 * of channel 10 using the Dma_ChUpdate API. */

/* The source and destination locations */
uint32 ChSourceAddress;
uint32 ChDestinationAddress;

/* Here, 'DmaChannelConfigVar' is a structure of type Dma_ConfigUpdateType,
 * holding the channel configuration to be used for updating the channel */
Dma_ConfigUpdateType DmaChannelConfigVar;

/* Step 1: Set the source and destination address pointers.
 * Step 2: Set the corresponding bit fields indicating that an update is needed
 * for the source and destination addresses */
DmaChannelConfigVar.SourceAddress = &ChSourceAddress;
DmaChannelConfigVar.DestAddress = &ChDestinationAddress;
DmaChannelConfigVar.UpdateAddressCrc = 0;
DmaChannelConfigVar.UpdateConfig = 0;
DmaChannelConfigVar.UpdateControlAdicr = 0;
DmaChannelConfigVar.UpdateControlChcsr = 0;
DmaChannelConfigVar.UpdateDataCrc = 0;
DmaChannelConfigVar.UpdateDestAddress = 1;
DmaChannelConfigVar.UpdateShadowConfig = 0;
DmaChannelConfigVar.UpdateSourceAddress = 1;

/* Perform the update using the driver API */
Dma_ChUpdate(10, &DmaChannelConfigVar, NULL_PTR);
```

Setting up of the linked list during configuration

The linked list for the DMA transactions can be setup during the configuration in Tresos. An example sequence is provided as follows, depicting the relevant parameters:

- **Step 1:** Configure the number of Transaction Control Sets (TCS) needed for the DMA channel.

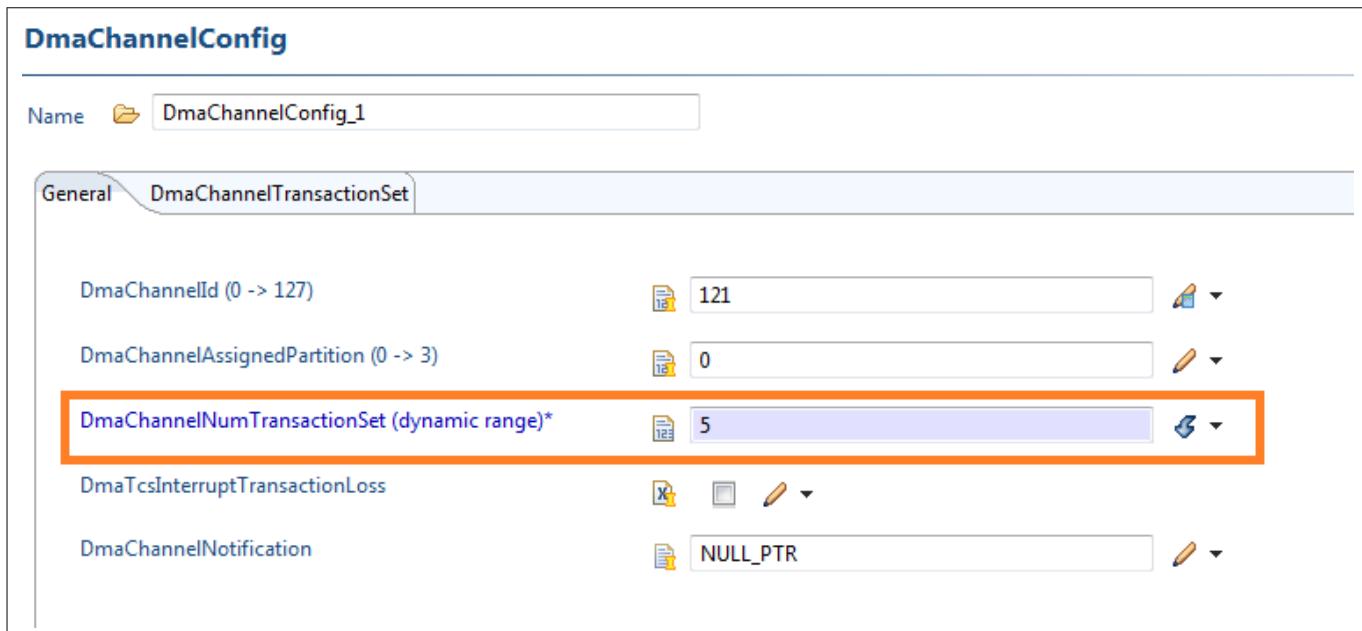
DMA driver

Figure 4 Step 1: Configure the number of Transaction Control Sets (TCS) needed for the DMA channel.

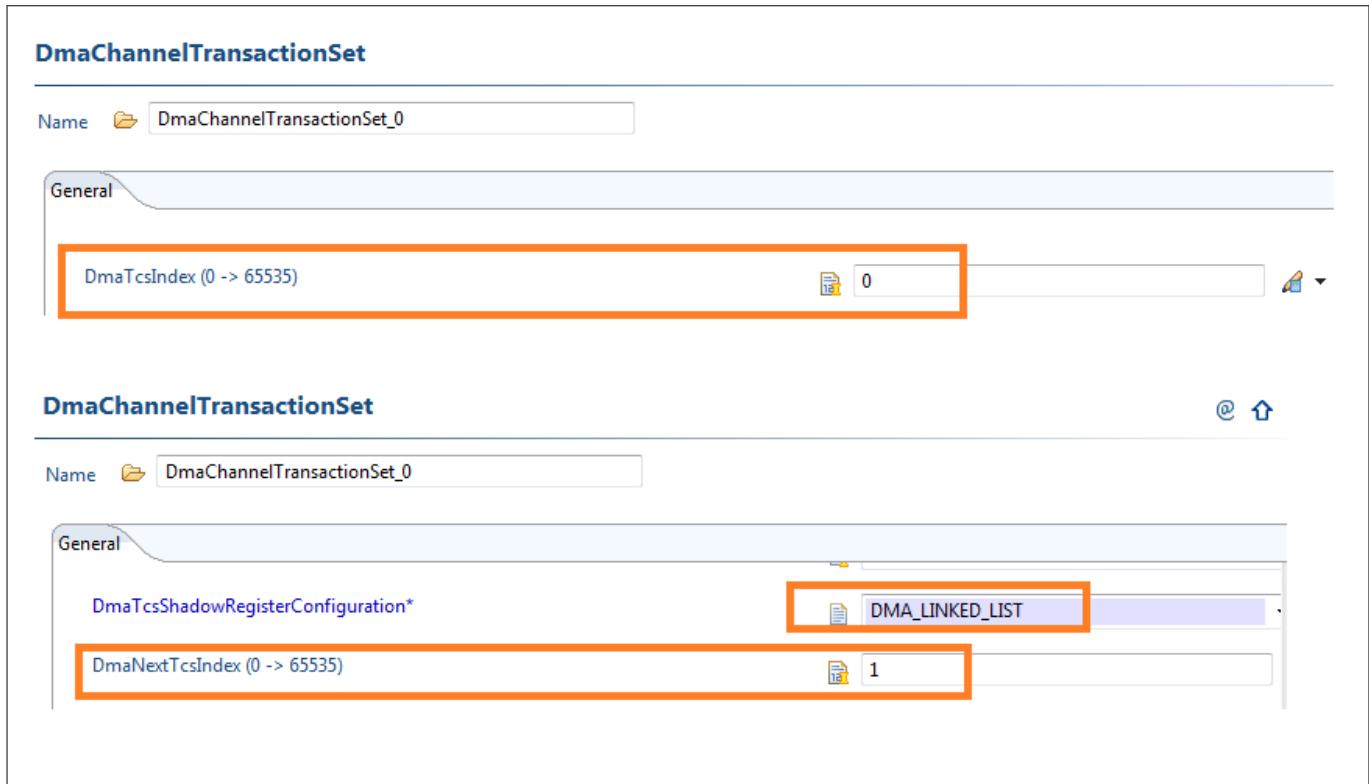
- **Step 2:** Configure and add the TCSes to the configuration.

The screenshot shows the 'DmaChannelConfig' dialog box with the 'DmaChannelTransactionSet' tab selected. Below it is a table titled 'DmaChannelTransactionSet*' with five rows, each representing a transaction set. The first row, 'DmaChannelTransactionSet_0', is highlighted with an orange border. The columns are labeled: Index, Name, DmaTcsIndex, DmaTcsS..., DmaTcsD..., and DmaTcsD... . All values in the table are currently set to 0x0.

Index	Name	DmaTcsIndex	DmaTcsS...	DmaTcsD...	DmaTcsD...
0	DmaChannelTransactionSet_0	0	0x0	0x0	0x0
1	DmaChannelTransactionSet_1	1	0x0	0x0	0x0
2	DmaChannelTransactionSet_2	2	0x0	0x0	0x0
3	DmaChannelTransactionSet_3	3	0x0	0x0	0x0
4	DmaChannelTransactionSet_4	4	0x0	0x0	0x0

Figure 5 Step 2: Configure and add the TCSes to the configuration.

- **Step 3:** Configure the order of the TCSes in Tresos.

DMA driver**Figure 6 Step 3: Configure the order of the TCSes in Tresos.**

Add the channel-specific configurations (source address, destination address and so on) to the TCSes and the channel can be used for the linked list.

Setting up of the linked list during runtime

DMA driver

The linked list for the DMA transactions can be setup during the runtime also. An example sequence is provided as follows, depicting the relevant parameters:

```

/*
 * Configuration values mandatory for below code snippet:-
 * DmaTcsShadowRegisterConfiguration = DMA_LINKED_LIST or
 * DMA_ACCUMULATED_LINKED_LIST or
 * DMA_SAFE_LINKED_LIST or
 * DMA_CONDITIONAL_LINKED_LIST
 * -> This is needed to enable the linked list option.
 */

/* The initial configuration of the channel has to be
 * created using the datatype Dma_ConfigUpdateType.
 * The order of the data to be provided and hints for the data
 * to be entered are mentioned as comments below. */

Dma_ConfigUpdateType Dma_ChannelInitialConfiguration =
{
    /* Provide Source address here. E.g. (uint32)&Dma_Source_01 */
    /* Provide Destination address here. E.g. (uint32)&Dma_Destination_01 */
    /* Provide ADICR register value here */
    /* Provide CHCSR register value here */
    /* Provide CHCFGR register value here */
    /* Provide the contents of the SHADR here.
    This would be the address of the next TCS E.g. (uint32)&Dma_Tcs_01 */
    /* Provide the SDCRC register value here */
    /* Provide the RDCRC register value here */
    /* Provide bit wise information on which of the
    above registers are to be updated E.g. 1,1,1,1,1,1,0,0 */
};

/* The further TCSes are to be declared using the
 * DMA provided datatype - Dma_TransactionCtrlSetType
 * The order of the data to be provided and hints for the data
 * to be entered are mentioned as comments below. */

Dma_TransactionCtrlSetType Dma_Tcs_01 =
{
    /* Provide RDCRC register value for TCS */
    /* Provide SDCRC register value for TCS */
    /* Provide SADR register value for TCS */
    /* Provide DADR register value for TCS */
    /* Provide ADICR register value for TCS */
    /* Provide CHCFGR register value for TCS */
    /* Provide SHADR register value for TCS,
    which is the address of the next TCS e.g. (uint32)&Dma_Tcs_02,
    or zero if it is the last TCS. */
    /* Provide CHCSR register value for TCS */
};

Dma_TransactionCtrlSetType Dma_Tcs_02 =
{

```

DMA driver

```

/* Provide RDCRC register value for TCS */
/* Provide SDCRC register value for TCS */
/* Provide SADR register value for TCS */
/* Provide DADR register value for TCS */
/* Provide ADICR register value for TCS */
/* Provide CHCFG register value for TCS */
/* Provide SHADR register value for TCS */
/* Provide CHCSR register value for TCS */
};

/* Update the desired channel with the TCS made ready for the linked list */
Dma_ChUpdate(10, &Dma_ChannelInitialConfiguration, NULL_PTR);

```

1.1.5 Key architectural considerations

1.1.5.1 Usage of General Purpose Software Request (GPSR)

In a multicore environment when DMA error occurs, the error interrupt can get serviced only by any one of the cores. This can create problems, since the cause of the error might be a transaction which was triggered from another core. To circumvent this problem, it is decided to use the General Purpose Software Requests (GPSR). The error interrupts shall be routed to the appropriate core using GPSRs. The error handler, which services the error interrupt finds the core to which the error-prone channel has been allocated, and triggers a GPSR to that core. The GPSR, in turn, calls the notification function, which was configured by the user.

DMA driver

1.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **DMA address CRC and data CRC verification**

The user of the DMA driver shall compare the expected CRC with the CRC calculated by the DMA driver to find any mismatch.

[cover parentID DMA={9697DA85-FEC0-4ed0-B41A-98A6EA438E5E}]

- **DMA timestamp feature**

If the timestamp feature of the DMA is used, the user shall compare the timestamp with expected timestamp and take appropriate measures. The user should also allocate a 4 byte space (at the next 32 bit aligned address, after the last byte copied to the destination) for the DMA to store the timestamp.

[cover parentID DMA={1FFDBE80-D8EE-434b-A6D4-BD27986E1DAF}]

- **Double buffering: software switch usage**

When using double buffering in the software switch mode, a TRL event is raised if the transaction completes before the DMA receives a software switch. In a typical use case, the user can set the RROAT bit to 0, so that each transfer would require a trigger.

[cover parentID DMA={946532FE-2B5D-438d-9A1B-01B98321ECA8}]

- **Monitoring the lost transactions of DMA**

The user of the DMA driver is recommended to check for any lost DMA transaction, using the interrupt handlers. If there are any requests lost, the user shall take appropriate actions.

[cover parentID DMA={246B1D83-BF24-4f82-8726-5A01D143233A}]

- **Non-receipt of the interrupts to DMA**

If the DMA channel is configured to receive the interrupts from the hardware as triggers for the transfer/transaction, the user has to ensure the following: 1. The appropriate interrupt settings should be set to route the interrupts to DMA. 2. The user should also configure and monitor the channel transfer/transaction complete interrupts. These interrupts should be monitored with a timeout mechanism to ensure the correct reception of the hardware interrupts. The user can take appropriate error handling mechanism in the event of a timeout.

[cover parentID DMA={F8377F12-31F4-4e39-B3C8-ACC568522748}]

- **Protection of DMA data and registers**

The user shall ensure that adequate memory protections is made available, so that the DMA global data and registers are not overwritten by any other software running in the system.

[cover parentID DMA={34CC22B0-38D9-436c-B0E3-2438A6DCAE7C}]

- **Usage of APIs for enabling/disabling hardware triggers**

The enable/disable hardware trigger APIs should not be called for a channel which is configured as 'no hardware trigger'

[cover parentID DMA={E7EB27D9-C3BB-4652-910B-4FE393BF8326}]

- **Usage of DMA interrupts**

It is recommended that the user enable the interrupts and configures the respective handlers as well. This would enable the user to detect the events and errors occurring during the DMA transactions and take appropriate measures for handling them. Note 1: Polling should not be done for detecting the channel events or error events. This is not supported by the driver. Note 2: The user has to configure and invoke the channel interrupts for the used channels (for detecting channel events), the resource partition interrupts and the GPSR00 interrupt (for detecting the errors).The user has to ensure that the Dma_MEInterruptDispatcher() interrupt handler is called from the GPSR00 interrupt handler.

[cover parentID DMA={D84CE84B-2D57-40bb-8DC0-C9EB17817712}]

- **Usage of Dma_ChUpdate API**

DMA driver

The Dma_ChUpdate API should not be called by the user with invalid TCS contents (reserved bit fields selected, unused features selected, read-only or status bits set). The API also should not be used to update the TCS of an active/pending/halted(frozen) channel.

[cover parentID DMA={9960DEFF-E8A7-4a6d-A830-45F3BD872C96}]

- **Usage of Dma_IsInitDone API**

The Dma_InitDone API only indicates whether the DMA driver initialization was invoked or not. In case channels were stopped, the user shall not rely or use this API to derive the initialization status of the channel

[cover parentID DMA={FC698CE5-041C-4033-98BB-9D89E5202455}]

- **User/Supervisor mode context usage**

The user shall ensure that the DMA APIs are invoked in the same context for which it is configured to be used. For example, if the DMA is configured to be used in the Supervisor mode, the user should invoke the DMA APIs in the Supervisor mode.

[cover parentID DMA={AD934C16-FC65-4d74-AE9D-47431A53A3E1}]

- **Pattern detection not supported**

Pattern detection feature is not supported by the DMA driver

[cover parentID DMA={76A849BA-6602-4f03-905B-DE829AA3954A}]

- **Config Check**

The user shall ensure that the generated configuration structures are correct against the intended GUI configurations.

[cover parentID DMA={C9DFDA58-CB2E-4a50-9AF1-98102D9A3760}]

- **Configuration of GPSR interrupts**

The GPSR00 interrupt is used by the DMA to report the error events of DMA IP. The user should also ensure that the 'Dma_MEInterruptDispatcher' interrupt handler is invoked from the GPSR00 interrupt handler. This is needed for the error events to be notified.

[cover parentID DMA={C24D28F2-FDC4-413e-8E16-416F8B119C5B}]

- **Usage and configuration of interrupts**

When the DMA channels are allocated by the user to any particular core, the user has to ensure that the TOS (in the IRQ configuration) of the corresponding channel interrupts is also assigned to the same core.

[cover parentID DMA={B05BDE05-CEA3-45df-AE66-B87C00256F2B}]

- **Usage of cache and DMA**

If the user requires the DMA to transfer the data to and from the memory, the related memory and the Transaction Control Sets (TCS) should be accessed from the CPU without using the cache. If the cache is used, the consistency of the data cannot be ensured.

[cover parentID DMA={711073AB-D2BA-4145-97EC-F8DEB1DE82A2}]

- **Usage of Init Check feature**

The user has to ensure that the Dma_InitCheck API of the driver is invoked from each core where the DMA initialization is performed. The Dma_InitCheck API invocation should take place after the initialization.

[cover parentID DMA={5407022A-8D01-4646-A8E6-0D4716A6DD0}]

- **Usage of TRL feature**

If the TRL event has to be reported for a particular channel, the user shall ensure that the ETRL bit is enabled for the channel. This is done by setting the 'DmaTcsInterruptTransactionLoss' configuration parameter in the configuration tool.

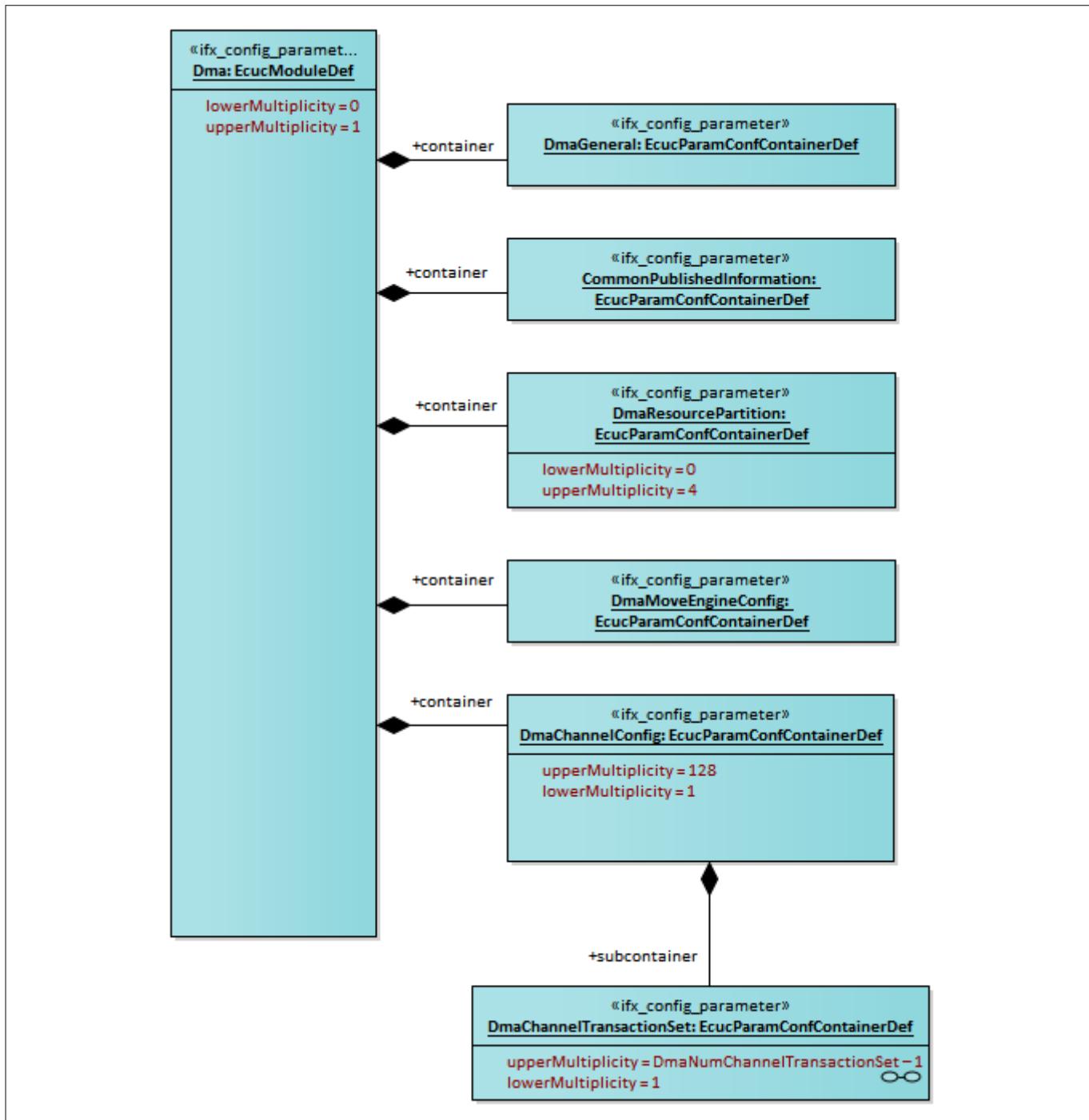
[cover parentID DMA={6880790B-2476-4a53-AA66-9CB3565BE1AB}]

- **Software access protection feature usage**

DMA driver

The user has to configure and use the hardware functionality of bus access protection (DMA ACCEN registers) to ensure freedom from interference of the software from different bus masters. The DMA driver provides means to configure these registers through the EB Tresos tool.

[cover parentID DMA={EFBDACD1-8935-40e9-B0D6-1A0870A1A1C5}]

DMA driver**1.3 Reference information****1.3.1 Configuration interfaces****Figure 7 Container hierarchy along with their configuration parameters****1.3.1.1 Container: CommonPublishedInformation**

This is the general configuration of DMA driver common container. It contains published information about vendor and versions.

Post-Build Variant Multiplicity: -

DMA driver

Multiplicity Configuration Class: -

1.3.1.1.1 ArMajorVersion

Table 5 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	-		

1.3.1.1.2 ArMinorVersion

Table 6 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	-		

1.3.1.1.3 ArPatchVersion

Table 7 Specification for ArPatchVersion

Name	ArPatchVersion
Description	This parameter provides the patch version of the AUTOSAR specification.

DMA driver**Table 7 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	-		

1.3.1.1.4 ModuleId**Table 8 Specification for ModuleId**

Name	ModuleId		
Description	This parameter provides the module ID of DMA		
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	-		

1.3.1.1.5 Release**Table 9 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	-

DMA driver**Table 9 Specification for Release (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.1.6 SwMajorVersion**Table 10 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	This parameter provides the major 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	-		

1.3.1.1.7 SwMinorVersion**Table 11 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	-		

DMA driver**1.3.1.1.8 SwPatchVersion****Table 12 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	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.1.9 VendorId**Table 13 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	-		

1.3.1.2 Container: Dma

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

1.3.1.3 Container: DmaChannelConfig

This container holds the channel wise configuration data for the DMA driver.

Post-Build Variant Multiplicity: FALSE

DMA driver

Multiplicity Configuration Class: Pre-Compile

1.3.1.3.1 DmaChannelAssignedPartition

Table 14 Specification for DmaChannelAssignedPartition

Name	DmaChannelAssignedPartition		
Description	This parameter determines the hardware partition which this channel is allocated to. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
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	-		

1.3.1.3.2 DmaChannelId

Table 15 Specification for DmaChannelId

Name	DmaChannelId		
Description	This parameter determines the physical channel number Note: The default value is an incremental value starting from zero, to make it unique automatically.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 127		
Default value	Incremental value starting with 0 and unique among all the channels		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

DMA driver**1.3.1.3.3 DmaChannelNotification****Table 16 Specification for DmaChannelNotification**

Name	DmaChannelNotification		
Description	<p>The DmaChannelNotification is used by the DMA driver to invoke the user-defined function for notification purposes. The parameter can be a name or the address(numeric value) of the notification function.</p> <p>Note1: By default, the notification parameter will be NULL.</p> <p>Note2: The DMA driver does not validate the configured function name or address for correctness and the responsibility falls on the user.</p>		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range			
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	-		

1.3.1.3.4 DmaChannelNumTransactionSet**Table 17 Specification for DmaChannelNumTransactionSet**

Name	DmaChannelNumTransactionSet		
Description	<p>This parameter indicates the number of transaction sets which can be configured to the particular channel. Linked list feature is offered in Transaction set configuration only if the value of this parameter is greater than one.</p> <p>The number of transaction sets is capped at DmaMaxTransactionSetPerChannel (to a max 0xFFFF).</p> <p>Note: Minimum TCS number is selected as the default value.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - DmaMaxTransactionSetPerChannel (capped to a max 0xFFFF)		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

DMA driver**Table 17 Specification for DmaChannelNumTransactionSet (continued)**

Dependency	DmaMaxTransactionSetPerChannel		
-------------------	--------------------------------	--	--

1.3.1.3.5 DmaTcsInterruptTransactionLoss**Table 18 Specification for DmaTcsInterruptTransactionLoss**

Name	DmaTcsInterruptTransactionLoss		
Description	<p>This parameter enables/disables transaction loss interrupt. Notification for the TRL interrupt would be provided only if this TRL bit is enabled.</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	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4 Container: DmaChannelTransactionSet

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Pre-Compile

1.3.1.4.1 DmaNextTcsIndex**Table 19 Specification for DmaNextTcsIndex**

Name	DmaNextTcsIndex		
Description	<p>This parameter defines the next TCS node in case of linked list feature.</p> <p>Note: The next transaction set index is selected as the default value of the parameter.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - Number of DmaTcsIndex available in Dma channel (max 0xFFFF)		
Default value	DmaTcsIndex+1 (Incrementing index)		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DMA driver**Table 19 Specification for DmaNextTcsIndex (continued)**

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

1.3.1.4.2 DmaTcsAppendTimeStamp**Table 20 Specification for DmaTcsAppendTimeStamp**

Name	DmaTcsAppendTimeStamp		
Description	This parameter determines if time stamp is to be appended after the last DMA move of a transaction. 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	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.3 DmaTcsAutoStartEnable**Table 21 Specification for DmaTcsAutoStartEnable**

Name	DmaTcsAutoStartEnable		
Description	This parameter enables/disables the autostart feature in the case of a linked list Note: The optional features are kept disabled by default. To be enabled by user as per the need.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

DMA driver**Table 21 Specification for DmaTcsAutoStartEnable (continued)**

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

1.3.1.4.4 DmaTcsCircularBufferDestinationEnable**Table 22 Specification for DmaTcsCircularBufferDestinationEnable**

Name	DmaTcsCircularBufferDestinationEnable		
Description	<p>This parameter determines if a circular buffering scheme is to be implemented on the destination buffer.</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	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.5 DmaTcsCircularBufferDestinationLength**Table 23 Specification for DmaTcsCircularBufferDestinationLength**

Name	DmaTcsCircularBufferDestinationLength		
Description	<p>This parameter determines how many bits of the destination address of a circular buffer can actually be modified. Remaining address bits stay unchanged.</p> <p>This parameter is grayed out if DmaTcsCircularBufferDestinationEnable is set to FALSE</p> <p>Note 1: The user has to ensure that the destination address is aligned accordingly (For e.g. if the circular buffer size is chosen to be 16, the address has to be aligned to the 16 byte boundary).</p> <p>Note 2: The default value of this parameter is set to the reset value of the corresponding SFR.</p>		

DMA driver**Table 23 Specification for DmaTcsCircularBufferDestinationLength (continued)**

Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DMA_CIRCULAR_BUFFER_LENGTH_1024BYTE: Circular buffer length of 1024 byte DMA_CIRCULAR_BUFFER_LENGTH_128BYTE: Circular buffer length of 128 byte DMA_CIRCULAR_BUFFER_LENGTH_16384BYTE: Circular buffer length of 16384 byte DMA_CIRCULAR_BUFFER_LENGTH_16BYTE: Circular buffer length of 16 byte DMA_CIRCULAR_BUFFER_LENGTH_1BYTE: Circular buffer length of 1 byte DMA_CIRCULAR_BUFFER_LENGTH_2048BYTE: Circular buffer length of 2048 byte DMA_CIRCULAR_BUFFER_LENGTH_256BYTE: Circular buffer length of 256 byte DMA_CIRCULAR_BUFFER_LENGTH_2BYTE: Circular buffer length of 2 byte DMA_CIRCULAR_BUFFER_LENGTH_32768BYTE: Circular buffer length of 32768 byte DMA_CIRCULAR_BUFFER_LENGTH_32BYTE: Circular buffer length of 32 byte DMA_CIRCULAR_BUFFER_LENGTH_4096BYTE: Circular buffer length of 4096 byte DMA_CIRCULAR_BUFFER_LENGTH_4BYTE: Circular buffer length of 4 byte DMA_CIRCULAR_BUFFER_LENGTH_512BYTE: Circular buffer length of 512 byte DMA_CIRCULAR_BUFFER_LENGTH_64BYTE: Circular buffer length of 64 byte DMA_CIRCULAR_BUFFER_LENGTH_8192BYTE: Circular buffer length of 8192 byte DMA_CIRCULAR_BUFFER_LENGTH_8BYTE: Circular buffer length of 8 byte		
Default value	DMA_CIRCULAR_BUFFER_LENGTH_1BYTE		
Post-build variant value	TRUE		
Value configuration class	Post-Build		
Origin	IFX		
Dependency	DmaTcsCircularBufferDestinationEnable		

1.3.1.4.6 DmaTcsCircularBufferSourceEnable**Table 24 Specification for DmaTcsCircularBufferSourceEnable**

Name	DmaTcsCircularBufferSourceEnable		
Description	This parameter determines if a circular buffering scheme is to be implemented on the source buffer. 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		

DMA driver**Table 24 Specification for DmaTcsCircularBufferSourceEnable (continued)**

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

1.3.1.4.7 DmaTcsCircularBufferSourceLength**Table 25 Specification for DmaTcsCircularBufferSourceLength**

Name	DmaTcsCircularBufferSourceLength		
Description	<p>This parameter determines how many bits of the source address of a circular buffer can actually be modified. Remaining address bits stay unchanged.</p> <p>This parameter is greyed out if DmaTcsCircularBufferSourceEnable is set to FALSE.</p> <p>Note 1: The user has to ensure that the source address is aligned accordingly (For e.g. if the circular buffer size is chosen to be 16, the address has to be aligned to the 16 byte boundary).</p> <p>Note 2: The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DMA_CIRCULAR_BUFFER_LENGTH_1024BYTE: Circular buffer length of 1024 byte DMA_CIRCULAR_BUFFER_LENGTH_128BYTE: Circular buffer length of 128 byte DMA_CIRCULAR_BUFFER_LENGTH_16384BYTE: Circular buffer length of 16384 byte DMA_CIRCULAR_BUFFER_LENGTH_16BYTE: Circular buffer length of 16 byte DMA_CIRCULAR_BUFFER_LENGTH_1BYTE: Circular buffer length of 1 byte DMA_CIRCULAR_BUFFER_LENGTH_2048BYTE: Circular buffer length of 2048 byte DMA_CIRCULAR_BUFFER_LENGTH_256BYTE: Circular buffer length of 256 byte DMA_CIRCULAR_BUFFER_LENGTH_2BYTE: Circular buffer length of 2 byte DMA_CIRCULAR_BUFFER_LENGTH_32768BYTE: Circular buffer length of 32768 byte DMA_CIRCULAR_BUFFER_LENGTH_32BYTE: Circular buffer length of 32 byte DMA_CIRCULAR_BUFFER_LENGTH_4096BYTE: Circular buffer length of 4096 byte DMA_CIRCULAR_BUFFER_LENGTH_4BYTE: Circular buffer length of 4 byte DMA_CIRCULAR_BUFFER_LENGTH_512BYTE: Circular buffer length of 512 byte DMA_CIRCULAR_BUFFER_LENGTH_64BYTE: Circular buffer length of 64 byte DMA_CIRCULAR_BUFFER_LENGTH_8192BYTE: Circular buffer length of 8192 byte DMA_CIRCULAR_BUFFER_LENGTH_8BYTE: Circular buffer length of 8 byte		
Default value	DMA_CIRCULAR_BUFFER_LENGTH_1BYTE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DMA driver**Table 25 Specification for DmaTcsCircularBufferSourceLength (continued)**

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

1.3.1.4.8 DmaTcsDaisyChaining**Table 26 Specification for DmaTcsDaisyChaining**

Name	DmaTcsDaisyChaining		
Description	<p>This parameter determines whether the adjacent lower channel is daisy chained with this channel. If so, after a transaction successfully completes on this channel, an internal hardware trigger is forwarded to the adjacent channel.</p> <p>Note: The optional features are kept disabled by default.</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	-		

1.3.1.4.9 DmaTcsDataTransferInterrupt**Table 27 Specification for DmaTcsDataTransferInterrupt**

Name	DmaTcsDataTransferInterrupt		
Description	<p>This parameter enables/disables the data transfer/transaction interrupt.</p> <p>Note: The interrupts are kept disabled by default. To be enabled by user as per the need.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		

DMA driver**Table 27 Specification for DmaTcsDataTransferInterrupt (continued)**

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

1.3.1.4.10 DmaTcsDestinationAddress**Table 28 Specification for DmaTcsDestinationAddress**

Name	DmaTcsDestinationAddress		
Description	<p>This parameter defines the start address of the destination memory. The address is to be provided either as an address value in hexadecimal or as the pointer to the location (with the ampersand symbol included in the name, if needed).</p> <p>Note: The code generator would indicate error in the following cases:</p> <ul style="list-style-type: none"> - If the address field is left blank - If there are spaces in the name - If a numeric value is used which does not end with 'U', indicating unsigned value. <p>Other than these checks, there are no other checks performed by the UI or code generator on the validity of the content entered.</p> <p>The DMA driver does not validate the configured address for correctness or data alignment and the responsibility falls on the user.</p> <p>If double buffering is enabled, this parameter indicates the first of the two buffers.</p> <p>Note: Since the address is user configurable, the default value is kept as NULL_PTR.</p>		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Link-Time	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

DMA driver**1.3.1.4.11 DmaTcsDestinationAddressModificationFactor****Table 29 Specification for DmaTcsDestinationAddressModificationFactor**

Name	DmaTcsDestinationAddressModificationFactor		
Description	<p>This parameter defines how the destination address changes after every move operation. If DmaTcsCircularBufferDestinationLength is 0 and DmaTcsCircularBufferDestinationEnable is TRUE, the UI widget of this parameter is grayed out.</p> <p>The description field of Tresos is used to explain the significance of each of the values is provided.</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	DMA_FACTOR_128: Address offset is 128 x DmaChBitsPerMove DMA_FACTOR_16: Address offset is 16 x DmaChBitsPerMove DMA_FACTOR_1: Address offset is 1 x DmaChBitsPerMove DMA_FACTOR_2: Address offset is 2 x DmaChBitsPerMove DMA_FACTOR_4: Address offset is 4 x DmaChBitsPerMove DMA_FACTOR_64: Address offset is 64 x DmaChBitsPerMove DMA_FACTOR_8: Address offset is 8 x DmaChBitsPerMove		
Default value	DMA_FACTOR_1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.12 DmaTcsDestinationAddressMovement**Table 30 Specification for DmaTcsDestinationAddressMovement**

Name	DmaTcsDestinationAddressMovement		
Description	<p>This parameter determines if the destination address after every move operation is to be increased or decreased.</p> <p>Note: The default value is chosen as 'increasing' for the natural progression. User can choose other values as needed.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DMA_DECREASING: Address decrements DMA_INCREASING: Address increments		
Default value	DMA_INCREASING		

DMA driver**Table 30 Specification for DmaTcsDestinationAddressMovement (continued)**

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

1.3.1.4.13 DmaTcsDoubleBuffer**Table 31 Specification for DmaTcsDoubleBuffer**

Name	DmaTcsDoubleBuffer		
Description	<p>This parameter defines the start address of the double buffer.</p> <p>If double buffering is enabled, this parameter gets enabled and indicates the second buffer. The address is to be provided either as an address value in hexadecimal or as the pointer to the location (with the ampersand symbol included in the name, if needed). With double buffering disabled, this parameter is grayed out.</p> <p>Note: The code generator would indicate error in the following cases:</p> <ul style="list-style-type: none"> - If the address field is left blank - If there are spaces in the name - If a numeric value is used which does not end with 'U', indicating unsigned value <p>Other than these checks, there are no other checks performed by the UI or code generator on the validity of the content entered.</p> <p>The DMA driver does not validate the configured address for correctness or data alignment and the responsibility falls on the user.</p> <p>Note: Since the address is user configurable, the default value is kept as NULL_PTR.</p>		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Link-Time	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DmaTcsShadowRegisterConfiguration		

DMA driver**1.3.1.4.14 DmaTcsHardwareTrigger****Table 32 Specification for DmaTcsHardwareTrigger**

Name	DmaTcsHardwareTrigger		
Description	<p>This parameter determines whether hardware initiated trigger is to be supported or not and if affirmative its mode</p> <p>Note: Since the hardware trigger is user configurable, the default value is kept as disabled.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>DMA_HARDWARE_TRIGGER_CONTINUOUS_MODE: After a DMA transaction, bit TSR.HTRE remains set, ready to accept the next trigger.</p> <p>DMA_HARDWARE_TRIGGER_SINGLE_MODE: After a DMA transaction, the DMA channel is disabled for further hardware requests</p> <p>DMA_NO_HARDWARE_TRIGGER: No hardware trigger would be used</p>		
Default value	DMA_NO_HARDWARE_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.15 DmaTcsIndex**Table 33 Specification for DmaTcsIndex**

Name	DmaTcsIndex		
Description	<p>This parameter defines the index of current TCS node in the list of TCS nodes.</p> <p>Note: The current transaction set index is selected as the default value of the parameter.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - DmaChannelNumTransactionSet -1 (max 0xFFFF)		
Default value	DmaChannelTransactionSet current Index		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

DMA driver**1.3.1.4.16 DmaTcsInterruptDataTransfer****Table 34 Specification for DmaTcsInterruptDataTransfer**

Name	DmaTcsInterruptDataTransfer		
Description	Interrupts can be generated either for every transfer or even transaction. Additionally, an interrupt can be generated after a certain number of transfers equaling a threshold have been completed Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DMA_INTERRUPT_AFTER_THRESHOLD: Interrupt is triggered when TCOUNT equals IRDV DMA_INTERRUPT_PER_TRANSACTION: Interrupt is triggered after the transaction is complete. DMA_INTERRUPT_PER_TRANSFER: Interrupt is triggered after the transfer is complete.		
Default value	DMA_INTERRUPT_PER_TRANSACTION		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.17 DmaTcsInterruptDataTransferThreshold**Table 35 Specification for DmaTcsInterruptDataTransferThreshold**

Name	DmaTcsInterruptDataTransferThreshold		
Description	If the parameter DmaTcsInterruptDataTransfer is set to a value of INTERRUPT_AFTER_THRESHOLD, this parameter determines the threshold of transfer following which an interrupt is generated. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 15		
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	DmaTcsInterruptDataTransfer		

DMA driver**1.3.1.4.18 DmaTcsInterruptDestinationAddressWrap****Table 36 Specification for DmaTcsInterruptDestinationAddressWrap**

Name	DmaTcsInterruptDestinationAddressWrap		
Description	This parameter enables/disables destination address wrap around interrupt. Note: The interrupts are kept disabled by default. To be enabled by user as per the need.		
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	-		

1.3.1.4.19 DmaTcsInterruptSourceAddressWrap**Table 37 Specification for DmaTcsInterruptSourceAddressWrap**

Name	DmaTcsInterruptSourceAddressWrap		
Description	This parameter enables/disables source address wrap around interrupt Note: The interrupts are kept disabled by default. To be enabled by user as per the need.		
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	-		

DMA driver**1.3.1.4.20 DmaTcsMoveLength****Table 38 Specification for DmaTcsMoveLength**

Name	DmaTcsMoveLength		
Description	<p>This parameter defines the amount of data transferred per move operation. The description field of Tresos tool explains the significance of the range of input values.</p> <p>Note: The channel data width has to be chosen depending on whether the SRI or SPB bus is used. User has to ensure that the appropriate channel width is chosen. Refer the Target Specification for the valid configurations.</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	DMA_WIDTH_128BITS: Number of bits per move is 128 bits DMA_WIDTH_16BITS: Number of bits per move is 16 bits DMA_WIDTH_256BITS: Number of bits per move is 256 bits DMA_WIDTH_32BITS: Number of bits per move is 32 bits DMA_WIDTH_64BITS: Number of bits per move is 64 bits DMA_WIDTH_8BITS: Number of bits per move is 8 bits		
Default value	DMA_WIDTH_8BITS		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.21 DmaTcsReferenceAddressCrc**Table 39 Specification for DmaTcsReferenceAddressCrc**

Name	DmaTcsReferenceAddressCrc		
Description	<p>This parameter defines the reference address CRC calculated by the user. No checks are performed by the UI or code generator on the validity of the CRC entered. Value is input in hexadecimal format.</p> <p>Note: Since the CRC value is user configurable, the default value is kept as zero.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0x00000000 - 0xFFFFFFFF		
Default value	0x00000000		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DMA driver**Table 39 Specification for DmaTcsReferenceAddressCrc (continued)**

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

1.3.1.4.22 DmaTcsReferenceDataCrc**Table 40 Specification for DmaTcsReferenceDataCrc**

Name	DmaTcsReferenceDataCrc		
Description	<p>This parameter defines the reference data CRC calculated by the user. No checks are performed by the UI or code generator on the validity of the CRC entered. Value is input in hexadecimal format.</p> <p>Note: Since the CRC value is user configurable, the default value is kept as zero.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0x00000000 - 0xFFFFFFFF		
Default value	0x00000000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.23 DmaTcsShadowRegisterConfiguration**Table 41 Specification for DmaTcsShadowRegisterConfiguration**

Name	DmaTcsShadowRegisterConfiguration		
Description	<p>This parameter determines how the shadow register is to be used. Details are in the Internal Target Specification.</p> <p>If the parameter DmaChannelNumTransactionSet is set to a value greater than one, user is allowed to choose only the linked list options.</p> <p>Further, it is only for the first node of the linked list that the user is allowed to choose the specific linked list type. All other non-leaf nodes of this linked list inherit the property of the root transaction set. The leaf node of the linked list has SHADOWING_DISABLED configured</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

DMA driver**Table 41 Specification for DmaTcsShadowRegisterConfiguration (continued)**

Range	DMA_ACCUMULATED_LINKED_LIST: Accumulated Linked List (ACLL) DMA_CONDITIONAL_LINKED_LIST: Conditional Linked List (CONLL) DMA_DEST_ADDRESS_BUFFERING_RO: Shadow Operation Read Only Mode Destination Address DMA_DEST_ADDRESS_BUFFERING_RW: Shadow Operation Direct Write Mode Destination Address DMA_DEST_DOUBLE_BUFFERING_HW_SW_SWITCH: DMA Double Destination Buffering with Software Switch and Automatic Hardware Switch DMA_DEST_DOUBLE_BUFFERING_SW_SWITCH: DMA Double Destination Buffering with Software Switch Only DMA_LINKED_LIST: DMA Linked List (DMALL) DMA_SAFE_LINKED_LIST: Safe Linked List (SAFLL) DMA_SHADOWING_DISABLED: Shadow operation disabled DMA_SOURCE_ADDRESS_BUFFERING_RO: Shadow Operation Read Only Mode Source Address DMA_SOURCE_ADDRESS_BUFFERING_RW: Shadow Operation Direct Write Mode Source Address DMA_SOURCE_DOUBLE_BUFFERING_HW_SW_SWITCH: DMA Double Source Buffering with Software Switch and Automatic Hardware Switch DMA_SOURCE_DOUBLE_BUFFERING_SW_SWITCH: DMA Double Source Buffering with Software Switch Only		
Default value	DMA_SHADOWING_DISABLED		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.24 DmaTcsSourceAddress**Table 42 Specification for DmaTcsSourceAddress**

Name	DmaTcsSourceAddress
-------------	---------------------

DMA driver**Table 42 Specification for DmaTcsSourceAddress (continued)**

Description	<p>This parameter defines the start address of the source memory. The address is to be provided either as an address value in hexadecimal or as the pointer to the location (with the ampersand symbol included in the name, if needed).</p> <p>Note: The code generator would indicate error in the following cases:</p> <ul style="list-style-type: none"> - If the address field is left blank - If there are spaces in the name - If a numeric value is used which does not end with 'U', indicating unsigned value <p>Other than these checks, there are no other checks performed by the UI or code generator on the validity of the content entered.</p> <p>The DMA driver does not validate the configured address for correctness or data alignment and the responsibility falls on the user.</p> <p>If double buffering is enabled, this parameter indicates the first of the two buffers.</p> <p>Note: Since the address is user configurable, the default value is kept as NULL_PTR.</p>		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Link-Time	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.25 DmaTcsSourceAddressModificationFactor**Table 43 Specification for DmaTcsSourceAddressModificationFactor**

Name	DmaTcsSourceAddressModificationFactor		
Description	<p>This parameter defines how the source address changes after every move operation. If DmaTcsCircularBufferSourceLength is 0 and DmaTcsCircularBufferSourceEnable is TRUE, the UI widget of this parameter is grayed out.</p> <p>The description field of Tresos is used to explain the significance of each of the values is provided.</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

DMA driver**Table 43 Specification for DmaTcsSourceAddressModificationFactor (continued)**

Range	DMA_FACTOR_128: Address offset is 128 x DmaChBitsPerMove DMA_FACTOR_16: Address offset is 16 x DmaChBitsPerMove DMA_FACTOR_1: Address offset is 1 x DmaChBitsPerMove DMA_FACTOR_2: Address offset is 2 x DmaChBitsPerMove DMA_FACTOR_32: Address offset is 32 x DmaChBitsPerMove DMA_FACTOR_4: Address offset is 4 x DmaChBitsPerMove DMA_FACTOR_64: Address offset is 64 x DmaChBitsPerMove DMA_FACTOR_8: Address offset is 8 x DmaChBitsPerMove		
Default value	DMA_FACTOR_1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.26 DmaTcsSourceAddressMovement**Table 44 Specification for DmaTcsSourceAddressMovement**

Name	DmaTcsSourceAddressMovement		
Description	This parameter determines if the source address after every move operation is to be increased or decreased. Note: The default value is chosen as 'increasing' for the natural progression. User can choose other values as needed.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DMA_DECREASING: Address decrements DMA_INCREASING: Address increments		
Default value	DMA_INCREASING		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

DMA driver**1.3.1.4.27 DmaTcsSwapDataCRCByteOrder****Table 45 Specification for DmaTcsSwapDataCRCByteOrder**

Name	DmaTcsSwapDataCRCByteOrder		
Description	<p>This parameter enables/disables the swap CRC byte order feature.</p> <p>Note: The optional features are kept disabled by default. To be enabled by user as per the need.</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	-		

1.3.1.4.28 DmaTcsTransactionLength**Table 46 Specification for DmaTcsTransactionLength**

Name	DmaTcsTransactionLength		
Description	<p>This parameter defines how many transfers are performed per DMA transaction.</p> <p>Value 0 and 1 will have same impact (i.e. one transfer) as DMA hardware executes at least one DMA transfer on a channel start.</p> <p>Note: Since the length value is user configurable, the default value is kept as zero.</p>		
Multiplicity	1..1	Type	EcuCIIntegerParamDef
Range	0 - 16383		
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	-		

DMA driver**1.3.1.4.29 DmaTcsTransferLength****Table 47 Specification for DmaTcsTransferLength**

Name	DmaTcsTransferLength		
Description	<p>This parameter defines how many moves are performed per DMA transfer. The description field of Tresos tool explains the significance of the range of input values.</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	DMA_MOVES_16: One transfer has 16 move DMA_MOVES_1: One transfer has 1 move DMA_MOVES_2: One transfer has 2 move DMA_MOVES_3: One transfer has 3 move DMA_MOVES_4: One transfer has 4 move DMA_MOVES_5: One transfer has 5 move DMA_MOVES_8: One transfer has 8 move DMA_MOVES_9: One transfer has 9 move		
Default value	DMA_MOVES_1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.4.30 DmaTcsTriggerFrequency**Table 48 Specification for DmaTcsTriggerFrequency**

Name	DmaTcsTriggerFrequency		
Description	<p>This parameter determines how much of data will the move engine move upon a trigger request (either hardware or software trigger). The move engine, upon receipt of a trigger request, could effectuate a transfer or even a transaction based on the choice made.</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	DMA_TRANSACTION_PER_TRIGGER: One DMA request starts a complete DMA transaction DMA_TRANSFER_PER_TRIGGER: A DMA request is required for each DMA transfer		
Default value	DMA_TRANSFER_PER_TRIGGER		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DMA driver**Table 48 Specification for DmaTcsTriggerFrequency (continued)**

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

1.3.1.4.31 DmaUserHeaderFileWithExternDeclarations**Table 49 Specification for DmaUserHeaderFileWithExternDeclarations**

Name	DmaUserHeaderFileWithExternDeclarations		
Description	This parameter takes the header file which contains the extern declarations of the variable names used to define the source address, destination address or double buffer address. Note: The default value is kept as NULL. This field gets enabled only when a non-address value is entered for the source address or destination address or double buffer address.		
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	IFX	Scope	LOCAL
Dependency	DmaTcsDoubleBuffer, DmaTcsDestinationAddress, DmaTcsSourceAddress		

1.3.1.5 Container: DmaGeneral

This container contains the general configuration parameters needed for the DMA driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 DmaBufferSwitchApiConfiguration**Table 50 Specification for DmaBufferSwitchApiConfiguration**

Name	DmaBufferSwitchApiConfiguration		
Description	This parameter results in generation of a preprocessor macro that conditionally includes code related to buffer switch interface Note: The optional APIs are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef

DMA driver**Table 50 Specification for DmaBufferSwitchApiConfiguration (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	IFX	Scope	LOCAL
Dependency	-		

1.3.1.5.2 DmaChDeInitApiConfiguration**Table 51 Specification for DmaChDeInitApiConfiguration**

Name	DmaChDeInitApiConfiguration		
Description	Adds / removes the service Dma_ChDeInit() from the code. true: Dma_ChDeInit() can be used. false: Dma_ChDeInit() cannot be used. 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	IFX	Scope	LOCAL
Dependency	-		

1.3.1.5.3 DmaDataPendingApiConfiguration**Table 52 Specification for DmaDataPendingApiConfiguration**

Name	DmaDataPendingApiConfiguration
-------------	--------------------------------

DMA driver**Table 52 Specification for DmaDataPendingApiConfiguration (continued)**

Description	Adds / removes the service Dma_ChGetRemainingData() from the code. true: Dma_ChGetRemainingData() can be used. false: Dma_ChGetRemainingData() cannot be used. 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	IFX	Scope	LOCAL
Dependency	-		

1.3.1.5.4 DmaDevErrorDetect**Table 53 Specification for DmaDevErrorDetect**

Name	DmaDevErrorDetect		
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	IFX	Scope	LOCAL
Dependency	-		

DMA driver**1.3.1.5.5 DmaGetVersionInfoApiConfiguration****Table 54 Specification for DmaGetVersionInfoApiConfiguration**

Name	DmaGetVersionInfoApiConfiguration		
Description	Adds / removes the API Dma_GetVersionInfo() from the code. true: Dma_GetVersionInfo() can be used. false: Dma_GetVersionInfo() cannot be used. 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	IFX	Scope	LOCAL
Dependency	-		

1.3.1.5.6 DmaInitCheckApi**Table 55 Specification for DmaInitCheckApi**

Name	DmaInitCheckApi		
Description	Adds / removes the service Dma_InitCheck() from the code. true: Dma_InitCheck() can be used. false: Dma_InitCheck() cannot be used. 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	IFX	Scope	LOCAL

DMA driver**Table 55 Specification for DmaInitCheckApi (continued)**

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

1.3.1.5.7 DmaInitDeInitApiMode**Table 56 Specification for DmaInitDeInitApiMode**

Name	DmaInitDeInitApiMode		
Description	<p>This configuration parameter gives the mode in which the Dma Init and De-Init APIs will be used.</p> <p>Note: Since DMA driver accesses the SFRs, it is more efficient to operate the DMA driver in supervisor mode. Hence, the default mode of operation is supervisor.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DMA_MCAL_SUPERVISORMODE: Operating mode used is Supervisor DMA_MCAL_USER1MODE: Operating mode used is USER1		
Default value	DMA_MCAL_SUPERVISORMODE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DmaRuntimeApiMode		

1.3.1.5.8 DmaMaxTransactionSetPerChannel**Table 57 Specification for DmaMaxTransactionSetPerChannel**

Name	DmaMaxTransactionSetPerChannel		
Description	<p>This parameter controls how many transaction sets are allowed to be defined per channel.</p> <p>No code is generated based on this parameter</p> <p>Note: Minimum TCS number is selected as the default value.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	1 - 65535		
Default value	1		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

DMA driver**Table 57 Specification for DmaMaxTransactionSetPerChannel (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.5.9 DmaMultiCoreErrorDetect**Table 58 Specification for DmaMultiCoreErrorDetect**

Name	DmaMultiCoreErrorDetect		
Description	The parameter enables or disables the multi core related default error tracer (DET) detection and reporting. It is applicable only when DET/Safety is 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	DmaDevErrorDetect		

1.3.1.5.10 DmaRuntimeApiMode**Table 59 Specification for DmaRuntimeApiMode**

Name	DmaRuntimeApiMode		
Description	This configuration parameter gives the mode in which the Runtime API will be used. Note: Since DMA driver accesses the SFRs, it is more efficient to operate the DMA driver in supervisor mode. Hence, the default mode of operation is supervisor.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DMA_MCAL_SUPERVISORMODE: Operating mode used is Supervisor DMA_MCAL_USER1MODE: Operating mode used is USER1		
Default value	DMA_MCAL_SUPERVISORMODE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-

DMA driver**Table 59 Specification for DmaRuntimeApiMode (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.5.11 DmaSafetyEnable**Table 60 Specification for DmaSafetyEnable**

Name	DmaSafetyEnable		
Description	Switch to enable/disable the safety check and reporting. true: Enable safety check and reporting false: Disable safety check and reporting 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	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DmaInitCheckApi		

1.3.1.5.12 DmaSuspendApiConfiguration**Table 61 Specification for DmaSuspendApiConfiguration**

Name	DmaSuspendApiConfiguration		
Description	Adds / removes the services Dma_ChTransferFreeze() and Dma_ChTransferResume() from the code. true: Dma_ChTransferFreeze() and Dma_ChTransferResume() can be used. false: Dma_ChTransferFreeze() and Dma_ChTransferResume() cannot be used. Note: The optional APIs are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		

DMA driver**Table 61 Specification for DmaSuspendApiConfiguration (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	-		

1.3.1.5.13 DmaTriggerApiConfiguration**Table 62 Specification for DmaTriggerApiConfiguration**

Name	DmaTriggerApiConfiguration		
Description	Adds / removes the services Dma_ChEnableHardwareTrigger() and Dma_ChDisableHardwareTrigger() from the code. true: Dma_ChEnableHardwareTrigger() and Dma_ChDisableHardwareTrigger() can be used. false: Dma_ChEnableHardwareTrigger() and Dma_ChDisableHardwareTrigger() cannot be used. 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	IFX	Scope	LOCAL
Dependency	-		

1.3.1.6 Container: DmaMoveEngineConfig

DmaMoveEngineConfig contains the configuration parameters for the notifications from the move engine.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

DMA driver**1.3.1.6.1 DmaMEDestinationErrorInterrupt****Table 63 Specification for DmaMEDestinationErrorInterrupt**

Name	DmaMEDestinationErrorInterrupt		
Description	This parameter enables/disables Move engines destination error interrupt. Note: The interrupts are kept disabled by default. To be enabled by user as per the need.		
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	-		

1.3.1.6.2 DmaMELinkedListErrorInterrupt**Table 64 Specification for DmaMELinkedListErrorInterrupt**

Name	DmaMELinkedListErrorInterrupt		
Description	This parameter enables/disables linked list error interrupts Note: The interrupts are kept disabled by default. To be enabled by user as per the need.		
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	-		

DMA driver**1.3.1.6.3 DmaMESourceErrorInterrupt****Table 65 Specification for DmaMESourceErrorInterrupt**

Name	DmaMESourceErrorInterrupt		
Description	This parameter enables/disables Move engines source error interrupt. Note: The interrupts are kept disabled by default. To be enabled by user as per the need.		
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	-		

1.3.1.7 Container: DmaResourcePartition

DMAResourcePartition contains the configuration related to the resource partitions of DMA.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.7.1 DmaPermittedBusMaster**Table 66 Specification for DmaPermittedBusMaster**

Name	DmaPermittedBusMaster		
Description	This parameter is a 32 bit hexadecimal mask and represents the bus masters that are allowed to access this partition. Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0x00000000 - 0xFFFFFFFF		
Default value	0xFFFFFFFF		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

DMA driver**Table 66 Specification for DmaPermittedBusMaster (continued)**

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

1.3.1.7.2 DmaResourcePartitionBusMode**Table 67 Specification for DmaResourcePartitionBusMode**

Name	DmaResourcePartitionBusMode		
Description	This parameter defines the bus access mode of the resource partition Note: The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DMA_RP_SUPERVISOR_MODE: Resource partition would be in the supervisor mode. DMA_RP_USER_MODE: Resource partition would be in the user mode.		
Default value	DMA_RP_SUPERVISOR_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.7.3 DmaResourcePartitionErrorNotifRoutine**Table 68 Specification for DmaResourcePartitionErrorNotifRoutine**

Name	DmaResourcePartitionErrorNotifRoutine		
Description	The DmaResourcePartitionErrorNotifRoutine is used by the DMA driver to invoke the user-defined function for notification of resource partition error. The parameter can be a name or the address(numeric value) of the notification function. Note 1: By default, the notification parameter will be NULL. Note 2: The DMA driver does not validate the configured function name or address for correctness and the responsibility falls on the user.		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range			
Default value	NULL		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DMA driver**Table 68 Specification for DmaResourcePartitionErrorNotifRoutine (continued)**

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

1.3.2 Functions - Type definitions**1.3.2.1 Dma_ChannelNotificationPtrType****Table 69 Specification for Dma_ChannelNotificationPtrType**

Syntax	Dma_ChannelNotificationPtrType
Type	Pointer to a function of type void Function_Name (const uint8 Channel, const uint32 Event)
File	Dma.h
Description	This function pointer type would hold the address of the function which has to be invoked when there is a channel event.
Source	IFX

1.3.2.2 Dma_ConfigType**Table 70 Specification for Dma_ConfigType**

Syntax	Dma_ConfigType
Type	Structure
File	Dma.h
Range	--
Description	Defines the type for data structure containing the set of configuration parameters required for initializing the DMA driver and DMA hardware unit(s).
Source	IFX

1.3.2.3 Dma_ConfigUpdateType**Table 71 Specification for Dma_ConfigUpdateType**

Syntax	Dma_ConfigUpdateType	
Type	Structure	
File	Dma.h	
Range	uint32 AddressCrc	Contents of SDCRC register

DMA driver**Table 71 Specification for Dma_ConfigUpdateType (continued)**

	uint32 Config	Contents of CHCFG register
	uint32 Control	Contents of ADICR register
	uint32 DestAddress	Start of the destination buffer address
	unsigned_int UpdateAddressCrc:1	Indicates whether the SDCRC should be updated or not
	unsigned_int UpdateConfig:1	Indicates whether the CHCFG should be updated or not
	unsigned_int UpdateControlAdicr:1	Indicates whether the ADICR should be updated or not
	unsigned_int UpdateDestAddress:1	Indicates whether the Destination address should be updated or not
	unsigned_int UpdateShadowConfig:1	Indicates whether the SHADR should be updated or not
	unsigned_int UpdateSourceAddress:1	Indicates whether the Source address should be updated or not
	uint32 ShadowConfig	Contents of SHADR register
	uint32 SourceAddress	Start of the source buffer address
	unsigned_int UpdateControlChcsr:1	Indicates whether the chcsr register should be updated or not
	uint32 DataCrc	Contents of RDCRC register
Description	Dma_ConfigUpdateType contains the elements which are needed in updating the channel settings of an already configured channel.	
Source	IFX	

1.3.2.4 Dma_CrcType**Table 72 Specification for Dma_CrcType**

Syntax	Dma_CrcType	
Type	Enumeration	
File	Dma.h	
Range	0 - DMA_NO_CRC_TYPE	Invalid option
	1 - DMA_DATA_CRC_TYPE	Refers to Data CRC
	2 - DMA_ADDRESS_CRC_TYPE	Refers to Address CRC
Description	This enum holds the list of the CRC types available	
Source	IFX	

DMA driver**1.3.2.5 Dma_ErrorNotificationPtrType****Table 73 Specification for Dma_ErrorNotificationPtrType**

Syntax	Dma_ErrorNotificationPtrType
Type	Pointer to a function of type void Function_Name (const uint8 Channel, const uint32 Event)
File	Dma.h
Description	This function pointer holds the function which needs to be invoked if there is an error event from the move engine. There would be 1 interrupt per resource partition.
Source	IFX

1.3.2.6 Dma_EventsType**Table 74 Specification for Dma_EventsType**

Syntax	Dma_EventsType	
Type	Enumeration	
File	Dma.h	
Range		
0 - DMA_EVENT_NONE		No channel events
1 - DMA_EVENT_CH_RUNNING		Channel is active and running
2 - DMA_EVENT_CH_TRANSFER_COMPLETE		Desired transfers completed
4 - DMA_EVENT_CH_BUFFER_WRAP_SOURCE		Source circular buffer wrap detected
8 - DMA_EVENT_CH_BUFFER_WRAP_DEST		Destination circular buffer wrap detected
16 - DMA_EVENT_CH_UNKNOWN_EVENT		Unknown event due to hardware limitations
32 - DMA_EVENT_CH_TRL_ERROR		Transaction request lost error
64 - DMA_EVENT_ME_SOURCE_ERROR		Bus error while moving data from source
128 - DMA_EVENT_ME_DESTINATION_ERROR		Bus error while moving data to destination
256 - DMA_EVENT_ME_SPB_ERROR		Bus error-SPB
512 - DMA_EVENT_ME_SRI_ERROR		Bus error-SRI
1028 - DMA_EVENT_ME_RAM_ERROR		DMARAM access error
2048 - DMA_EVENT_ME_SAFE_LINKEDLIST_ERROR		CRC comparison error in safe linked list
4096 - DMA_EVENT_ME_DMA_LINKEDLIST_ERROR		DMARAM access error (during over write of TCS in linked list scenario)
Description	Dma_EventsType is an enumeration holding the list of DMA events, a no-event scenario and an active channel condition.	
Source	IFX	

DMA driver**1.3.2.7 Dma_MoveEngineListType****Table 75 Specification for Dma_MoveEngineListType**

Syntax	Dma_MoveEngineListType	
Type	Enumeration	
File	Dma.h	
Range	0 - DMA_ME_NONE	Invalid option
	1 - DMA_ME_0	Refers to Move Engine 0
	2 - DMA_ME_1	Refers to Move Engine 1
	4 - DMA_ME_ALL	Refers to both Move Engines
Description	This enum holds the list of the Move Engines available	
Source	IFX	

1.3.3 Functions - APIs**1.3.3.1 Dma_Init****Table 76 Specification for Dma_Init API**

Syntax	void Dma_Init (const Dma_ConfigType * const ConfigPtr)	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to root data structure of all post build configurations
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface is meant to be called by the ECU initialization routine. The move engines, the resource partitions and the channels are initialized.	
Source	IFX	

DMA driver**Table 76 Specification for Dma_Init API (continued)**

Error handling	<p>DET:</p> <p>DMA_E_NULL_POINTER: NULL pointer</p> <p>DMA_E_ALREADY_INITIALIZED: DMA driver initialization requested despite the driver already initialized</p> <p>DMA_E_MASTER_UNINIT: A slave core initialization is attempted, before initializing the master core.</p> <p>DMA_E_CORE_NOT_CONFIGURED: Channel not configured for the particular core 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	-

1.3.3.2 Dma_IsInitDone**Table 77 Specification for Dma_IsInitDone API**

Syntax	Std_ReturnType Dma_IsInitDone (void)	
Service ID	0x02	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK if the driver has been initialized, E_NOT_OK otherwise
Description	This interface is meant to provide the status whether the initialization process for the DMA driver has been executed or not.	
Source	IFX	

DMA driver**Table 77 Specification for Dma_IsInitDone 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	Note: Dma_InitDone API only indicates if the module initialization was invoked or not. In case channels were stopped, the user shall not rely or use this API to derive the initialization status of the channel.

1.3.3.3 Dma_ChInit**Table 78 Specification for Dma_ChInit API**

Syntax	void Dma_ChInit (const uint8 Channel)	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface is meant to be called to initialize the specified DMA channel with parameters defined in DMAChannelTransactionSet container. This interface is called by the owner (e.g. ADC driver) of the specified channel. A typical use case would be that an ongoing transfer was aborted by invocation of Dma_ChStopTransfer entailing re-initialization of the channel. Any previously triggered interrupt statuses would be cleared when the channel init is done.	
Source	IFX	

DMA driver**Table 78 Specification for Dma_ChInit API (continued)**

Error handling	<p>DET:</p> <ul style="list-style-type: none"> DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_CH_ALREADY_INITIALIZED: DMA channel initialization requested despite the channel already initialized DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked <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	Hint: If a DMA channel is reset (for e.g. due to invocation of 'Dma_ChStopTransfer' or 'Dma_ChDelInit' APIs), the user has to invoke 'Dma_ChInit' to restore the initialization status of the channel.

1.3.3.4 Dma_ChUpdate**Table 79 Specification for Dma_ChUpdate API**

Syntax	<pre>void Dma_ChUpdate (const uint8 Channel, const Dma_ConfigUpdateType * const Config, const uint32 * const NodeAddress)</pre>	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel Config NodeAddress	DMA channel number Pointer to configuration data Start of a memory block which shall be formatted with information in "Config" object
Parameters (out)	-	-
Parameters (in-out)	-	-
Return	void	-

DMA driver**Table 79 Specification for Dma_ChUpdate API (continued)**

Description	In certain cases, applications wishing to use the DMA are informed of transfer attributes only at run time. In such cases, it is not possible to use the post-build configuration data. This interface may be used to program transfer characteristics at run time.
Source	IFX
Error handling	<p>DET:</p> <p>DMA_E_NULL_POINTER: NULL pointer</p> <p>DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init</p> <p>DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter</p> <p>DMA_E_INVALID_SHADOW_CONFIG_REQ: Dma_ChUpdate API requested for Shadow register update while ADICR is configured in non-shadow configuration</p> <p>DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked</p> <p>DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress</p> <p>DMA_E_CHANNEL_INVALID_START_REQ: Dma_ChUpdate API invoked with invalid CHCSR configuration, since SCH bit should not be set.</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	Note: Dma_ChUpdate API should not be called by the user with invalid TCS contents (reserved bit fields selected, unused features selected, read-only or status bits set).

1.3.3.5 Dma_ChDeInit**Table 80 Specification for Dma_ChDeInit API**

Syntax	<pre>void Dma_ChDeInit (const uint8 Channel)</pre>	
Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-

DMA driver**Table 80 Specification for Dma_ChDeInit API (continued)**

Parameters (in - out)	-	-
Return	void	-
Description	This interface is meant to be called to de-initialize a previously configured channel.	
Source	IFX	
Error handling	<p>DET:</p> <ul style="list-style-type: none"> DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress Runtime Errors: None DEM: None Safety Errors: None <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	DmaChDeInitApiConfiguration	
User hints	-	

1.3.3.6 Dma_ChTransferFreeze**Table 81 Specification for Dma_ChTransferFreeze API**

Syntax	void Dma_ChTransferFreeze (const uint8 Channel)	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-

DMA driver**Table 81 Specification for Dma_ChTransferFreeze API (continued)**

Return	void	-
Description	This interface may be used to temporarily freeze an ongoing transfer. This is typically useful in situations where a block of memory which is an endpoint in a DMA transaction is required to undergo memory testing by a diagnostics task scheduled periodically.	
Source	IFX	
Error handling	<p>DET:</p> <ul style="list-style-type: none"> DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_NO_TRANSFERS_PENDING: Freeze requested when no transfers are pending DMA_E_TIMEOUT: Timeout detected while waiting for a certain value of critical register bit fields DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked <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	DmaSuspendApiConfiguration	
User hints	-	

1.3.3.7 Dma_ChTransferResume**Table 82 Specification for Dma_ChTransferResume API**

Syntax	<pre>void Dma_ChTransferResume (const uint8 Channel)</pre>	
Service ID	0x07	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-

DMA driver**Table 82 Specification for Dma_ChTransferResume API (continued)**

Return	void	-
Description	This interface may be used to resume a previously frozen channel.	
Source	IFX	
Error handling	<p>DET:</p> <ul style="list-style-type: none"> DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_NOT_IN_FREEZE_STATE: Resume requested without a prior freeze DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked <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	DmaSuspendApiConfiguration	
User hints	The resume interface would put the channel from the freeze (halt) state to the idle state. To continue any DMA operations, which was in progress, the user has to call the start transfer interface.	

1.3.3.8 Dma_ChEnableHardwareTrigger**Table 83 Specification for Dma_ChEnableHardwareTrigger API**

Syntax	void Dma_ChEnableHardwareTrigger (const uint8 Channel)	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

DMA driver**Table 83 Specification for Dma_ChEnableHardwareTrigger API (continued)**

Description	This interface is meant to be used to enable (or re-enable) hardware trigger recognition capability. This is particularly important if the channel has been configured for a DMA trigger to be asserted upon detection of a hardware trigger from a peripheral. In "Hardware Trigger-Single" mode of operation, the hardware trigger detection capability is lost after a transaction and shall be reinstalled. This API is precisely meant to do just that.
Source	IFX
Error handling	<p>DET:</p> <ul style="list-style-type: none"> DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_NOT_IN_FREEZE_STATE: Resume requested without a prior freeze DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked DMA_E_FREEZE_STATE: Data transfer start/channel update requested by application during freeze state DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress Runtime Errors: None DEM: None Safety Errors: None <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	DmaTriggerApiConfiguration
User hints	Note: The Dma_ChEnableHardwareTrigger API should not be called for a channel which is configured as 'no hardware trigger'

1.3.3.9 Dma_ChDisableHardwareTrigger**Table 84 Specification for Dma_ChDisableHardwareTrigger API**

Syntax	<pre>void Dma_ChDisableHardwareTrigger (const uint8 Channel)</pre>	
Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number

DMA driver**Table 84 Specification for Dma_ChDisableHardwareTrigger API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface disables hardware trigger detection capability.	
Source	IFX	
Error handling	<p>DET:</p> <ul style="list-style-type: none"> DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked DMA_E_FREEZE_STATE: Data transfer start/channel update requested by application during freeze state DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress Runtime Errors: None DEM: None Safety Errors: None <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	DmaTriggerApiConfiguration	
User hints	Note: The Dma_ChDisableHardwareTrigger API should not be called for a channel which is configured as 'no hardware trigger'	

1.3.3.10 Dma_ChStartTransfer**Table 85 Specification for Dma_ChStartTransfer API**

Syntax	<pre>void Dma_ChStartTransfer (const uint8 Channel)</pre>	
Service ID	0x0A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number

DMA driver**Table 85 Specification for Dma_ChStartTransfer API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used by users of DMA channels (e.g. SPI driver) to manually start a DMA transfer.	
Source	IFX	
Error handling	DET: DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_FREEZE_STATE: Data transfer start/channel update requested by application during freeze state DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

1.3.3.11 Dma_ChStopTransfer**Table 86 Specification for Dma_ChStopTransfer API**

Syntax	<pre>void Dma_ChStopTransfer (const uint8 Channel)</pre>	
Service ID	0x0B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number

DMA driver**Table 86 Specification for Dma_ChStopTransfer API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used by users of DMA channels (e.g. ADC driver) to abort an ongoing transfer.	
Source	IFX	
Error handling	DET: DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_TIMEOUT: Timeout detected while waiting for a certain value of critical register bit fields DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked 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 channel stoppage would result in the hardware resetting the registers of the channel being used. To use the channel again, the user should call the 'Dma_ChInit' API, to get the registers re-initialized.	

1.3.3.12 Dma_ChGetRemainingData**Table 87 Specification for Dma_ChGetRemainingData API**

Syntax	<pre>uint32 Dma_ChGetRemainingData (const uint8 Channel)</pre>	
Service ID	0x0C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number

DMA driver**Table 87 Specification for Dma_ChGetRemainingData API (continued)**

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Pending number of Dma transfers. Note: <i>If the channel is active (transfers are ongoing) or if there are DET/safety errors present, the API will return the value 0xFFFFFFFF. If 0xFFFFFFFF is returned by the API, the caller should check for the presence of any DET/Safety error.</i>
Description	This interface may be used by users of DMA channels to find out how many DMA transfers are pending to copy data from source to destination address.	
Source	IFX	
Error handling	DET: DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	DmaDataPendingApiConfiguration	
User hints	-	

1.3.3.13 Dma_ChSwitchBuffer**Table 88 Specification for Dma_ChSwitchBuffer API**

Syntax	void Dma_ChSwitchBuffer (const uint8 Channel)
Service ID	0x0D
Sync/Async	Synchronous
ASIL Level	B

DMA driver**Table 88 Specification for Dma_ChSwitchBuffer API (continued)**

Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This interface may be used in a double buffering scheme, to redirect the data from the source to destination in either of the following ways:</p> <ol style="list-style-type: none"> 1. From two source buffers to a destination buffer or 2. From a source buffer to two destination buffers. 	
Source	IFX	
Error handling	<p>DET:</p> <ul style="list-style-type: none"> DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel DMA_E_CHANNEL_INVALID_SWITCH_REQ: Switch Buffer API invoked despite the feature not configured DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress Runtime Errors: None DEM: None Safety Errors: None <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	DmaBufferSwitchApiConfiguration	
User hints	-	

1.3.3.14 Dma_GetEvents**Table 89 Specification for Dma_GetEvents API**

Syntax	<pre>uint32 Dma_GetEvents (const uint8 Channel)</pre>
Service ID	0x0E

DMA driver**Table 89 Specification for Dma_GetEvents API (continued)**

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	<p>The 32bit status word containing the consolidated channel events.</p> <p><i>Note:</i> If the DMA channel is active, the API would return 'DMA_EVENT_CH_RUNNING'. In that case, the caller should call the API again to get the events information from DMA.</p> <p><i>Note:</i> The API would return zero in case: (1) If there are any DET or safety errors present or (2) If there are no events recorded in the SFRs. When the return value is zero, the caller should check for the presence of DET or safety errors.</p>
Description	This interface may be used by users of DMA channels to get all the events which occurred for the DMA channel. The events include the channel events and the move engine events.	
Source	IFX	
Error handling	<p>DET:</p> <p>DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init</p> <p>DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel</p> <p>DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked</p> <p>DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note:</i> All DET IDs are also reported as safety errors.</p>	
Configuration dependencies	-	
User hints	-	

DMA driver**1.3.3.15 Dma_ChStatusClear****Table 90 Specification for Dma_ChStatusClear API**

Syntax	<pre>void Dma_ChStatusClear (const uint8 Channel, const Dma_EventsType Event)</pre>	
Service ID	0x0F	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
	Event	Event whose status is to be cleared
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used by users of DMA channels (e.g. SPI driver) to reset the specified status event.	
Source	IFX	
Error handling	<p>DET:</p> <p>DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init</p> <p>DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel</p> <p>DMA_E_CHANNEL_INVALID_EVENT: Incorrect channel event</p> <p>DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter</p> <p>DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked</p> <p>DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress</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	-	

DMA driver**1.3.3.16 Dma_ChInterruptEnable****Table 91 Specification for Dma_ChInterruptEnable API**

Syntax	<pre>void Dma_ChInterruptEnable (const uint8 Channel, const Dma_EventsType Event)</pre>	
Service ID	0x10	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
	Event	Event whose status is to be cleared
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used by users of DMA channels (e.g. SPI driver) to enable interrupt generation upon occurrence of the specified event.	
Source	IFX	
Error handling	<p>DET:</p> <p>DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init</p> <p>DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel</p> <p>DMA_E_CHANNEL_INVALID_EVENT: Incorrect channel event</p> <p>DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter</p> <p>DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked</p> <p>DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress</p> <p>DMA_E_CH_NO_NOTIFICATION_CONFIGURED: No notification function is configured for channel interrupt</p> <p>DMA_E_FREEZE_STATE: Data transfer start/channel update requested by application during freeze 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	-	

DMA driver**Table 91 Specification for Dma_ChInterruptEnable API (continued)**

User hints	-
------------	---

1.3.3.17 Dma_ChInterruptDisable**Table 92 Specification for Dma_ChInterruptDisable API**

Syntax	<pre>void Dma_ChInterruptDisable (const uint8 Channel, const Dma_EventsType Event)</pre>	
Service ID	0x11	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
	Event	Event whose status is to be cleared
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used by users of DMA channels (e.g. SPI driver) to disable interrupt generation upon occurrence of the specified event.	
Source	IFX	
Error handling	<p>DET:</p> <p>DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init</p> <p>DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel</p> <p>DMA_E_CHANNEL_INVALID_EVENT: Incorrect channel event</p> <p>DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter</p> <p>DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked</p> <p>DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress</p> <p>DMA_E_FREEZE_STATE: Data transfer start/channel update requested by application during freeze 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>	

DMA driver**Table 92 Specification for Dma_ChInterruptDisable API (continued)**

Configuration dependencies	-
User hints	-

1.3.3.18 Dma_GetVersionInfo**Table 93 Specification for Dma_GetVersionInfo API**

Syntax	void Dma_GetVersionInfo (Std_VersionInfoType * const VersionInfoPtr)	
Service ID	0x12	
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	This function provides the version information of the DMA driver	
Source	IFX	
Error handling	DET: DMA_E_NULL_POINTER: NULL pointer Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	DmaGetVersionInfoApiConfiguration,ModuleId,VendorId,SwPatchVersion,SwMinorVersion,SwMajorVersion	
User hints	-	

DMA driver**1.3.3.19 Dma_MEStatusClear****Table 94 Specification for Dma_MEStatusClear API**

Syntax	<pre>Std_ReturnType Dma_MEStatusClear (const Dma_MoveEngineListType MoveEngineId)</pre>	
Service ID	0x13	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	MoveEngineld	The Move Engine number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	<p>The return value would be E_OK if the ME status clear could be done, E_NOT_OK if the clear could not be done due to DET or safety errors.</p> <p><i>Note:</i> If E_NOT_OK is returned by the API, the caller should check for the presence of any DET/Safety errors.</p>
Description	This interface may be used by users of DMA channels clear move engine events. The error flags of both the move engines would be cleared by this interface.	
Source	IFX	
Error handling	<p>DET:</p> <p>DMA_E_MOVE_ENGINE_INVALID_ID: Incorrect channel passed as parameter</p> <p>DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note:</i> All DET IDs are also reported as safety errors.</p>	
Configuration dependencies	-	
User hints	-	

DMA driver**1.3.3.20 Dma_InitCheck****Table 95 Specification for Dma_InitCheck API**

Syntax	<pre>Std_ReturnType Dma_InitCheck (const Dma_ConfigType ConfigPtr)</pre>	
Service ID	0x14	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to root data structure of all post build configurations
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	Indication as to whether the init check was successful or not.
Description	<p>This interface is meant to be called by the upper layer using the DMA, to check if the initialization has been successfully performed or not. For the same reason, this interface is expected to be called after the initialization is done. After the checks, the interface would indicate the success or failure in the return value.</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	DmalInitCheckApi,DmaDevErrorDetect	
User hints	-	

1.3.3.21 Dma_GetCrcValue**Table 96 Specification for Dma_GetCrcValue API**

Syntax	<pre>uint32 Dma_GetCrcValue (const uint8 ChannelId, const Dma_CrcType CrcType)</pre>
Service ID	0x15
Sync/Async	Synchronous

DMA driver**Table 96 Specification for Dma_GetCrcValue API (continued)**

ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	ChannelId	DMA channel ID
	CrcType	Address CRC or Data CRC selection
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	CRC value calculated by the hardware. Note: <i>The API would return zero if there are any DET/Safety errors. If a zero is returned by the API, the caller should check for the presence of DET/Safety errors.</i>
Description	This API allows the user to read the CRC value calculated by the DMA hardware.	
Source	IFX	
Error handling	DET: DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter DMA_E_CH_ALREADY_INITIALIZED: DMA channel initialization requested despite the channel already initialized DMA_E_INVALID_CRC_TYPE_REQ: Dma_GetCrcValue API requested with an invalid CRC type, which is not either Address CRC or Data CRC DMA_E_CRC_NOT_SUPPORTED: Dma_GetCrcValue API requested for a channel which does not support the CRC calculation DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked DMA_E_NULL_POINTER: NULL pointer DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress DMA_E_FREEZE_STATE: Data transfer start/channel update requested by application during freeze state 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 'Dma_GetCrcValue' API provides the CRC value computed by the DMA hardware. Computing the reference CRC and comparing the calculated CRC and reference CRC should be done by the user.	

DMA driver**1.3.3.22 Dma_GetCurrentTimeStamp****Table 97 Specification for Dma_GetCurrentTimeStamp API**

Syntax	uint32 Dma_GetCurrentTimeStamp (void)	
Service ID	0x16	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	The current timestamp value which can range from 0 to 0xFFFFFFFF. <i>Note:</i> If a DET or safety error occurs, the timestamp would be returned as zero. If a zero is returned by the API, the caller should check for the presence of any DET or safety errors.
Description	This API allows the user to read the current time stamp in the DMA hardware. This time stamp can be used to compare against the time stamp appended by the DMA when the data is moved.	
Source	IFX	
Error handling	DET: DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init 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	

DMA driver**1.3.3.23 Dma_IsChannelInitDone****Table 98 Specification for Dma_IsChannelInitDone API**

Syntax	<pre>Std_ReturnType Dma_IsChannelInitDone (const uint8 ChannelId)</pre>	
Service ID	0x17	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	ChannelId	The DMA channel ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	<p>The initialization status of the channel E_OK - The initialization is done E_NOT_OK - The initialization is not done</p> <p><i>Note:</i> The API would return E_NOT_OK if there are any DET/Safety errors. If E_NOT_OK is returned by the API, the caller should check for the presence of any DET/Safety error.</p>
Description	This API allows the user to check if the channel being used is in initialized state or not.	
Source	IFX	
Error handling	<p>DET:</p> <p>DMA_E_DRIVER_NOT_INITIALIZED: An API called before invocation of Dma_Init</p> <p>DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter</p> <p>DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors: None</p> <p><i>Note:</i> All DET IDs are also reported as safety errors.</p>	
Configuration dependencies	-	
User hints	-	

DMA driver**1.3.4 Notifications and Callbacks**

There are no notifications and callbacks for the DMA driver.

1.3.5 Scheduled functions

There are no scheduled functions for the DMA driver.

1.3.6 Interrupt service routines**1.3.6.1 Dma_ChInterruptHandler**

Table 99 Specification for Dma_ChInterruptHandler API

Syntax	<pre>void Dma_ChInterruptHandler (const uint8 Channel)</pre>	
Service ID	Not Applicable	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This is the interrupt service routine of a channel interrupt invoked by the interrupt frame (installed in the interrupt vector table). It identifies the cause of the interrupt, clears the status and invokes registered notification routines.</p> <p>Note: If there are no status flags set in the interrupt registers, the notification function would be given the reason as 'unknown reason'. The user can trigger the appropriate error handling.</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	-	

DMA driver**Table 99 Specification for Dma_ChInterruptHandler API (continued)**

User hints	-
------------	---

1.3.6.2 Dma_MEInterruptDispatcher**Table 100 Specification for Dma_MEInterruptDispatcher API**

Syntax	void Dma_MEInterruptDispatcher (void)	
Service ID	Not Applicable	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Not Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This is the interrupt service routine meant to be called by the interrupt frame (installed in the interrupt vector table). The ISR identifies the move engine responsible for the error interrupt and calls the interrupt handler.	
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	-	

1.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

DMA driver**1.3.7.1 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 101 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
An API called before invocation of Dma_Init.	IFX	DMA_E_DRIVER_NOT_INITIALIZED=1	Dma_GetEvents, Dma_GetCurrentTimeStamp, Dma_GetCrcValue, Dma_IsChannelInitDone, Dma_ChInterruptDisable, Dma_ChInterruptEnable, Dma_ChStatusClear, Dma_ChSwitchBuffer, Dma_ChGetRemainingData, Dma_ChStopTransfer, Dma_ChStartTransfer, Dma_ChDisableHardwareTrigger, Dma_ChEnableHardwareTrigger, Dma_ChTransferResume, Dma_ChTransferFreeze, Dma_ChDeInit, Dma_ChUpdate, Dma_ChInit,Dma_MEStatusClear
A channel API invoked on a channel before initialization of that channel.	IFX	DMA_E_CHANNEL_NOT_INITIALIZED =2	Dma_GetEvents, Dma_ChInterruptDisable, Dma_ChInterruptEnable, Dma_ChStatusClear, Dma_ChSwitchBuffer, Dma_ChGetRemainingData, Dma_ChStopTransfer, Dma_ChStartTransfer, Dma_ChDisableHardwareTrigger, Dma_ChEnableHardwareTrigger, Dma_ChTransferResume, Dma_ChTransferFreeze, Dma_ChDeInit

DMA driver**Table 101 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
Incorrect channel passed as parameter.	IFX	DMA_E_CHANNEL_INVALID_ID=3	Dma_GetEvents, Dma_GetCrcValue, Dma_IsChannelInitDone, Dma_ChInterruptDisable, Dma_ChInterruptEnable, Dma_ChStatusClear, Dma_ChSwitchBuffer, Dma_ChGetRemainingData, Dma_ChStopTransfer, Dma_ChStartTransfer, Dma_ChDisableHardwareTrigger, Dma_ChEnableHardwareTrigger, Dma_ChTransferResume, Dma_ChTransferFreeze, Dma_ChDeInit, Dma_ChUpdate, Dma_ChInit
No notification function is configured for channel interrupt.	IFX	DMA_E_CH_NO_NOTIFICATION_CONFIGURED=18	Dma_ChInterruptEnable
Incorrect channel passed as parameter.	IFX	DMA_E_MOVE_ENGINE_INVALID_ID=17	Dma_MEStatusClear
Incorrect channel event.	IFX	DMA_E_CHANNEL_INVALID_EVENT=4	Dma_ChInterruptDisable, Dma_ChInterruptEnable, Dma_ChStatusClear
NULL pointer.	IFX	DMA_E_NULL_POINTER=5	Dma_GetCrcValue, Dma_GetVersionInfo, Dma_ChUpdate, Dma_Init
Data transfer start/channel update requested by application during freeze state.	IFX	DMA_E_FREEZE_STATE=6	Dma_ChInterruptDisable, Dma_ChInterruptEnable, Dma_ChDisableHardwareTrigger, Dma_ChEnableHardwareTrigger, Dma_GetCrcValue, Dma_ChStartTransfer
Freeze requested when no transfers are pending.	IFX	DMA_E_NO_TRANSFERS_PENDING=7	Dma_ChTransferFreeze
Resume requested without a prior freeze.	IFX	DMA_E_NOT_IN_FREEZE_STATE=8	Dma_ChEnableHardwareTrigger, Dma_ChTransferResume

DMA driver**Table 101 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
Start transfer requested when a data transfer is already in progress.	IFX	DMA_E_DATA_TRANSFER_IN_PROGRESS=9	Dma_ChStatusClear, Dma_ChSwitchBuffer, Dma_ChDisableHardwareTrigger, Dma_ChEnableHardwareTrigger, Dma_GetCrcValue, Dma_ChGetRemainingData, Dma_ChDeInit, Dma_ChUpdate, Dma_ChInterruptEnable, Dma_ChInterruptDisable, Dma_ChStartTransfer
DMA driver initialization requested despite the driver already initialized.	IFX	DMA_E_ALREADY_INITIALIZED=10	Dma_Init
DMA channel initialization requested despite the channel already initialized.	IFX	DMA_E_CH_ALREADY_INITIALIZED=11	Dma_GetCrcValue, Dma_ChInit
Timeout detected while waiting for a certain value of critical register bit fields.	IFX	DMA_E_TIMEOUT=12	Dma_ChStopTransfer, Dma_ChTransferFreeze
Dma_ChUpdate API invoked with invalid CHCSR configuration, since SCH bit should not be set.	IFX	DMA_E_CHANNEL_INVALID_START_REQ=19	Dma_ChUpdate
Switch Buffer API invoked despite the feature not configured.	IFX	DMA_E_CHANNEL_INVALID_SWITCH_REQ=13	Dma_ChSwitchBuffer
Dma_GetCrcValue API requested for a channel which does not support the CRC calculation.	IFX	DMA_E_CRC_NOT_SUPPORTED=16	Dma_GetCrcValue
Dma_GetCrcValue API requested with an invalid CRC type, which is not either Address CRC or Data CRC.	IFX	DMA_E_INVALID_CRC_TYPE_REQ=15	Dma_GetCrcValue
Dma_ChUpdate API requested for Shadow register update while ADICR is configured in non-shadow configuration.	IFX	DMA_E_INVALID_SHADOW_CONFIG_REQ=14	Dma_ChUpdate

DMA driver**Table 101 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
Channel not configured for the particular core is invoked.	IFX	DMA_E_CORE_NOT_CONFIGURED=100	Dma_Init
Channel not configured for the particular core is invoked.	IFX	DMA_E_CORE_CHANNEL_MISMATCH =101	Dma_GetEvents, Dma_GetCrcValue, Dma_IsChannelInitDone, Dma_ChDelInit, Dma_ChInterruptDisable, Dma_ChStartTransfer, Dma_ChStopTransfer, Dma_ChGetRemainingData, Dma_ChSwitchBuffer, Dma_ChInterruptEnable, Dma_ChStatusClear, Dma_ChUpdate, Dma_ChTransferFreeze, Dma_ChTransferResume, Dma_ChEnableHardwareTrigger, Dma_ChDisableHardwareTrigger, Dma_ChInit
A slave core initialization is attempted, before initializing the master core.	IFX	DMA_E_MASTER_UNINIT=102	Dma_Init

1.3.7.2 Production errors

The driver does not report any production errors.

1.3.7.3 Safety errors

The driver does not report any safety errors.

1.3.7.4 Runtime errors

The driver does not report any runtime errors.

1.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

1.3.8.1 Deviations

The section describes the deviations from software specification.

DMA driver

Table 102 Known deviations

Reference	Deviation
DMA driver writes into the interrupt control registers for the GPSR interrupts.	DMA driver requires accessing the SRC registers of the Interrupt Router in a multicore scenario. The access is needed to route the resource partition interrupts to the appropriate CPU core when an error event occurs.
De-initialization of DMA not supported.	The DMA driver does not support the driver de-initialization feature. Note: De-initialization of individual DMA channels is supported by the driver.

1.3.8.2 Limitations

The section describes the limitations from software specification.

Table 103 Known limitations

Reference	Limitation
The last TCS node of a linked list, cannot be configured as 'autostart'.	This limitation is applicable only if the linked list feature is being used in the autostart mode. If the linked list feature with autostart is used, the last node must not be configured as autostart. For more information, refer to Chapter 21.3.4.9.3 in the Target Specification.
Global addresses should be used when specifying the addresses to be used by DMA.	When specifying the addresses for the DMA transactions, the addresses to be used should be in the global address range and not in the local address range, as the DMA hardware can handle only the global addresses. User should ensure this when configuring the DMA channels.
Central allocation of DMA resources.	The DMA driver does not provide a central allocation of the DMA channels. Therefore, the driver does not validate if the channel allocated to one driver/application software is invoked by another. The user should invoke the DMA APIs with the channel IDs intended to be used by the application software.
Multiple ME interrupts for source and destination errors.	If there are ME errors for source or destination memory, the channel would continue the transaction till all transfers are completed. Because of this, there would be multiple error interrupt notifications for the same transaction.

DMA driver**Table 103 Known limitations (continued)**

Reference	Limitation
Handling of multiple RP error interrupts	<p>As per the hardware design, the DMA IP stores only the last error channel (LEC field of ERRSRm of move engine register) when there are multiple DMA errors. So, if there are multiple DMA RP error interrupts triggered at the same time (faster than the DMA driver RP error ISR processing), there is a possibility that some of the DMA RP error interrupts could be lost. This could result in missing the error notifications to the application which uses DMA.</p> <p>Workaround hint:</p> <p>The application should check for the interrupt overflow flag (IOV bit in SRC register of the DMA RP error interrupt) before invoking the RP error interrupt service routine to the DMA driver. If the IOV flag is set, the application shall take additional measures to evaluate the impact to the users of DMA. (For example, for a missed SPI reception DMA error, the application can monitor the SPI Rx FIFO overflow error flag). The application should clear the IOV bit using the IOVCLR bit of the SRC register, once the handling is complete.</p>
DMA errors when using of SPI and DMA drivers together (Ref: ESM[SW]:DMA:ERROR_HANDLING)	<p>The below limitation is applicable if SPI uses the DMA driver for asynchronous transmission:</p> <p>Whenever the DMA channel used by SPI driver encounters an error, the DMA driver notifies the error along with the channel information to the SPI driver. However, due to the limitation in the error notification interface from SPI to the application, the SPI driver notifies the same error to the application without the DMA channel information.</p> <p>Workaround:</p> <p>The application shall supervise the DMA RP Error Interrupt Service Request. The application shall read the DMA move engine error registers ERRSR0 and ERRSR1 before invoking the interrupt handler provided by DMA. This is to determine the error cause and the DMA channel number which caused the RP error. The sequence is as below:</p> <ul style="list-style-type: none"> - Move engine error occurs in hardware - RP ISR is triggered - Application reads the error flags in ERRSRm for determining the error cause and ERRSRm.LEC for determining the DMA channel number that caused the error. - Call Dma_MEInterruptDispatcher - On return from Dma_MEInterruptDispatcher, Application shall take necessary action.

1.3.9 Unsupported hardware features

The following hardware features are not supported by the DMA driver:

- Pattern detection feature and the corresponding interrupts

DMA driver

- Conditional linked list
- Activation of error interrupts using Error Interrupt Set register

DSADC driver

2 DSADC driver

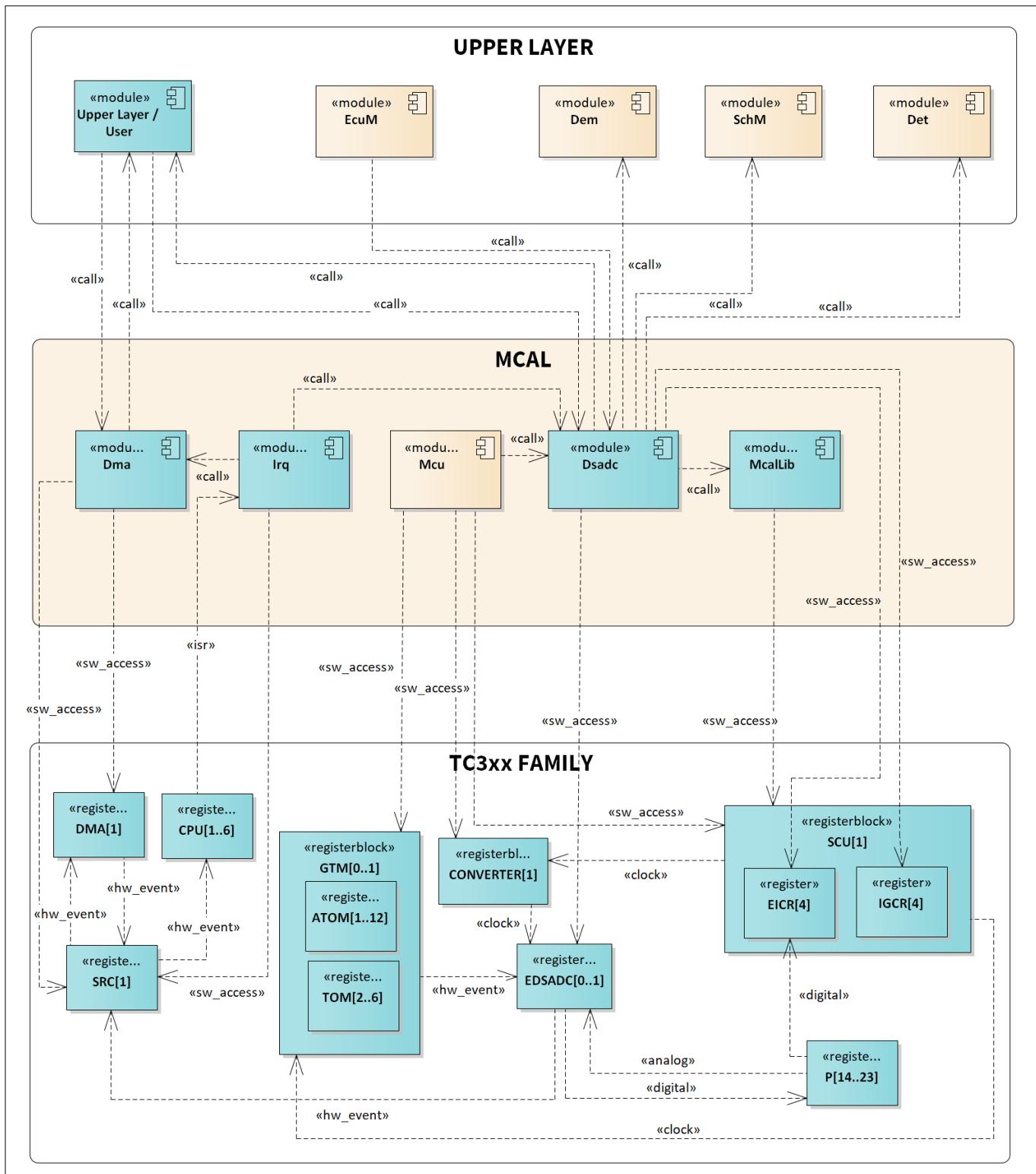
2.1 User information

2.1.1 Description

The DSADC driver provides analog-to-digital conversion based on the Delta Sigma (DS) conversion principle. The DSADC driver provides configurations for various parameters of the functional blocks of the EDSADC IP. The driver is responsible for the initialization and configuration of the channels (internal modulators, demodulators, filter chain) in the EDSADC IP, thus providing interfaces to convert analog input signals to digital data streams at a selectable output rate. The DSADC driver does not support multicore processing. The driver is delivered as a post-build variant.

2.1.2 Hardware-software mapping

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

DSADC driver

Figure 8 **Mapping of hardware-software interfaces**
2.1.2.1 EDSADC: primary hardware peripheral
Hardware functional features

The DSADC driver uses the EDSADC IP for converting the analog signals to digital values. The key hardware functional features used are:

DSADC driver

- Service request generation using external trigger
- Filter chain configuration
- Calibration support
- Timestamp capture
- Carrier signal generation
- Integrator support
- Configuration support for return signal synchronization for resolver support
- Configuration support for limit checking feature

The unsupported features of EDSADC are:

- External modulator support
- Automatic power control
- Trigger signal from port pin
- Trigger signal from GTM ADC trigger lines
- Event handling for Limit checking
- Event handling for return signal synchronization

Users of the hardware

The DSADC driver exclusively utilizes the EDSADC IP.

Hardware diagnostic features

The SMU alarms configured for the EDSADC IP are not monitored by the DSADC driver.

Hardware events

The DSADC driver uses the following hardware events from the EDSADC IP:

- Result event: to trigger the conversion result transfer through DMA in DMA mode or CPU in interrupt mode
- Timestamp trigger event: to read the timestamp information for external trigger event (rising/falling)

2.1.2.2 GTM: dependent hardware peripheral

Hardware functional features

The DSADC driver depends on the GTM IP for realizing the gating features. The DSADC driver uses the compare-match event and the channel output signal for starting and stopping the conversion result acquisition of a DSADC channel. The selection of GTM trigger line for the DSADC channel is done by the DSADC driver and the corresponding TOM/ATOM selection for the GTM trigger line is done by the MCU driver.

Users of the hardware

The GTM IP is used by the PWM, OCU, ICU, WDG, GPT and ADC drivers. The GTM resources used by each driver are reserved through the configuration interface of the MCU driver to avoid resource conflict. The GTM TOM/ATOM configuration is done by the PWM driver to generate the gate signal for the DSADC driver.

Hardware diagnostic features

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

Hardware events

- Compare-match event: to prepare the channel for the result acquisition or raise a window close notification
- Channel output level: to start/stop the conversion result acquisition

2.1.2.3 SCU: dependent hardware peripheral

Hardware functional features

DSADC driver

The DSADC driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB and fPER 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 DSADC driver.

Hardware events

Hardware events from the SCU are not used by the DSADC driver.

2.1.2.4 EICR/IGCR: primary hardware peripheral

Hardware functional features

The DSADC driver uses the ERU IP for realizing the gating feature. The following features of the ERU are used by the driver:

- Pattern Detection
- Generation of interrupt based on the pattern detection output
- External resource selection for input channel
- Input channel trigger logic selection
- Selection of input channels for pattern detection logic per output unit

The unsupported features of the ERU IP are:

- Generation of interrupt based on the trigger output

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 DSADC driver.

Hardware events

- Pattern match/miss event: to prepare the channel for the result acquisition or raise a window close notification
- Pattern detection output level: to start/stop the conversion result acquisition

2.1.2.5 SRC: dependent hardware peripheral

Hardware functional features

The DSADC driver depends on interrupt router for raising an interrupt to the CPU based on the result event which indicates the end of conversion of a channel.

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 DSADC driver.

Hardware events

DSADC driver

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

2.1.2.6 DMA: dependent hardware peripheral

Hardware functional features

The DSADC driver depends on the DMA IP for transferring the conversion results to the application buffer using the DMA channel in the DMA mode of result handling.

Users of the hardware

The DMA channels are configured by the DMA driver.

Hardware Diagnostic Features

The SMU alarms configured for the DMA IP are not monitored by the DSADC driver.

Hardware events

Hardware events from DMA channels are not used by the DSADC driver.

2.1.2.7 CONVERTER: dependent hardware peripheral

Hardware functional features

The DSADC driver depends on the converter control block for the clock synchronization signal. The clock synchronization signal synchronizes the analog clocks of all EDSADC hardware channels.

Users of the hardware

The converter control block is configured by the MCU driver.

Hardware diagnostic features

The SMU alarms configured for the converter control block are not monitored by the DSADC driver.

Hardware events

Not applicable.

2.1.2.8 Port: dependent hardware peripheral

Hardware functional features

The analog signals are routed to the EDSADC through the analog port pads. The external trigger events for the channel are routed through the digital port pad. The generated carrier signal from the EDSADC are routed through the digital port pad. 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

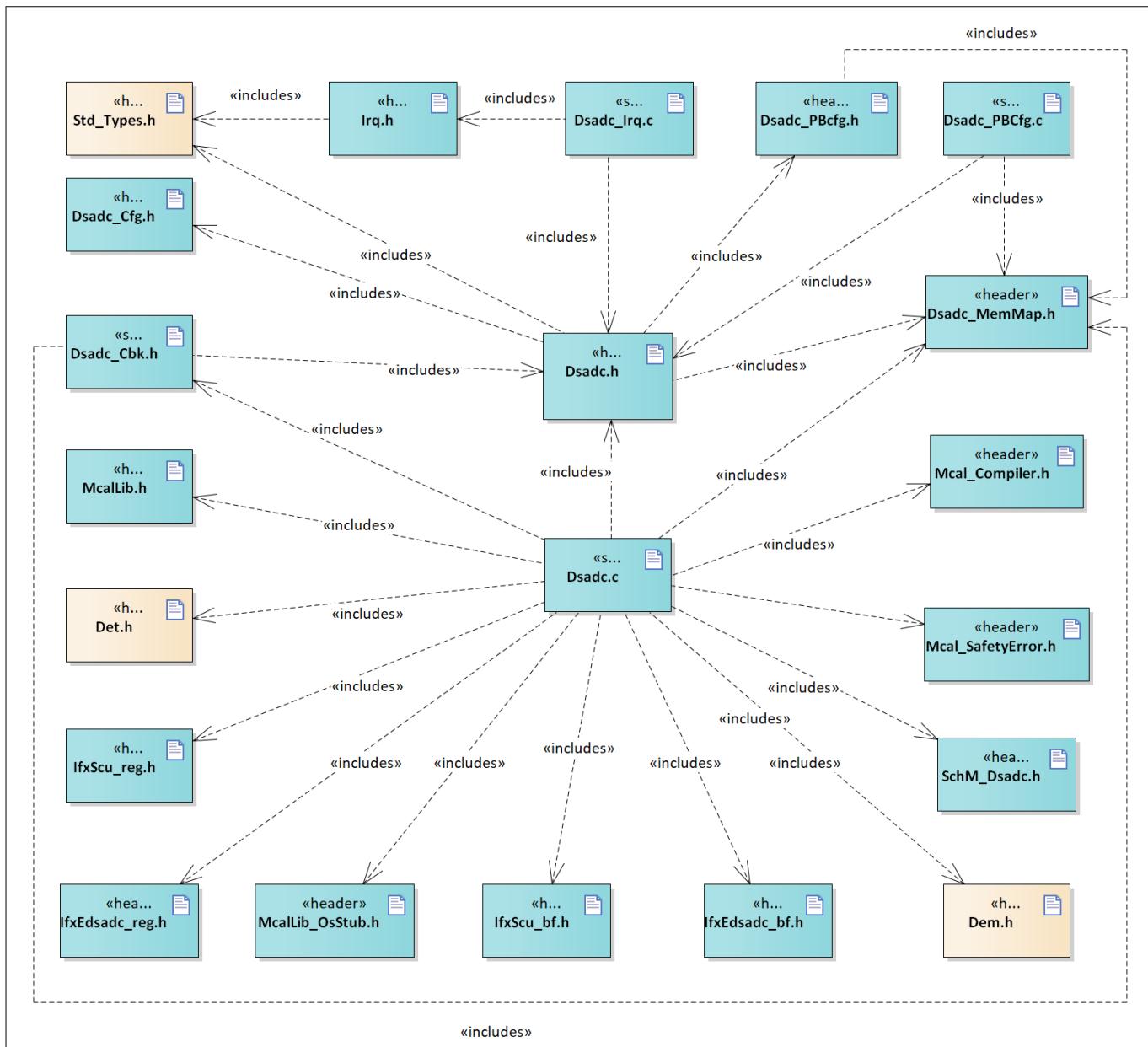
Not applicable.

Hardware events

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

2.1.3 File structure

2.1.3.1 C File Structure

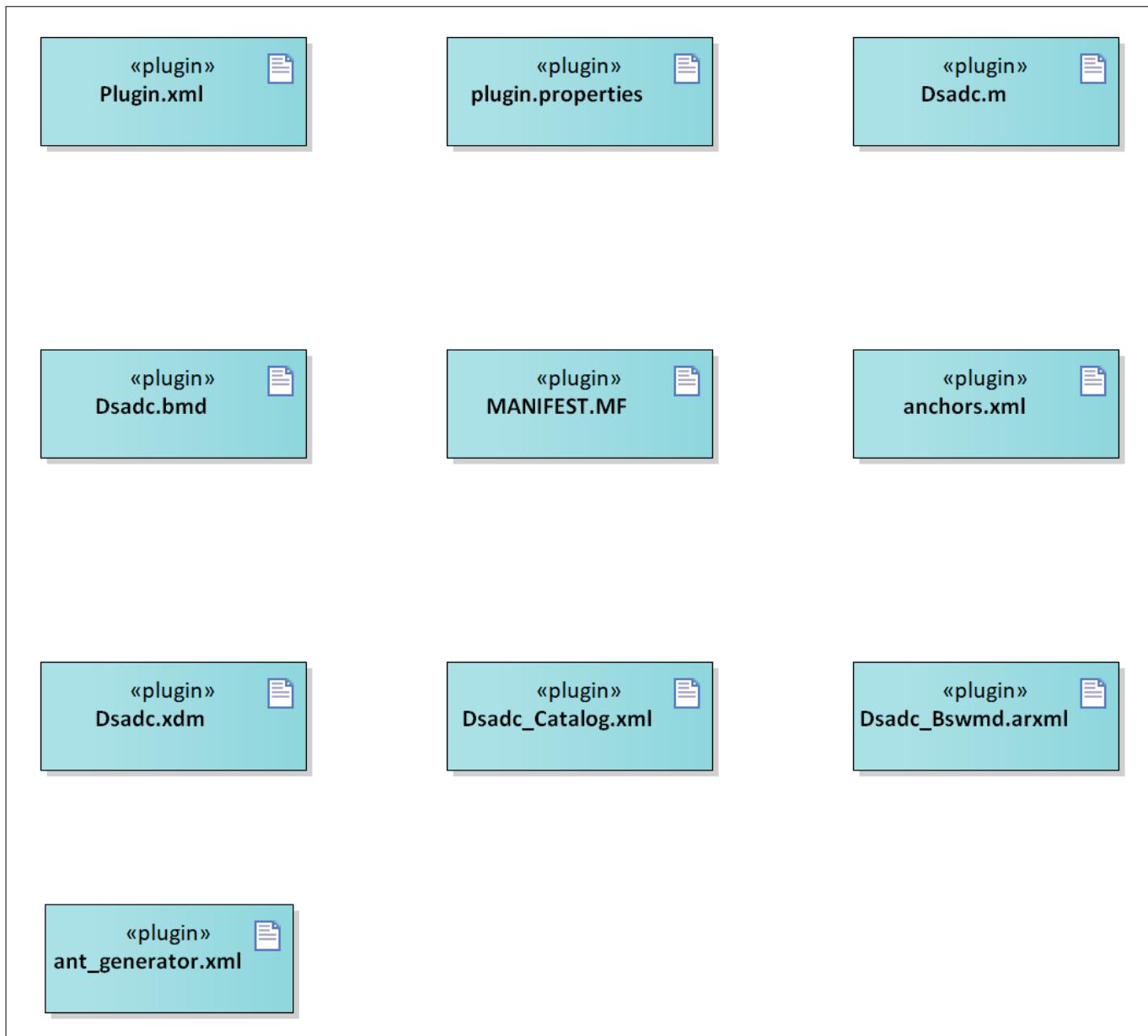
DSADC driver

Figure 9 **C File Structure**
Table 104 **C File Structure**

File name	Description
<code>Dem.h</code>	Provides the exported interfaces of Diagnostic Event Manager
<code>Det.h</code>	Provides the exported interfaces of Development Error Tracer
<code>Dsadc.c</code>	File (Static) containing implementation of APIs
<code>Dsadc.h</code>	Header file (Static) defining prototypes of data structures, APIs and interrupt handlers
<code>Dsadc_Cbk.h</code>	Header file to declare the DSADC callback APIs
<code>Dsadc_Cfg.h</code>	Header file (Generated) containing constants and pre-processor macros as #defines

DSADC driver**Table 104 C File Structure (continued)**

File name	Description
Dsadc_Irq.c	Interrupt handler file for DSADC
Dsadc_MemMap.h	File (Static) containing the memory section definitions used by the DSADC driver
Dsadc_PBCfg.c	File (Generated) containing declaration of the post-build configuration data structures
Dsadc_PBcfg.h	File (Generated) containing declaration of the post-build configuration data structures
IfxEdsadc_bf.h	SFR header file for EDSADC
IfxEdsadc_reg.h	SFR header file for EDSADC
IfxScu_bf.h	SFR header file for SCU
IfxScu_reg.h	SFR header file for SCU
Irq.h	The file exports Mcal compiler specific functions and macros
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
SchM_Dsadc.h	Export header for SchM functions of DSADC
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

2.1.3.2 **Code Generator Plugin Files**

DSADC driver**Figure 10 Code Generator Plugin Files****Table 105 Code Generator Plugin Files**

File name	Description
Dsadc.bmd	AUTOSAR format XML data model schema file
Dsadc.m	Code template macro file for the DSADC driver
Dsadc.xdm	Tresos format XML data model schema file
Dsadc_Bswmd.arxml	AUTOSAR format module description file
Dsadc_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the meta data for the DSADC driver
Plugin.xml	Tresos plugin support file for the DSADC driver
anchors.xml	Tresos anchors support file for the DSADC driver

DSADC driver
Table 105 Code Generator Plugin Files (continued)

File name	Description
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 DSADC driver

2.1.4 Integration hints

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

2.1.4.1 Integration with AUTOSAR stack

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

- **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 `Dsadc_MemMap.h` file.

The `Dsadc_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:

DSADC driver

```

/*To be used for all global or static variables.*/
#if defined DSADC_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
/* User Pragma here */
#define DSADC_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
#define MEMMAP_ERROR
#elif defined DSADC_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
/* User Pragma here */
#define DSADC_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
#define MEMMAP_ERROR
#elif defined DSADC_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_8
/* User Pragma here */
#define DSADC_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_8
#define MEMMAP_ERROR
#elif defined DSADC_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_8
/* User Pragma here */
#define DSADC_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_8
#define MEMMAP_ERROR

/* DSADC module configuration data */
#elif defined DSADC_START_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
/* User Pragma here */
#define DSADC_START_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined DSADC_STOP_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
/* User Pragma here */
#define DSADC_STOP_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR

/* Code Sections */
#elif defined DSADC_START_SEC_CODE_ASIL_B_LOCAL
/* your Pragma here */
#define DSADC_START_SEC_CODE_ASIL_B_LOCAL
#define MEMMAP_ERROR
#elif defined DSADC_STOP_SEC_CODE_ASIL_B_LOCAL
/* your Pragma here */
#define DSADC_STOP_SEC_CODE_ASIL_B_LOCAL
#define MEMMAP_ERROR
#endif

#if defined MEMMAP_ERROR
#error "Dsadc_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 DSADC driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the DMA 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.

DSADC driver

- **DEM**

The DEM module is a part of the AUTOSAR stack that handles all the production errors reported by the BSW modules. The DSADC driver reports all the production errors to the DEM modules through the `Dem_ReportErrorStatus()` API. The user of the DSADC 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 `Dsadc_ReadStreamResults()` and `Dsadc_ReadResult()` APIs is dependent on the re-entrancy of the `Dem_ReportErrorStatus()` API. As per their design, the module APIs are re-entrant for different channels. 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 DSADC driver uses the exclusive areas defined in the `SchM_Dsadc.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchM identified for the DSADC driver is: `ChannelData`

The `SchM_Dsadc.h` and `SchM_Dsadc.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 DSADC 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_Dsadc.c *****/
#include "Os.h"
void SchM_Enter_Dsadc_ChannelData(void)
{
/* Start of Critical Section */
SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}
void SchM_Exit_Dsadc_ChannelData(void)
{
/* End of Critical Section */
ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

The DSADC 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 DSADC driver does not implement any notifications. However, the DSADC driver reports the detection of new conversion result (for access mode not configured for DMA Access), window close event (for trigger mode window) and buffer full event (for access mode linear buffer) through notification functions. These notification functions can be configured by the user in Tresos for each DSADC Channel.

DSADC driver

The driver does not expect any callbacks from the application, however the driver requires the callback ISR from the MCU.

- **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.

2.1.4.2 Multicore and Resource Manager

The DSADC driver does not support execution on multiple cores simultaneously.

2.1.4.3 MCU support

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

- The Phase synchronizer inside the CONVCTRL block must be programmed according to the required EDSADC modulator frequency, that is, $f_{PHSYNC} = f_{MOD}$ when the synchronized mode is selected in DSADC driver. The configuration and programming of the CONVCTRL block is managed directly by the MCU driver.
- DSADC channel may require gate signal generated by a GTM timer for result data acquisition. DSADC driver shall not configure the GTM channel to generate the gate signal. PWM driver may reserve the GTM channel and generate gate signal for DSADC channel result data acquisition. The same GTM channel needs to be configured in MCU in corresponding GtmchannelForDsadc parameter
- DSADC channel may require gate signal generated by EICR-IGCR for result data acquisition. The EICR-IGCR channels used by the DSADC driver must be reserved in the MCU configuration for exclusive use by the DSADC.

2.1.4.4 Port support

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

2.1.4.5 DMA support

The DSADC 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 channel-wise feature.

The result register event from EDSADC 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.

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

DSADC driver

- Configuration to enable the DMA-based result transfer must be done through the EB tresos parameter: DsadcAccessMode.
- DMA channels intended to be used for the DSADC channel must be reserved and configured through the DMA driver in the EB tresos.
- DSADC driver does not configure the DMA channels. The user of the DSADC should invoke proper DMA APIs to start/stop the DMA channels before starting/stopping an DSADC channel.
- DSADC channel result register address must be configured as a source address in the corresponding DMA channel.
- 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 DSADC 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.

2.1.4.6 Interrupt connections

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

- **Result handling in interrupt mode**

Conversion result transfer in interrupt is selected when DsadcAccessMode is not equal to DSADC_DMA_ACCESS. In this mode, the conversion results are transferred from result registers to the application buffers in the ISR. The following figure depicts the interrupt connections required by the DSADC driver.

Note: *User shall ensure that correct hardware channel id is passed while invoking Dsadc_Isr() from the interrupt frame. That is, in DSADC0SRGM_ISR, Dsadc_Isr(0) shall be invoked, in DSADC13SRGM_ISR, Dsadc_Isr(13) shall be invoked*

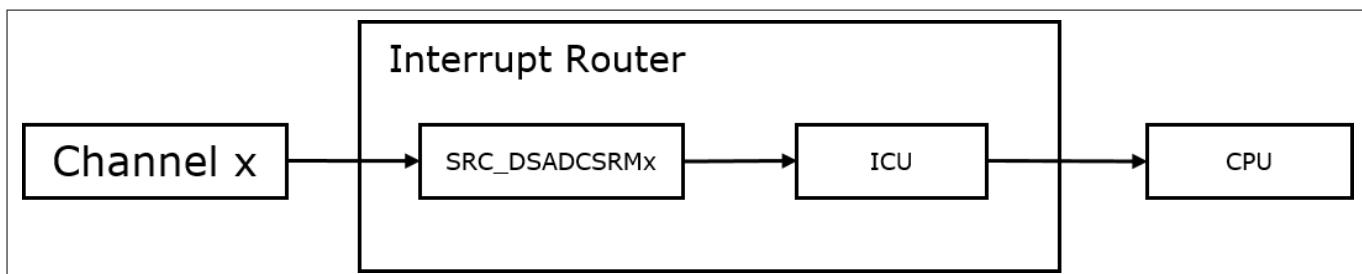


Figure 11 Result handling in the interrupt mode

DSADC driver

Invoking the interrupt handlers provided by the driver must be done by the user. A sample invocation for Channel 0 and channel 6 is shown as follows:

```
#include "Dsadc.h"

/* EDSADC Channel 0 */
/****************** DSADC Channel 0 Main service Request*****/
ISR(DSADC0SRGM_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    #if(DSADC_ALL_CH_RESULT_HANDLING_DMA != STD_ON)
        /* Call Dsadc Main Service request Interrupt function*/
        Dsadc_Isr(0); /* 0 indicates the HW channel number */
    #endif
}

/* EDSADC Channel 6 */
/****************** DSADC Channel 6 Main service Request*****/
ISR(DSADC6SRGM_ISR)
{
    /* Enable Global Interrupts */
    ENABLE();
    #if(DSADC_ALL_CH_RESULT_HANDLING_DMA != STD_ON)
        /* Call Dsadc Main Service request Interrupt function*/
        Dsadc_Isr(6); /* 6 indicates the HW channel number */
    #endif
}
```

- **Result handling in the DMA mode:**

Conversion result transfer through DMA is selected when DsadcAccessMode is equal to DSADC_DMA_ACCESS. 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.

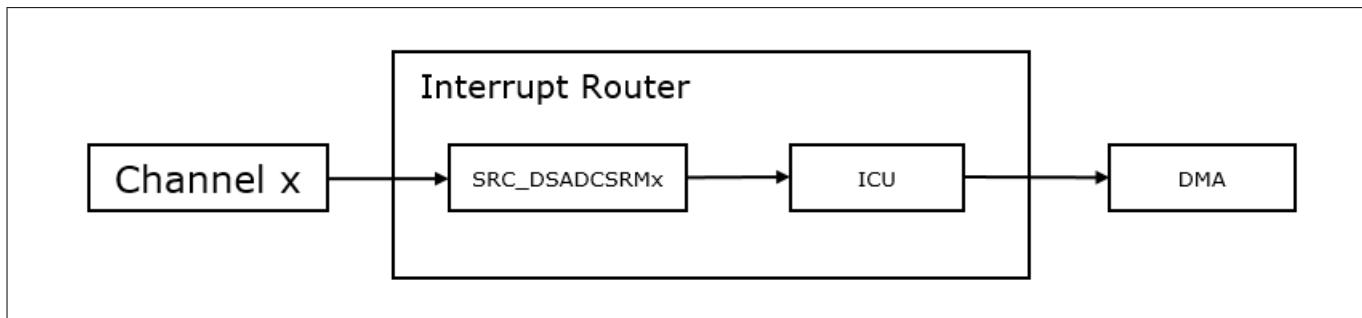


Figure 12 Result handling in the DMA mode

DSADC driver**2.1.4.7 Example usage**

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

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

Initialization of the driver

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

```

/*
Configuration values mandatory for below code snippet:-
DsadcAccessMode = DSADC_DMA_ACCESS: Then Dma_Init() is required prior to use
of runtime DSADC services.
*/

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

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

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

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

/* DSADC Initialization */
Dsadc_Init(&Dsadc_Config);

/* Enable Interrupts for used Dsadc Hardware (x) */
SRC_DSADC_DSADCx_SRM.B.SRE = 1U;
/* Further APIs of DSADC driver can be called now */

```

Calibration for Configured channel

DSADC driver

The code sequence for starting the calibration and checking for the calibration status is follows:

```

/* Each Dsadc channel has to be calibrated using the service
Dsadc_StartCalibration provided by the Dsadc Driver after the Reset.*/

#include 'Dsadc.h'

Dsadc_ChannelType ChannelId;
Dsadc_CalibrationStatusType CalibStatus;
Std_ReturnType lRetVal;

/* DsadcChannel_DsadcChannelConfiguration_x is a valid SW channel ID macro
Generated in Dsadc_Cfg.h */

ChannelId = DsadcChannel_DsadcChannelConfiguration_x;

lRetVal = Dsadc_StartCalibration(ChannelId);

if(lRetVal!=E_NOT_OK)
{
    Do
    {
        CalibStatus = Dsadc_GetCalibrationStatus (ChannelId);
        /* Wait till the Start Calibration is over*/
        }while(CalibStatus== DSADC_CALIBRATION_RUNNING);

    If(CalibStatus == DSADC_CALIBRATION_DONE)
    {
        /* Calibration is successful */
    }
    else
    {
        /* Calibration is failed */
    }
}
else
{
    /*Could not start the calibration*/
}

```

Software-triggered stream result read (linear buffer)

DSADC driver

The code sequence for setting up the linear buffer and reading the conversion results from the buffer is follows:

```

/*
Configuration values mandatory for below code snippet:-
1. DsadcAccessMode = DSADC_STREAM_LINEAR_BUFFER: Dsadc Channel Access mode is
Linear buffer*/
2. DsadcTriggerMode = DSADC_TRIGGER_MODE_NORMAL: Conversion Result Acquisition
will start Dsadc_StartModulation API calls
3. DsadcBufferFullNotification = FunctionRead: In this Example Buffer Full
notification is enabled. So once the Buffer is Full user shall get this
function call
*/

#include 'Dsadc.h'

Dsadc_ChannelType ChannelId;
Dsadc_ResultType DataBufferPtr[25]; // Assuming buffer size of 25 is
sufficient for the Dsadc Channel
Dsadc_ResultType UserBufferPtr[25]
Std_ReturnType lRetVal;

/* DsadcChannel_DsadcChannelConfiguration_x is a valid SW channel ID macro
Generated in Dsadc_Cfg.h */
ChannelId = DsadcChannel_DsadcChannelConfiguration_x;

lRetVal = Dsadc_SetupResultBuffer(ChannelId, &DataBufferPtr[0],25);
if(lRetVal!=E_NOT_OK)
{
    Dsadc_EnableNotifications(ChannelId);
    lRetVal = Dsadc_StartModulation(ChannelId);
    if(lRetVal!=E_NOT_OK)
    {
        /* Result Acquisition is started. Result data can be read once the buffer is
Full. Assuming that only buffer full notification is enabled for this channel
*/
    }
    Else
    {
        /* Result Acquisition is not started */
    }
}
else
{
    /* Could not setup the result buffer */
}
/* Buffer Full notification function */
In FunctionRead()
{
    Dsadc_ReadStreamResults(ChannelId, &UserBufferPtr);
}

```

DSADC driver**Software-triggered single result read**

The code sequence for setting up the channel for single access and reading the result is follows:

```

/*
Configuration values mandatory for below code snippet:-
1. DsadcAccessMode = DSADC_SINGLE_READ: Dsadc Channel Access mode is Single
Read/
2. DsadcTriggerMode = DSADC_TRIGGER_MODE_NORMAL: Conversion Result Acquisition
will start after Dsadc_StartModulation API calls
3. DsadcNewResultNotification = FunctionRead: In this Example New Result
notification event is enabled. So once the conversion is completed user shall
get this function call
*/

#include 'Dsadc.h'

Dsadc_ChannelType ChannelId;
Dsadc_ResultType ConversionResult;
Std_ReturnType lRetVal;

/* DsadcChannel_DsadcChannelConfiguration_x is a valid SW channel ID macro
Generated in Dsadc_Cfg.h */
ChannelId = DsadcChannel_DsadcChannelConfiguration_x;

Dsadc_EnableNotifications(ChannelId);
lRetVal = Dsadc_StartModulation(ChannelId);
if(lRetVal!=E_NOT_OK)
{
    /* Result Acquisition is started. Result data can be read once the conversion
    is completed.Assuming that New Result notification is enabled for this channel
*/
}
Else
{
    /* Result Acquisition is not started */
}

```

Software-triggered circular buffer read

DSADC driver

The code sequence for setting up the circular buffer and reading the conversion results from the buffer is follows:

```

/*
Configuration values mandatory for below code snippet:-
1. DsadcAccessMode = DSADC_CIRCULAR_BUFFER: Dsadc Channel Access mode is
circular buffer*
2. DsadcTriggerMode = DSADC_TRIGGER_MODE_NORMAL: Conversion Result Acquisition
will start Dsadc_StartModulation API calls
*/

#include 'Dsadc.h'

Dsadc_ChannelType ChannelId;
Dsadc_ResultType DataBufferPtr[25]; // Assuming buffer size of 25 is
sufficient for the Dsadc Channel
Dsadc_ResultType ConversionResult;
Dsadc_ChannelstatusType ChannelStatus;
Std_ReturnType lRetVal;

/* DsadcChannel_DsadcChannelConfiguration_x is a valid SW channel ID macro
Generated in Dsadc_Cfg.h */
ChannelId = DsadcChannel_DsadcChannelConfiguration_x;

lRetVal = Dsadc_SetupResultBuffer(ChannelId, &DataBufferPtr[0],25);
if(lRetVal!=E_NOT_OK)
{
    lRetVal = Dsadc_StartModulation(ChannelId);
    if(lRetVal!=E_NOT_OK)
    {
        /* Result Acquisition is started. Result data can be read anytime after the
first conversion results are available*/
    }
    Else
    {
        /* Result Acquisition is not started */
    }
}
else
{
    /* Could not setup the result buffer */
}

/* Read Single conversion results from the circular buffer */
In FunctionRead()
{
    ChannelStatus = Dsadc_GetStatus(ChannelId);
    If(ChannelStatus == DSADC_RESULT_READY)
    {
        /* read the circular buffer data */
        Dsadc_ReadResult (ChannelId, &ConversionResult);
    }
}

```

DSADC driver

```
Else
{
/* Buffer is Empty */
}
```

Hardware-triggered stream result read (linear buffer)

DSADC driver

The code sequence for setting up the linear buffer and reading the conversion results from the buffer is follows:

```

/*
Configuration values mandatory for below code snippet:-
1. DsadcAccessMode = DSADC_STREAM_LINEAR_BUFFER: Dsadc Channel Access mode is
Linear buffer*/
2. DsadcTriggerMode = DSADC_TRIGGER_MODE_WINDOW: Conversion Result Acquisition
will start after the Window Open Event
3. DsadcWindowCloseNotification = FunctionRead: In this Example Window close
notification is enabled. So once the window close event, user shall get this
function call
*/

#include 'Dsadc.h'

Dsadc_ChannelType ChannelId;
Dsadc_ResultType DataBufferPtr[25]; // Assuming buffer size of 25 is
sufficient for the Dsadc Channel
Dsadc_ResultType UserBufferPtr[25]
Std_ReturnType lRetVal;

/* DsadcChannel_DsadcChannelConfiguration_x is a valid SW channel ID macro
Generated in Dsadc_Cfg.h */
ChannelId = DsadcChannel_DsadcChannelConfiguration_x;

lRetVal = Dsadc_SetupResultBuffer(ChannelId, &DataBufferPtr[0],25);
if(lRetVal!=E_NOT_OK)
{
    Dsadc_EnableNotifications(ChannelId);
    lRetVal = Dsadc_StartModulation(ChannelId);
    if(lRetVal!=E_NOT_OK)
    {
        /* Result Acquisition will start after the window Open event*/
    }
    Else
    {
        /* Result Acquisition is not started */
    }
}
else
{
    /* Could not setup the result buffer */
}
/* Window close notification function */
In FunctionRead()
{
    Dsadc_ReadStreamResults(ChannelId, &UserBufferPtr);
}

```

Stop the result data acquisition

DSADC driver

The code sequence for stopping the Result data acquisition is follows:

```
#include 'Dsadc.h'

Dsadc_ChannelType ChannelId;
Std_ReturnType lRetVal;

/* DsadcChannel_DsadcChannelConfiguration_x is a valid SW channel ID macro
Generated in Dsadc_Cfg.h */
ChannelId = DsadcChannel_DsadcChannelConfiguration_x;

/* Make sure Channelhas already start result data acquisition by calling
Dsadc_StartModulation API */

/* Disable the result data acquisition */
lRetVal = Dsadc_StopModulation(ChannelId);
if(lRetVal!=E_NOT_OK)
{
    /* Result data Acquisition is stopped */
}
Else
{
    /* Result Acquisition is not stoped */
}
```

Read the timestamp for the last read result event

DSADC driver

The code sequence to read the timestamp value for read result event is as follows:

```

/*
Configuration values mandatory for below code snippet:-
1. DsadcAccessMode = DSADC_SINGLE_READ: Dsadc Channel Access mode is Single
Read
2. DsadcTriggerMode = DSADC_TRIGGER_MODE_NORMAL: Conversion Result Acquisition
will start after invoking the Dsadc_StartModulation API.
3. DsadcTimestampFeature = DSADC_TIMESTAMP_ENABLED: Timestamp is enabled
*/
#include 'Dsadc.h'

Dsadc_ChannelType ChannelId;
Dsadc_ResultType ConversionResult;
Dsadc_TimeStampType Timestamp;
Std_ReturnType lRetVal;

/* DsadcChannel_DsadcChannelConfiguration_x is a valid SW channel ID macro
Generated in Dsadc_Cfg.h */
ChannelId = DsadcChannel_DsadcChannelConfiguration_x;

lRetVal = Dsadc_StartModulation(ChannelId);
if(lRetVal!=E_NOT_OK)
{
    /* Result Acquisition is started. Result data can be read once the conversion
is completed. */
}
Else
{
    /* Result Acquisition is not started */
}

/* Read Single conversion results from the circular buffer */
In FunctionRead()
{
    ChannelStatus = Dsadc_GetStatus(ChannelId);
    If(ChannelStatus == DSADC_RESULT_READY)
    {
        /* read the circular buffer data */
        lRetVal = Dsadc_ReadResult (ChannelId, & ConversionResult);
        if(lRetVal!=E_NOT_OK)
        {
            ConversionResult = Dsadc_GetTimestamp(ChannelId);
        }
    }
    Else
    {
        /* Result is not available */
    }
}

```

DSADC driver

Start the carrier signal

The code sequence to start the carrier signal Generation is follows:

```
#include 'Dsadc.h'

Std_ReturnType lRetVal;

lRetVal = Dsadc_StartCarrierSignal();

if(lRetVal!=E_NOT_OK)
{
    /* Carrier signal Generation is started. */
}
Else
{
    /* Carrier signal Generation is not started. */
}
```

Stop the carrier signal

The code sequence to start the carrier signal generation is as follows:

```
#include 'Dsadc.h'

Std_ReturnType lRetVal;

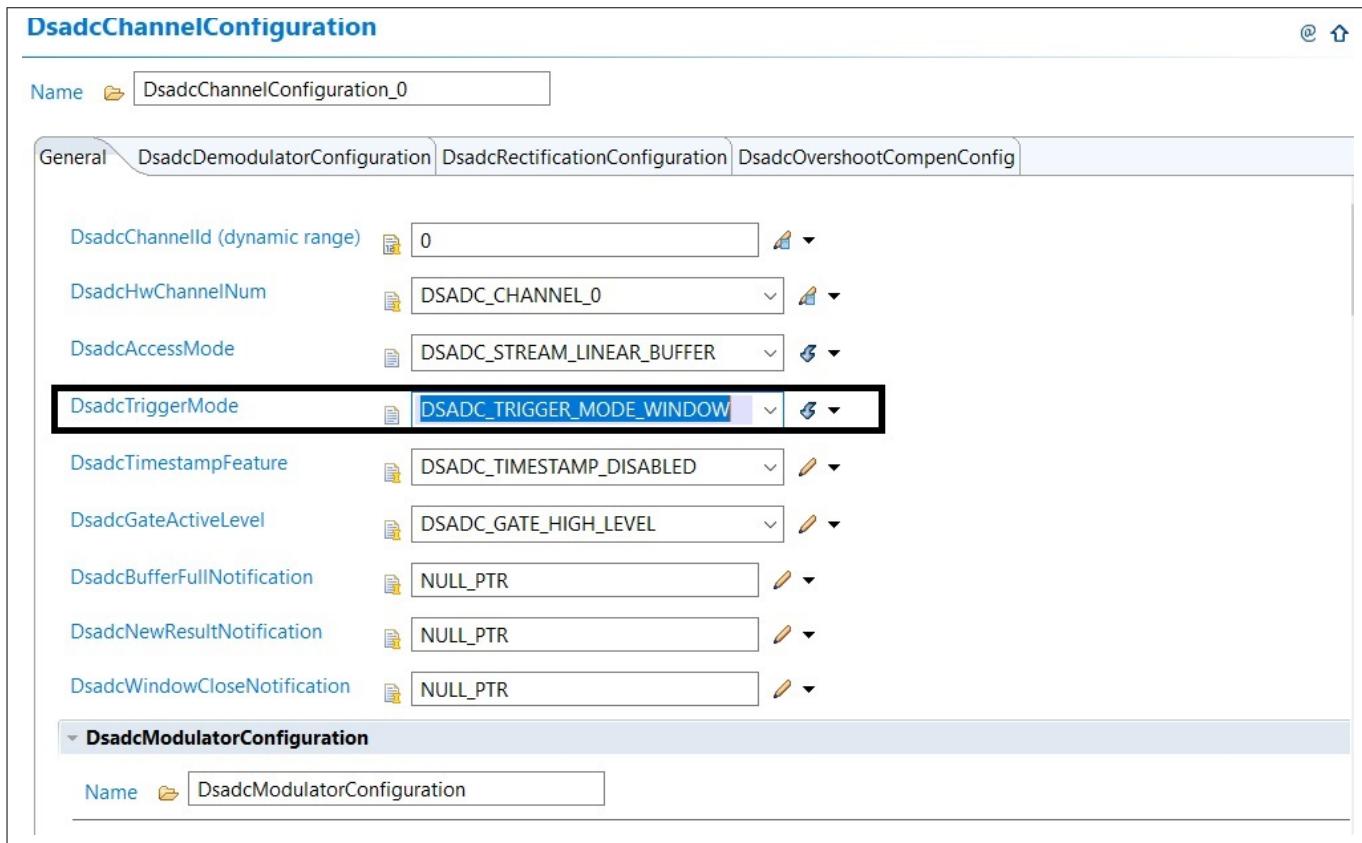
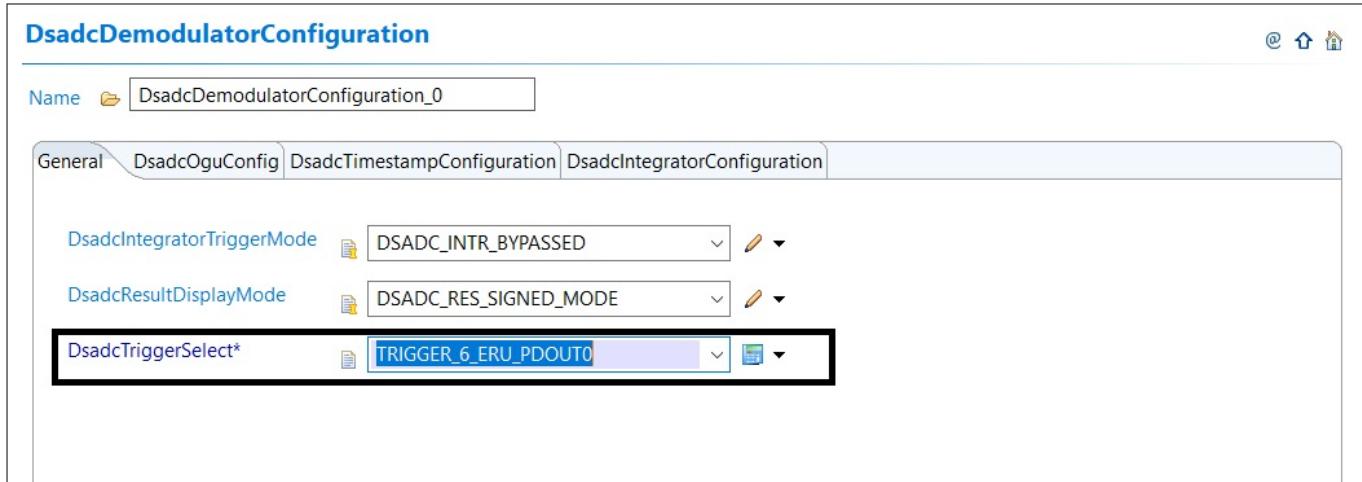
/* Ensure that the Carrier generation is already started using the service
Dsadc_StartCarrierSignal */

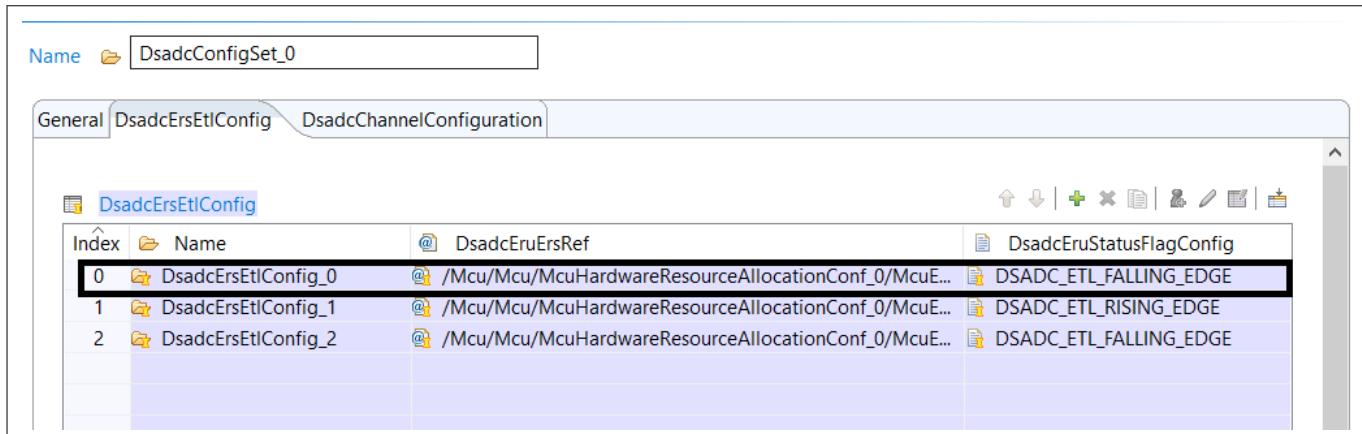
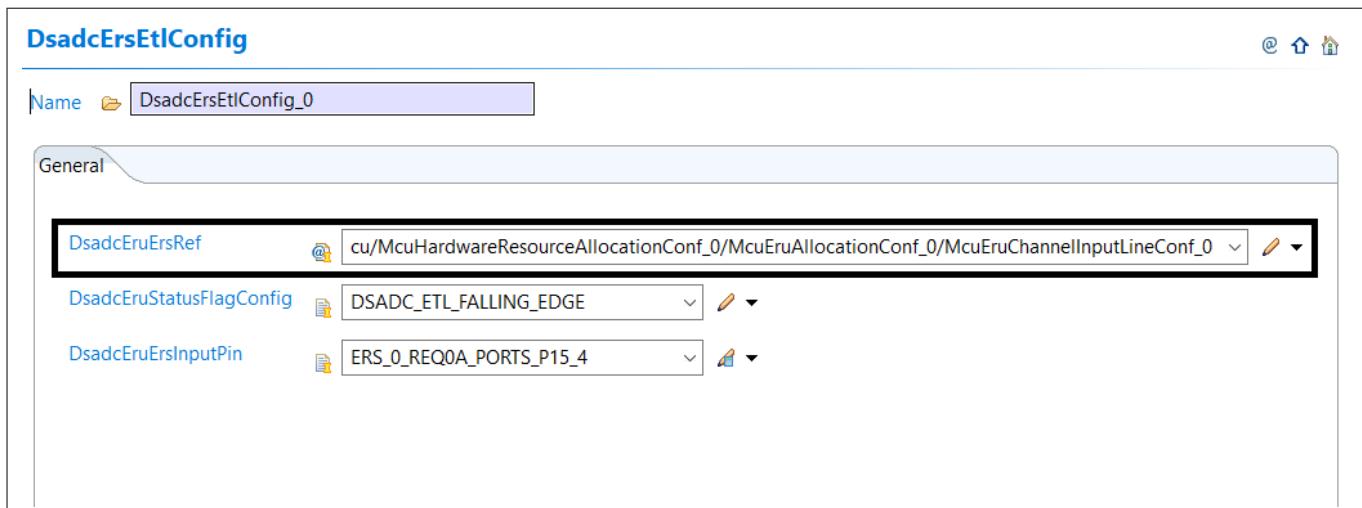
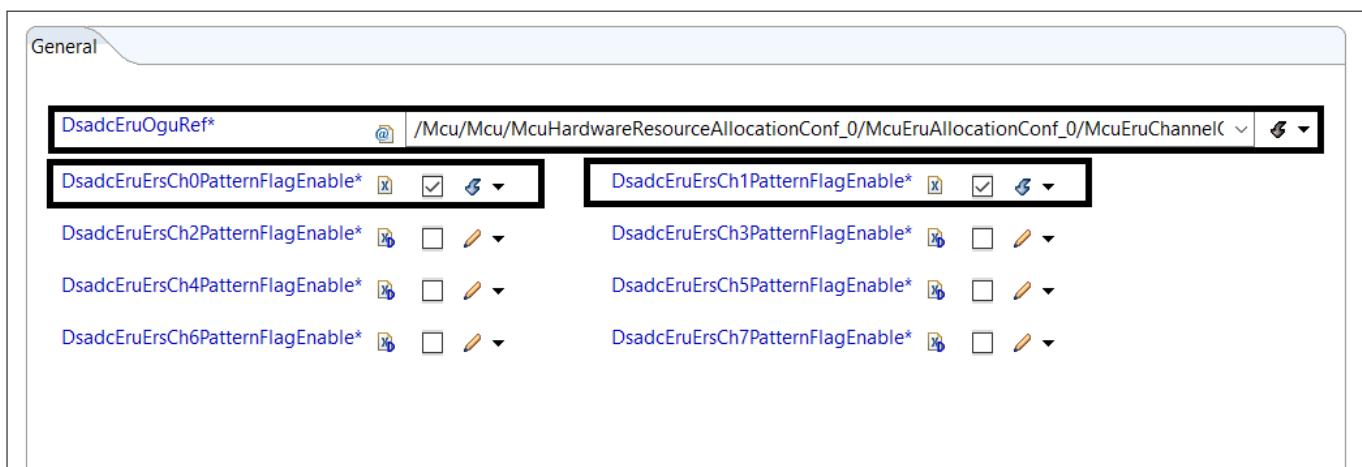
lRetVal = Dsadc_StopCarrierSignal();

if(lRetVal!=E_NOT_OK)
{
    /* Carrier signal Generation is started. */
}
Else
{
    /* Carrier signal Generation is not started. */
}
```

Configuration example for trigger mode window

Below Example shows how to configure the trigger mode window using ERU/GTM

DSADC driver

Figure 13 Configuration: Trigger mode window

Figure 14 Configuration: Trigger source ERU pattern detection output

DSADC driver

Figure 15 Configuration: Add ERS container

Figure 16 Configuration: ERS channel selection

Figure 17 Configuration: Select the OGU channel, select the ERS input channels for pattern detection
Trigger source as GTM

DSADC driver

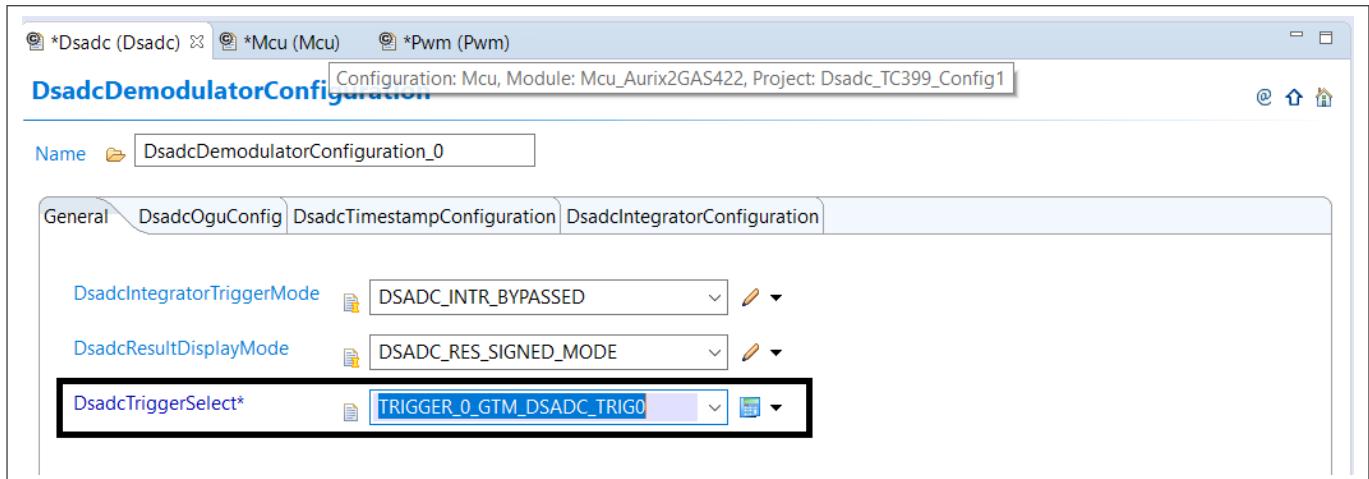


Figure 18 Configuration: Trigger source is GTM DSADC trigger line 0

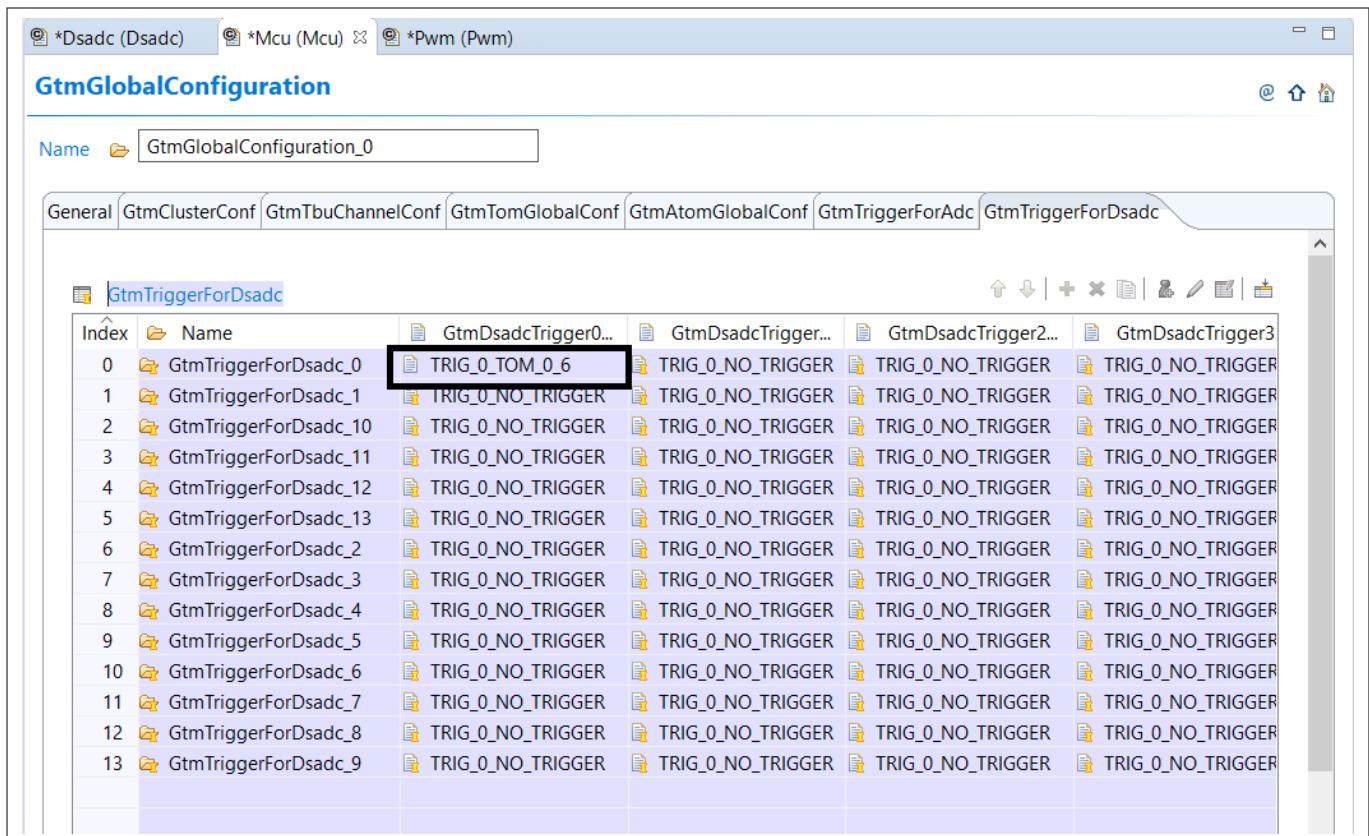


Figure 19 Configuration: Resource selection for GTM DSADC trigger line 0 in MCU driver

DSADC driver

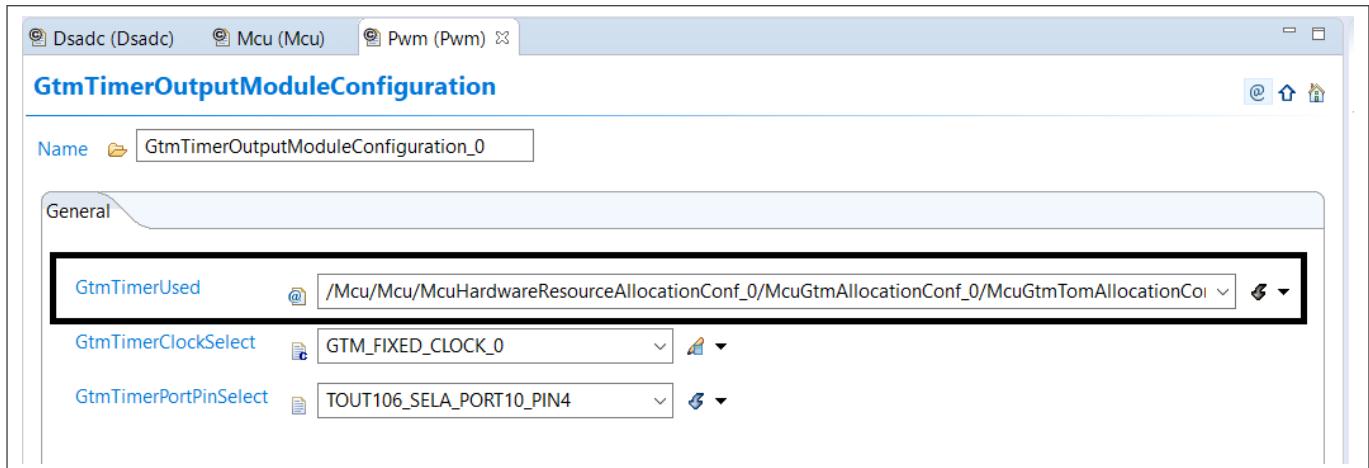


Figure 20 Configuration: GTM resource(TOM/ATOM) configuration in PWM driver

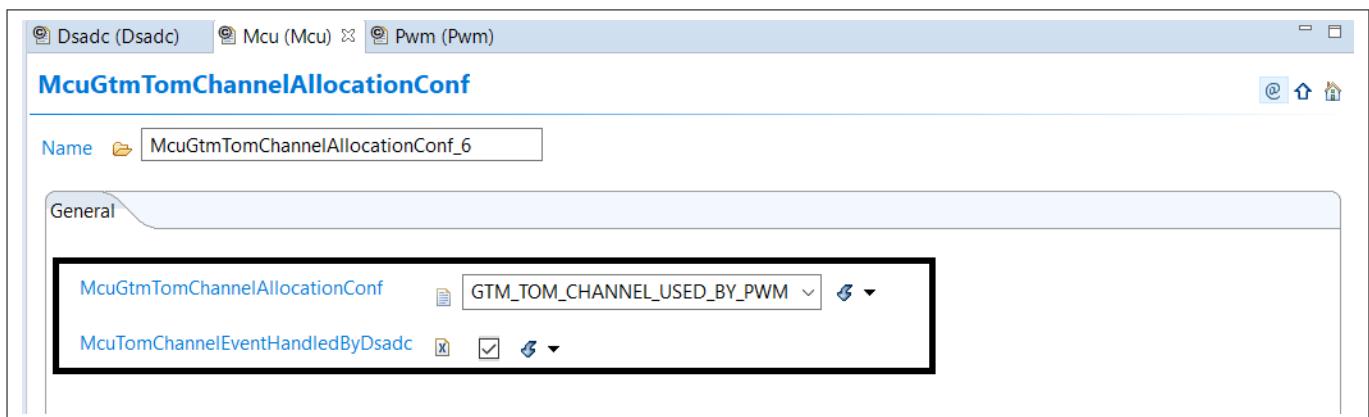


Figure 21 Configuration: GTM reservation in MCU driver.

Trigger mode window for GTM with PWM driver

DSADC driver

The code sequence to initialize and run the trigger mode window functionality for GTM with PWM driver is follows:

```

/*
Configuration values mandatory for below code snippet:-
1. DsadcAccessMode = DSADC_STREAM_LINEAR_BUFFER/DSADC_SINGLE_READ: Dsadc
Channel Access mode is Linear buffer/single read*
2. DsadcTriggerMode = DSADC_TRIGGER_MODE_WINDOW: Conversion Result Acquisition
will start after the Window Open Event
*/

#include 'Dsadc.h'
#include 'Pwm_17_GtmCcu6.h'
#include 'Irq.h'

Dsadc_ChannelType ChannelId;
Dsadc_ResultType DataBufferPtr[25]; // Assuming buffer size of 25 is
sufficient for the Dsadc Channel
Std_ReturnType lRetVal;

/* DsadcChannel_DsadcChannelConfiguration_x is a valid SW channel ID macro
Generated in Dsadc_Cfg.h */
ChannelId = DsadcChannel_DsadcChannelConfiguration_x;

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

/*DSADOC Initialization*/
Dsadc_Init(&Dsadc_Config);

/*PWM Initialization*/
Pwm_17_GtmCcu6_Init(&Pwm_17_GtmCcu6_Config)

/*Enable Interrupt*/
IrqDsadc_Init();
IrqGtm_Init();
MODULE_SRC.DSADC.DSADC[x].SRM.B.SRE = 1;
MODULE_SRC.GTM.GTMTOMxx.B.SRE = 1;

/* make sure that window is not started */
Pwm_17_GtmCcu6_SetOutputToIdle(ChannelNumber);

lRetVal = Dsadc_SetupResultBuffer(ChannelId, &DataBufferPtr[0],25);
if(lRetVal!=E_NOT_OK)
{
    lRetVal = Dsadc_StartModulation(ChannelId);
    if(lRetVal!=E_NOT_OK)
    {
        /* To enable the GTM interrupt since window open and close event is required

```

DSADC driver

```

in DDSADC driver */
Pwm_17_GtmCcu6_EnableNotification(ChannelNumber);
/* To start generating the PWM signal from the GTM */
Pwm_17_GtmCcu6_SetDutyCycle(ChannelNumber,xxxx)
/* Result Acquisition will start after the window Open event*/
}
Else
{
/* Result Acquisition is not started */
}
}
else
{
/* Could not setup the result buffer */
}

/* Sequence when stop modulation is required*/
/* To stop the gate signal generation */
Pwm_17_GtmCcu6_SetOutputToIdle(ChannelNumber);
Dsadc_StopModulation(Dsadc_ChannelNumber)

```

2.1.5 Key architectural considerations

2.1.5.1 Mode of operation: result acquisition

The DSADC driver supports two modes of operation related to result data acquisition. The modes can be selectable for each channel by assigning appropriate value to the DsadcTriggerMode parameter. These modes are as follows:

- **Result acquisition using gate signal**

In this mode of operation, once the `Dsadc_StartModulation()` API is called the conversion results are acquired after the window open event. When the `Dsadc_StopModulation()` API is called or in case of a window close event the results acquisition shall be stopped.

The DsadcGateActiveLevel configuration parameter will define the gate signal level where it needs to acquire the result data. If this parameter is configured as DSADC_GATE_HIGH_LEVEL then the gate signal rising edge will be considered as window open and falling edge will be considered as window close. If this parameter is configured as DSADC_GATE_LOW_LEVEL then the gate signal falling edge will be considered as window open and rising edge will be considered as window close.

If the GTM TOM/ATOM is used as a trigger source, then user has to configure and control the TOM/ATOM channel using the PWM driver. In the DSADC driver, TOM/ATOM CCU0 event is considered as window open event and CCU1 event is considered as a window close event. User has to configure the signal level (SL) bit of the TOM/ATOM accordingly.

When the ERU pattern detection is used as a trigger source, If the DsadcGateActiveLevel configuration parameter is configured as DSADC_GATE_HIGH_LEVEL then the pattern match event will be considered as window open and pattern miss event will be considered as window close event. If the DsadcGateActiveLevel configuration parameter is configured as DSADC_GATE_LOW_LEVEL then the pattern miss event will be considered as window open and pattern match event will be considered as window close event.

Configuration settings

DsadcTriggerMode: DSADC_TRIGGER_MODE_WINDOW

DSADC driver

DsadcTriggerSelect: Can be GTM or ERU Trigger

DsadcGateActiveLevel: DSADC_GATE_HIGH_LEVEL/DSADC_GATE_LOW_LEVEL

- **Result acquisition without using gate signal**

In this mode of operation, The conversion results are acquired after invoking the Dsadc_StartModulation API. The result acquisition is stopped after invoking the Dsadc_StopModulation API.

Configuration settings

DsadcTriggerMode: DSADC_TRIGGER_MODE_NORMAL

DsadcTriggerSelect: TRIGGER_0_NO_DSADC_TRIG

2.1.5.2 Mode of operation: result handling

The DSADC driver supports two modes of operation related to result handling. The modes can be selectable for each channel by assigning appropriate value to DsadcAccessMode configuration parameter. These modes are 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.

Configuration Settings

DsadcAccessMode: DSADC_SINGLE_READ or DSADC_CIRCULAR_BUFFER or DSADC_STREAM_LINEAR_BUFFER

- **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. Each channel may have a flexibility to select DMA or interrupt mode.

For more information, refer to the DMA support section.

Configuration settings

DsadcAccessMode: DSADC_DMA_ACCESS

2.1.5.3 Accessing shared SFR

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

- **Accessing ERU registers:** The DSADC driver updates MODULE_SCU.EICR and MODULE_SCU.IGCR to configure the trigger signal. 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 DSADC 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.

2.1.5.4 Settling time after the filter chain restart

The filter chain is restarted once the calibration is done. So user has to wait for the settling time before calling Dsadc_StartModulation () API. Settling time is mathematically defined by the step response of the related filter chain configuration. The step response has to be considered only for analog sensor signals which have a

DSADC driver

DC component. For sinusoidal like signals, only the group delay has to be considered. The step response for the four different possible filter chain configurations are defined by following characteristics:

- When the CIC filter is enabled then the step response is the time taken to generate 4 output samples.
- when the CIC filter and FIR0 filters are enabled then the step response is the time taken to generate 5 output samples.
- when the CIC filter, FIR0 filter and FIR1 filter are enabled then the step response is the time taken to generate 15 output samples.
- When the FIR1 filter is configured with a decimation rate of 1 then the step response is the time taken to generate 30 output samples.

Every of the above described filter chain configurations can be extended by the integrator stage when it is enabled. The step response of the integrator is related to the configured number of accumulation steps.

2.1.5.5 Timestamp

DSADC driver provides the timestamp information in two cases.

- **Timestamp for trigger mode window**

DSADC driver provides timestamp for the window open event. In this case timestamp is the time between the last conversion result and the window open event.

Configuration settings

DsadcTriggerMode: DSADC_TRIGGER_MODE_WINDOW

DsadcTimestampFeature: DSADC_TIMESTAMP_ENABLED

Note: By using the conversion result before the window open event, timestamp and the conversion result after the window open event, user can interpolate the conversion result for the window open event.

Timestamp with conversion results for different access mode is shown as follows:

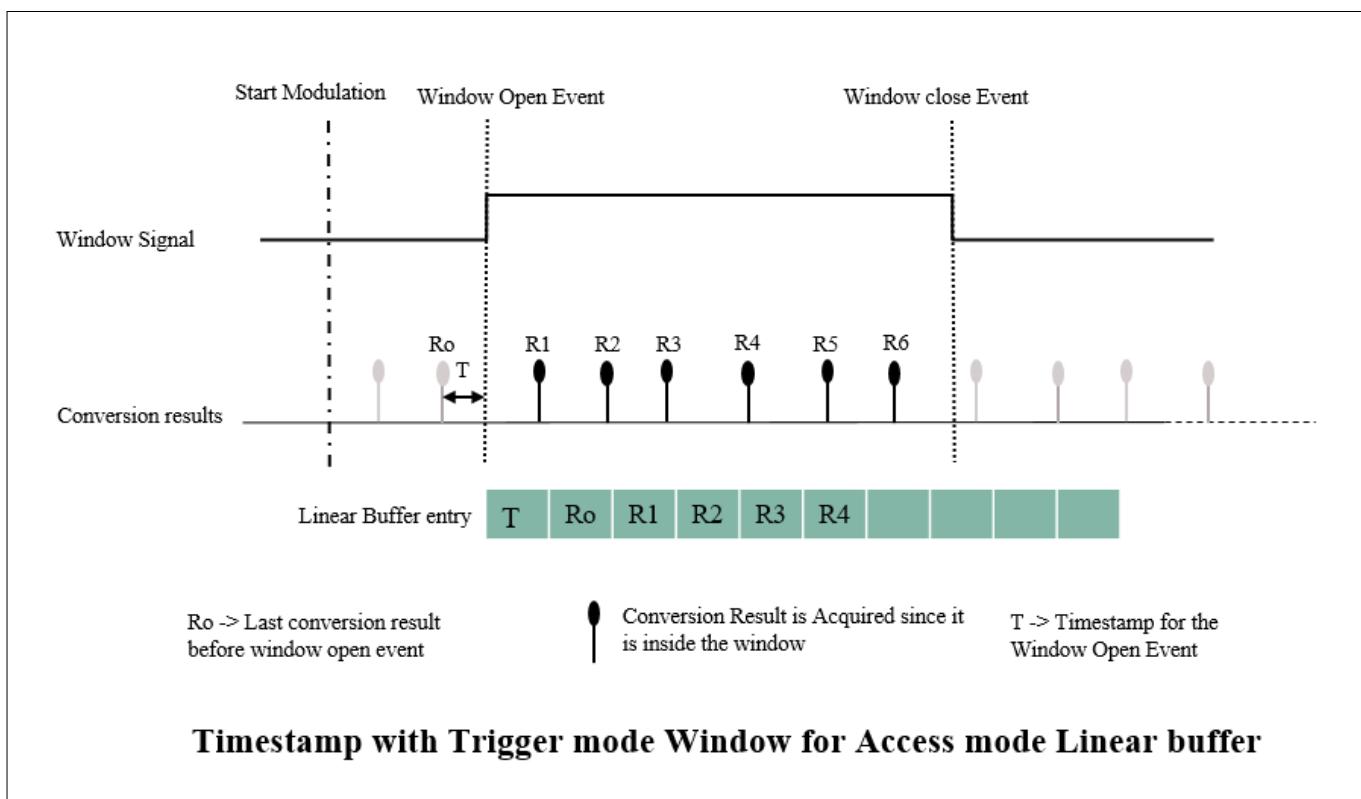
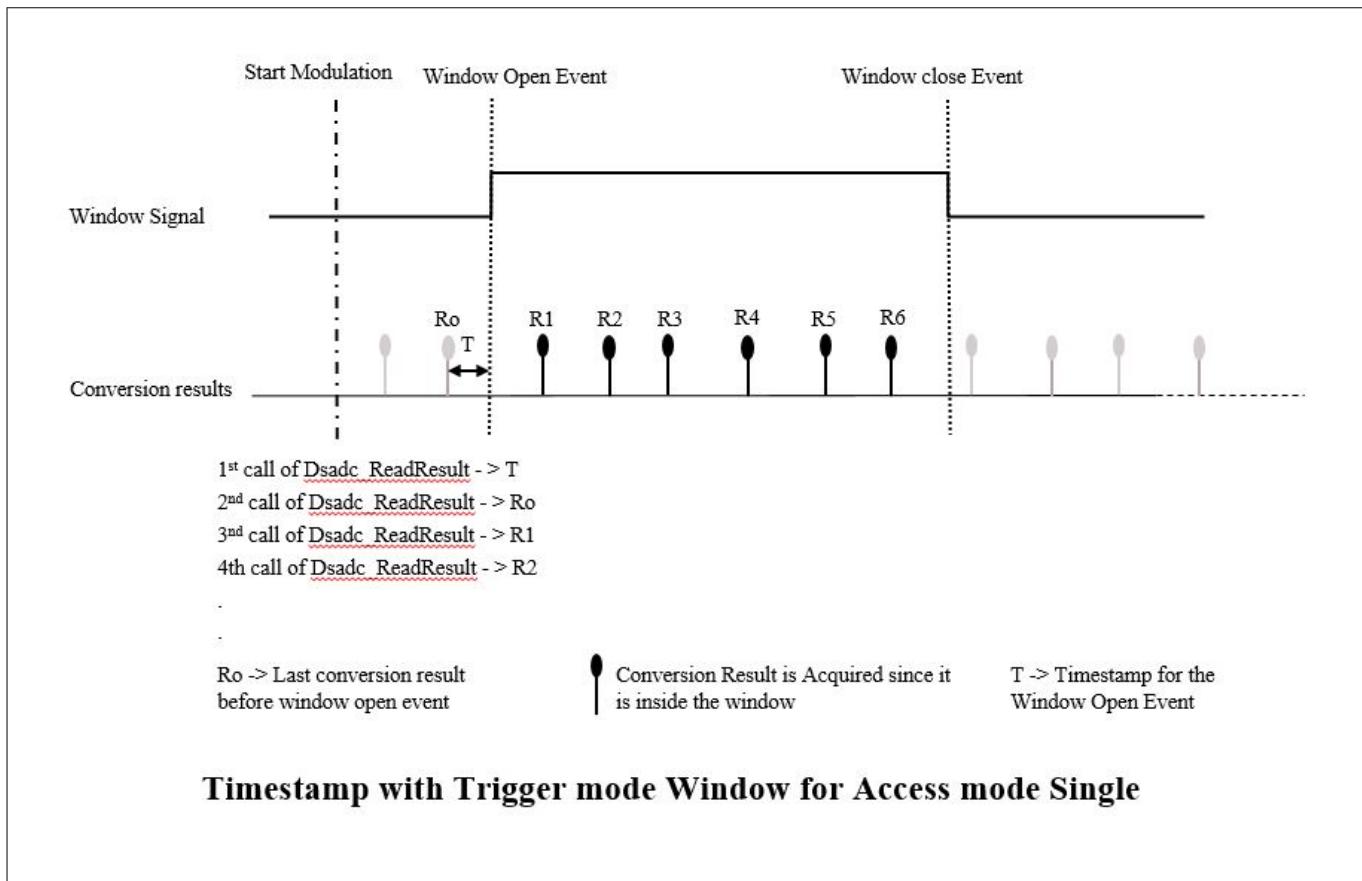


Figure 22

Timestamp with Trigger mode Window for Access mode Linear buffer

DSADC driver

Figure 23 Timestamp with Trigger mode Window for Access mode Single

- **Timestamp for single access mode with trigger mode normal**

DSADC driver provides the timestamp for the `Dsadc_ReadResult()` API event. In this case timestamp is the time between the conversion result and the `Dsadc_ReadResult()` API reads the result from the hardware result register.

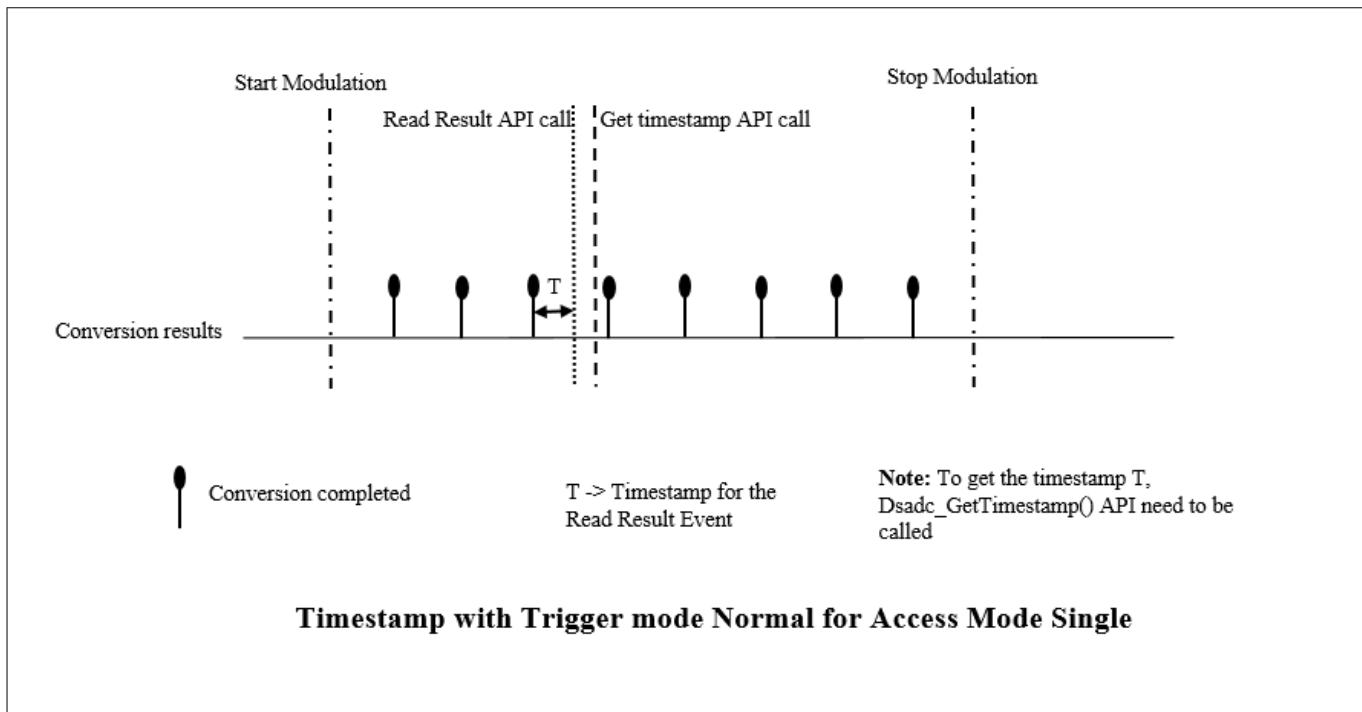
Configuration settings

`DsadcTriggerMode: DSADC_TRIGGER_MODE_NORMAL`

`DsadcTimestampFeature: DSADC_TIMESTAMP_ENABLED`

`DsadcAccessMode: DSADC_SINGLE_READ`

Note: By using this timestamp user can interpolate the exact conversion result when the `Dsadc_ReadResult()` API read the result from the result register.

DSADC driver**Figure 24****Timestamp with Trigger mode Normal for Access Mode Single**

DSADC driver

2.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Call sequence for proper window event handling**

User shall follow the following call sequences when the IFX MCAL PWM driver is used to generate the gate signal.

- Pwm_17_GtmCcu6_SetOutputToldle API shall be called before invoking Dsadc_StartModulation API. -

Pwm_17_GtmCcu6_SetDutyCycle/Pwm_17_GtmCcu6_SetPeriodAndDuty API shall be called after invoking the

Dsadc_StartModulation API. - Pwm_17_GtmCcu6_EnableNotification API shall be called before invoking the

Pwm_17_GtmCcu6_SetDutyCycle/Pwm_17_GtmCcu6_SetPeriodAndDuty API. -

Pwm_17_GtmCcu6_EnableNotification API shall be called after invoking the

Pwm_17_GtmCcu6_SetOutputToldle API. - Pwm_17_GtmCcu6_SetOutputToldle API shall be called before

invoking the Dsadc_StopModulation API.

[cover parentID DSADC={62B5AE66-5D29-4030-AF77-9E5B0F804888}]

- **Wrong duty cycle passed in PWM driver to generate gate signal**

User shall be aware that the duty cycle can not be set to 0 or 100 percentage after the Dsadc_StartModulation API and before the Dsadc_StopModulation API when the IFX MCAL PWM driver is used to generate the gate signal.

[cover parentID DSADC={DD8039D7-C386-4261-8056-0A3A03F2EF23}]

- **Mcu_Init shall be called before Dsadc_Init**

User shall ensure that the MCU driver is initialized before invoking the Dsadc_Init API.

[cover parentID DSADC={526D4F74-EFD6-4c71-B6D2-A71E39407C4A}]

- **Correct configuration pointer for initialization of the DSADC driver**

User shall ensure that the correct configuration pointer is passed for initializing the DSADC driver.

[cover parentID DSADC={5D8325C1-B202-472d-8D8D-429D520FF70B}]

- **User shall pass a pointer different from channel buffer**

User shall ensure that while invoking the Dsadc_ReadStreamResults() API, the passed result buffer pointer shall be different from the channel buffer pointer, which was used during setting up the result buffer.

[cover parentID DSADC={379669C4-790C-4652-8988-4AF381AB63EC}]

- **Stop modulation call sequence**

User shall ensure that the results are read(using Dsadc_ReadResult/ Dsadc_ReadStreamResults) before the Dsadc_StopModulation function is invoked, else the results will be lost.

[cover parentID DSADC={6BD82066-F2AB-4c00-8715-CC6813647BE2}]

- **Setup result buffer call sequence**

User shall ensure that the Dsadc_SetupResultBuffer API is called before invoking the Dsadc_StartModulation API.

[cover parentID DSADC={2456266F-E919-49bf-B217-D90EBD47E7B7}]

- **Past event not notified**

User shall ensure that the Dsadc_EnableNotifications API is invoked before the window close event. Else the notification will be missed and will be provided only for the future window close event.

[cover parentID DSADC={4DB24530-3475-4275-8D1E-57B552D59920}]

- **DMA usage**

User shall use the DMA mode for high output rate (output rate is greater than the ISR execution frequency).

Note: ISR execution frequency = 1/WCET of ISR execution time.

[cover parentID DSADC={AE9B2A95-7D71-40e3-A1DD-D51F2F94FA42}]

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

DSADC driver

User shall verify that the macros generated with the symbolic name of channels in the Dsadc_Cfg.h file contain correct values and should use these macros while invoking APIs of the DSADC driver.

[cover parentID DSADC={E34FB178-2046-4013-9F27-FC90430E1C68}]

- **Valid result buffer parameters are passed**

User shall ensure that the passed pointer parameter and the buffer size contain the valid values.

[cover parentID DSADC={AC081E71-E51A-4bcf-840F-15187E0E948E}]

- **Trigger signal priority higher than DSADC ISR**

User shall ensure that the priority of trigger signal ISR is higher than the DSADC conversion completed ISR.

[cover parentID DSADC={AC15A256-9820-4383-9C27-04AE9A3E68A7}]

- **Start modulation call sequence**

User shall ensure that after invoking the Dsadc_Init API, the Dsadc_StartCalibration API is invoked before calling the Dsadc_StartModulation API.

[cover parentID DSADC={5566A5CB-793D-4edc-A6C3-1B82EE0E020E}]

- **Settling time after calibration**

User shall ensure that there is a delay of settling time between the completion of the Dsadc_StartCalibration API and invocation of the Dsadc_StartModulation API.

[cover parentID DSADC={9BF1B1AD-2960-40e3-91C8-3A96C33C65F8}]

- **Result status check for DMA**

Result ready status is not applicable for the DMA mode and the user should check for the availability of data using the DMA transfer count value.

[cover parentID DSADC={E2E282BE-DD91-4eea-BA7F-79A3F07551D0}]

- **DMA channel initialization sequence**

User shall ensure that the DMA channel is set up before calling the Dsadc_StartModulation API and also DMA channel is stopped/de-initialized after calling the Dsadc_StopModulation API.

[cover parentID DSADC={3F7B0823-89D9-4280-B15D-701DC528CBA6}]

- **Loss of last two samples**

User shall be aware that when the hardware trigger is configured and the timestamp feature is enabled, the last two samples before the window close event are lost.

[cover parentID DSADC={9A59E04C-F733-488b-BB14-73A3D1F00ADB}]

- **Trigger signal start after start modulation call**

User shall ensure that the trigger signal is started after the Dsadc_StartModulation API execution is completed.

[cover parentID DSADC={B161CA40-73AA-4a80-8807-4DB5EEDB39F6}]

- **Correct configuration for calibration parameters**

User shall ensure that the calibration-related configuration parameters are configured correctly for the successful calibration.

[cover parentID DSADC={0B6282A3-B21F-495d-B197-CF1634AEB982}]

DSADC driver

2.3 Reference information

2.3.1 Configuration interfaces

DSADC driver

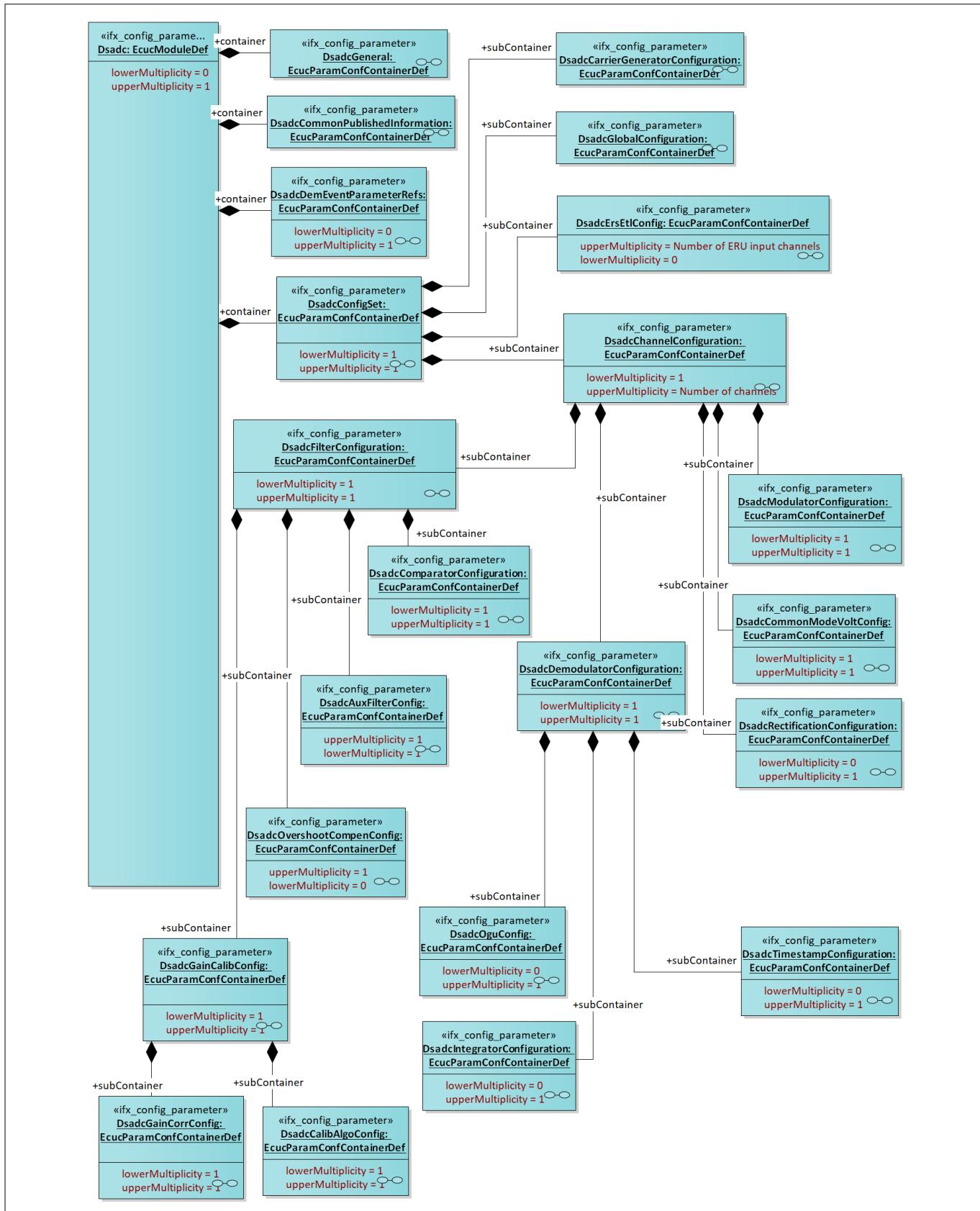


Figure 25

Container hierarchy along with their configuration parameters

DSADC driver**2.3.1.1 Container: Dsadc**

Configuration of the DSADC (Delta Sigma Analog Digital Conversion) module

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: -

2.3.1.2 Container: DsadcAuxFilterConfig

This configuration container provides parameters related to DSADC auxiliary filter configuration.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.2.1 DsadcAuxCicFilterEnable**Table 106 Specification for DsadcAuxCicFilterEnable**

Name	DsadcAuxCicFilterEnable		
Description	This parameter defines the availability of DSADC auxiliary filter. 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	-		

2.3.1.2.2 DsadcAuxFilterCicDecimationFactor**Table 107 Specification for DsadcAuxFilterCicDecimationFactor**

Name	DsadcAuxFilterCicDecimationFactor		
Description	This parameter defines the over sampling rate/decimation factor for the Auxiliary CIC filter. This parameter is set the default value when the parameter DsadcAuxCicFilterEnable is configured as false. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

DSADC driver**Table 107 Specification for DsadcAuxFilterCicDecimationFactor (continued)**

Range	DSADC_AUXCIC_OSR_16: DSADC auxiliary CIC filter over sampling rate is 16 DSADC_AUXCIC_OSR_32: DSADC auxiliary CIC filter over sampling rate is 32		
Default value	DSADC_AUXCIC_OSR_16		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcAuxCicFilterEnable		

2.3.1.3 Container: DsadcCalibAlgoConfig

This container provides configuration parameters for Gain calibration.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.3.1 DsadcCICDecimationRate**Table 108 Specification for DsadcCICDecimationRate**

Name	DsadcCICDecimationRate		
Description	This parameter defines Decimation factor for the CIC filter. This parameter value will be considered only during the execution of the calibration algorithm. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_CIC_DECIMATION_RATE_128: Decimation factor for CIC is 128 during calibration time. DSADC_CIC_DECIMATION_RATE_16: Decimation factor for CIC is 16 during calibration time. DSADC_CIC_DECIMATION_RATE_256: Decimation factor for CIC is 256 during calibration time. DSADC_CIC_DECIMATION_RATE_32: Decimation factor for CIC is 32 during calibration time. DSADC_CIC_DECIMATION_RATE_512: Decimation factor for CIC is 512 during calibration time. DSADC_CIC_DECIMATION_RATE_64: Decimation factor for CIC is 64 during calibration time. DSADC_CIC_DECIMATION_RATE_8: Decimation factor for CIC is 8 during calibration time.		
Default value	DSADC_CIC_DECIMATION_RATE_8		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DSADC driver**Table 108 Specification for DsadcCICDecimationRate (continued)**

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

2.3.1.3.2 DsadcCalibAlgoTargetValue**Table 109 Specification for DsadcCalibAlgoTargetValue**

Name	DsadcCalibAlgoTargetValue		
Description	<p>This parameter defines the full scale target value for calibration algorithm.</p> <p>Maximum target value allowed for this parameter depends on DsadcInputGain. When DsadcInputGain is selected as 2 or 4 then DsadcCalibAlgoTargetValue must not go beyond 22757.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 32767		
Default value	25000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcInputGain		

2.3.1.3.3 DsadcCalibCICFilterOutputShiftPos**Table 110 Specification for DsadcCalibCICFilterOutputShiftPos**

Name	DsadcCalibCICFilterOutputShiftPos		
Description	<p>This parameter defines the position of the CIC output shifter, which is to be used to select the valid output bits from the CIC filter. This is valid only during execution of calibration algorithm.</p> <p>Default value for this parameter is BITS_0_TO_16. The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef

DSADC driver**Table 110 Specification for DsadcCalibCICFilterOutputShiftPos (continued)**

Range	BITS_i_TO_j: Valid output bit selection from the CIC filter output. Possible values of i and j are i : 0 to 28 j : 16 to 44		
Default value	BITS_0_TO_16		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.3.4 DsadcCalibGainCorrMulFactor**Table 111 Specification for DsadcCalibGainCorrMulFactor**

Name	DsadcCalibGainCorrMulFactor		
Description	This parameter defines multiplication factor for Gain correction. This is valid only during execution of the calibration algorithm. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0000 - 1.9999		
Default value	1.0000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.3.5 DsadcGainCalibMulFactor**Table 112 Specification for DsadcGainCalibMulFactor**

Name	DsadcGainCalibMulFactor
Description	This parameter defines multiplication factor for Gain calibration. This is valid only during the execution of calibration algorithm. The default value of this parameter is set to the reset value of the corresponding SFR.

DSADC driver
Table 112 Specification for DsadcGainCalibMulFactor (continued)

Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0000 - 1.9999		
Default value	1.0000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.4 Container: DsadcCarrierGeneratorConfiguration

This container contains the Carrier Generation related Parameters.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.4.1 DsadcCarrierFrequencyClockDiv

Table 113 Specification for DsadcCarrierFrequencyClockDiv

Name	DsadcCarrierFrequencyClockDiv		
Description	This parameter defines the divider factor, which is used to define the frequency of the carrier signal generator, which is derived from the selected internal clock source and divider factor. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

DSADC driver**Table 113 Specification for DsadcCarrierFrequencyClockDiv (continued)**

Range	DSADC(CG)_CLOCKDIVIDER_DIV10: Input clock is divided by 10 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV12: Input clock is divided by 12 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV14: Input clock is divided by 14 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV16: Input clock is divided by 16 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV18: Input clock is divided by 18 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV20: Input clock is divided by 20 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV22: Input clock is divided by 22 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV24: Input clock is divided by 24 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV26: Input clock is divided by 26 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV28: Input clock is divided by 28 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV2: Input clock is divided by 2 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV30: Input clock is divided by 30 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV32: Input clock is divided by 32 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV4: Input clock is divided by 4 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV6: Input clock is divided by 6 for carrier generator DSADC(CG)_CLOCKDIVIDER_DIV8: Input clock is divided by 8 for carrier generator		
Default value	DSADC(CG)_CLOCKDIVIDER_DIV2		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.4.2 DsadcCarrierSignalPolarity**Table 114 Specification for DsadcCarrierSignalPolarity**

Name	DsadcCarrierSignalPolarity		
Description	This parameter defines the starting polarity of the carrier signal. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_CARR_SIG_INVERTED: Carrier signal begins with -1 i.e. LOW DSADC_CARR_SIG_NORMAL: Carrier signal begins with +1 i.e. HIGH		
Default value	DSADC_CARR_SIG_NORMAL		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DSADC driver**Table 114 Specification for DsadcCarrierSignalPolarity (continued)**

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

2.3.1.4.3 DsadcCarrierSignalType**Table 115 Specification for DsadcCarrierSignalType**

Name	DsadcCarrierSignalType		
Description	This parameter determines the carrier signal type to be generated. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_CARR_SIG_SINEWAVE: Carrier Generator generates the Sinewave. DSADC_CARR_SIG_SQUAREWAVE: Carrier Generator generates the Square wave. DSADC_CARR_SIG_STOPPED: Carrier signals are stopped. DSADC_CARR_SIG_TRIANGLE: Carrier Generator generates the Triangle wave		
Default value	DSADC_CARR_SIG_STOPPED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.4.4 DsadcPwmGenerationMode**Table 116 Specification for DsadcPwmGenerationMode**

Name	DsadcPwmGenerationMode		
Description	This parameter defines the mode in which the Carrier Generator signal is generated. In case of Bit Reverse generation mode, it increases the frequency spectrum to yield a smoother induced sine signal. This is done by distributing the 0 and 1 bits over the 32 cycles of a PWM period. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

DSADC driver
Table 116 Specification for DsadcPwmGenerationMode (continued)

Range	DSADC_BIT_REVERSE_MODE: Carrier generated in bit reverse mode DSADC_NORMAL_MODE: Normal Carrier generation		
Default value	DSADC_NORMAL_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.5 Container: DsadcChannelConfiguration

This container contains the channel configuration (parameters) depending on the hardware capability.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

2.3.1.5.1 DsadcAccessMode

Table 117 Specification for DsadcAccessMode

Name	DsadcAccessMode		
Description	<p>This parameter determines the Result Access Mode selection for a DSADC channel.</p> <p>The available access mode depends on the parameter DsadcTriggerMode.</p> <p>If DsadcTriggerMode is Configured as TRIGGER_MODE_WINDOW then DsadcAccessMode can not be set as DSADC_CIRCULAR_BUFFER.</p> <p>The default value of this parameter is set to DSADC_SINGLE_READ to minimize the execution time of ISR</p>		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	DSADC_CIRCULAR_BUFFER: Configure buffer as circular buffer (i.e. the DSADC Driver wraps around if the end of the stream buffer is reached) DSADC_DMA_ACCESS: Result data has to be transferred via DMA DSADC_SINGLE_READ: Hardware result register value is read and returned to the user without buffering. DSADC_STREAM_LINEAR_BUFFER: Configure buffer as linear buffer (i.e. Once the Buffer is full, the subsequent results are discarded.).		
Default value	DSADC_SINGLE_READ		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DSADC driver**Table 117 Specification for DsadcAccessMode (continued)**

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

2.3.1.5.2 DsadcBufferFullNotification**Table 118 Specification for DsadcBufferFullNotification**

Name	DsadcBufferFullNotification		
Description	<p>Callback function for buffer full event.</p> <p>This parameter is set to the default value when the parameter DsadcAccessMode is not configured as DSADC_STREAM_LINEAR_BUFFER</p> <p>Dsadc_NotifyFnPtrType is the data type for this callback function.</p> <p>By default, the notification parameter will be NULL_PTR , to remove the dependency from the user defined functions.</p> <p>The DSADC driver does not validate the configured function name or address for correctness and hence the responsibility falls on the user.</p>		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcAccessMode		

2.3.1.5.3 DsadcChannelId**Table 119 Specification for DsadcChannelId**

Name	DsadcChannelId		
Description	<p>Unique number to identify the channel.</p> <p>The Default value for this parameter is the index value for the container.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - (Number of channels - 1) where Number of channels depends on derivative		

DSADC driver**Table 119 Specification for DsadcChannelId (continued)**

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

2.3.1.5.4 DsadcGateActiveLevel**Table 120 Specification for DsadcGateActiveLevel**

Name	DsadcGateActiveLevel		
Description	<p>This parameter is used to define the active level for the gate signal.</p> <p>If this Parameter is configured as DSADC_GATE_HIGH_LEVEL then high level of the gate signal is considered as active phase.</p> <p>If this Parameter Configured as DSADC_GATE_LOW_LEVEL then low level of the gate signal is considered as the active phase.</p> <p>This Parameter is configurable only if the Parameter DsadcTriggerMode is configured as DSADC_TRIGGER_MODE_WINDOW.</p> <p>The default value of this parameter is set to DSADC_GATE_HIGH_LEVEL since many applications want to acquire result when the signal is High</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>DSADC_GATE_HIGH_LEVEL: The Gate is active when the signal level is High</p> <p>DSADC_GATE_LOW_LEVEL: The Gate is active when the signal level is Low</p>		
Default value	DSADC_GATE_HIGH_LEVEL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcTriggerMode		

2.3.1.5.5 DsadcHwChannelNum**Table 121 Specification for DsadcHwChannelNum**

Name	DsadcHwChannelNum
-------------	-------------------

DSADC driver**Table 121 Specification for DsadcHwChannelNum (continued)**

Description	Hardware EDSADC channel number. The default value for this parameter is set to DSADC_CHANNEL_0, since it is the first physical channel number.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_CHANNEL_x: where x stands for the channel number.		
Default value	DSADC_CHANNEL_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.5.6 DsadcNewResultNotification**Table 122 Specification for DsadcNewResultNotification**

Name	DsadcNewResultNotification		
Description	Callback function for new result event. This parameter is set to the default value when the parameter DsadcAccessMode is configured as DSADC_DMA_ACCESS. Dsadc_NotifyFnPtrType is the data type for this callback function. By default, the notification parameter will be NULL_PTR , to remove the dependency from the user defined functions. The DSADC driver does not validate the configured function name or address for correctness and hence the responsibility falls on the user.		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcAccessMode		

DSADC driver

2.3.1.5.7 DsadcTimestampFeature

Table 123 Specification for DsadcTimestampFeature

Name	DsadcTimestampFeature		
Description	<p>This parameter is used to define the timestamp function availability for the DSADC Channel. The Timestamp Functionality varies depending upon the access mode.</p> <p>If the parameter DsadcTriggerMode is configured as DSADC_TRIGGER_MODE_WINDOW then the timestamp is the, Timestamp count from the last HW result event till the gate signal open event.</p> <p>If the Parameter DsadcAccessMode is configured as DSADC_SINGLE_READ and the DsadcTriggerMode is configured as DSADC_TRIGGER_MODE_NORMAL then the timestamp is the, Timestamp count from the result event till the Dsadc_ReadResult API reads the result value from the Hw result register.</p> <p>This parameter is configured as DSADC_TIMESTAMP_DISABLED by default and it can not be configure as DSADC_TIMESTAMP_ENABLED, if DsadcTriggerMode is configured as DSADC_TRIGGER_MODE_NORMAL and the parameter DsadcAccessMode is not configured as DSADC_SINGLE_READ.</p> <p>The default value for this parameter is set to DSADC_TIMESTAMP_DISABLED to avoid the configuration dependency error for the default configuration.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_TIMESTAMP_DISABLED: Disable the timestamp functionality DSADC_TIMESTAMP_ENABLED: Enable the timestamp functionality		
Default value	DSADC_TIMESTAMP_DISABLED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcTriggerMode , DsadcAccessMode		

2.3.1.5.8 DsadcTriggerMode

Table 124 Specification for DsadcTriggerMode

Name	DsadcTriggerMode		
Description	<p>This parameter is used to define the trigger mode. Possible Result data acquisition modes are DSADC_TRIGGER_MODE_NORMAL and DSADC_TRIGGER_MODE_WINDOW.</p> <p>The default value of this parameter is set to DSADC_TRIGGER_MODE_NORMAL to avoid the configuration dependency error for the default configuration.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef

DSADC driver**Table 124 Specification for DsadcTriggerMode (continued)**

Range	DSADC_TRIGGER_MODE_NORMAL: Result data acquisition starts after calling Dsadc_StartModulation() API. DSADC_TRIGGER_MODE_WINDOW: Result data acquisition starts after the gate signal (GTM, ERU) is open.		
Default value	DSADC_TRIGGER_MODE_NORMAL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.5.9 DsadcWindowCloseNotification**Table 125 Specification for DsadcWindowCloseNotification**

Name	DsadcWindowCloseNotification		
Description	<p>Callback function for window close event.</p> <p>This Parameter is set to the default value if the Parameter DsadcTriggerMode is configured as DSADC_TRIGGER_MODE_NORMAL.</p> <p>Dsadc_NotifyFnPtrType is the data type for this callback function.</p> <p>By default, the notification parameter will be NULL_PTR , to remove the dependency from the user defined functions.</p> <p>The DSADC driver does not validate the configured function name or address for correctness and hence the responsibility falls on the user.</p>		
Multiplicity	1..1	Type	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcTriggerMode		

DSADC driver

2.3.1.6 Container: DsadcCommonModeVoltConfig

This container provides configuration parameters related to common mode voltage application to the inputs of the channel modulator.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.6.1 DsadcComModeVoltNegAEnable

Table 126 Specification for DsadcComModeVoltNegAEnable

Name	DsadcComModeVoltNegAEnable		
Description	<p>This parameter defines if the negative analog line connected to position A in the MUX needs to be connected to the Common Mode Voltage.</p> <p>This parameter is set to the default value if the Parameter DsadcCommonModeVoltageEnable is configured as False or the selected Hardware Channel DsadcHwChannelNum does not have the connection.</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	DsadcCommonModeVoltageEnable, DsadcHwChannelNum		

2.3.1.6.2 DsadcComModeVoltNegBEnable

Table 127 Specification for DsadcComModeVoltNegBEnable

Name	DsadcComModeVoltNegBEnable		
Description	<p>This parameter defines if the negative analog line connected to position B in the MUX needs to be connected to the common mode voltage.</p> <p>This Parameter is set to the default value if the Parameter DsadcCommonModeVoltageEnable is configured as False or the selected Hardware Channel DsadcHwChannelNum does not have the connection.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef

DSADC driver**Table 127 Specification for DsadcComModeVoltNegBEnable (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	DsadcCommonModeVoltageEnable, DsadcHwChannelNum		

2.3.1.6.3 DsadcComModeVoltNegCEnable**Table 128 Specification for DsadcComModeVoltNegCEnable**

Name	DsadcComModeVoltNegCEnable		
Description	<p>This parameter defines if the negative analog line connected to position C in the MUX needs to be connected to the Common Mode Voltage.</p> <p>This Parameter is set to the default value if the Parameter DsadcCommonModeVoltageEnable is configured as False or the selected Hardware Channel DsadcHwChannelNum does not have the connection.</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	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	DsadcCommonModeVoltageEnable, DsadcHwChannelNum		

2.3.1.6.4 DsadcComModeVoltNegDEnable**Table 129 Specification for DsadcComModeVoltNegDEnable**

Name	DsadcComModeVoltNegDEnable
-------------	----------------------------

DSADC driver**Table 129 Specification for DsadcComModeVoltNegDEnable (continued)**

Description	<p>This parameter defines if the negative analog line connected to position D in the MUX needs to be connected to the Common Mode Voltage.</p> <p>This Parameter is set to the default value if the Parameter DsadcCommonModeVoltageEnable is configured as False or the selected Hardware Channel DsadcHwChannelNum does not have the connection.</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	DsadcCommonModeVoltageEnable, DsadcHwChannelNum		

2.3.1.6.5 DsadcComModeVoltPosAEnable**Table 130 Specification for DsadcComModeVoltPosAEnable**

Name	DsadcComModeVoltPosAEnable		
Description	<p>This parameter defines if the positive analog line connected to position A in the MUX needs to be connected to the Common Mode Voltage.</p> <p>This Parameter is set to the default value if the Parameter DsadcCommonModeVoltageEnable is configured as False or the selected Hardware Channel DsadcHwChannelNum does not have the connection.</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

DSADC driver**Table 130 Specification for DsadcComModeVoltPosAEnable (continued)**

Dependency	DsadcCommonModeVoltageEnable, DsadcHwChannelNum
-------------------	---

2.3.1.6.6 DsadcComModeVoltPosBEnable**Table 131 Specification for DsadcComModeVoltPosBEnable**

Name	DsadcComModeVoltPosBEnable		
Description	<p>This parameter defines if the positive analog line connected to position B in the MUX needs to be connected to the Common Mode Voltage.</p> <p>This Parameter is set to the default value if the Parameter DsadcCommonModeVoltageEnable is configured as False or the selected Hardware Channel DsadcHwChannelNum does not have the connection.</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	DsadcCommonModeVoltageEnable, DsadcHwChannelNum		

2.3.1.6.7 DsadcComModeVoltPosCEnable**Table 132 Specification for DsadcComModeVoltPosCEnable**

Name	DsadcComModeVoltPosCEnable		
Description	<p>This parameter defines if the positive analog line connected to position C in the MUX needs to be connected to the Common Mode Voltage.</p> <p>This Parameter is set to the default value if the Parameter DsadcCommonModeVoltageEnable is configured as False or the selected Hardware Channel DsadcHwChannelNum does not have the connection.</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>		

DSADC driver**Table 132 Specification for DsadcComModeVoltPosCEnable (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	DsadcCommonModeVoltageEnable, DsadcHwChannelNum		

2.3.1.6.8 DsadcComModeVoltPosDEnable**Table 133 Specification for DsadcComModeVoltPosDEnable**

Name	DsadcComModeVoltPosDEnable		
Description	<p>This parameter defines if the positive analog line connected to position D in the MUX needs to be connected to the Common Mode Voltage.</p> <p>This Parameter is set to the default value if the Parameter DsadcCommonModeVoltageEnable is configured as False or the selected Hardware Channel DsadcHwChannelNum does not have the connection.</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	DsadcCommonModeVoltageEnable, DsadcHwChannelNum		

2.3.1.6.9 DsadcCommonModeVoltageEnable**Table 134 Specification for DsadcCommonModeVoltageEnable**

Name	DsadcCommonModeVoltageEnable
Description	<p>This parameter defines the availability of Common Mode voltage to the input pins.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>

DSADC driver**Table 134 Specification for DsadcCommonModeVoltageEnable (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	-		

2.3.1.6.10 DsadcCommonModeVoltageSelect**Table 135 Specification for DsadcCommonModeVoltageSelect**

Name	DsadcCommonModeVoltageSelect		
Description	<p>This parameter defines the voltage level configured as Common Mode voltage, which can be connected to the input pins.</p> <p>This Parameter is set to the Default value if the Parameter DsadcCommonModeVoltageEnable is configured as False.</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	DSADC_VCM_VREFX_16: Common mode voltage is configured as (VREFX/16) DSADC_VCM_VREFX_2: Common mode voltage is configured as (VREFX/2) DSADC_VCM_VREFX_4: Common mode voltage is configured as (VREFX/4) DSADC_VCM_VREFX_8: Common mode voltage is configured as (VREFX/8)		
Default value	DSADC_VCM_VREFX_2		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcCommonModeVoltageEnable		

2.3.1.7 Container: CommonPublishedInformation

This container contains the published information of the DSADC driver.

DSADC driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.7.1 ArMajorVersion**Table 136 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Major version number of AUTOSAR specification on which the appropriate 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	-		

2.3.1.7.2 ArMinorVersion**Table 137 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	-		

DSADC driver**2.3.1.7.3 ArPatchVersion****Table 138 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	Patch version number of AUTOSAR specification on which the appropriate 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	-		

2.3.1.7.4 ModuleId**Table 139 Specification for ModuleId**

Name	ModuleId		
Description	Module ID of this module from Module List		
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	-		

2.3.1.7.5 Release**Table 140 Specification for Release**

Name	Release		
Description	Aurix2G derivative used for the implementation.		
Multiplicity	1..1	Type	EcucStringParamDef

DSADC driver**Table 140 Specification for Release (continued)**

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	-		

2.3.1.7.6 SwMajorVersion**Table 141 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	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	-		

2.3.1.7.7 SwMinorVersion**Table 142 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	EcuIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	Post-build variant multiplicity	-

DSADC driver**Table 142 Specification for SwMinorVersion (continued)**

Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.7.8 SwPatchVersion**Table 143 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.		
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	-		

2.3.1.7.9 VendorId**Table 144 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	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

DSADC driver**Table 144 Specification for VendorId (continued)**

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

2.3.1.8 Container: DsadcComparatorConfiguration

This container provides configuration parameters for comparator configuration.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.8.1 DsadcComparatorEventSelect**Table 145 Specification for DsadcComparatorEventSelect**

Name	DsadcComparatorEventSelect		
Description	<p>This parameter defines the comparator mode selected to generate an alarm event (and also a service request, if the alternate service request 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>DSADC_RESULT_ALWAYS: Event is generated on each new result generated.</p> <p>DSADC_RESULT_INSIDE_RANGE: Event is generated if the result is within the boundaries defined.</p> <p>DSADC_RESULT_OUTSIDE_RANGE: Event is generated if the result is outside the boundaries defined.</p>		
Default value	DSADC_RESULT_ALWAYS		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.8.2 DsadcLowerBoundaryValue**Table 146 Specification for DsadcLowerBoundaryValue**

Name	DsadcLowerBoundaryValue		
Description	<p>This parameter defines the lower boundary used for limit checking.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	-32768 - +32767		

DSADC driver
Table 146 Specification for DsadcLowerBoundaryValue (continued)

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	-		

2.3.1.8.3 DsadcUpperBoundaryValue
Table 147 Specification for DsadcUpperBoundaryValue

Name	DsadcUpperBoundaryValue		
Description	This parameter defines the upper boundary used for limit checking. The value of this Parameter should be greater than the parameter DsadcLowerBoundaryValue. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	-32768 - +32767		
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	DsadcLowerBoundaryValue		

2.3.1.9 Container: DsadcConfigSet

This is the base container that contains Dsadc module related parameters

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.10 Container: DsadcDemEventParameterRefs

Container list down the production errors supported by the DSADC driver. This container must be present when safety check is enabled.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

DSADC driver

2.3.1.10.1 DsadcClcFailureNotification

Table 148 Specification for DsadcClcFailureNotification

Name	DsadcClcFailureNotification		
Description	Parameter defines whether CLC failure DEM notification is enabled or not. This parameter must be present when safety check is enabled. The default value of this parameter is set to NULL to minimize the executable code size.		
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	IFX	Scope	LOCAL
Dependency	DsadcSafetyEnable		

2.3.1.10.2 DsadcFifoFailureNotification

Table 149 Specification for DsadcFifoFailureNotification

Name	DsadcFifoFailureNotification		
Description	Parameter defines whether HW FIFO failure DEM notification is enabled or not. This parameter must be present when safety check is enabled. The default value of this parameter is set to NULL to minimize the executable code size.		
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	IFX	Scope	LOCAL
Dependency	DsadcSafetyEnable		

2.3.1.11 Container: DsadcDemodulatorConfiguration

This container contains configuration parameters related to the de-modulator, input data stream and trigger selection.

Post-Build Variant Multiplicity: -

DSADC driver

Multiplicity Configuration Class: -

2.3.1.11.1 DsadcIntegratorTriggerMode

Table 150 Specification for DsadcIntegratorTriggerMode

Name	DsadcIntegratorTriggerMode		
Description	<p>This parameter defines the Integrator trigger mode.</p> <p>If DsadcTriggerMode is configured as TRIGGER_MODE_WINDOW and the DsadcGateActiveLevel is configured as HIGH, then DsadcIntegratorTriggerMode should be DSADC_INTR_RISING_EDGE or DSADC_INTR_BYPASSED.</p> <p>If DsadcTriggerMode is configured as TRIGGER_MODE_WINDOW and the DsadcGateActiveLevel is configured as LOW, then DsadcIntegratorTriggerMode should be DSADC_INTR_FALLING_EDGE or DSADC_INTR_BYPASSED.</p> <p>If DsadcTriggerMode is configured as TRIGGER_MODE_NORMAL then DsadcIntegratorTriggerMode should be Configured as DSADC_INTR_BYPASSED or DSADC_INTR_ALWAYS_ACTIVE</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>DSADC_INTR_ALWAYS_ACTIVE: No trigger required, integrator is active all the time</p> <p>DSADC_INTR_BYPASSED: No integration trigger, integrator bypassed</p> <p>DSADC_INTR_FALLING_EDGE: Trigger event upon a falling edge i.e. Integrator is activated on the falling edge of the trigger.</p> <p>DSADC_INTR_RISING_EDGE: Trigger event upon a rising edge i.e. Integrator is activated on the rising edge of the trigger.</p>		
Default value	DSADC_INTR_BYPASSED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcTriggerMode , DsadcGateActiveLevel		

2.3.1.11.2 DsadcResultDisplayMode

Table 151 Specification for DsadcResultDisplayMode

Name	DsadcResultDisplayMode
Description	<p>This parameter defines the ranges of the result values i.e. result display modes.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>

DSADC driver**Table 151 Specification for DsadcResultDisplayMode (continued)**

Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_RES_SIGNED_MODE: The result values range is -32768 to +32767 DSADC_RES_UNSIGNED_MODE: The result values range is 0 to +65535		
Default value	DSADC_RES_SIGNED_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.11.3 DsadcTriggerSelect**Table 152 Specification for DsadcTriggerSelect**

Name	DsadcTriggerSelect		
Description	<p>This parameter defines EDSADC channel trigger source.</p> <p>The Trigger source is depends on the DSADC Hardware Channel selected by the parameter DsadcHwChannelNum.</p> <p>If the Parameter DsadcTriggerMode is configured as DSADC_TRIGGER_MODE_WINDOW then the parameter DsadcTriggerSelect should not be configured as TRIGGER_0_NO_DSADC_TRIG.</p> <p>If the Parameter DsadcTriggerMode is configured as DSADC_TRIGGER_MODE_NORMAL then the parameter DsadcTriggerSelect should be configured as TRIGGER_0_NO_DSADC_TRIG.</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>TRIGGER_0_NO_DSADC_TRIG: No trigger is selected for DSADC channel.</p> <p>TRIGGER_x_y: x: The available number of trigger selection.</p> <p>y: depends on the DSADC channel selected and trigger input selected from the available selection.</p>		
Default value	TRIGGER_0_NO_DSADC_TRIG		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

DSADC driver
Table 152 Specification for DsadcTriggerSelect (continued)

Dependency	DsadcTriggerMode , DsadcHwChannelNum
-------------------	--------------------------------------

2.3.1.12 Container: DsadcErsEtlConfig

This container configures the parameters for ERU input triggering. If the Parameter DsadcTriggerSelect selects the ERU as a Trigger source then only this container should be present.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

2.3.1.12.1 DsadcEruErsInputPin

Table 153 Specification for DsadcEruErsInputPin

Name	DsadcEruErsInputPin		
Description	<p>This parameter determines the input pin for the selected ERS.</p> <p>The default value for this parameter is set depends on the configured ERS channel and the first physical input pin for the multiplexer.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ERS_X_Y: X stands for the input connection number Y stands for the input source		
Default value	ERS_X_Y		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.12.2 DsadcEruErsRef

Table 154 Specification for DsadcEruErsRef

Name	DsadcEruErsRef
-------------	----------------

DSADC driver**Table 154 Specification for DsadcEruErsRef (continued)**

Description	<p>This parameter is a reference to the ERU container in the MCU. It lists down all the ERU-ERS channels available.</p> <p>If referred ERU-ERS channel is not marked as ERU_CHANNEL_INP_USED_BY_DSADC_DRIVER in MCU an error message will be raised.</p> <p>If the ERU ERS input channel referenced in one container is already referenced in another container, then an error message is provided.</p> <p>The default value for this parameter is set to NULL to avoid the configuration dependency error for the default configuration</p>		
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	-		

2.3.1.12.3 DsadcEruStatusFlagConfig**Table 155 Specification for DsadcEruStatusFlagConfig**

Name	DsadcEruStatusFlagConfig		
Description	<p>This parameters defines the condition on which the status flag in ETL block of ERU is set. On the inverse of the edge selected, the status flag is cleared.</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>DSADC_ETL_FALLING_EDGE: Status flag of ERU channel is set on the detection of a falling edge on input channel</p> <p>DSADC_ETL_RISING_EDGE: Status flag of ERU channel is set on the detection of a rising edge on input channel</p>		
Default value	DSADC_ETL_FALLING_EDGE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

DSADC driver**Table 155 Specification for DsadcEruStatusFlagConfig (continued)**

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

2.3.1.13 Container: DsadcFilterConfiguration

This container provides configuration parameter related to main filter chain.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.13.1 DsadcAlternateServiceReq**Table 156 Specification for DsadcAlternateServiceReq**

Name	DsadcAlternateServiceReq		
Description	<p>This parameter is used to generate alternate service request for any one of the following events, comparator event, time stamp event or alternate source.</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>DSADC_ALT_SERVICE_DISABLE: Service request is disabled.</p> <p>DSADC_COMPARATOR_EVENT: comparator event generates the service request.</p> <p>DSADC_RESOLVER_EVENT: Alternate source (capturing of a sign delay value to register carrier generator synchronization register) Service request.</p> <p>DSADC_TIMESTAMP_EVENT: Timestamp event generates service request.</p>		
Default value	DSADC_ALT_SERVICE_DISABLE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.13.2 DsadcCICFilterDecimationFactor**Table 157 Specification for DsadcCICFilterDecimationFactor**

Name	DsadcCICFilterDecimationFactor		
Description	<p>This parameter defines the oversampling rate/Decimation factor for the CIC filter.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	4 - 512		

DSADC driver**Table 157 Specification for DsadcCICFilterDecimationFactor (continued)**

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	-		

2.3.1.13.3 DsadcCICFilterStartValue**Table 158 Specification for DsadcCICFilterStartValue**

Name	DsadcCICFilterStartValue		
Description	<p>This parameter defines the starting value of decimation counter, when the CIC filter is started/restarted.</p> <p>If the value of DsadcCICFilterStartValue is set higher than the value of DsadcCICFilterDecimationFactor, then an error message is provided.</p> <p>Starting value exceeding the value specified in DsadcCICFilterDecimationFactor can lead to overflow of CIC filter.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	4 - 512		
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	DsadcCICFilterDecimationFactor		

2.3.1.13.4 DsadcFIR0FilterEnable**Table 159 Specification for DsadcFIR0FilterEnable**

Name	DsadcFIR0FilterEnable		
Description	<p>This parameter defines the availability of FIR0 filter in the filter chain of the DSADC channel.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef

DSADC driver**Table 159 Specification for DsadcFIR0FilterEnable (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	-		

2.3.1.13.5 DsadcFIR1FilterDecimationEnable**Table 160 Specification for DsadcFIR1FilterDecimationEnable**

Name	DsadcFIR1FilterDecimationEnable		
Description	<p>This parameter defines the decimation rate of FIR1 filter. If selected as TRUE, then the filter decimates with a ratio 2:1.</p> <p>This Parameter takes the default value when the parameter DsadcFIR1FilterEnable set to FALSE.</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	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	DsadcFIR1FilterEnable		

2.3.1.13.6 DsadcFIR1FilterEnable**Table 161 Specification for DsadcFIR1FilterEnable**

Name	DsadcFIR1FilterEnable
-------------	-----------------------

DSADC driver**Table 161 Specification for DsadcFIR1FilterEnable (continued)**

Description	This parameter defines the availability of FIR1 filter in the filter chain of the DSADC channel. This Parameter takes the default value when the parameter DsadcFIR0FilterEnable set to FALSE. 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	DsadcFIR0FilterEnable		

2.3.1.13.7 DsadcOffsetCompFilterEnable**Table 162 Specification for DsadcOffsetCompFilterEnable**

Name	DsadcOffsetCompFilterEnable		
Description	This parameter indicates availability of an Offset Compensation (IIR) filter (and its various operation modes) in Offset Compensation block. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

DSADC driver**Table 162 Specification for DsadcOffsetCompFilterEnable (continued)**

Range	DSADC_OFFSETCOMP_FILTER_DISABLE: Offset compensation filter is disabled. DSADC_OFFSETCOMP_FILTER_RATE_1: Offset compensation filter is enabled and it adjusts OFFCOMP register with offset Compensation filter with Rate 1 DSADC_OFFSETCOMP_FILTER_RATE_2: Offset compensation filter is enabled and it adjusts OFFCOMP register with offset Compensation filter with Rate 2 DSADC_OFFSETCOMP_FILTER_RATE_3: Offset compensation filter is enabled and it adjusts OFFCOMP register with offset Compensation filter with Rate 3 DSADC_OFFSETCOMP_FILTER_RATE_4: Offset compensation filter is enabled and it adjusts OFFCOMP register with offset Compensation filter with Rate 4 DSADC_OFFSETCOMP_FILTER_RATE_5: Offset compensation filter is enabled and it adjusts OFFCOMP register with offset Compensation filter with Rate 5 DSADC_OFFSETCOMP_FILTER_RATE_6: Offset compensation filter is enabled and it adjusts OFFCOMP register with offset Compensation filter with Rate 6 DSADC_OFFSETCOMP_FILTER_RATE_7: Offset compensation filter is enabled and it adjusts OFFCOMP register with offset Compensation filter with Rate 7		
Default value	DSADC_OFFSETCOMP_FILTER_DISABLE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.13.8 DsadcOffsetCompValue**Table 163 Specification for DsadcOffsetCompValue**

Name	DsadcOffsetCompValue		
Description	<p>This parameter defines the offset component value to be removed (subtracted) from each result from the filter chain.</p> <p>This Parameter is set to the default value '0' if the parameter DsadcOffsetCompFilterEnable is not configured as DSADC_OFFSETCOMP_FILTER_DISABLE.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	-32768 - +32767		
Default value	0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

DSADC driver**Table 163 Specification for DsadcOffsetCompValue (continued)**

Origin	IFX	Scope	LOCAL
Dependency	DsadcOffsetCompFilterEnable		

2.3.1.13.9 DsadcOffsetCompValueProtect**Table 164 Specification for DsadcOffsetCompValueProtect**

Name	DsadcOffsetCompValueProtect		
Description	<p>This parameter defines the protection of the Offset Compensation register from the calibration algorithm.</p> <p>This Parameter should not set to false when the parameter DsadcOffsetCompFilterEnable is not configured as OFFCOMP_FILTER_DISABLE.</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	DsadcOffsetCompFilterEnable		

2.3.1.13.10 DsadcOvershootCompensationEn**Table 165 Specification for DsadcOvershootCompensationEn**

Name	DsadcOvershootCompensationEn		
Description	<p>This parameter defines the availability of Overshoot compensation block in the filter chain of the DSADC channel.</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		

DSADC driver
Table 165 Specification for DsadcOvershootCompensationEn (continued)

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

2.3.1.13.11 DsadcPreFilterEnable
Table 166 Specification for DsadcPreFilterEnable

Name	DsadcPreFilterEnable		
Description	This parameter defines the availability of Prefilter in the filter chain of the DSADC channel 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	-		

2.3.1.14 Container: DsadcGainCalibConfig

This container provides configuration parameters for Gain calibration.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.15 Container: DsadcGainCorrConfig

This container provides configuration parameters for Gain correction.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

DSADC driver

2.3.1.15.1 DsadcCICFilterOutputShiftPos

Table 167 Specification for DsadcCICFilterOutputShiftPos

Name	DsadcCICFilterOutputShiftPos		
Description	<p>This parameter defines the position of the CIC output shifter, which is to be used to select the valid output bits from the CIC filter.</p> <p>Default value for this parameter is BITS_0_TO_16. The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	BITS_i_TO_j: Valid output bits selection from the CIC Filter output. Possible values of i and j are i : 0 to 28 j : 16 to 44		
Default value	BITS_0_TO_16		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.15.2 DsadcGainCorrMulFactor

Table 168 Specification for DsadcGainCorrMulFactor

Name	DsadcGainCorrMulFactor		
Description	<p>This parameter defines multiplication factor for Gain correction.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0.0000 - 1.9999		
Default value	1.0000		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

DSADC driver**2.3.1.16 Container: DsadcGeneral**

This Container contains all the general configuration parameters for the DSADC driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.16.1 DsadcDeInitApi**Table 169 Specification for DsadcDeInitApi**

Name	DsadcDeInitApi		
Description	This Parameter adds or removes the service Dsadc_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	IFX	Scope	LOCAL
Dependency	-		

2.3.1.16.2 DsadcDevErrorDetect**Table 170 Specification for DsadcDevErrorDetect**

Name	DsadcDevErrorDetect		
Description	This Parameter Enables/Disables the Development Error Detection and reporting in the DSADC Driver. The default value for this parameter is set to TRUE to ensure that the Development Errors 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	-

DSADC driver**Table 170 Specification for DsadcDevErrorDetect (continued)**

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

2.3.1.16.3 DsadcInitCheckApi**Table 171 Specification for DsadcInitCheckApi**

Name	DsadcInitCheckApi		
Description	This Parameter adds or removes the service Dsadc_InitCheck() 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	IFX	Scope	LOCAL
Dependency	-		

2.3.1.16.4 DsadcInitDeInitApiMode**Table 172 Specification for DsadcInitDeInitApiMode**

Name	DsadcInitDeInitApiMode		
Description	This parameter defines the privilege mode in which the Initialization and De-initialization APIs would operate. Since the DSADC driver accesses the SFRs, it is efficient to operate the DSADC driver in supervisory mode than the USER1 mode. Hence, the default mode of operation is the supervisory mode.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_MCAL_SUPERVISOR: Operating mode used is Supervisory. DSADC_MCAL_USER1: Operating mode used is USER1.		

DSADC driver**Table 172 Specification for DsadcInitDeInitApiMode (continued)**

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

2.3.1.16.5 DsadcRuntimeApiMode**Table 173 Specification for DsadcRuntimeApiMode**

Name	DsadcRuntimeApiMode		
Description	<p>This Parameter defines the privilege mode in which the runtime APIs would operate.</p> <p>Since the DSADC driver accesses the SFRs, it is efficient to operate the DSADC driver in supervisory mode. Hence, the default mode of operation is supervisory mode.</p> <p>DsadcRuntimeApiMode must be configured as User-1 mode if DsadcInitDeInitApiMode is configured as User-1 mode.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>DSADC_MCAL_SUPERVISOR: Operating mode used is Supervisory.</p> <p>DSADC_MCAL_USER1: Operating mode used is USER1.</p>		
Default value	DSADC_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcInitDeInitApiMode		

2.3.1.16.6 DsadcSafetyEnable**Table 174 Specification for DsadcSafetyEnable**

Name	DsadcSafetyEnable
Description	<p>This Parameter determines whether to Enable/Disable the safety check and reporting.</p> <p>The default value of this parameter is set TRUE to ensure that safety issues are addressed during the product lifecycle.</p>

DSADC driver**Table 174 Specification for DsadcSafetyEnable (continued)**

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	-		

2.3.1.16.7 DsadcVersionInfoApi**Table 175 Specification for DsadcVersionInfoApi**

Name	DsadcVersionInfoApi		
Description	This Parameter adds or removes the Dsadc_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	IFX	Scope	LOCAL
Dependency	-		

2.3.1.17 Container: DsadcGlobalConfiguration

This container contains the parameters to configure DSADC IP global configuration.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

DSADC driver**2.3.1.17.1 DsadcDitheringTrimValue****Table 176 Specification for DsadcDitheringTrimValue**

Name	DsadcDitheringTrimValue		
Description	<p>This Parameter defines the trimming value for internal dithering function. This trimming value is used for all the modulators of DSADC.</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>DSADC_DITHERING_HIGH_400_MILVLT: Dithering intensity is high volts which is 400 millivolts.</p> <p>DSADC_DITHERING_LOW_100_MILVLT: Dithering intensity is Low volts which is 100 millivolts.</p> <p>DSADC_DITHERING_MED_200_MILVLT: Dithering intensity is medium volts which is 200 millivolts</p> <p>DSADC_DITHERING_MIN_50_MILVLT: Dithering intensity is minimum volts which is 50 millivolts.</p>		
Default value	DSADC_DITHERING_MIN_50_MILVLT		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.17.2 DsadcSleepMode**Table 177 Specification for DsadcSleepMode**

Name	DsadcSleepMode		
Description	<p>This Parameter defines EDSADC reaction to the Sleep mode requests.</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>SLEEP_DISABLE: Disable the sleep mode for DSADC module</p> <p>SLEEP_ENABLE: Enable the sleep mode for DSADC module</p>		
Default value	SLEEP_ENABLE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-

DSADC driver**Table 177 Specification for DsadcSleepMode (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.17.3 DsadcSupplyVoltageLevel**Table 178 Specification for DsadcSupplyVoltageLevel**

Name	DsadcSupplyVoltageLevel		
Description	This Parameter defines the supply voltage level to be used for DSADC internal operations. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	VOLTAGESUPPLY_3_3V: 3.3V power supply is connected. VOLTAGESUPPLY_5V: 5V power supply is connected VOLTAGESUPPLY_AUTO: The voltage range is controlled by the power supply		
Default value	VOLTAGESUPPLY_AUTO		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.17.4 DsadcSyncClockGen**Table 179 Specification for DsadcSyncClockGen**

Name	DsadcSyncClockGen		
Description	This Parameter defines the influence of Analog Phase synchronizer on the clock generated. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SYNCHRONIZED_MODE: Rising clock edge is defined by the phase synchronizer. UNSYNCHRONIZED_MODE: Modulator clock is generated without the influence of phase synchronizer		
Default value	UNSYNCHRONIZED_MODE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DSADC driver
Table 179 Specification for DsadcSyncClockGen (continued)

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

2.3.1.18 Container: DsadcIntegratorConfiguration

This container provides configuration parameter related to the Integrator functional block. If the Parameter DsadcIntegratorTriggerMode is set to DSADC_INTR_BYPASSED, then this container cannot be added.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

2.3.1.18.1 DsadcDiscardCount

Table 180 Specification for DsadcDiscardCount

Name	DsadcDiscardCount		
Description	This parameter defines the number of result values to be discarded before the start of the integration cycle. The default value of this parameter is set to the reset value of the corresponding SFR.		
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	IFX	Scope	LOCAL
Dependency	-		

2.3.1.18.2 DsadcIntegrationCount

Table 181 Specification for DsadcIntegrationCount

Name	DsadcIntegrationCount		
Description	This parameter defines the number of result values to be integrated. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	2 - 64		

DSADC driver**Table 181 Specification for DsadcIntegrationCount (continued)**

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	-		

2.3.1.19 Container: DsadcModulatorConfiguration

This container contains configuration parameters related to the On-chip modulator, input pin selection and modulator clock configuration.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

2.3.1.19.1 DsadcAnalogClockSyncDelay**Table 182 Specification for DsadcAnalogClockSyncDelay**

Name	DsadcAnalogClockSyncDelay		
Description	<p>This parameter defines the delay in clock cycles after the sync signal provided from Phase synchronizer.</p> <p>This Parameter is set to the default value when the parameter DsadcSyncClockGen is set to UNSYNCHRONIZED_MODE.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
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	DsadcSyncClockGen		

2.3.1.19.2 DsadcClockDivider**Table 183 Specification for DsadcClockDivider**

Name	DsadcClockDivider
-------------	-------------------

DSADC driver**Table 183 Specification for DsadcClockDivider (continued)**

Description	This parameter defines the divider factor, which defines the frequency of the modulator clock, which is derived from the peripheral clock. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_CLOCKDIVIDER_DIV10: Input clock is divided by 10 for modulator DSADC_CLOCKDIVIDER_DIV12: Input clock is divided by 12 for modulator DSADC_CLOCKDIVIDER_DIV14: Input clock is divided by 14 for modulator DSADC_CLOCKDIVIDER_DIV16: Input clock is divided by 16 for modulator DSADC_CLOCKDIVIDER_DIV18: Input clock is divided by 18 for modulator DSADC_CLOCKDIVIDER_DIV4: Input clock is divided by 4 for modulator DSADC_CLOCKDIVIDER_DIV6: Input clock is divided by 6 for modulator DSADC_CLOCKDIVIDER_DIV8: Input clock is divided by 8 for modulator		
Default value	DSADC_CLOCKDIVIDER_DIV4		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build		
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.19.3 DsadcDitheringEnable**Table 184 Specification for DsadcDitheringEnable**

Name	DsadcDitheringEnable		
Description	This parameter defines the availability of internal dithering functionality for the modulator. 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		

DSADC driver**Table 184 Specification for DsadcDitheringEnable (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.19.4 DsadcInputGain**Table 185 Specification for DsadcInputGain**

Name	DsadcInputGain		
Description	This parameter defines multiplication Gain factor for the analog input signal. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_INPUT_GAIN_FACTOR_1: Gain factor is 1 DSADC_INPUT_GAIN_FACTOR_2: Gain factor is 2 DSADC_INPUT_GAIN_FACTOR_4: Gain factor is 4		
Default value	DSADC_INPUT_GAIN_FACTOR_1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.19.5 DsadcInputMuxActionMode**Table 186 Specification for DsadcInputMuxActionMode**

Name	DsadcInputMuxActionMode		
Description	This parameter defines the control mechanism for input multiplexer. It defines the action to be taken for the input multiplexer upon a trigger event for pin selection. This parameter is set to the default value if the Parameter DsadcTriggerMode is configured as DSADC_TRIGGER_MODE_NORMAL. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_INPUTMUX_PRESET_MODE: Load INMUX upon a trigger DSADC_INPUTMUX_SINGLE_STEP_MODE: Decrement INMUX value upon a trigger and wrap around the value specified in parameter DsadcInputPinSelection.		
Default value	DSADC_INPUTMUX_PRESET_MODE		

DSADC driver**Table 186 Specification for DsadcInputMuxActionMode (continued)**

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

2.3.1.19.6 DsadcInputMuxControlMode**Table 187 Specification for DsadcInputMuxControlMode**

Name	DsadcInputMuxControlMode		
Description	<p>This parameter defines the condition for a trigger event to control the input multiplexer.</p> <p>This parameter is set to the default value If the Parameter DsadcTriggerSelect is configured as DSADC_TRIGGER_MODE_NORMAL.</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>DSADC_INMUX_SOFTWARE_CONTROL: Multiplexer controlled by software</p> <p>DSADC_INMUX_TRIG_EVENT_BOTH_EDGES: Trigger event upon an any edge</p> <p>DSADC_INMUX_TRIG_EVENT_FALLING_EDGE: Trigger event upon a falling edge</p> <p>DSADC_INMUX_TRIG_EVENT_RISING_EDGE: Trigger event upon a raising edge</p>		
Default value	DSADC_INMUX_SOFTWARE_CONTROL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcTriggerMode		

2.3.1.19.7 DsadcInputPinSelection**Table 188 Specification for DsadcInputPinSelection**

Name	DsadcInputPinSelection
-------------	------------------------

DSADC driver**Table 188 Specification for DsadcInputPinSelection (continued)**

Description	<p>This parameter defines the initial/permanent setting for the input multiplexer, based on the operating mode selected.</p> <p>The Input pin selection is depends on the DSADC Hardware Channel selected by the parameter DsadcHwChannelNum.</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	INPUT_PIN_[x]: where [x] stands for the pin selected for the selected channel		
Default value	INPUT_PIN_[x]		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcHwChannelNum		

2.3.1.19.8 DsadcIntegratorResetEnable**Table 189 Specification for DsadcIntegratorResetEnable**

Name	DsadcIntegratorResetEnable		
Description	<p>This parameter defines the modulator overload handling.</p> <p>This Parameter is set to the Default value when the Parameter DsadcIntegratorTriggerMode is configured as DSADC_INTR_BYPASSED</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	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	DsadcIntegratorTriggerMode		

DSADC driver**2.3.1.19.9 DsadcNegativeInputLine****Table 190 Specification for DsadcNegativeInputLine**

Name	DsadcNegativeInputLine		
Description	<p>This parameter defines the modulator internal connection of the negative input.</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>DSADC_NEG_INPUT_PIN: Modulator negative input is connected to Input pin.</p> <p>DSADC_NEG_IN_COMMON_MODE_VOLT: Modulator negative input is connected to Common mode voltage V<sub>x</sub>REFX</sub>.</p> <p>DSADC_NEG_IN_REFERENCE_GROUND: Modulator negative input is connected to reference ground V<sub>x</sub>AGND</sub></p> <p>DSADC_NEG_IN_SUPPLY_VOLT: Modulator negative input is connected to Supply voltage V<sub>x</sub>AREF</sub>.</p>		
Default value	DSADC_NEG_INPUT_PIN		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.19.10 DsadcPositiveInputLine**Table 191 Specification for DsadcPositiveInputLine**

Name	DsadcPositiveInputLine		
Description	<p>This parameter defines the modulator internal connection of the positive input.</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>DSADC_POS_INPUT_PIN: Modulator positive input is connected to Input pin</p> <p>DSADC_POS_IN_COMMON_MODE_VOLT: Modulator positive input is connected to Common mode voltage V<sub>x</sub>REFX</sub>.</p> <p>DSADC_POS_IN_REFERENCE_GROUND: Modulator positive input is connected to reference ground V<sub>x</sub>AGND</sub></p> <p>DSADC_POS_IN_SUPPLY_VOLT: Modulator positive input is connected to Supply voltage V<sub>x</sub>AREF</sub>.</p>		
Default value	DSADC_POS_INPUT_PIN		

DSADC driver**Table 191 Specification for DsadcPositiveInputLine (continued)**

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

2.3.1.20 Container: DsadcOguConfig

This container configures the parameters for ERU Pattern Detection output. This container can be added only if the Parameter DsadcTriggerSelect selecting the ERU signal as a trigger source.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

2.3.1.20.1 DsadcEruErsCh0PatternFlagEnable**Table 192 Specification for DsadcEruErsCh0PatternFlagEnable**

Name	DsadcEruErsCh0PatternFlagEnable		
Description	<p>This parameter determines of the ERU ERS channel 0 is used for pattern detection for the selected OGU channel</p> <p>The default value of this parameter is set to FALSE to avoid the configuration dependency error for the default configuration.</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	-		

2.3.1.20.2 DsadcEruErsCh1PatternFlagEnable**Table 193 Specification for DsadcEruErsCh1PatternFlagEnable**

Name	DsadcEruErsCh1PatternFlagEnable
-------------	---------------------------------

DSADC driver**Table 193 Specification for DsadcEruErsCh1PatternFlagEnable (continued)**

Description	This parameter determines if the ERU ERS channel 1 is used for pattern detection for the selected OGU channel. The default value of this parameter is set to FALSE to avoid the configuration dependency error for the default configuration.		
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	-		

2.3.1.20.3 DsadcEruErsCh2PatternFlagEnable**Table 194 Specification for DsadcEruErsCh2PatternFlagEnable**

Name	DsadcEruErsCh2PatternFlagEnable		
Description	This parameter determines if the ERU ERS channel 2 is used for pattern detection for the selected OGU channel. The default value of this parameter is set to FALSE to avoid the configuration dependency error for the default configuration.		
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	-		

DSADC driver

2.3.1.20.4 DsadcEruErsCh3PatternFlagEnable

Table 195 Specification for DsadcEruErsCh3PatternFlagEnable

Name	DsadcEruErsCh3PatternFlagEnable		
Description	<p>This parameter determines if the ERU ERS channel 3 is used for pattern detection for the selected OGU channel.</p> <p>The default value of this parameter is set to FALSE to avoid the configuration dependency error for the default configuration.</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	-		

2.3.1.20.5 DsadcEruErsCh4PatternFlagEnable

Table 196 Specification for DsadcEruErsCh4PatternFlagEnable

Name	DsadcEruErsCh4PatternFlagEnable		
Description	<p>This parameter determines if the ERU ERS channel 4 is used for pattern detection for the selected OGU channel.</p> <p>The default value of this parameter is set to FALSE to avoid the configuration dependency error for the default configuration.</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	-		

DSADC driver

2.3.1.20.6 DsadcEruErsCh5PatternFlagEnable

Table 197 Specification for DsadcEruErsCh5PatternFlagEnable

Name	DsadcEruErsCh5PatternFlagEnable		
Description	<p>This parameter determines if the ERU ERS channel 5 is used for pattern detection for the selected OGU channel.</p> <p>The default value of this parameter is set to FALSE to avoid the configuration dependency error for the default configuration.</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	-		

2.3.1.20.7 DsadcEruErsCh6PatternFlagEnable

Table 198 Specification for DsadcEruErsCh6PatternFlagEnable

Name	DsadcEruErsCh6PatternFlagEnable		
Description	<p>This parameter determines if the ERU ERS channel 6 is used for pattern detection for the selected OGU channel.</p> <p>The default value of this parameter is set to FALSE to avoid the configuration dependency error for the default configuration.</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	-		

DSADC driver

2.3.1.20.8 DsadcEruErsCh7PatternFlagEnable

Table 199 Specification for DsadcEruErsCh7PatternFlagEnable

Name	DsadcEruErsCh7PatternFlagEnable		
Description	<p>This parameter determines if the ERU ERS channel 7 is used for pattern detection for the selected OGU channel.</p> <p>The default value of this parameter is set to FALSE to avoid the configuration dependency error for the default configuration.</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	-		

2.3.1.20.9 DsadcEruOguRef

Table 200 Specification for DsadcEruOguRef

Name	DsadcEruOguRef		
Description	<p>This parameter is a reference to the ERU container in the MCU. It lists down all the ERU-OGU channels available. The OGUs available is dependent on the DsadcHwChannelNum.</p> <p>The Trigger source ERU PD_OUT selected by the DsadcTriggerSelect should be connected to the corresponding DSADC Channel.</p>		
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	DsadcHwChannelNum, DsadcTriggerSelect		

DSADC driver

2.3.1.21 Container: DsadcOvershootCompenConfig

This container provides configuration parameters for the Overshoot Compensation block. If the Parameter DsadcOvershootCompensationEn is set to FALSE, then this container cannot be added

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

2.3.1.21.1 DsadcSlewRateFilterRunTime

Table 201 Specification for DsadcSlewRateFilterRunTime

Name	DsadcSlewRateFilterRunTime		
Description	This parameter defines the time constant for the slew rate filter. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_SLEWRATE_FILTR_RUNTIME_16: Slew Rate Filter runs for 16 Input cycle DSADC_SLEWRATE_FILTR_RUNTIME_2: Slew Rate Filter runs for 2 Input cycle DSADC_SLEWRATE_FILTR_RUNTIME_4: Slew Rate Filter runs for 4 Input cycle DSADC_SLEWRATE_FILTR_RUNTIME_8: Slew Rate Filter runs for 8 Input cycle		
Default value	DSADC_SLEWRATE_FILTR_RUNTIME_2		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.21.2 DsadcSlewRateFilterStrength

Table 202 Specification for DsadcSlewRateFilterStrength

Name	DsadcSlewRateFilterStrength		
Description	This parameter defines the filter strength for the slew rate filter. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_MAXIMUM_FILTER_EFFECT: Maximum filter effect for slew rate Filter DSADC_MEDIUM_FILTER_EFFECT: Medium filter effect for slew rate Filter DSADC_MINIMUM_FILTER_EFFECT: Minimum filter effect for slew rate Filter DSADC_WEAK_FILTER_EFFECT: Weak filter effect for slew rate Filter		
Default value	DSADC_MINIMUM_FILTER_EFFECT		

DSADC driver**Table 202 Specification for DsadcSlewRateFilterStrength (continued)**

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

2.3.1.21.3 DsadcStepDetectionMode**Table 203 Specification for DsadcStepDetectionMode**

Name	DsadcStepDetectionMode		
Description	This parameter defines the when the slew rate filter has to be activated. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_STEP_DETECT_CMP_LAST: Compare threshold to difference of current and last input to activate the slew rate filter DSADC_STEP_DETECT_CMP_SEC_LAST: Compare threshold to difference of current and second last input to activate the slew rate filter		
Default value	DSADC_STEP_DETECT_CMP_LAST		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.1.21.4 DsadcStepDetectionThreshold**Table 204 Specification for DsadcStepDetectionThreshold**

Name	DsadcStepDetectionThreshold		
Description	This parameter defines the threshold value (magnitude) used for step detection. The threshold value is DsadcStepDetectionThreshold multiplied with 32. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 2047		

DSADC driver
Table 204 Specification for DsadcStepDetectionThreshold (continued)

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	-		

2.3.1.22 Container: DsadcRectificationConfiguration

This container provides configuration parameters of the rectifier block, used for resolver support, to determine position of the motor. This Container is added only when the Parameter DsadcIntegratorTriggerMode is not configured as DSADC_INTR_BYPASSED.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

2.3.1.22.1 DsadcNegSignDelayValue

Table 205 Specification for DsadcNegSignDelayValue

Name	DsadcNegSignDelayValue		
Description	<p>This parameter determines the value of Sign Delay counter (number of result values), for which to generate a negative delayed sign signal.</p> <p>If the Parameter DsadcRectificationEnable is set to FALSE then this parameter is set to the Default value.</p> <p>This Parameter value needs to be configured greater than the parameter DsadcPosSignDelayValue.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
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	IFX	Scope	LOCAL
Dependency	DsadcPosSignDelayValue, DsadcRectificationEnable		

DSADC driver

2.3.1.22.2 DsadcPosSignDelayValue

Table 206 Specification for DsadcPosSignDelayValue

Name	DsadcPosSignDelayValue		
Description	<p>This parameter determines the value of Sign Delay counter (number of result values), for which to generate a positive delayed sign signal.</p> <p>If the Parameter DsadcRectificationEnable is set to FALSE then this parameter takes the Default value.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 255		
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	DsadcRectificationEnable		

2.3.1.22.3 DsadcRectificationEnable

Table 207 Specification for DsadcRectificationEnable

Name	DsadcRectificationEnable		
Description	<p>This parameter controls the action of the rectifier circuit on the input data.</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	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	-		

DSADC driver

2.3.1.22.4 DsadcSignSignalChannel

Table 208 Specification for DsadcSignSignalChannel

Name	DsadcSignSignalChannel		
Description	<p>This parameter selects the DSADC channel which provides the source of the sign signal, which is to be delayed for the purpose of rectification.</p> <p>The Selection of DSADC Channel as a Sign Signal source is possible only when the Parameter DsadcSignSignalSource is set to SRC_1_SIGNRESULT_FROM_DSADC_CHANNEL and DsadcRectificationEnable is set to TRUE.</p> <p>The default value for this parameter is DSADC_CHANNEL_0 and it is the result value for the SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	DSADC_CHANNEL_x: where x varies for channel number		
Default value	DSADC_CHANNEL_0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcSignSignalSource, DsadcRectificationEnable		

2.3.1.22.5 DsadcSignSignalSource

Table 209 Specification for DsadcSignSignalSource

Name	DsadcSignSignalSource		
Description	<p>This parameter selects the source of the sign signal, which is to be delayed for the purpose of rectification.</p> <p>If the Parameter DsadcRectificationEnable is set to FALSE then this parameter takes the Default value.</p> <p>The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef

DSADC driver**Table 209 Specification for DsadcSignSignalSource (continued)**

Range	SRC_0_ON_CHIP_CARRIER_GENERATOR: Sign signal is generated from the On-chip carrier generator SRC_1_SIGNRESULT_FROM_DSADC_CHANNEL: The Sign signal is generated from one of the DSADC channels SRC_2_EXTERNAL_SIGN_SIGNAL_PORT_A: Sign signal is provide from external source, through Port Pin A 'A': Port pin number depends on the device. refer device property files SRC_3_EXTERNAL_SIGN_SIGNAL_PORT_B: Sign signal is provide from external source, through Port Pin B 'B': Port pin number depends on the device. refer device property files		
Default value	SRC_0_ON_CHIP_CARRIER_GENERATOR		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	DsadcRectificationEnable		

2.3.1.23 Container: DsadcTimestampConfiguration

This container provides configuration parameters regarding the timestamp counter configuration. This Container is added only when the Parameter DsadcTimestampFeature is configured as DSADC_TIMESTAMP_ENABLED.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

2.3.1.23.1 DsadcInputMuxSetCopyEnable**Table 210 Specification for DsadcInputMuxSetCopyEnable**

Name	DsadcInputMuxSetCopyEnable		
Description	This parameter defines the availability of Analog MUX setting in the timestamp information. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucBooleanParamD ef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	Post-build variant multiplicity	-

DSADC driver**Table 210 Specification for DsadcInputMuxSetCopyEnable (continued)**

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

2.3.1.23.2 DsadcTimestampCounterClockSel**Table 211 Specification for DsadcTimestampCounterClockSel**

Name	DsadcTimestampCounterClockSel		
Description	<p>This parameter defines the divider factor, which defines the clock used for Timestamp counter.</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	CLOCKDIVIDER_DIV1: Modulator clock is divided by 1 for timestamp counter increment CLOCKDIVIDER_DIV2: Modulator clock is divided by 2 for timestamp counter increment CLOCKDIVIDER_DIV4: Modulator clock is divided by 4 for timestamp counter increment CLOCKDIVIDER_DIV8: Modulator clock is divided by 8 for timestamp counter increment		
Default value	CLOCKDIVIDER_DIV1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

2.3.2 Functions - Type definitions**2.3.2.1 Dsadc_CalibrationStatusType****Table 212 Specification for Dsadc_CalibrationStatusType**

Syntax	Dsadc_CalibrationStatusType	
Type	uint8	
File	Dsadc.h	
Range	0 - DSADC_CALIBRATION_NOT_STARTED	The DSADC channel Calibration activity not started.

DSADC driver**Table 212 Specification for Dsadc_CalibrationStatusType (continued)**

	1 - DSADC_CALIBRATION_RUNNING	Calibration algorithm is currently running
	2 - DSADC_CALIBRATION_DONE	Calibration is completed. Normal operations is possible
	3 - DSADC_CALIBRATION_ERROR	Calibration terminated incorrectly.
Description	This datatype is used to define the various states of the calibration algorithm for the DSADC channel.	
Source	IFX	

Dsadc_ChannelStatusType**Table 213 Specification for Dsadc_ChannelStatusType**

Syntax	Dsadc_ChannelStatusType	
Type	uint8	
File	Dsadc.h	
Range	0 - DSADC_IDLE	DSADC channel is in idle state
	1 - DSADC_BUSY	DSADC channel is in busy state
	2 - DSADC_RESULT_READY	Result is available for DSADC channel
Description	Gives the status of DSADC channel	
Source	IFX	

Dsadc_ChannelType**Table 214 Specification for Dsadc_ChannelType**

Syntax	Dsadc_ChannelType	
Type	uint8	
File	Dsadc.h	
Range	0 to 255	
Description	Numeric identifier of DSADC channel.	
Source	IFX	

Dsadc_ConfigType**Table 215 Specification for Dsadc_ConfigType**

Syntax	Dsadc_ConfigType	
Type	Structure	
File	Dsadc.h	

DSADC driver**Table 215 Specification for Dsadc_ConfigType (continued)**

Range	--
Description	This type defines the data structure used to store the configuration root for the DSADC driver
Source	IFX

2.3.2.5 Dsadc_NotifyFnPtrType**Table 216 Specification for Dsadc_NotifyFnPtrType**

Syntax	Dsadc_NotifyFnPtrType
Type	Pointer to a function of type void Function_Name (void)
File	Dsadc.h
Description	Defines the function pointer type for call back functions.
Source	IFX

2.3.2.6 Dsadc_ResultType**Table 217 Specification for Dsadc_ResultType**

Syntax	Dsadc_ResultType
Type	sint16
File	Dsadc.h
Range	-32768 to 32767
Description	DataType used for the result value generated from the DSADC channel.
Source	IFX

2.3.2.7 Dsadc_SizeType**Table 218 Specification for Dsadc_SizeType**

Syntax	Dsadc_SizeType
Type	uint16
File	Dsadc.h
Range	0 to 65535
Description	Datatype used to define the size of the buffer and also the same type is used to return the valid size entries
Source	IFX

DSADC driver**2.3.2.8 Dsadc_TimeStampType****Table 219 Specification for Dsadc_TimeStampType**

Syntax	Dsadc_TimeStampType
Type	uint16
File	Dsadc.h
Range	0 to 65535
Description	This data type is used to return the current timestamp value.
Source	IFX

2.3.3 Functions - APIs

This section lists all the APIs of the driver.

2.3.3.1 Dsadc_Init**Table 220 Specification for Dsadc_Init API**

Syntax	void Dsadc_Init (const Dsadc_ConfigType * const ConfigPtr)	
Service ID	0x1A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the DSADC driver configuration structure
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>This API initializes the EDSADC hardware as per the configuration pointer passed and sets all the global variables to their required initial values. The SFRs of EDSADC hardware are reset to their default values and then they are configured with the given configuration data. This API will set the DSADC module state to DSADC_INITIALIZED if the initialization is successful.</p> <p>It also enables the DSADC module by writing into the CLC register and enable the modulator and demodulator for the configured channels.</p>	
Source	IFX	

DSADC driver**Table 220 Specification for Dsadc_Init API (continued)**

Error handling	<p>DET:</p> <p>DSADC_E_PARAM_CONFIG: Error code is reported if Dsadc_Init has been called with incorrect configuration parameter (configuration pointer is NULL_PTR).</p> <p>DSADC_E_ALREADY_INITIALIZED: Error code is reported if the Dsadc_Init API is called while the DSADC driver is already in initialized state.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>DSADC_E_CLC_FAILURE: Error is reported when enabling/disabling 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. DSADC driver does not perform a NULL_PTR check on ConfigPtr, when DET is off. 2. Dsadc_StartCalibration API must be invoked by the user after the initialization is completed and before calling the Dsadc_StartModulation API . 3. Mcu_Init() should be called before calling this API. 4. Interrupts should be in a disabled state before calling this API.

2.3.3.2 Dsadc_DeInit**Table 221 Specification for Dsadc_DeInit API**

Syntax	<pre>void Dsadc_DeInit (void)</pre>	
Service ID	0x1B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

DSADC driver**Table 221 Specification for `Dsadc_DeInit` API (continued)**

Description	This API resets all SFRs of the EDSADC configured during initialization to their reset values. It sets DSADC module's state to DSADC_UNINIT_COMPLETED. It also disables the EDSADC hardware by writing into the CLC register. This API is available only when DsadcDeInitApi is configured as TRUE.
Source	IFX
Error handling	DET: DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization. Runtime Errors: None DEM: DSADC_E_CLC_FAILURE: Error is reported when enabling/disabling of CLC (module clock) fails. Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	DsadcDeInitApi
User hints	None

Dsadc_StartModulation**Table 222 Specification for `Dsadc_StartModulation` API**

Syntax	<pre>Std_ReturnType Dsadc_StartModulation (const Dsadc_ChannelType ChannelId)</pre>	
Service ID	0x1C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant for same channel. Reentrant for other channels	
Parameters (in)	ChannelId	Numeric Id of the requested DSADC Channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: DSADC channel result data acquisition is enabled E_NOT_OK: DSADC channel result data acquisition is disabled
Description	This API enables the result data acquisition for the given channel and enables the trigger source if it is configured. It sets the channel status to DSADC_BUSY.	

DSADC driver**Table 222 Specification for Dsadc_StartModulation API (continued)**

Source	IFX
Error handling	<p>DET:</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</p> <p>DSADC_E_BUSY: Error code is reported when the Dsadc_StartModulation API is called while the result acquisition is already started for the given channel.</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_CALIB_RUNNING: Error code is reported when the calibration algorithm is still running for the requested channel.</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	<p>In case of Trigger mode configured as Window,</p> <ol style="list-style-type: none"> 1. The data acquisition not started immediately after calling this function. 2. Data acquisition starts only after the window open event. 3. During window close event the result data acquisition will be stopped. <p>After calling the APIs Dsadc_Init or Dsadc_StartCalibration there must be a delay of 2 x Group delay shall be added before invoking the API Dsadc_StartModulation.</p>

Dsadc_StopModulation**Table 223 Specification for Dsadc_StopModulation API**

Syntax	<pre>Std_ReturnType Dsadc_StopModulation (const Dsadc_ChannelType ChannelId)</pre>	
Service ID	0x1D	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant for same channel, Reentrant for other channels	
Parameters (in)	ChannelId	Numeric Id of the requested DSADC Channel
Parameters (out)	-	-

DSADC driver**Table 223 Specification for Dsadc_StopModulation API (continued)**

Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: DSADC Channel Result data acquisition is stopped E_NOT_OK: DSADC Channel Result data acquisition is not stopped
Description	This API Disables the Result data acquisition for the given channel and disable the Trigger source if it is configured. It sets the Channel status to DSADC_IDLE.	
Source	IFX	
Error handling	<p>DET:</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</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	After calling this API, Result data acquisition is stopped immediately even though the window is in active state in case of Trigger mode window.	

2.3.3.5 Dsadc_ReadStreamResults**Table 224 Specification for Dsadc_ReadStreamResults API**

Syntax	<pre>Dsadc_SizeType Dsadc_ReadStreamResults (const Dsadc_ChannelType ChannelId, Dsadc_ResultType * const ResultLinearBufferPtr)</pre>	
Service ID	0x1E	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant for same channel, Reentrant for other channels	
Parameters (in)	ChannelId	Numeric Id of the requested DSADC Channel
Parameters (out)	-	-

DSADC driver**Table 224 Specification for `Dsadc_ReadStreamResults` API (continued)**

Parameters (in - out)	ResultLinearBufferPtr	Location to store the requested channel result.
Return	Dsadc_SizeType	65535 -> Read stream result failed. 0-> No Failure in the ReadStream Result, but no data available for read Other than 0 and 65535 -> Read stream result is successful and the return value indicate the size of valid data in the result Buffer.
Description	<p>This API reads the conversion results stored in the linear buffer. If the ResultLinearBufferPtr is not the channel Buffer what was configured in the Dsadc_SetupResultBuffer then this API copies the conversion results from channel buffer to the ResultLinearBufferPtr. This API returns the number of valid conversion results. The starting location of the conversion result is always 0 in the buffer.</p> <p>If the Parameter DsadcTimestampFeature is configured as DSADC_TIMESTAMP_ENABLED and the DsadcTriggerMode configured as TRIGGER_MODE_WINDOW then the first location of the buffer always contains the timestamp count for the window open event and the next value is the conversion result prior to window opening.</p> <p>Note: This API is available only if the DsadcAccessMode is not configured as DSADC_DMA_ACCESS for all the configured channels.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</p> <p>DSADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>DSADC_E_INV_LINEAR_BUFFER_CONFIG: Error code is reported when the access mode is not configured as Linear buffer and the API Dsadc_ReadStreamResults is called.</p> <p>DSADC_E_INVALID_BUFFER_POINTER: Error code is reported if the result buffer range is within the channel buffer range.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>DSADC_E_FIFO_FAILURE: Error is reported when hardware FIFO failure is detected.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	DsadcAccessMode	
User hints	Call this Function only when the Access Mode is configured as Linear buffer.	

DSADC driver**2.3.3.6 Dsadc_ReadResult****Table 225 Specification for Dsadc_ReadResult API**

Syntax	<pre>Std_ReturnType Dsadc_ReadResult (const Dsadc_ChannelType ChannelId, Dsadc_ResultType * const ResultPtr)</pre>	
Service ID	0x1F	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant for same channel, Reentrant for other channels	
Parameters (in)	ChannelId	Numeric Id of the requested DSADC Channel
Parameters (out)	-	-
Parameters (in - out)	ResultPtr	<p>Access mode single: Pointer to result data from the DSADC hardware result register</p> <p>Access mode circular Buffer: Pointer to result data from the circular buffer current read pointer location.</p> <p>In both modes if any error present data pointed by this pointer will not be updated. If there is no error but there is no data to read then data pointed by pointer will be updated with value 0.</p>
Return	Std_ReturnType	<p>E_OK: Requested DSADC channel result is read</p> <p>E_NOT_OK: Failed to read requested DSADC channel result.</p>
Description	<p>This API reads the result data for given DSADC channel. If the DsadcAccessMode is configured as DSADC_SINGLE_READ then this API read the DSADC hardware result register to update location pointed by ResultPtr. If the DsadcAccessMode is configured as DSADC_CIRCULAR_BUFFER then this API update the location pointed by ResultPtr with the circular Buffer data.</p> <p>Note: This API is available only if the DsadcAccessMode is not configured as DSADC_DMA_ACCESS for all the configured channels.</p>	
Source	IFX	

DSADC driver**Table 225 Specification for `Dsadc_ReadResult` API (continued)**

Error handling	<p>DET:</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</p> <p>DSADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>DSADC_E_INV_CIRCULAR_BUFFER_CONFIG: Error code is reported when the DsadcAccessMode is configured as DSADC_STREAM_LINEAR_BUFFER or DSADC_DMA_ACCESS</p> <p>DSADC_E_INVALID_BUFFER_POINTER: Error code is reported if the result buffer range is within the channel buffer range.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>DSADC_E_FIFO_FAILURE: Error is reported when hardware FIFO failure is detected.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	DsadcAccessMode
User hints	None

2.3.3.7 `Dsadc_GetStatus`**Table 226 Specification for `Dsadc_GetStatus` API**

Syntax	<pre>Dsadc_ChannelStatusType Dsadc_GetStatus (const Dsadc_ChannelType ChannelId)</pre>
Service ID	0x20
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Non Reentrant for same channel, Reentrant for other channels
Parameters (in)	ChannelId Numeric Id of the requested DSADC Channel
Parameters (out)	- -
Parameters (in-out)	- -

DSADC driver**Table 226 Specification for `Dsadc_GetStatus` API (continued)**

Return	Dsadc_ChannelStatusType	DSADC_IDLE: DSADC driver is in idle state. No action is performed. DSADC_BUSY: DSADC driver is processing the input signal. DSADC_RESULT_READY: DSADC driver is in ready state to read the converted results.
Description	This API returns the current status of the requested DSADC channel. In case of DET or SAFETY error the channel status is returned as DSADC_IDLE.	
Source	IFX	
Error handling	<p>DET: DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</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	when the DsadcResultHandlingImplementation is configured as DSADC_DMA_MODE for a channel, then the status DSADC_RESULT_READY will not be set by DSADC Driver. After the Dsadc_StartModulation API call and till the Dsadc_StopModulation API call, the channel status stays only at DSADC_BUSY	

2.3.3.8 `Dsadc_SetupResultBuffer`**Table 227 Specification for `Dsadc_SetupResultBuffer` API**

Syntax	Std_ReturnType Dsadc_SetupResultBuffer (const Dsadc_ChannelType ChannelId, const Dsadc_ResultType * const DataBufferPtr, const Dsadc_SizeType Size)
Service ID	0x21
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Non Reentrant for same channel, Reentrant for other channels

DSADC driver**Table 227 Specification for `Dsadc_SetupResultBuffer` API (continued)**

Parameters (in)	ChannelId DataBufferPtr Size	Numeric Id of the requested DSADC Channel Pointer to the start of result buffer(Channel buffer) for the requested channel. Result buffer size which defines the number of result values that can be stored in the result buffer. Maximum size of the buffer should be 65534.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Result buffer(Channel Buffer) initialization is successful E_NOT_OK: Result buffer Initialization failed
Description	<p>This API sets up the start address of Channel specific result buffers, where the conversion results will be stored. This API sets up the buffer only if the DsadcAccessMode is configured as DSADC_STREAM_LINEAR_BUFFER or DSADC_CIRCULAR_BUFFER.</p> <p>This API returns E_OK on successful initialization of result buffer.</p> <p>Note: This API is available only if the DsadcAccessMode is not configured as DSADC_DMA_ACCESS for all the configured channels.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>DSADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null-pointer as a parameter.</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</p> <p>DSADC_E_UNIDLE: Error code is reported when the current channel status is not DSADC_IDLE.</p> <p>DSADC_E_INVALID_BUFFER_CONFIG: Error code is reported when the configured DsadcAccessMode is neither DSADC_STREAM_LINEAR_BUFFER nor DSADC_CIRCULAR_BUFFER.</p> <p>DSADC_E_INV_BUFFER_SIZE: Error code is reported when the requested size of the buffer is zero or greater than the maximum buffer size.</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	DsadcAccessMode	
User hints	None	

DSADC driver**2.3.3.9 Dsadc_StartCarrierSignal****Table 228 Specification for Dsadc_StartCarrierSignal API**

Syntax	Std_ReturnType Dsadc_StartCarrierSignal (void)	
Service ID	0x22	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Carrier signal generation started successfully. E_NOT_OK: Failed to start the carrier signal generation.
Description	This API is used to start the generation of the carrier signal from the carrier generator based on the configured waveform properties for exciting the resolver coils. This interface returns E_OK on successful starting of the carrier signal.	
Source	IFX	
Error handling	DET: DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization. DSADC_E_CARRIER_ALREADY_RUNNING: Error code is reported when the Dsadc_StartCarrierSignal API is called but the carrier signal is already running. 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	

DSADC driver**2.3.3.10 Dsadc_StopCarrierSignal****Table 229 Specification for Dsadc_StopCarrierSignal API**

Syntax	Std_ReturnType Dsadc_StopCarrierSignal (void)	
Service ID	0x23	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Carrier signal generation stopped successfully. E_NOT_OK: Failed to stop the carrier signal generation.
Description	The interface is used to stop the generation of carrier signal from the carrier generator. This interface returns E_OK on successfully stopping of carrier signal	
Source	IFX	
Error handling	DET: DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module 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	Stopping of the carrier generator terminates the PWM output, after the completion of the current period.	

2.3.3.11 Dsadc_EnableNotifications**Table 230 Specification for Dsadc_EnableNotifications API**

Syntax	void Dsadc_EnableNotifications (const Dsadc_ChannelType ChannelId)
---------------	---

DSADC driver**Table 230 Specification for Dsadc_EnableNotifications API (continued)**

Service ID	0x24	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant for same channel, Reentrant for other channels	
Parameters (in)	ChannelId	Numeric Id of the requested DSADC Channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This API enables the notification mechanism for the requested DSADC Channel.	
Source	IFX	
Error handling	<p>DET:</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</p> <p>DSADC_E_NOTIF_CAPABILITY: Error code is reported when the enable/disable notification function for a channel is called but there is no notification function configured for that channel.</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	Any one of the Channel notification function should not be a NULL Pointer	

2.3.3.12 Dsadc_DisableNotifications**Table 231 Specification for Dsadc_DisableNotifications API**

Syntax	void Dsadc_DisableNotifications (const Dsadc_ChannelType ChannelId)
Service ID	0x25
Sync/Async	Synchronous

DSADC driver**Table 231 Specification for `Dsadc_DisableNotifications` API (continued)**

ASIL Level	B	
Re-entrancy	Non Reentrant for same channel, Reentrant for other channels	
Parameters (in)	ChannelId	Numeric Id of the requested DSADC Channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This API disables the notification mechanism for the requested DSADC Channel.	
Source	IFX	
Error handling	<p>DET:</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</p> <p>DSADC_E_NOTIF_CAPABILITY: Error code is reported when the enable/disable notification function for a channel is called but there is no notification function configured for that channel.</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	Any one of the Channel notification function should not be a NULL Pointer	

2.3.3.13 Dsadc_GetTimestamp**Table 232 Specification for `Dsadc_GetTimestamp` API**

Syntax	Dsadc_TimeStampType Dsadc_GetTimestamp (const Dsadc_ChannelType ChannelId)
Service ID	0x26
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Non Reentrant for same channel, Reentrant for other channels

DSADC driver**Table 232 Specification for Dsadc_GetTimestamp API (continued)**

Parameters (in)	ChannelId	Numeric Id of the requested DSADC Channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Dsadc_TimeStampType	Timestamp count for the last read result event. In case of error value 0 will be returned.
Description	This API returns the timestamp count value for the Dsadc_ReadResult API read result event. This timestamp count is the time from the HW result event till the Dsadc_ReadResult API reads the result value from the HW Result register.	
Source	IFX	
Error handling	<p>DET:</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</p> <p>DSADC_E_SINGLE_ACCESSMODE_TIMESTAMP: Error code is reported When Dsadc_GetTimestamp API is invoked and if the DsadcAccessMode is not configured as DSADC_SINGLE_READ or if the DsadcTimestampFeature is DSADC_TIMESTAMP_DISABLED or if the DsadcTriggerMode is DSADC_TRIGGER_MODE_WINDOW.</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	

2.3.3.14 Dsadc_StartCalibration**Table 233 Specification for Dsadc_StartCalibration API**

Syntax	Std_ReturnType Dsadc_StartCalibration (const Dsadc_ChannelType ChannelId)
Service ID	0x27
Sync/Async	Synchronous
ASIL Level	B

DSADC driver**Table 233 Specification for `Dsadc_StartCalibration` API (continued)**

Re-entrancy	Non Reentrant for same channel, Reentrant for other channels	
Parameters (in)	ChannelId	Numeric Id of the requested DSADC Channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Calibration algorithm started successfully E_NOT_OK: Failed to start the calibration algorithm
Description	<p>This API triggers the calibration algorithm. The calibration algorithm will be triggered only when the current status of the channel is DSADC_IDLE.</p> <p>This interface returns E_OK on successful start of the calibration algorithm</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before the module initialization.</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</p> <p>DSADC_E_UNIDLE: Error code is reported when the current channel status is not DSADC_IDLE.</p> <p>DSADC_E_CALIB_RUNNING: Error code is reported when the calibration algorithm is still running for the requested channel.</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 Calibration related parameters must be valid for the successful calibration.	

2.3.3.15 Dsadc_GetCalibrationStatus**Table 234 Specification for `Dsadc_GetCalibrationStatus` API**

Syntax	<code>Dsadc_CalibrationStatusType Dsadc_GetCalibrationStatus</code> (<code>const Dsadc_ChannelType ChannelId</code>)
Service ID	0x28
Sync/Async	Synchronous

DSADC driver**Table 234 Specification for `Dsadc_GetCalibrationStatus` API (continued)**

ASIL Level	B	
Re-entrancy	Non Reentrant for same channel, Reentrant for other channels	
Parameters (in)	ChannelId	Numeric Id of the requested DSADC Channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Dsadc_CalibrationStatusType	<p>DSADC_CALIBRATION_NOT_STARTED: DSADC driver calibration is not yet started for the given channel.</p> <p>DSADC_CALIBRATION_DONE: DSADC driver calibration is done for the given channel.</p> <p>DSADC_CALIBRATION_RUNNING: DSADC driver calibration is currently running for the given channel.</p> <p>DSADC_CALIBRATION_ERROR: DSADC driver calibration is failed for the given channel.</p>
Description	<p>This API returns the current calibration status for the given channel. In case of DET or SAFETY error the status DSADC_CALIBRATION_NOT_STARTED will be returned.</p> <p>when the Calibration status is DSADC_CALIBRATION_DONE then the status will be changed to DSADC_CALIBRATION_NOT_STARTED. So that when this function is called next time DSADC_CALIBRATION_NOT_STARTED is returned.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>DSADC_E_PARAM_CHANNEL: Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).</p> <p>DSADC_E_UNINIT: Error code is reported if the API service is invoked before 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>	
Configuration dependencies	-	
User hints	<p>Once the calibration is done and if it is success then this API will return DSADC_CALIBRATION_DONE. If this API is called again before calling Dsadc_StartCalibration API then it will return the calibration status as DSADC_CALIBRATION_NOT_STARTED.</p>	

DSADC driver**2.3.3.16 Dsadc_InitCheck****Table 235 Specification for Dsadc_InitCheck API**

Syntax	<pre>Std_ReturnType Dsadc_InitCheck (const Dsadc_ConfigType * const ConfigPtr)</pre>	
Service ID	0x29	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the DSADC Driver configuration structure
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : Initialization check passed E_NOT_OK : Initialization check failed
Description	<p>This API verifies the DSADC Module Initialization.</p> <p>This API returns E_NOT_OK when configuration pointer is NULL or DSADC driver is not initialized.</p> <p>Note: This API is available only when the parameter DsadcInitCheckApi is configured as TRUE.</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	DsadcInitCheckApi	
User hints	<p>The DSADC module environment should ensure the following calling sequence for this API.</p> <ol style="list-style-type: none"> 1. Dsadc_Init API is called. 2. Dsadc_Initcheck API is called. 	

DSADC driver**2.3.3.17 Dsadc_GetVersionInfo****Table 236 Specification for Dsadc_GetVersionInfo API**

Syntax	<pre>void Dsadc_GetVersionInfo (Std_VersionInfoType * const versioninfo)</pre>	
Service ID	0x2C	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where to store the version information of the DSADC driver.
Parameters (in - out)	-	-
Return	void	-
Description	<p>API returns the version information of this driver.</p> <p>Note: This API is available only when DsadcVersionInfoApi is configured as TRUE.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>DSADC_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	DsadcVersionInfoApi	
User hints	None	

2.3.4 Notifications and Callbacks

This section lists all the callbacks of the driver. The driver does not support any notifications from other drivers.

DSADC driver**2.3.4.1 Dsadc_TimerIsr****Table 237 Specification for Dsadc_TimerIsr API**

Syntax	<pre>void Dsadc_TimerIsr (const Dsadc_ChannelType ChannelId, const uint32 StatusFlags)</pre>	
Service ID	0x2B	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	ChannelId StatusFlags	Numeric Id of the DSADC Channel Source of the ISR: For GTM: CCU0 (1) or CCU1 (2) For ERU: Pattern match (1) or Pattern miss (0)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Handles the interrupt from GTM and ERU for window events(Open and Close).</p> <p>Note: This Callback function is available only if the DsadcTriggerSelect is configured as GTM or ERU resource for at least any one of the DSADC channel.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>DSADC_SE_INVALID_ISR: Error is reported if unintended interrupt is triggering the ISR.</p> <p>DSADC_SE_EARLY_WINDOW_ISR: Safety Error shall be reported when Timer Isr is Called before the Dsadc_StartModulation invoked.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	DsadcTriggerSelect	
User hints	<p>Setting of SRC Register for the corresponding ERU and GTM resource must be handled by the OS/Application.</p> <p>This function shall be called by application/user during the CCU0 and CCU1 interrupt event if the GTM timer for the DSADC trigger is handled by application. User shall not call this function if the IFX MCAL PWM driver is used to handle the GTM timer for the DSADC trigger.</p>	

DSADC driver**2.3.5 Scheduled functions**

The driver does not support any scheduled functions.

2.3.6 Interrupt service routines

This section lists all the interrupt handlers of the driver.

2.3.6.1 Dsadc_Isr**Table 238 Specification for Dsadc_Isr API**

Syntax	<pre>void Dsadc_Isr (const Dsadc_ChannelType HwChannelId)</pre>	
Service ID	0x2A	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel	
Parameters (in)	HwChannelId	Hardware channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Handles the interrupts from Main Service Request for the given DSADC HW Channel Id.</p> <p>Note: This ISR is available only if the DsadcAccessMode is not configured as DSADC_DMA_ACCESS for all the configured channels.</p>	
Source	IFX	
Error handling	<p>DET: None</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>DSADC_SE_INVALID_ISR: Error is reported if unintended interrupt is triggering the ISR.</p> <p>DSADC_SE_PARAM_HW_CHANNEL: Error code is reported if the HW Channel ID passed is not configured.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	DsadcAccessMode	

DSADC driver
Table 238 Specification for `Dsadc_Isr` API (continued)

User hints	User must call this interrupt handler from the ISR of DSADCxSRGM of each channel and pass the HW Channel Id as the parameter.
-------------------	---

2.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

2.3.7.1 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 239 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
Error code is reported if the result buffer range is within the channel buffer range.	IFX	DSADC_E_INVALID_BUFFER_POINTE R=0x10	<code>Dsadc_ReadResult</code> , <code>DsadcReadStreamResults</code>
Error code is reported when the requested size of the buffer is zero or greater than the maximum buffer size.	IFX	DSADC_E_INV_BUFFER_SIZE=0x0F	<code>Dsadc_SetupResultBuffer</code>
Error code is reported When <code>Dsadc_GetTimestamp</code> API is invoked and if the <code>DsadcAccessMode</code> is not configured as <code>DSADC_SINGLE_READ</code> or if the <code>DsadcTimestampFeature</code> is <code>DSADC_TIMESTAMP_DISABLED</code> or if the <code>DsadcTriggerMode</code> is <code>DSADC_TRIGGER_MODE_WINDOW</code> .	IFX	DSADC_E_SINGLE_ACCESSMODE_TIMESTAMP=0x0D	<code>Dsadc_GetTimestamp</code>
Error code is reported if <code>Dsadc_Init</code> has been called with incorrect configuration parameter (configuration pointer is <code>NULL_PTR</code>).	IFX	DSADC_E_PARAM_CONFIG=0x01	<code>Dsadc_Init</code>
Error code is reported if the <code>Dsadc_Init</code> API is called while the DSADC driver is already in initialized state.	IFX	DSADC_E_ALREADY_INITIALIZED=0x02	<code>Dsadc_Init</code>

DSADC driver**Table 239 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
Error code is reported if the API service is invoked before the module initialization.	IFX	DSADC_E_UNINIT=0x03	Dsadc_GetCalibrationStatus, Dsadc_StartCalibration, Dsadc_GetTimestamp, Dsadc_DisableNotifications, Dsadc_EnableNotifications, Dsadc_StopCarrierSignal, Dsadc_StartCarrierSignal, Dsadc_SetupResultBuffer, Dsadc_GetStatus, Dsadc_ReadResult, DsadcReadStreamResults, Dsadc_StopModulation, Dsadc_StartModulation, Dsadc_DeInit
Error code is reported when the Dsadc_StartModulation API is called while the result acquisition is already started for the given channel.	IFX	DSADC_E_BUSY=0x04	Dsadc_StartModulation
Error code is reported if the input channel ID is an invalid channel identifier i.e. its value is either equal to or greater than the number of DSADC channels configured (channel ID value starts from 0).	IFX	DSADC_E_PARAM_CHANNEL=0x05	Dsadc_GetCalibrationStatus, Dsadc_StartCalibration, Dsadc_GetTimestamp, Dsadc_DisableNotifications, Dsadc_EnableNotifications, Dsadc_SetupResultBuffer, Dsadc_GetStatus, Dsadc_ReadResult, DsadcReadStreamResults, Dsadc_StopModulation, Dsadc_StartModulation
Error code is reported if the API is invoked with null-pointer as a parameter.	IFX	DSADC_E_PARAM_POINTER=0x06	Dsadc_GetVersionInfo, Dsadc_SetupResultBuffer, Dsadc_ReadResult, DsadcReadStreamResults

DSADC driver**Table 239 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
Error code is reported when the access mode is not configured as Linear buffer and the API Dsadc_ReadStreamResults is called.	IFX	DSADC_E_INV_LINEAR_BUFFER_CONFIG=0x07	Dsadc_ReadStreamResults
Error code is reported when the DsadcAccessMode is configured as DSADC_STREAM_LINEAR_BUFFER or DSADC_DMA_ACCESS	IFX	DSADC_E_INV_CIRCULAR_BUFFER_CONFIG=0x08	Dsadc_ReadResult
Error code is reported when the configured DsadcAccessMode is neither DSADC_STREAM_LINEAR_BUFFER nor DSADC_CIRCULAR_BUFFER.	IFX	DSADC_E_INVALID_BUFFER_CONFIG=0x09	Dsadc_SetupResultBuffer
Error code is reported when the current channel status is not DSADC_IDLE.	IFX	DSADC_E_UNIDLE=0x0A	Dsadc_SetupResultBuffer, Dsadc_StartCalibration
Error code is reported when the Dsadc_StartCarrierSignal API is called but the carrier signal is already running.	IFX	DSADC_E_CARRIER_ALREADY_RUNNING=0x0B	Dsadc_StartCarrierSignal
Error code is reported when the enable/disable notification function for a channel is called but there is no notification function configured for that channel.	IFX	DSADC_E_NOTIF_CAPABILITY=0x0C	Dsadc_DisableNotifications, Dsadc_EnableNotifications
Error code is reported when the calibration algorithm is still running for the requested channel.	IFX	DSADC_E_CALIB_RUNNING=0xE	Dsadc_StartModulation, Dsadc_StartCalibration

2.3.7.2 Production errors

The following table lists all the production errors reported by the driver.

DSADC driver

Table 240 Description of production errors reported

Description	Source	Error code and value	Applicable APIs
Error is reported when enabling/disabling of CLC (module clock) fails.	IFX	DSADC_E_CLC_FAILURE=Assigned by DEM	Dsadc_DelInit, Dsadc_Init
Error is reported when hardware FIFO failure is detected.	IFX	DSADC_E_FIFO_FAILURE=Assigned by DEM	Dsadc_ReadResult, DsadcReadStreamResults

2.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 241 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
Safety Error shall be reported when Timer Lsr is Called before the Dsadc_StartModulation invoked.	IFX	DSADC_SE_EARLY_WINDOW_ISR=0x13	Dsadc_TimerLsr
Error code is reported if the HW Channel ID passed is not configured.	IFX	DSADC_SE_PARAM_HW_CHANNEL=0x12	Dsadc_Lsr
Error is reported if unintended interrupt is triggering the ISR.	IFX	DSADC_SE_INVALID_ISR=0x11	Dsadc_Lsr, Dsadc_TimerLsr

2.3.7.4 Runtime errors

The driver does not report any runtime errors.

2.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

2.3.8.1 Deviations

The section describes the deviations from software specification.

Table 242 Known deviations

Reference	Deviation
Address and Data CRC in the DMA mode	Since the Data CRC and Address CRC features of DMA are not used for DSADC 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.

DSADC driver
Table 242 Known deviations (continued)

Reference	Deviation
Spurious interrupt reporting when the channel is in inactive state	The DSADC driver cannot report the occurrence of a spurious interrupt when the channel is in the inactive state [that is, before invoking the Dsadc_StartModulation() API and after invoking the Dsadc_StopModulation() API].

2.3.8.2 Limitations

The section describes the limitations from software specification.

Table 243 Known limitations

Reference	Limitation
Channel has trigger mode window and timestamp enabled	The last two conversion results before the window close events are lost due to the hardware limitation.
Input parameter of Dsadc_InitCheck is used only to check CLC register.	Input parameter of Dsadc_Initcheck() API is used only to check CLC register. To check all other registers, parameter used during Dsadc_Init() API is used for Initcheck evaluation.

2.3.9 Unsupported hardware features

This section lists all the AURIX EDSADC hardware features that are not supported by the driver. These features are out of scope of the DSADC driver specification.

- External modulator support
- Automatic power control
- Trigger signal from port pin
- Trigger signal from GTM ADC trigger lines

FLSLOADER driver

3 FLSLOADER driver

3.1 User information

3.1.1 Description

Two types of non-volatile memory (NVM) are instantiated in the 2nd Generation AURIX™ (TC3xx) microcontroller.

- Program Flash (PFlash) stores the program code and constant data
- Data Flash (DFlash) stores the application-specific data

The FLSLOADER driver provides the following services:

- Initialization and de-initialization of the Flash
- Writing the program and data to the Flash
- Erasing the contents of the Flash
- Locking and unlocking the Flash

These services of the driver are operable on DFlash bank 0 and all PFlash banks of the microcontroller. All references to DFlash and PFlash, in this section, are meant for bank 0 of the DFlash and all banks of the PFlash, respectively. The driver is delivered as a pre-compile variant. Therefore, the driver supports configuration parameters with only pre-compile configuration class.

3.1.2 Hardware-software mapping

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

FLSLOADER driver

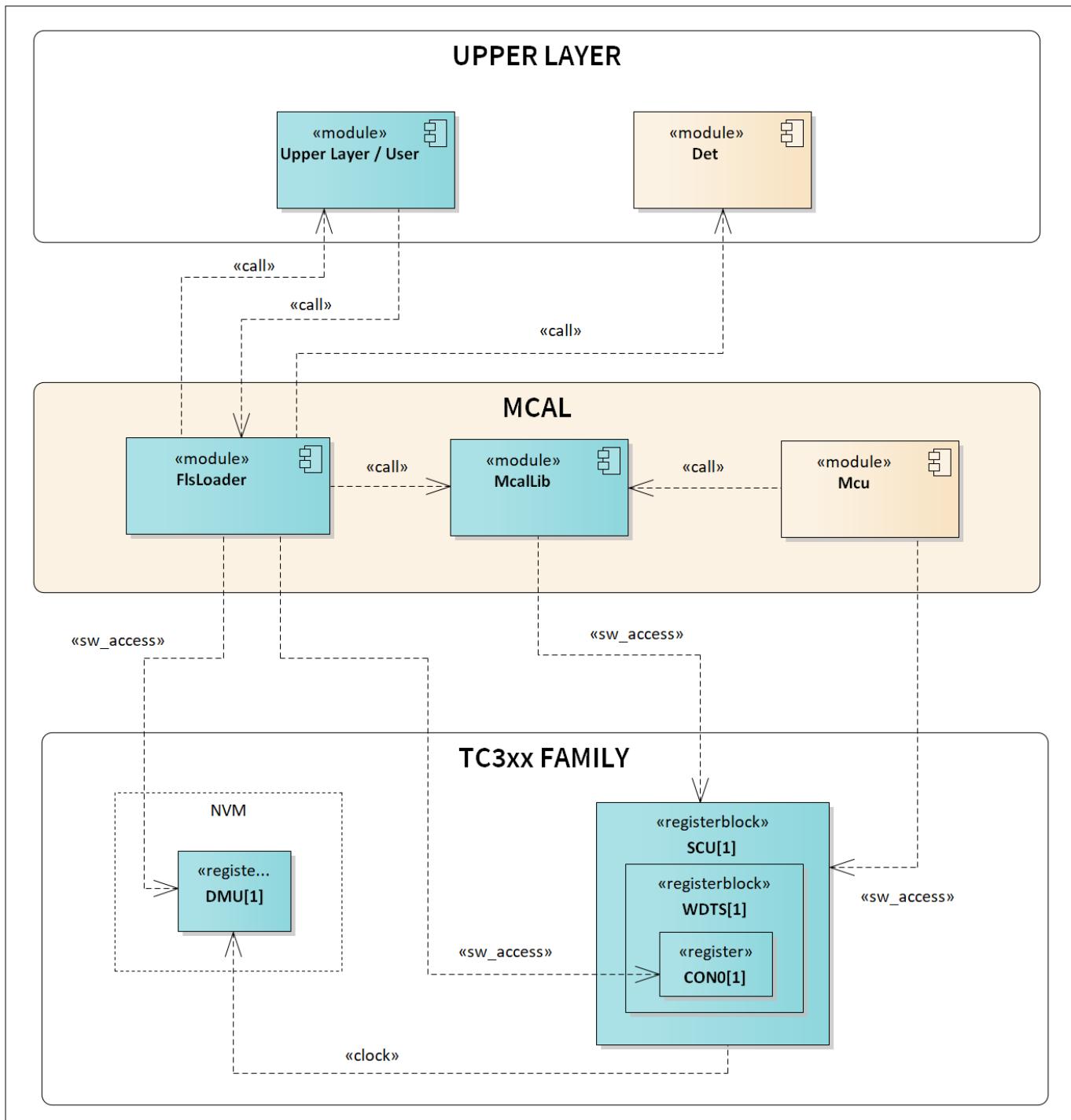


Figure 26 Mapping of hardware-software interfaces

3.1.2.1 DMU: primary hardware peripheral

Hardware functional features

FLSLOADER driver

The FLSLOADER driver uses the DMU for various operations on the PFlash/DFlash memories. The key hardware functional features used by the driver are:

- Write and erase to PFlash:
 - 32 bytes page programming and 256 bytes burst programming
 - Erase by multi-sector erase commands
- Write and erase to DFlash0 and UCB:
 - 8 bytes page programming and 32 bytes burst programming
 - Erase by multi-sector erase commands
 - Single ended mode support for DFlash0
 - Password based protection of PFlash/DFlash0 banks through UCBs

The unsupported features of the DMU are:

- Compliment sensing mode for DFlash0
- Suspend and resume operations
- ECC error reporting to safety management unit (SMU)
- Interrupt service requests

Users of the hardware

The FLSLOADER and FLS drivers utilize the DMU module. FLSLOADER driver is used during the boot and FLS driver is used during the runtime. Therefore, access to the DMU registers is not concurrent.

Hardware diagnostic features

The SMU alarms configured for the DMU are not monitored by the FLSLOADER driver.

Hardware events

The FLSLOADER driver uses the following hardware events for error event from the DMU IP:

- 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 hardware when write or erase command executed on protected memory section
- Operation Error (OPER): This flag is set by hardware when Flash standard interface (FSI) encounters any error
- Sequence Error (SQER): This flag is set by hardware when improper DMU command sequences are executed

3.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The FLSLOADER driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB, fSRI and fFSI 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. The FLSLOADER driver accesses the SCU register to disable the Safety Endinit protection temporarily for PFlash write and erase operations. Since the driver is used during the boot, access to the SCU register will be not be concurrent with other drivers in runtime.

Hardware diagnostic features

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

Hardware events

Hardware events from the SCU are not used by the FLSLOADER driver.

FLSLOADER driver

3.1.3 File structure

3.1.3.1 C file structure

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

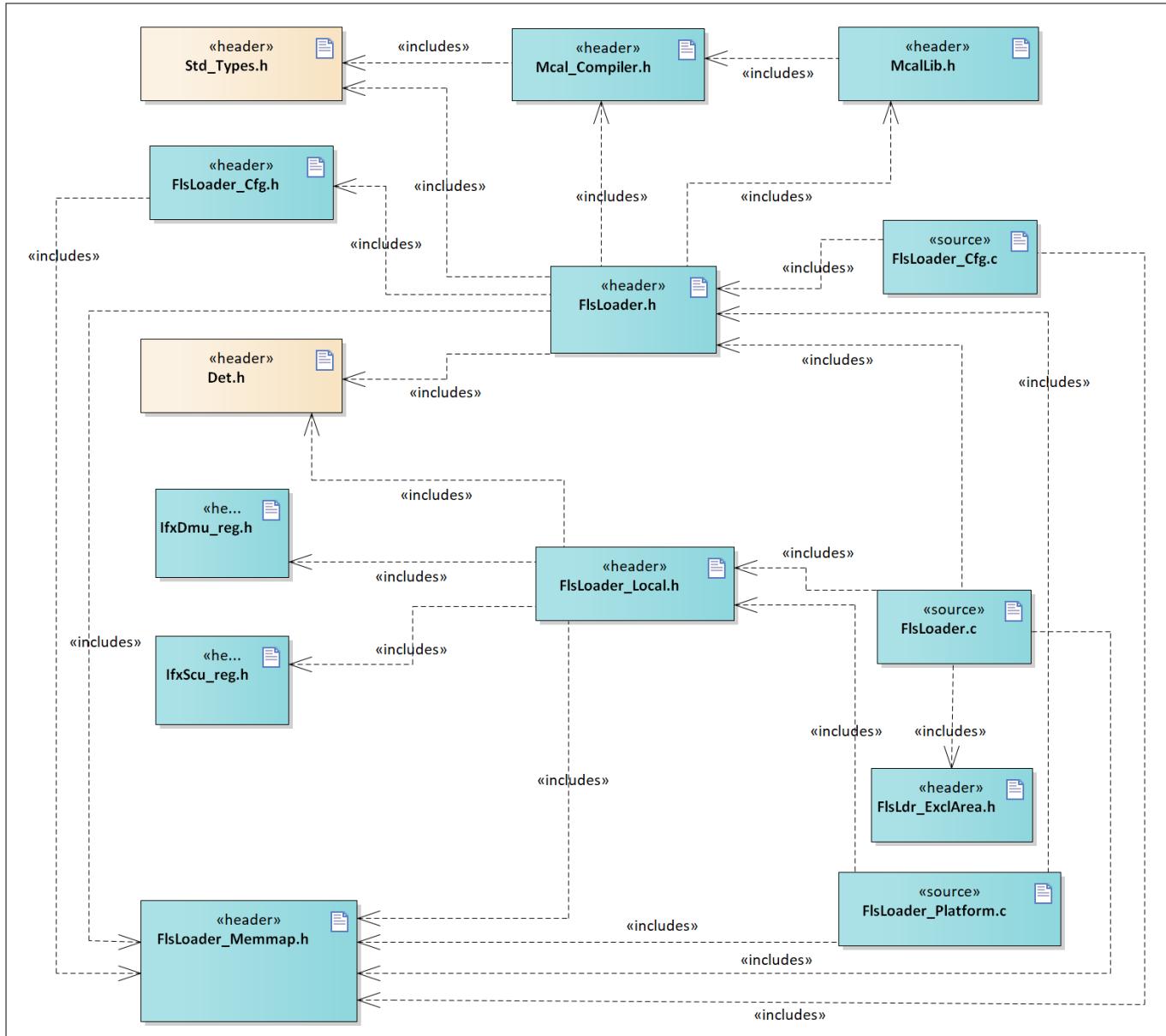


Figure 27 C file structure

Table 244 C file structure

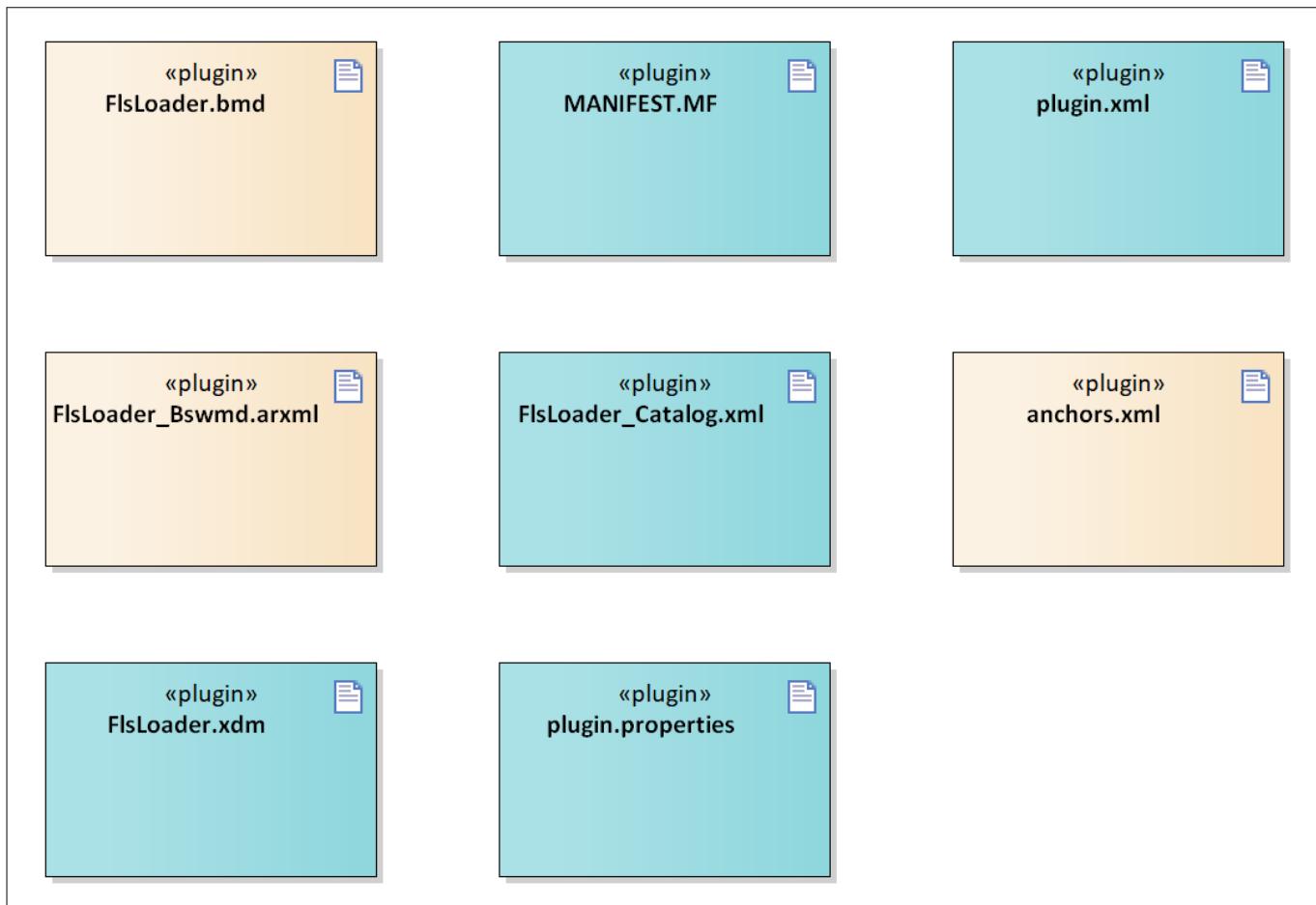
File name	Description
Det.h	Provides the exported interfaces of Development Error Tracer
FlsLoader.c	Implementation of FLSLOADER driver functionality
FlsLoader.h	FLSLOADER driver header definition file
FlsLoader_Cfg.c	Pre-compile configuration data file for the FLSLOADER driver functionality

FLSLOADER driver
Table 244 C file structure (continued)

File name	Description
FlsLoader_Cfg.h	FLSLOADER driver configuration generated out of ECUC file
FlsLdr_ExclArea.h	Header file containing the prototype of the APIs to define the start and the end of an exclusive area of FLSLOADER module.
FlsLoader_Local.h	FLSLOADER driver local header definition file
FlsLoader_Memmap.h	Mapping of code and data (variables, constant variables) used by FLSLOADER driver to specific memory sections
FlsLoader_Platform.c	FLSLOADER driver platform-specific source file
IfxDmu_reg.h	SFR header file for Dmu
IfxScu_reg.h	SFR header file for SCU
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
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 FLSLOADER driver.

FLSLOADER driver

Figure 28 Code generator plugin files
Table 245 Code generator plugin files

File name	Description
FlsLoader.bmd	Code template macro file for FLSLOADER driver
FlsLoader.xdm	Tresos format XML data model schema file
FlsLoader_Bswmd.arxml	AUTOSAR format module description file
FlsLoader_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the metadata for the FLSLOADER driver
anchors.xml	AUTOSAR format module description file
plugin.properties	Tresos plugin support file for the FLSLOADER driver
plugin.xml	Tresos plugin support file for the FLSLOADER driver

3.1.4 Integration hints

This section describes the key points that the integrator or user of the FLSLOADER driver must consider.

FLSLOADER driver
Flash reprogramming

- Complete driver code or write and erase APIs can be placed in the RAM or in another Flash bank for which Flash operations are not being executed.
For example, if the erase or write operations need to be executed on PFlash bank0 sectors, place the code in either PFlash bank 1 to 5 or the RAM.
- The driver does not provide any APIs or functions for moving the write and erase APIs to the RAM. User of this API has to take care in the application.

Safety watchdog configuration

- Application shall configure the safety watchdog timer in the static and time independent password mode.

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 FLSLOADER driver.

- **EcuM**

The FLSLOADER driver is designed for the boot loader application, which may not contain EcuM.

Therefore, the application software must ensure that the initialization and de-initialization of the FLSLOADER driver are invoked before and after using its services.

- **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 `FlsLoader_MemMap.h` file.

The `FlsLoader_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

FLSLOADER driver

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

```
#define MEMMAP_ERROR

/** GLOBAL RAM DATA **/
#if defined FLSLOADER_START_SEC_VAR_CLEARED_QM_LOCAL_8
/* User pragmas here for DSPr of CPU core *****/
#undef FLSLOADER_START_SEC_VAR_CLEARED_QM_LOCAL_8
#undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
/* User pragmas here for DSPr of CPU core *****/
#undef FLSLOADER_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
#undef MEMMAP_ERROR
#elif defined FLSLOADER_START_SEC_VAR_CLEARED_QM_LOCAL_32
/* User pragmas here for DSPr of CPU core *****/
#undef FLSLOADER_START_SEC_VAR_CLEARED_QM_LOCAL_32
#undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_VAR_CLEARED_QM_LOCAL_32
/* User pragmas here for DSPr of CPU core *****/
#undef FLSLOADER_STOP_SEC_VAR_CLEARED_QM_LOCAL_32
#undef MEMMAP_ERROR

/** CONSTANT DATA **/
#elif defined FLSLOADER_START_SEC_CONST_QM_LOCAL_8
/* User pragmas here for PFlash*****/
#undef FLSLOADER_START_SEC_CONST_QM_LOCAL_8
#undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_CONST_QM_LOCAL_8
/* User pragmas here for PFlash*****/
#undef FLSLOADER_STOP_SEC_CONST_QM_LOCAL_8
#undef MEMMAP_ERROR
#elif defined FLSLOADER_START_SEC_CONST_QM_LOCAL_32
/* User pragmas here for PFlash*****/
#undef FLSLOADER_START_SEC_CONST_QM_LOCAL_32
#undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_CONST_QM_LOCAL_32
/* User pragmas here for PFlash*****/
#undef FLSLOADER_STOP_SEC_CONST_QM_LOCAL_32
#undef MEMMAP_ERROR

/** CODE **/
#elif defined FLSLOADER_START_SEC_CODE_QM_LOCAL
/* User pragmas here for PFlash*****/
#undef FLSLOADER_START_SEC_CODE_QM_LOCAL
#undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_CODE_QM_LOCAL
/* User pragmas here for PFlash*****/
#undef FLSLOADER_STOP_SEC_CODE_QM_LOCAL
#undef MEMMAP_ERROR
#elif defined FLSLOADER_START_SEC_WRITEERASE_CODE_QM_LOCAL
/* User pragmas here for PFlash*****/
#undef FLSLOADER_START_SEC_WRITEERASE_CODE_QM_LOCAL

```

FLSLOADER driver

```

#define MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_WRITEERASE_CODE_QM_LOCAL
/******User pragmas here for PFlash*****/
#define FLSLOADER_STOP_SEC_WRITEERASE_CODE_QM_LOCAL
#define MEMMAP_ERROR
#endif

#if defined MEMMAP_ERROR
#error "Flsloader_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 FLSLOADER driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the FLSLOADER 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 the integration of the FLSLOADER driver.

- **SchM**

The SchM module is not required for the integration of the FLSLOADER driver.

- **Safety error**

The FLSLOADER driver does not report safety errors.

- **Callout**

The FLSLOADER driver provides optional callout function to the application while looping for hardware busy status during write and erase operations.

To enable the callout function, user shall define callout function name using the `FlsLoaderCallOutFunction` configuration and a timeout interval using the `FlsLoaderCallOutTime` configuration.

If enabled, the callout function is invoked at configured timeout interval during FLSLOADER write and erase operations. Application shall define the callout function in application software.

The following sequence diagram shows callout functionality during erase operation. Similarly callout shall be invoked by the driver during the write operations.

FLSLOADER driver

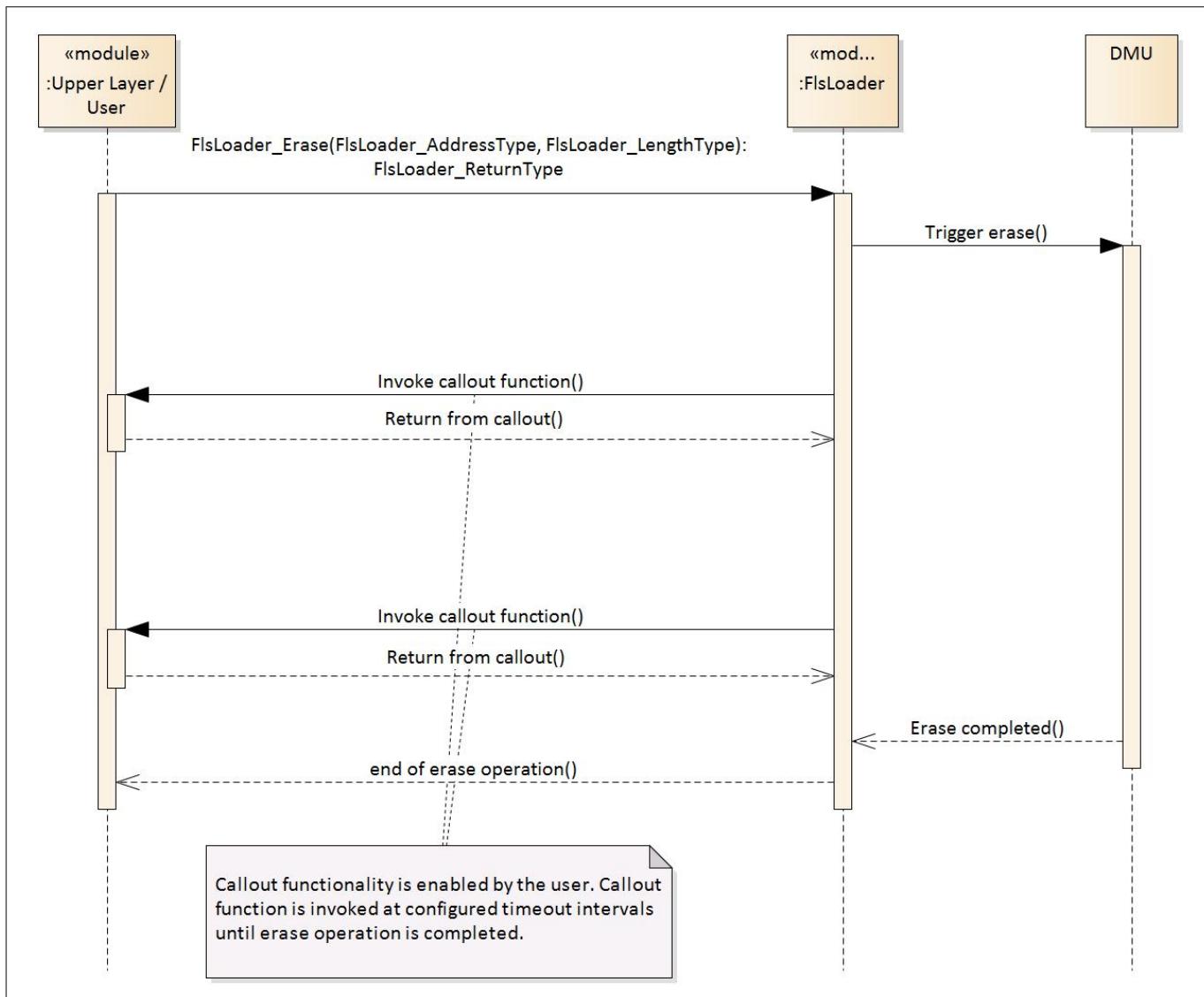


Figure 29 Erase operation with callout enabled

- **Operating system (OS)**

The OS is not required for the integration of the FLSLOADER driver.

- **Notifications and callbacks**

The FLSLOADER driver does not provide callbacks and notifications.

3.1.4.2 Multicore and Resource Manager

The FLSLOADER driver does not support execution on multiple cores in simultaneously.

3.1.4.3 MCU support

The FLSLOADER driver is dependent on the MCU driver for clock configuration. The initialization of the FLSLOADER driver must be started only after completing the MCU initialization.

3.1.4.4 Port support

The FLSLOADER driver does not use services provided by the Port driver.

FLSLOADER driver**3.1.4.5 DMA support**

The FLSLOADER driver does not use services provided by the DMA driver.

3.1.4.6 Interrupt connections

The FLSLOADER driver does not use any interrupt source. All write and erase functions are implemented using the polling method.

FLSLOADER driver

3.1.4.7 Example usage

This section describes how the FLSLOADER driver can be configured and how to use different APIs provided by it. All APIs should be provided with valid input parameters. To detect the invalid function parameters, DET (Development Error Tracer) should be enabled. Behavior of APIs undefined if DET is disabled and wrong parameters passed.

Driver APIs are designed to be non-reentrant, which means if there is an ongoing Flash operation and then an interrupt occurs or driver invokes its callout function if enabled, the driver APIs should not be called again within this callout function or interrupt context.

For more details on program or data Flash wait cycle configuration refer to the hardware user manual section “Configuring Flash Read Access Cycles”.

Configuration

Configuration of the FLSLOADER driver involves the following steps:

- Configuration to generate system clock of the required frequency. This configuration is done using the MCU driver.
- General guidelines for configuration of the FLSLOADER driver.
 - Lock check is an optional functionality for erase and write APIs and can be switched off.
 - For write and erase of Flash a minimum set of configuration without DET, Deinit, Lock and Unlock can be configured.

Refer to section 1.3 for all EB tresos configuration interfaces of the FLSLOADER driver.

Initialization of the driver

Initialize the MCU driver so that system clock is up and running. Initialize the FLSLOADER driver by calling the FlsLoader_Init API with the NULL pointer.

The following code listing shows FLSLOADER driver initialization.

```

/* MCU driver initialization */
Mcu_Init(ConfigPtr);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Initialize the FLSLOADER driver */
retval = FlsLoader_Init(NULL_PTR);

/* Check if driver initialized successfully */
if (retval == FLSLOADER_E_OK)
{
    /* FlsLoader_Init returned FLSLOADER_E_OK. Driver initialized successfully.*/
}

```

Erase and write PFlash and DFlash0

PFlash and DFlash0 are erased sector wise and are written page wise.

For erasing the Flash, erase API should be called with start address of a sector in Flash and number of sectors to be erased.

For writing the Flash, write API should be called with start address of a page in Flash, number of bytes to be written along with source data address. Number of bytes to be written should be multiple of the page size.

Refer to the hardware user manual sections **Program Flash Banks** and **Data Flash Bank DFLASH0** regarding more details on structure of PFlash and DFlash0 banks.

FLSLOADER driver

The driver provides an optional lock-check functionality for erase and write APIs, which when enabled checks if a sector or bank falling under current erase or write request is protected and it exits from the API safely if sector or bank found to be protected. User can enable this functionality by enabling the configuration `FlsLoaderEnableLockCheck`.

The driver provides another optional callout functionality for erase and write APIs, which when enabled invokes callout function to application at regular time intervals. Configuration and details of callout functionality is explained under **Callout** subsection of **Integration with AUTOSAR stack** section.

If a user requirement is to enable lock-check and callout functionalities with callout function name and callout time configured as `FlsLoader_LoopCallOut` and `5ms` respectively, following configuration should be made in configuration tool is shown as follows.

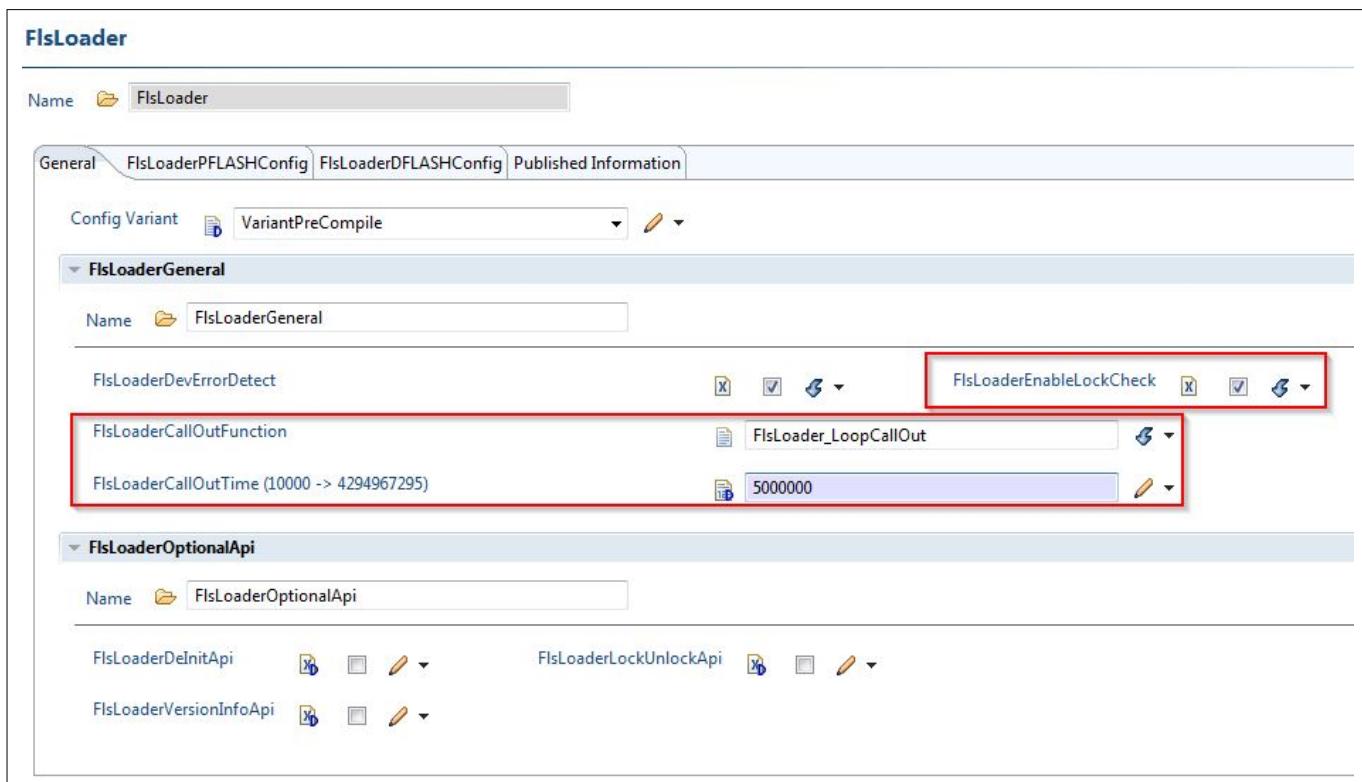
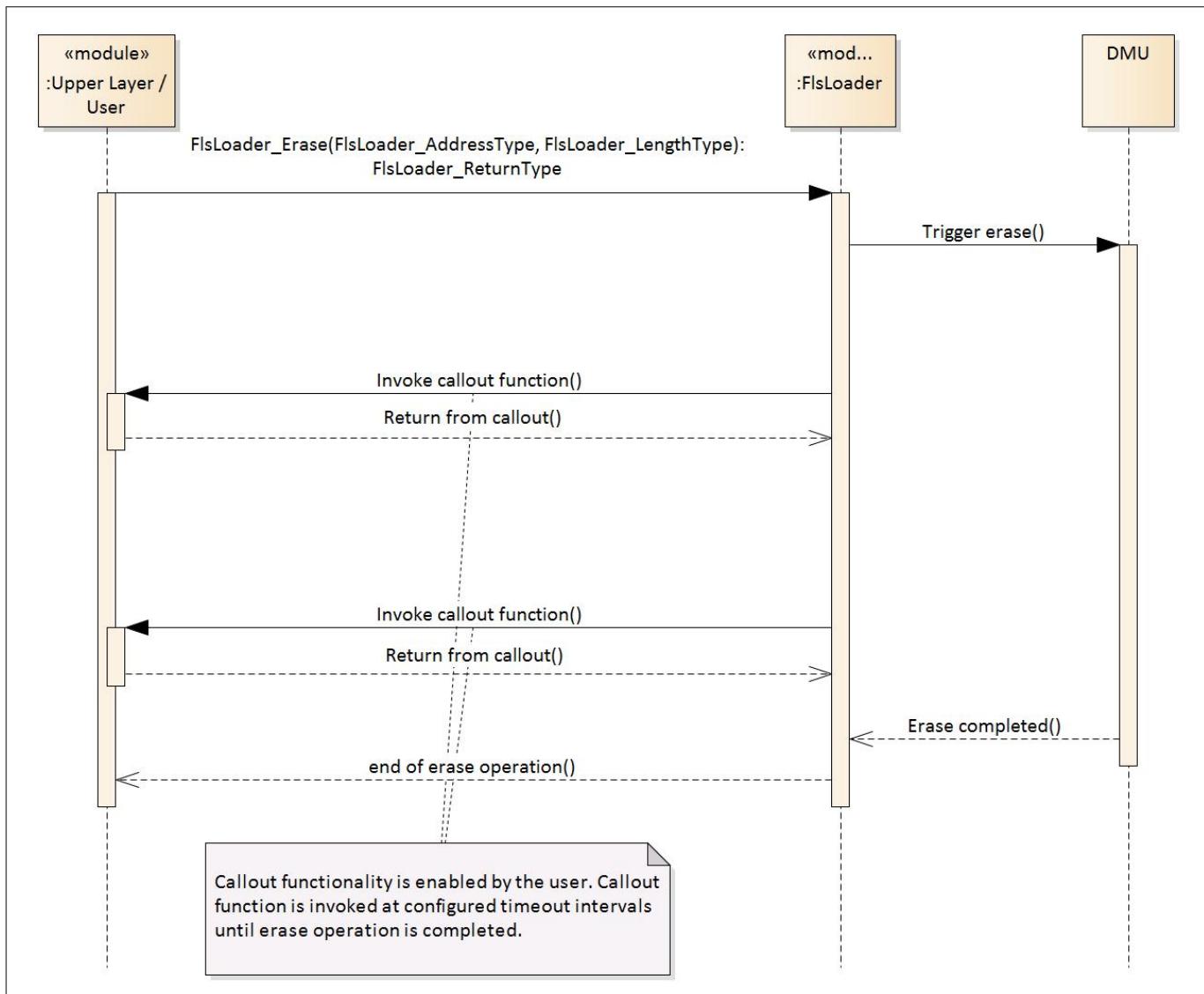
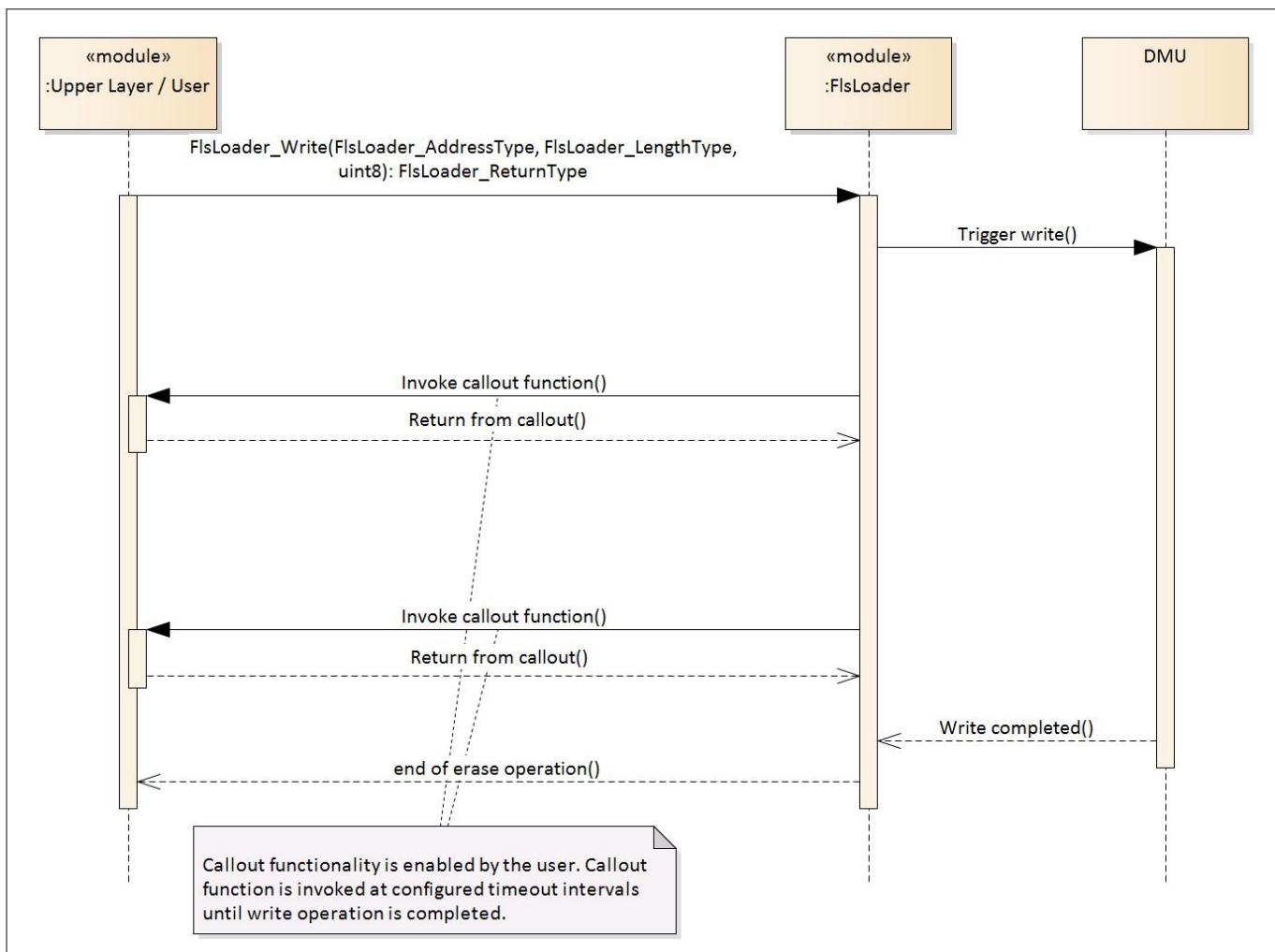


Figure 30 Configuration: Enable lock-check, Callout function FlsLoader_LoopCallOut, Callout time 52000000 ns

Sequence for the FLSLOADER erase operation with callout enabled is shown as follows.

FLSLOADER driver

Figure 31 Sequence for FLSLOADER erase operation with callout enabled

Sequence for the FLSLOADER write operation with callout enabled is shown as follows.

FLSLOADER driver

Figure 32 Sequence for FLSLOADER write operation with callout enabled

FLSLOADER driver

Sample code to erase the first two sectors of the PFlash bank 2 and write 1024 bytes of data to sector 0 is as follows:

```

/* Erase operation start.. */
/* Erase 2 sectors starting from 0xA0600000U */

/* Erase Start address */
FlsLoader_Address = 0xA0600000U;

/*Erase Length*/
EraseLength = 2U;

retval = FlsLoader_Erase((uint32)FlsLoader_Address, EraseLength);

/* Check if sectors erased successfully */
if(retval == FLSLOADER_E_OK)
{
    /* FlsLoader_Erase API returned with FLSLOADER_E_OK. Sectors erased
    successfully */

    /* Programming operation start.. */
    /* Writing 1024byte data to 0xA0600000U */

    /*Write Start Address*/
    FlsLoader_Address = 0xA0600000U;

    /*Write Length*/
    WriteLength = 1024U;

    /* Buffer[1024] is source data buffer*/
    retval = FlsLoader_Write((uint32)FlsLoader_Address, WriteLength, &Buffer[0]);

    /* Check if data written successfully */
    if(retval == FLSLOADER_E_OK)
    {
        /* FlsLoader_Write API returned with FLSLOADER_E_OK. Data written
        successfully */
    }
}

```

Erase and write an user configuration block (UCB)

User can update an UCB using the FLSLOADER erase and write APIs.

For erasing an UCB, the start address should be a valid UCB sector address and length should be 1 sector.

For writing an UCB, the start address should be a valid UCB sector address and length should be size of a UCB sector

(512 bytes) and data should contain valid confirmation code on specified offset (refer to **6.10.2 UCB Address Map** in the hardware user manual).

Before updating an UCB user shall ensure that the UCB is not protected or is in ERRORED state. Refer to section **6.6.8.3.1 UCB Confirmation** in the hardware user manual to know about UCB confirmation states.

FLSLOADER driver

While updating a dual UCB user should follow sequence mentioned in **Dual Password UCB ORIG and COPY Re-programming** under section **6.6.8.3.1 UCB Confirmation** in the hardware user manual.

FLSLOADER driver

Sample code to erase and write UCB_DFLASH_ORIG and UCB_DFLASH_COPY is as follows:

```

/* Erase UCB_DFLASH_COPY */

/* Erase Start address */
FlsLoader_Address = 0xAF403200U;

/*Erase Length*/
EraseLength = 1U;

retval = FlsLoader_Erase((uint32)FlsLoader_Address, EraseLength);

/* Check if UCB_DFLASH_COPY erased successfully */
if(retval == FLSLOADER_E_OK)
{
    /* FlsLoader_Erase API returned with FLSLOADER_E_OK. UCB_DFLASH_COPY erased
    successfully */

    /* Writing UCB_DFLASH_COPY */

    /*Write Start Address*/
    FlsLoader_Address = 0xAF403200U;

    /*Write Length*/
    WriteLength = 512U;

    /* Buffer[512] is source data buffer*/
    retval = FlsLoader_Write((uint32)FlsLoader_Address, WriteLength, &Buffer[0]);

    /* Check if UCB_DFLASH_COPY written successfully */
    if(retval == FLSLOADER_E_OK)
    {
        /* FlsLoader_Write API returned with FLSLOADER_E_OK. Data written
        successfully to UCB_DFLASH_COPY*/
    }
}

/* Erase UCB_DFLASH_ORIG */

/* Erase Start address */
FlsLoader_Address = 0xAF402200U;

/*Erase Length*/
EraseLength = 1U;

retval = FlsLoader_Erase((uint32)FlsLoader_Address, EraseLength);

/* Check if UCB_DFLASH_ORIG erased successfully */
if(retval == FLSLOADER_E_OK)
{
    /* FlsLoader_Erase API returned with FLSLOADER_E_OK. UCB_DFLASH_ORIG erased
    successfully */
}

```

FLSLOADER driver

```

/* Writing UCB_DFLASH_ORIG */

/*Write Start Address*/
FlsLoader_Address = 0xAF402200U;

/*Write Length*/
WriteLength = 512U;

/* Buffer[512] is source data buffer*/
retval = FlsLoader_Write((uint32)FlsLoader_Address, WriteLength, &Buffer[0]);

/* Check if UCB_DFLASH_ORIG written successfully */
if(retval == FLSLOADER_E_OK)
{
    /* FlsLoader_Write API returned with FLSLOADER_E_OK. Data written
    successfully to UCB_DFLASH_ORIG*/
}
}

```

FLSLOADER lock operation

The lock API installs the protections configured for the Flash banks during pre-compile time.

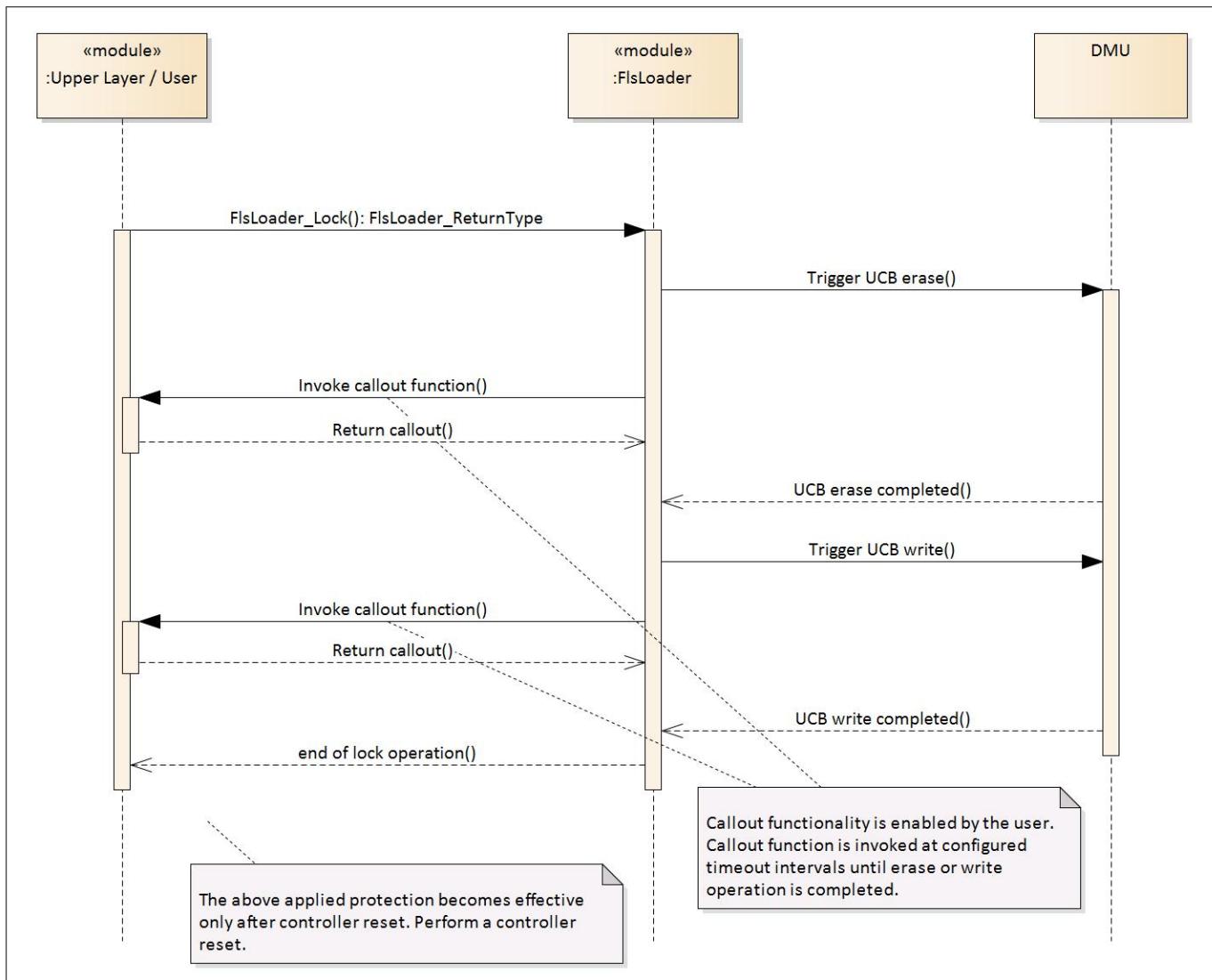
DFlash0 protections are configurable at bank level. Supported protections for DFlash0 are read and write protections.

PFlash protections are configurable at sector level. Supported protections for PFlash are write protection, write once protection (WOP), one time programmable (OTP) protection.

A controller reset is required after execution of lock API for the installed protections to become effective in the hardware.

If callout functionality is enabled by the user, the driver invokes the callout function to application at configured timeout intervals while executing the lock API.

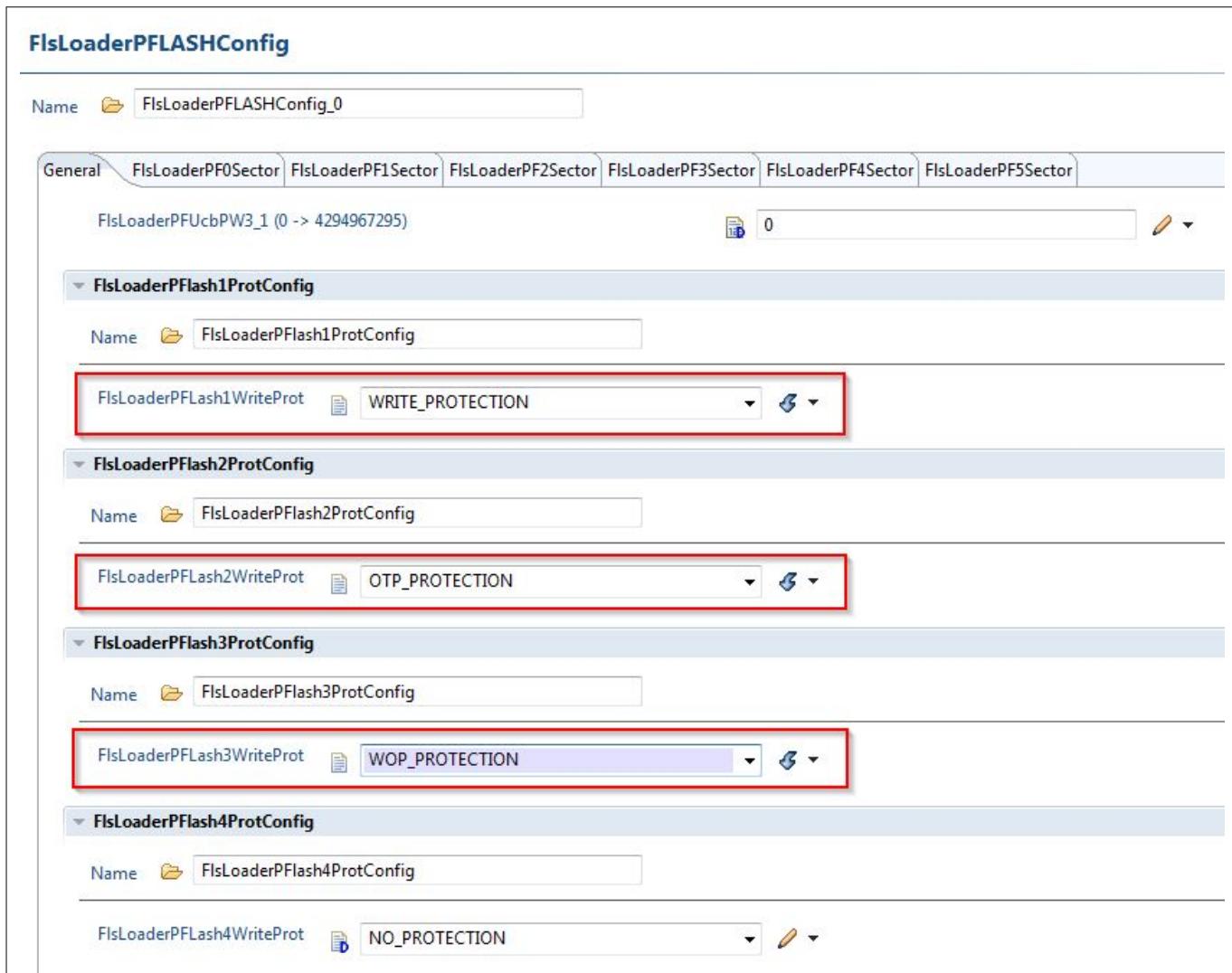
Following sequence diagram depicts FLSLOADER lock operation with callout enabled.

FLSLOADER driver

Figure 33 Sequence for FLSLOADER lock operation with callout enabled

If requirement is to enable following protections for Flash banks:

- Write protection for sector 0 of PFlash bank 1
- OTP protection for sector 0 of PFlash bank 2
- WOP protection for sector 0 of PFlash bank 3
- Write protection for DFlash bank 0

Then following configuration should be set in the configuration tool:

FLSLOADER driver
**Figure 34**

Step1: Select PFlash 1, PFlash 2, PFlash 3 bank protections as write, OTP and WOP protections

FLSLOADER driver

General FlsLoaderPF0Sector FlsLoaderPF1Sector FlsLoaderPF2Sector FlsLoaderP																										
FlsLoaderPF1Sector																										
<table border="1"> <thead> <tr> <th>Index</th> <th>Name</th> <th>FlsLoaderPFSectorWriteProtection</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>FlsLoaderPF1Sector0</td> <td>WRITE_PROTECTION</td> </tr> <tr> <td>1</td> <td>FlsLoaderPF1Sector1</td> <td>NO_PROTECTION</td> </tr> <tr> <td>2</td> <td>FlsLoaderPF1Sector2</td> <td>NO_PROTECTION</td> </tr> <tr> <td>3</td> <td>FlsLoaderPF1Sector3</td> <td>NO_PROTECTION</td> </tr> <tr> <td>4</td> <td>FlsLoaderPF1Sector4</td> <td>NO_PROTECTION</td> </tr> <tr> <td>5</td> <td>FlsLoaderPF1Sector5</td> <td>NO_PROTECTION</td> </tr> </tbody> </table>			Index	Name	FlsLoaderPFSectorWriteProtection	0	FlsLoaderPF1Sector0	WRITE_PROTECTION	1	FlsLoaderPF1Sector1	NO_PROTECTION	2	FlsLoaderPF1Sector2	NO_PROTECTION	3	FlsLoaderPF1Sector3	NO_PROTECTION	4	FlsLoaderPF1Sector4	NO_PROTECTION	5	FlsLoaderPF1Sector5	NO_PROTECTION			
Index	Name	FlsLoaderPFSectorWriteProtection																								
0	FlsLoaderPF1Sector0	WRITE_PROTECTION																								
1	FlsLoaderPF1Sector1	NO_PROTECTION																								
2	FlsLoaderPF1Sector2	NO_PROTECTION																								
3	FlsLoaderPF1Sector3	NO_PROTECTION																								
4	FlsLoaderPF1Sector4	NO_PROTECTION																								
5	FlsLoaderPF1Sector5	NO_PROTECTION																								
General FlsLoaderPF0Sector FlsLoaderPF1Sector FlsLoaderPF2Sector FlsLoaderP																										
FlsLoaderPF2Sector																										
<table border="1"> <thead> <tr> <th>Ind...</th> <th>Name</th> <th>FlsLoaderPFSectorWriteProtection</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>FlsLoaderPF2Sector0</td> <td>OTP_PROTECTION</td> </tr> <tr> <td>1</td> <td>FlsLoaderPF2Sector1</td> <td>NO_PROTECTION</td> </tr> <tr> <td>2</td> <td>FlsLoaderPF2Sector2</td> <td>NO_PROTECTION</td> </tr> <tr> <td>3</td> <td>FlsLoaderPF2Sector3</td> <td>NO_PROTECTION</td> </tr> <tr> <td>4</td> <td>FlsLoaderPF2Sector4</td> <td>NO_PROTECTION</td> </tr> <tr> <td>5</td> <td>FlsLoaderPF2Sector5</td> <td>NO_PROTECTION</td> </tr> <tr> <td>6</td> <td>FlsLoaderPF2Sector6</td> <td>NO_PROTECTION</td> </tr> </tbody> </table>			Ind...	Name	FlsLoaderPFSectorWriteProtection	0	FlsLoaderPF2Sector0	OTP_PROTECTION	1	FlsLoaderPF2Sector1	NO_PROTECTION	2	FlsLoaderPF2Sector2	NO_PROTECTION	3	FlsLoaderPF2Sector3	NO_PROTECTION	4	FlsLoaderPF2Sector4	NO_PROTECTION	5	FlsLoaderPF2Sector5	NO_PROTECTION	6	FlsLoaderPF2Sector6	NO_PROTECTION
Ind...	Name	FlsLoaderPFSectorWriteProtection																								
0	FlsLoaderPF2Sector0	OTP_PROTECTION																								
1	FlsLoaderPF2Sector1	NO_PROTECTION																								
2	FlsLoaderPF2Sector2	NO_PROTECTION																								
3	FlsLoaderPF2Sector3	NO_PROTECTION																								
4	FlsLoaderPF2Sector4	NO_PROTECTION																								
5	FlsLoaderPF2Sector5	NO_PROTECTION																								
6	FlsLoaderPF2Sector6	NO_PROTECTION																								
General FlsLoaderPF0Sector FlsLoaderPF1Sector FlsLoaderPF2Sector FlsLoaderPF3Sector																										
FlsLoaderPF3Sector																										
<table border="1"> <thead> <tr> <th>Index</th> <th>Name</th> <th>FlsLoaderPFSectorWriteProtection</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>FlsLoaderPF3Sector0</td> <td>WOP_PROTECTION</td> </tr> <tr> <td>1</td> <td>FlsLoaderPF3Sector1</td> <td>NO_PROTECTION</td> </tr> <tr> <td>2</td> <td>FlsLoaderPF3Sector2</td> <td>NO_PROTECTION</td> </tr> <tr> <td>3</td> <td>FlsLoaderPF3Sector3</td> <td>NO_PROTECTION</td> </tr> <tr> <td>4</td> <td>FlsLoaderPF3Sector4</td> <td>NO_PROTECTION</td> </tr> <tr> <td>5</td> <td>FlsLoaderPF3Sector5</td> <td>NO_PROTECTION</td> </tr> </tbody> </table>			Index	Name	FlsLoaderPFSectorWriteProtection	0	FlsLoaderPF3Sector0	WOP_PROTECTION	1	FlsLoaderPF3Sector1	NO_PROTECTION	2	FlsLoaderPF3Sector2	NO_PROTECTION	3	FlsLoaderPF3Sector3	NO_PROTECTION	4	FlsLoaderPF3Sector4	NO_PROTECTION	5	FlsLoaderPF3Sector5	NO_PROTECTION			
Index	Name	FlsLoaderPFSectorWriteProtection																								
0	FlsLoaderPF3Sector0	WOP_PROTECTION																								
1	FlsLoaderPF3Sector1	NO_PROTECTION																								
2	FlsLoaderPF3Sector2	NO_PROTECTION																								
3	FlsLoaderPF3Sector3	NO_PROTECTION																								
4	FlsLoaderPF3Sector4	NO_PROTECTION																								
5	FlsLoaderPF3Sector5	NO_PROTECTION																								

Figure 35

Step2: Select sector 0 of PFlash 1, PFlash 2, PFlash 3 banks as write, OTP and WOP protections

FLSLOADER driver

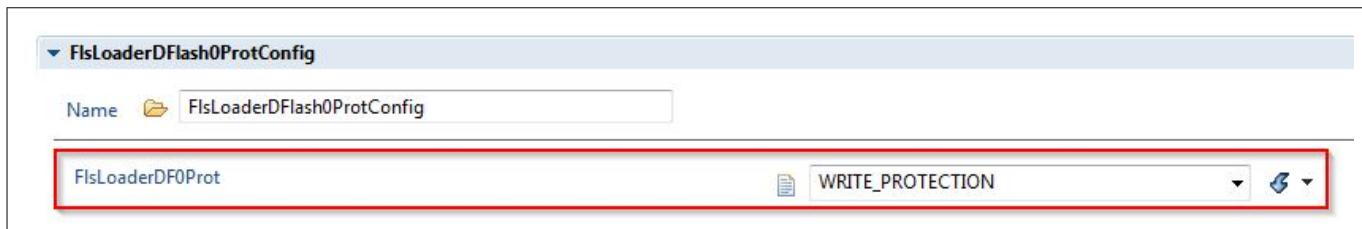


Figure 36 Step3: Select DFlash0 bank protection as write protection

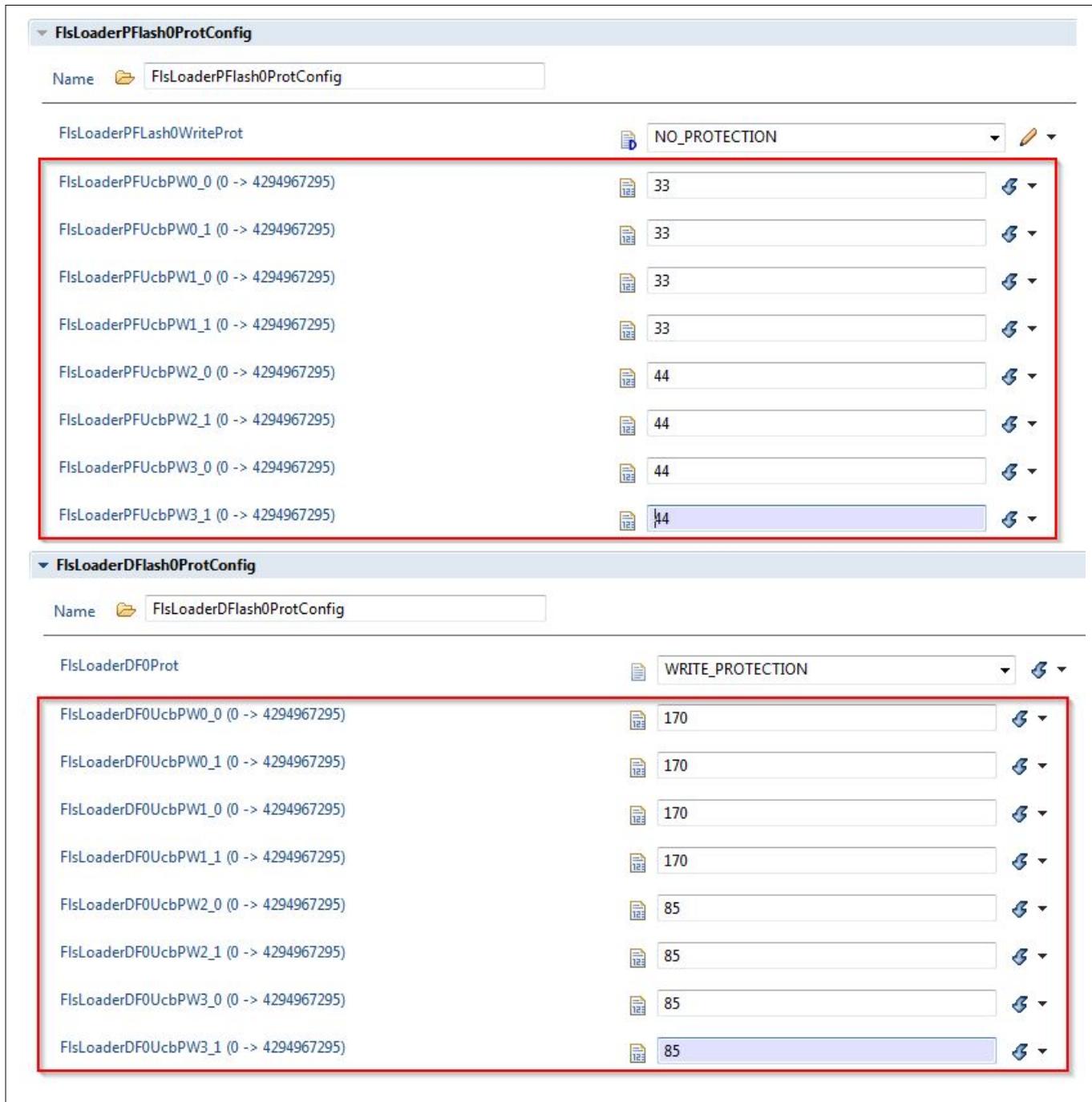


Figure 37 Step4: Configure passwords PFlash and DFlash0 access

FLSLOADER driver

Sample code to depict lock operation is as follows:

```

/* FlsLoader_Lock execution starts.. */
retval = FlsLoader_Lock();

/*Lock executed successfully*/
if(retval == FLSLOADER_E_OK)
{
    /* FlsLoader_Lock Execution Passed */
    /* Trigger controller reset for HW to install protections */
}

/* Following part should be executed after micro-controller reset */

/* Erasing the Sector-0 of PFlash Bank-1 where write Protection is enabled */

/*Address of Sector 0 of PFlash bank 1*/
FlsLoader_Address = 0xA0300000U;

/*Number of sectors to be erased is 1*/
Length = 1U;

retval = FlsLoader_Erase(FlsLoader_Address, Length);

/* Check if erase operation returned with FLSLOADER_E_LOCKED */
if (retval == FLSLOADER_E_LOCKED)
{
    /* FlsLoader_Erase returned FLSLOADER_E_LOCKED. Sector is locked. */
}

```

FLSLOADER unlock operation

The unlock API is used to temporarily (until next controller reset) disable the protections installed by lock API.

Only read and write protections installed by the lock API are disabled. OTP and WOP protections can't be disabled.

Sequence for FLSLOADER unlock operation is depicted as below:

FLSLOADER driver

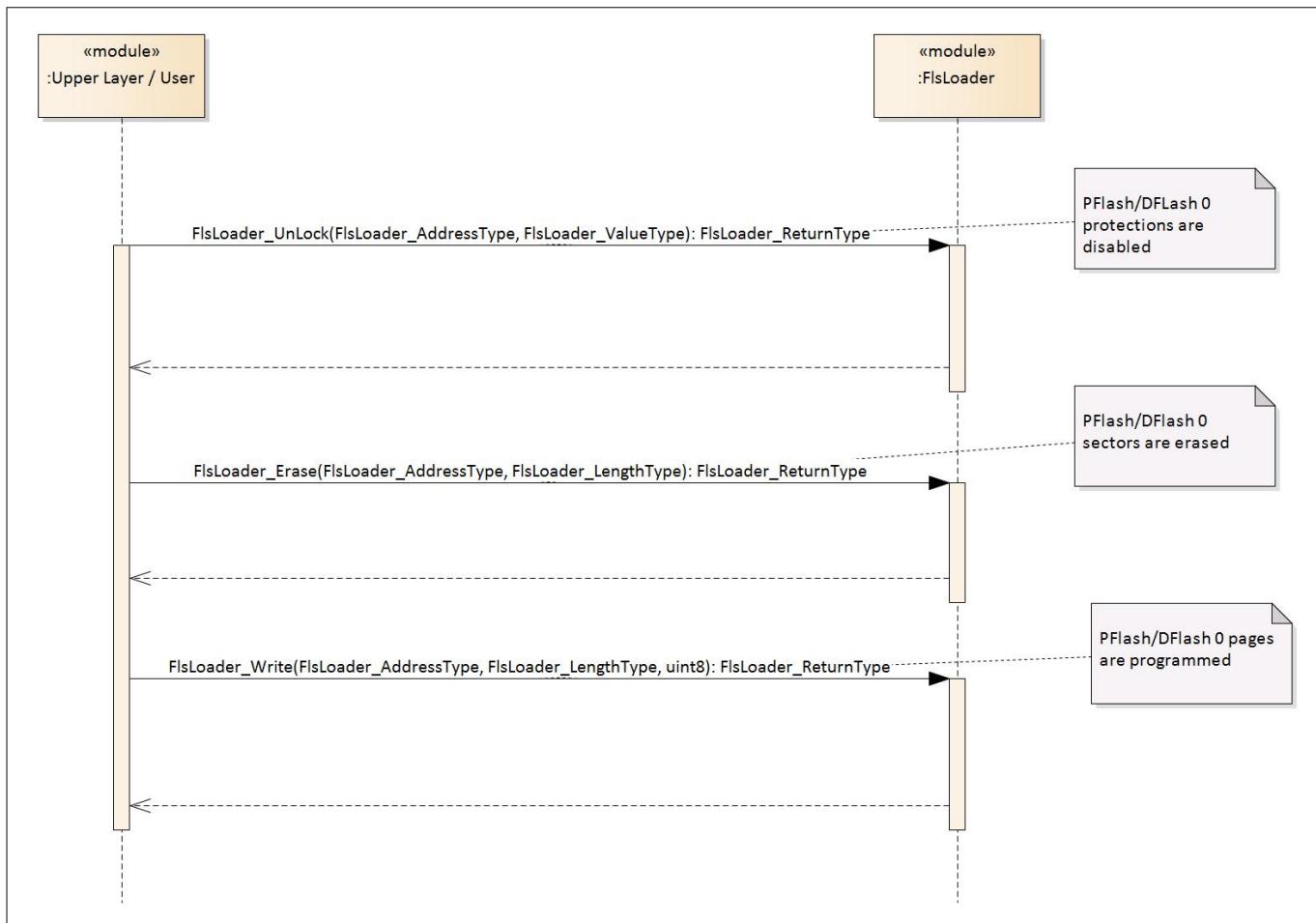


Figure 38 Sequence for FLSLOADER unlock operation

FLSLOADER driver

Sample code to depict FLSLOADER unlock operation is as follows:

```

/* FlsLoader_Unlock execution starts.. */

/* Unlocking the Write Protections installed for PFlash sectors.*/

/* PFLASH_ORIG ucb address*/
FlsLoader_Address = 0xAF402000U;

/*PfPassword[8] is 8-word password for PFlash */
retval = FlsLoader_Unlock(FlsLoader_Address, &PfPassword[0]);

/* Check if PFlash Write protection Disabled */
if(retval == FLSLOADER_E_OK)
{
    /* Write Protections installed for PFlash sectors are disabled*/
    /* Erasing the Sector-0 of PFlash Bank-1 where write Protection is Disabled.*/

    /*Address of Sector 0 of PFlash bank 1*/
    FlsLoader_Address = 0xA0300000U;

    /*Number of sectors to be erased is 1*/
    Length = 1U;

    retval = FlsLoader_Erase(FlsLoader_Address, Length);

    /* Check if sector erased successfully */
    if (retval == FLSLOADER_E_OK)
    {
        /* FlsLoader_Erase returned FLSLOADER_E_OK. Sector erased successfully.*/
    }
}
}

```

Load FLSLOADER write and erase routines to RAM

- In driver code, the FLSLOADER write and erase routines alone are encapsulated under memory map section FLSLOADER_START_SEC_WRITEERASE_CODE_QM_LOCAL and FLSLOADER_STOP_SEC_WRITEERASE_CODE_QM_LOCAL. The constant data required for the write and erase routines are encapsulated under memory map section FLSLOADER_START_SEC_CONST_QM_LOCAL_32 and FLSLOADER_STOP_SEC_CONST_QM_LOCAL_32.
- The dependent MCALLIB APIs Mcal_DelayTickResolution() and Mcal_DelayGetTick() for write and erase routines are encapsulated under memory map section MCALLIB_START_SEC_CODE_ASIL_B_GLOBAL and MCALLIB_STOP_SEC_CODE_ASIL_B_GLOBAL
- In order to load complete write and erase routines to RAM, above mentioned sections need to be moved to RAM. This can be done using linker settings. Copying of RAM can be done either by user application or can make use of the init functions provided by the compiler.
- The DET should be disabled if FLSLOADER write and erase routines are executed from the RAM.
- If callout functionality is enabled, user shall ensure that configured callout function is placed in the RAM.

E_NOT_OK return value for FlsLoader_Erase and FlsLoader_Write API

FlsLoader_Erase and FlsLoader_Write API operation may return E_NOT_OK under either of the following conditions.

FLSLOADER driver

- Hardware operation takes more time than the calculated timeout value for the operation.
- Hardware reports an operational error while an operation is being performed.

In such a situation, if the hardware continues to be in the busy state then further calls to any of the following APIs FlsLoader_DelInit, FlsLoader_Write, FlsLoader_Erase, FlsLoader_Lock, FlsLoader_UnLock will cause FLSLOADER_E_BUSY to be returned. As hardware continues to be in busy state a hardware reset is required.

Exclusive area for PFlash erase and write operation

- Safety Endint protection is disabled for the duration of PFlash erase and write command cycle execution. These operations are performed within an exclusive area enforced with the help of following exclusive area interfaces :-

```
FlsLdr_ExclArea_PfProg_Enter()
FlsLdr_ExclArea_PfProg_Exit()
FlsLdr_ExclArea_PfErase_Enter()
FlsLdr_ExclArea_PfErase_Exit()
```

Provided by files FlsLdr_ExclArea.h and FlsLdr_ExclArea.c.

- The user needs to implement these functions so as to prevent the interruption of operations within the exclusive area. This will allow the code in exclusive area to execute quickly and not reach a safety watchdog timeout condition on the disabling of Safety Endinit protection. For above mentioned functions, please refer following example code.

```
void FlsLdr_ExclArea_PfProg_Enter(void)
{
    SuspendAllInterrupts();
}

void FlsLdr_ExclArea_PfProg_Exit(void)
{
    ResumeAllInterrupts();
}

void FlsLdr_ExclArea_PfErase_Enter(void)
{
    SuspendAllInterrupts();
}

void FlsLdr_ExclArea_PfErase_Exit(void)
{
    ResumeAllInterrupts();
}
```

3.1.5 Key architectural considerations

3.1.5.1 User mode is not supported by the driver

The FLSLOADER driver does not support user mode configuration for any of its APIs. The driver is meant to be used in the boot loader application for Flash programming. Boot loader is not a part of the MCAL runtime component. Therefore, all APIs of the driver shall be executed in the supervisory mode.

[cover parentID FLSLOADER={DC8DF695-F796-4e0e-9514-DEF2CE8C2AA4}]

FLSLOADER driver

3.2 Assumptions of Use (AoUs)

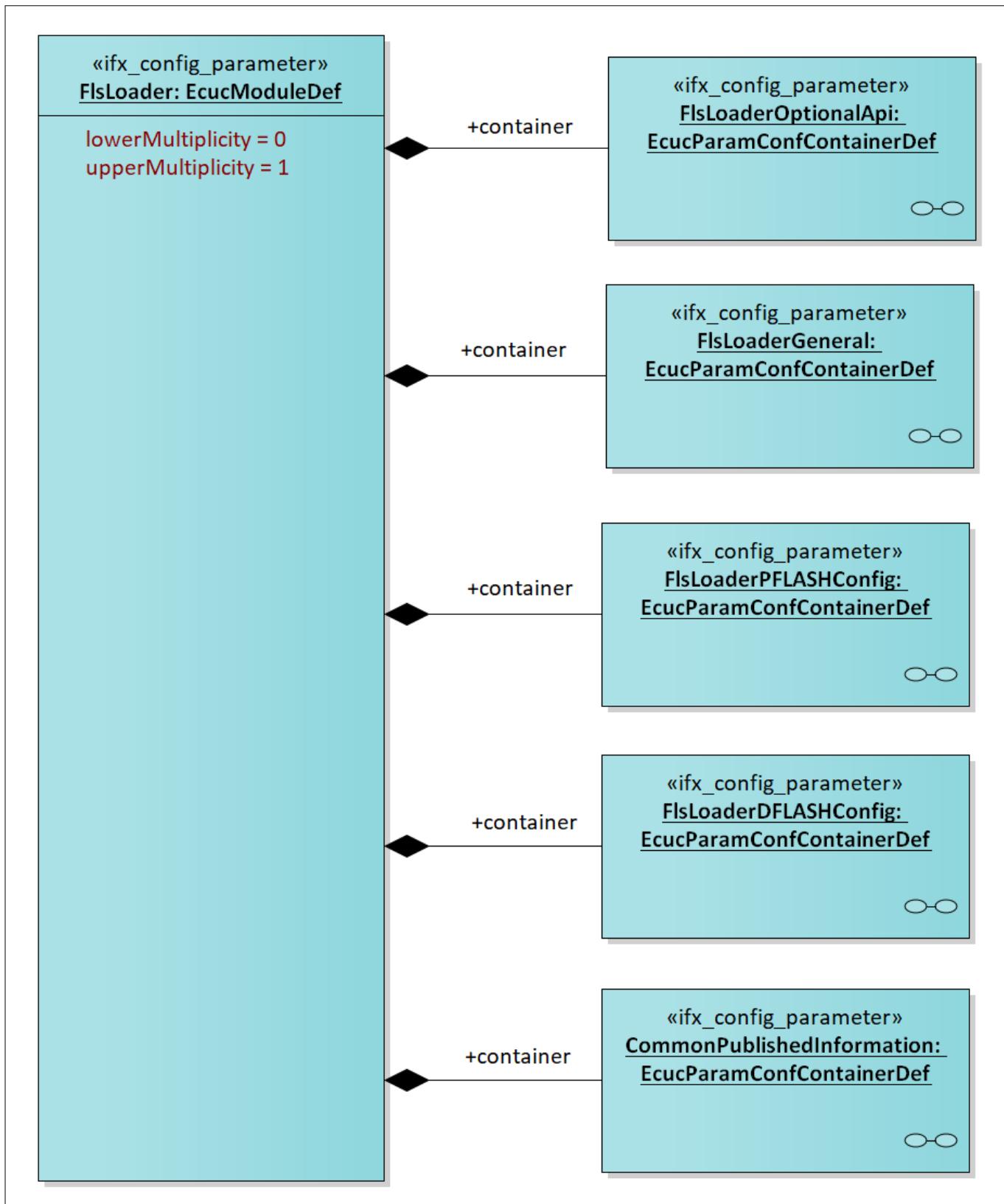
There are no AoUs for the driver.

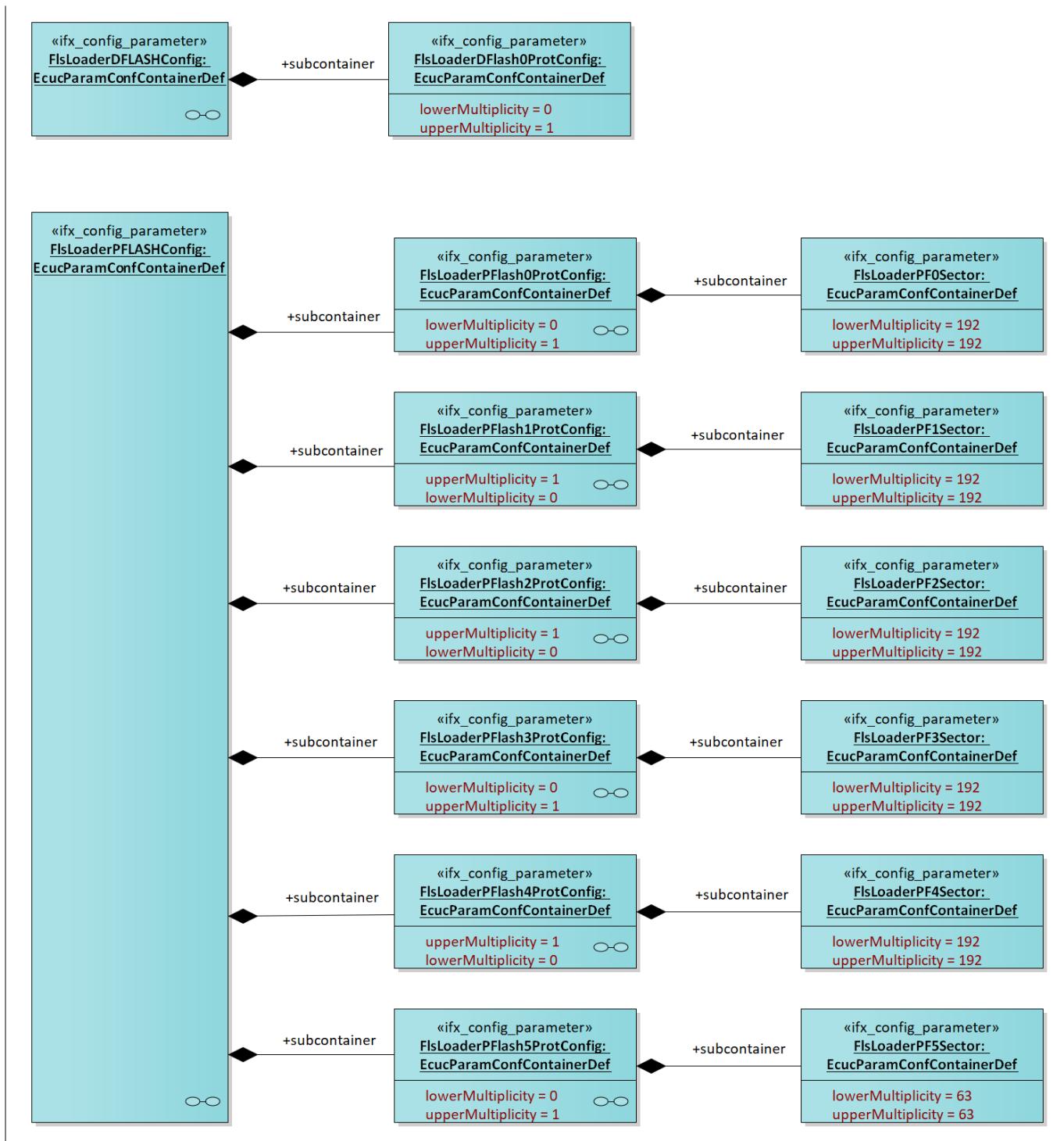
FLSLOADER driver

3.3 Reference information

3.3.1 Configuration interfaces

FLSLOADER driver



FLSLOADER driver

Figure 39 Container hierarchy along with their configuration parameters
3.3.1.1 Container: CommonPublishedInformation

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

FLSLOADER driver**3.3.1.1.1 ArMajorVersion****Table 246 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	-		

3.3.1.1.2 ArMinorVersion**Table 247 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		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.1.3 ArPatchVersion**Table 248 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	This parameter provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		

FLSLOADER driver**Table 248 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	-		

3.3.1.1.4 ModuleId**Table 249 Specification for ModuleId**

Name	ModuleId		
Description	This parameter provides the Module ID.		
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	-		

3.3.1.1.5 Release**Table 250 Specification for Release**

Name	Release		
Description	The configuration parameter defines the TC3xx derivative used for the implementation.		
Multiplicity	1..1	Type	EcucStringParamDef
Range	String		
Default value	As per selected TC3xx derivative		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-

FLSLOADER driver**Table 250 Specification for Release (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.1.6 SwMajorVersion**Table 251 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	This 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	-		

3.3.1.1.7 SwMinorVersion**Table 252 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		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

FLSLOADER driver
3.3.1.1.8 SwPatchVersion
Table 253 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		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.1.9 VendorId
Table 254 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	-		

3.3.1.2 Container: FlsLoader

Configuration of the FLSLOADER driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.3 Container: FlsLoaderDFlash0ProtConfig

Container for configuring DFlash bank 0 protection. Container is available only if DFlash bank 0 is available in the TC3xx device selected by configuration else the container is not available.

FLSLOADER driver

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.3.1 FlsLoaderDF0Prot**Table 255 Specification for FlsLoaderDF0Prot**

Name	FlsLoaderDF0Prot		
Description	Configures DFlash bank 0 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection READ_PROTECTION: Read protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.3.2 FlsLoaderDF0UcbPW0_0**Table 256 Specification for FlsLoaderDF0UcbPW0_0**

Name	FlsLoaderDF0UcbPW0_0		
Description	PW0: First least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	FlsLoaderDF0Prot		

FLSLOADER driver**3.3.1.3.3 FlsLoaderDF0UcbPW0_1****Table 257 Specification for FlsLoaderDF0UcbPW0_1**

Name	FlsLoaderDF0UcbPW0_1		
Description	PW1: Second least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	IFX	Scope	LOCAL
Dependency	FlsLoaderDF0Prot		

3.3.1.3.4 FlsLoaderDF0UcbPW1_0**Table 258 Specification for FlsLoaderDF0UcbPW1_0**

Name	FlsLoaderDF0UcbPW1_0		
Description	PW2: Third least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	IFX	Scope	LOCAL
Dependency	FlsLoaderDF0Prot		

3.3.1.3.5 FlsLoaderDF0UcbPW1_1**Table 259 Specification for FlsLoaderDF0UcbPW1_1**

Name	FlsLoaderDF0UcbPW1_1
-------------	----------------------

FLSLOADER driver**Table 259 Specification for FlsLoaderDF0UcbPW1_1 (continued)**

Description	PW3: Fourth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	IFX	Scope	LOCAL
Dependency	FlsLoaderDF0Prot		

3.3.1.3.6 FlsLoaderDF0UcbPW2_0**Table 260 Specification for FlsLoaderDF0UcbPW2_0**

Name	FlsLoaderDF0UcbPW2_0		
Description	PW4: Fifth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	IFX	Scope	LOCAL
Dependency	FlsLoaderDF0Prot		

3.3.1.3.7 FlsLoaderDF0UcbPW2_1**Table 261 Specification for FlsLoaderDF0UcbPW2_1**

Name	FlsLoaderDF0UcbPW2_1		
Description	PW5: Sixth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4294967295		

FLSLOADER driver**Table 261 Specification for FlsLoaderDF0UcbPW2_1 (continued)**

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	FlsLoaderDF0Prot		

3.3.1.3.8 FlsLoaderDF0UcbPW3_0**Table 262 Specification for FlsLoaderDF0UcbPW3_0**

Name	FlsLoaderDF0UcbPW3_0		
Description	PW6: Seventh least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	FlsLoaderDF0Prot		

3.3.1.3.9 FlsLoaderDF0UcbPW3_1**Table 263 Specification for FlsLoaderDF0UcbPW3_1**

Name	FlsLoaderDF0UcbPW3_1		
Description	PW7: Eighth least significant 32-bit word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-

FLSLOADER driver
Table 263 Specification for FlsLoaderDF0UcbPW3_1 (continued)

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsLoaderDF0Prot		

3.3.1.4 Container: FlsLoaderDFLASHConfig

This container contains the configuration parameters and sub-containers for configuration of DFlash.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.5 Container: FlsLoaderGeneral

This container contains the general configuration parameters of the FLSLOADER driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.5.1 FlsLoaderCallOutFunction
Table 264 Specification for FlsLoaderCallOutFunction

Name	FlsLoaderCallOutFunction		
Description	<p>FLSLOADER provides a call out function to the user while performing Flash write and erase operations using its APIs. It is invoked at every user defined time rate (FlsLoaderCallOutTime).</p> <p>Call out function should be defined by the user using this parameter. User can configure either the name or the valid address of the call out function in Flash.</p> <p>Call out function is enabled only if valid value as mentioned above is configured for the parameter. By default this parameter contains NULL_PTR and call out function is disabled to reduce the executable code size.</p> <p>When call out is enabled user should configure call out time using parameter FlsLoaderCallOutTime.</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

FLSLOADER driver**Table 264 Specification for FlsLoaderCallOutFunction (continued)**

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

3.3.1.5.2 FlsLoaderCallOutTime**Table 265 Specification for FlsLoaderCallOutTime**

Name	FlsLoaderCallOutTime		
Description	<p>Specifies the maximum time in nanoseconds to wait before invoking call out function to application while looping for status during write and erase operations.</p> <p>The default value of this parameter is set to 5000000 as an example value within the range.</p> <p>This parameter is valid only if FlsLoaderCallOutFuntion is configured with a value other than NULL_PTR.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	10000 - 4294967295		
Default value	5000000		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsLoaderCallOutFunction		

3.3.1.5.3 FlsLoaderDevErrorDetect**Table 266 Specification for FlsLoaderDevErrorDetect**

Name	FlsLoaderDevErrorDetect		
Description	<p>Switch for enabling the development error tracer (DET).</p> <p>True: DET enabled</p> <p>False: DET disabled</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	-

FLSLOADER driver
Table 266 Specification for FlsLoaderDevErrorDetect (continued)

Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.5.4 FlsLoaderEnableLockCheck
Table 267 Specification for FlsLoaderEnableLockCheck

Name	FlsLoaderEnableLockCheck		
Description	Switch for enabling the Lock-check functionality True: Enable the Lock-check routine in write /erase APIs. False: Disable the Lock-check routine in write/erase APIs. This optional feature is disabled by default to reduce 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	-		

3.3.1.6 Container: FlsLoaderOptionalApi

This container contains the configuration parameters for optional APIs of the FLSLOADER driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.6.1 FlsLoaderDeInitApi
Table 268 Specification for FlsLoaderDeInitApi

Name	FlsLoaderDeInitApi
Description	Switch for enabling the FlsLoader_DeInit API. True: FlsLoader_DeInit API enabled False: FlsLoader_DeInit API disabled This optional API is disabled by default to reduce the executable code size.

FLSLOADER driver**Table 268 Specification for FlsLoaderDeInitApi (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	-		

3.3.1.6.2 FlsLoaderLockUnlockApi**Table 269 Specification for FlsLoaderLockUnlockApi**

Name	FlsLoaderLockUnlockApi		
Description	Switch for enabling the FlsLoader_Lock and FlsLoader_Unlock APIs. True: FlsLoader_Lock and FlsLoader_Unlock APIs enabled False: FlsLoader_Lock and FlsLoader_Unlock APIs disabled These optional APIs are disabled by default to reduce 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	-		

3.3.1.6.3 FlsLoaderVersionInfoApi**Table 270 Specification for FlsLoaderVersionInfoApi**

Name	FlsLoaderVersionInfoApi
-------------	-------------------------

FLSLOADER driver**Table 270 Specification for FlsLoaderVersionInfoApi (continued)**

Description	Switch for enabling the FlsLoaderVersionInfo API. True: FlsLoaderVersionInfo API enabled False: FlsLoaderVersionInfo API disabled This optional API is disabled by default to reduce 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	-		

3.3.1.7 Container: FlsLoaderPF0Sector

Container for configuration of PFlash bank 0 sectors

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.7.1 FlsLoaderPFSectorWriteProtection**Table 271 Specification for FlsLoaderPFSectorWriteProtection**

Name	FlsLoaderPFSectorWriteProtection		
Description	Configuration of PFlash bank 0 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-

FLSLOADER driver**Table 271 Specification for FlsLoaderPFSectorWriteProtection (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsLoaderPFlash0WriteProt		

3.3.1.8 Container: FlsLoaderPF1Sector

Container for configuration of PFlash bank 1 sectors.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.8.1 FlsLoaderPFSectorWriteProtection**Table 272 Specification for FlsLoaderPFSectorWriteProtection**

Name	FlsLoaderPFSectorWriteProtection		
Description	Configuration of PFlash bank 1 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsLoaderPFlash1WriteProt		

3.3.1.9 Container: FlsLoaderPF2Sector

Container for configuration of PFlash bank 2 sectors.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

FLSLOADER driver**3.3.1.9.1 FlsLoaderPFSectorWriteProtection****Table 273 Specification for FlsLoaderPFSectorWriteProtection**

Name	FlsLoaderPFSectorWriteProtection		
Description	Configuration of PFlash bank 2 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protection		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsLoaderPFlash2WriteProt		

3.3.1.10 Container: FlsLoaderPF3Sector

Container for configuration of PFlash bank 3 sectors.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.10.1 FlsLoaderPFSectorWriteProtection**Table 274 Specification for FlsLoaderPFSectorWriteProtection**

Name	FlsLoaderPFSectorWriteProtection		
Description	Configuration of PFlash bank 3 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		

FLSLOADER driver**Table 274 Specification for FlsLoaderPFSectorWriteProtection (continued)**

Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsLoaderPFlash3WriteProt		

3.3.1.11 Container: FlsLoaderPF4Sector

Container for configuration of PFlash bank 4 sectors.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.1.11.1 FlsLoaderPFSectorWriteProtection**Table 275 Specification for FlsLoaderPFSectorWriteProtection**

Name	FlsLoaderPFSectorWriteProtection		
Description	Configuration of PFlash bank 4 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsLoaderPFlash4WriteProt		

3.3.1.12 Container: FlsLoaderPF5Sector

Container for configuration of PFlash bank 5 sectors.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

FLSLOADER driver**3.3.1.12.1 FlsLoaderPFSectorWriteProtection****Table 276 Specification for FlsLoaderPFSectorWriteProtection**

Name	FlsLoaderPFSectorWriteProtection		
Description	Configuration of PFlash bank 5 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsLoaderPFlash5WriteProt		

3.3.1.13 Container: FlsLoaderPFlash0ProtConfig

Container for configuring PFlash bank 0 protection. Container is available only if PFlash bank 0 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.13.1 FlsLoaderPF0UcbPW0_0**Table 277 Specification for FlsLoaderPF0UcbPW0_0**

Name	FlsLoaderPF0UcbPW0_0		
Description	PW0: First least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	Post-build variant multiplicity	-

FLSLOADER driver**Table 277 Specification for FlsLoaderPF0UcbPW0_0 (continued)**

Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	FlsLoaderPFlash0WriteProt		

3.3.1.13.2 FlsLoaderPF0UcbPW0_1**Table 278 Specification for FlsLoaderPF0UcbPW0_1**

Name	FlsLoaderPF0UcbPW0_1		
Description	PW1: Second least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	FlsLoaderPFlash0WriteProt		

3.3.1.13.3 FlsLoaderPF0UcbPW1_0**Table 279 Specification for FlsLoaderPF0UcbPW1_0**

Name	FlsLoaderPF0UcbPW1_0		
Description	PW2: Third least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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

FLSLOADER driver**Table 279 Specification for FlsLoaderPF0UcbPW1_0 (continued)**

Dependency	FlsLoaderPFlash0WriteProt
-------------------	---------------------------

3.3.1.13.4 FlsLoaderPF0UcbPW1_1**Table 280 Specification for FlsLoaderPF0UcbPW1_1**

Name	FlsLoaderPF0UcbPW1_1		
Description	PW3: Fourth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	Type	EcuclIntegerParamDef
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	FlsLoaderPFlash0WriteProt		

3.3.1.13.5 FlsLoaderPF0UcbPW2_0**Table 281 Specification for FlsLoaderPF0UcbPW2_0**

Name	FlsLoaderPF0UcbPW2_0		
Description	PW4: Fifth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	Type	EcuclIntegerParamDef
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	FlsLoaderPFlash0WriteProt		

FLSLOADER driver**3.3.1.13.6 FlsLoaderPF0UcbPW2_1****Table 282 Specification for FlsLoaderPF0UcbPW2_1**

Name	FlsLoaderPF0UcbPW2_1		
Description	PW5: Sixth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	IFX	Scope	LOCAL
Dependency	FlsLoaderPFlash0WriteProt		

3.3.1.13.7 FlsLoaderPF0UcbPW3_0**Table 283 Specification for FlsLoaderPF0UcbPW3_0**

Name	FlsLoaderPF0UcbPW3_0		
Description	PW6: Seventh least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	IFX	Scope	LOCAL
Dependency	FlsLoaderPFlash0WriteProt		

3.3.1.13.8 FlsLoaderPF0UcbPW3_1**Table 284 Specification for FlsLoaderPF0UcbPW3_1**

Name	FlsLoaderPF0UcbPW3_1
-------------	----------------------

FLSLOADER driver
Table 284 Specification for FlsLoaderPF0UcbPW3_1 (continued)

Description	PW7: Eighth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
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	FlsLoaderPFlash0WriteProt		

3.3.1.13.9 FlsLoaderPFlash0WriteProt

Table 285 Specification for FlsLoaderPFlash0WriteProt

Name	FlsLoaderPFlash0WriteProt		
Description	Configuration of PFlash bank 0 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.14 Container: FlsLoaderPFlash1ProtConfig

Container for configuring PFlash bank 1 protection. Container is available only if PFlash bank 1 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

FLSLOADER driver

Multiplicity Configuration Class: Pre-Compile

3.3.1.14.1 FlsLoaderPFlash1WriteProt
Table 286 Specification for FlsLoaderPFlash1WriteProt

Name	FlsLoaderPFlash1WriteProt		
Description	Configuration of PFlash bank 1 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.15 Container: FlsLoaderPFlash2ProtConfig

Container for configuring PFlash bank 2 protection. Container is available only if PFlash bank 2 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.15.1 FlsLoaderPFlash2WriteProt
Table 287 Specification for FlsLoaderPFlash2WriteProt

Name	FlsLoaderPFlash2WriteProt		
Description	Configuration of PFlash bank 2 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

FLSLOADER driver**Table 287 Specification for FlsLoaderPFlash2WriteProt (continued)**

Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
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: FlsLoaderPFlash3ProtConfig

Container for configuring PFlash bank 3 protection. Container is available only if PFlash bank 3 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.16.1 FlsLoaderPFlash3WriteProt**Table 288 Specification for FlsLoaderPFlash3WriteProt**

Name	FlsLoaderPFlash3WriteProt		
Description	Configuration of PFlash bank 3 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

FLSLOADER driver
Table 288 Specification for FlsLoaderPFlash3WriteProt (continued)

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

3.3.1.17 Container: FlsLoaderPFlash4ProtConfig

Container for configuring PFlash bank 4 protection. Container is available only if PFlash bank 4 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

3.3.1.17.1 FlsLoaderPFlash4WriteProt

Table 289 Specification for FlsLoaderPFlash4WriteProt

Name	FlsLoaderPFlash4WriteProt		
Description	Configuration of PFlash bank 4 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.18 Container: FlsLoaderPFlash5ProtConfig

Container for configuring PFlash bank 5 protection. Container is available only if PFlash bank 5 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

FLSLOADER driver**3.3.1.18.1 FlsLoaderPFlash5WriteProt****Table 290 Specification for FlsLoaderPFlash5WriteProt**

Name	FlsLoaderPFlash5WriteProt		
Description	Configuration of PFlash bank 5 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

3.3.1.19 Container: FlsLoaderPFLASHConfig

This container contains the configuration parameters and sub-containers for configuration of PFlash.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

3.3.2 Functions - Type definitions**3.3.2.1 FlsLoader_AddressType****Table 291 Specification for FlsLoader_AddressType**

Syntax	FlsLoader_AddressType	
Type	uint32	
File	FlsLoader.h	
Range	0 to 4294967295	Target address in Flash
Description	This type specifies the logical destination address of Flash in PFlash or DFlash	
Source	IFX	

FLSLOADER driver**3.3.2.2 FlsLoader_CallOutFunc****Table 292 Specification for FlsLoader_CallOutFunc**

Syntax	FlsLoader_CallOutFunc
Type	Pointer to a function of type void Function_Name (void)
File	FlsLoader.h
Description	Call out function to application which is called at every user defined time rate
Source	IFX

3.3.2.3 FlsLoader_ConfigType**Table 293 Specification for FlsLoader_ConfigType**

Syntax	FlsLoader_ConfigType
Type	void
File	FlsLoader.h
Range	None
Description	This defines the void configuration type as the module supports single configuration variant
Source	IFX

3.3.2.4 FlsLoader_LengthType**Table 294 Specification for FlsLoader_LengthType**

Syntax	FlsLoader_LengthType	
Type	uint32	
File	FlsLoader.h	
Range	0 to 4294967295	Length information for write and erase operations
Description	This type specifies length information for write and erase operations as following: Write: Number of bytes to be written Erase: Number of logical sectors to be erased	
Source	IFX	

3.3.2.5 FlsLoader_ReturnType**Table 295 Specification for FlsLoader_ReturnType**

Syntax	FlsLoader_ReturnType	
Type	uint32	
File	FlsLoader.h	
Range	0 - FLSLOADER_E_OK	Successful execution

FLSLOADER driver
Table 295 Specification for FlsLoader_ReturnType (continued)

	1 - FLSLOADER_E_NOT_OK	Development error occurred
	2 - FLSLOADER_E_LOCKED	Sectors are read/write protected
	3 - FLSLOADER_E_ROMVERSION	All sectors are protected under OTP
	5 - FLSLOADER_E_BUSY	Device is busy
Description	This specifies the various Return types that can be specified by the APIs. This type is used for the errors detected by the APIs	
Source	IFX	

3.3.2.6 FlsLoader_ValueType

Table 296 Specification for FlsLoader_ValueType

Syntax	FlsLoader_ValueType	
Type	uint32	
File	FlsLoader.h	
Range	0 to 4294967295	Password (8 words)
Description	The type specifies values for Flash (PFlash or DFlash) protection password (8 words).	
Source	IFX	

3.3.3 Functions - APIs

This section describes all the APIs available to the upper layer.

3.3.3.1 FlsLoader_DeInit

Table 297 Specification for FlsLoader_DeInit API

Syntax	FlsLoader_ReturnType FlsLoader_DeInit (void)	
Service ID	0x30	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-

FLSLOADER driver**Table 297 Specification for FlsLoader_DeInit API (continued)**

Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Successful execution. FLSLOADER_E_BUSY: Flash is busy with erase/write operation. FLSLOADER_E_NOT_OK: Development error occurred.
Description	This function de-initializes the Flash module. This Function sets the registers to their default state and executes the reset to read command.	
Source	IFX	
Error handling	DET: FLSLOADER_E_BUSY: API service called while driver still busy. FLSLOADER_E_UNINIT: APIs are invoked without initialization of the driver. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FlsLoaderDelInitApi	
User hints	-	

3.3.3.2 FlsLoader_Erase**Table 298 Specification for FlsLoader_Erase API**

Syntax	FlsLoader_ReturnType FlsLoader_Erase (const FlsLoader_AddressType TargetAddress, const FlsLoader_LengthType Length)	
Service ID	0x32	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress Length	Target address in Flash memory. It should be aligned to the following sector sizes of the selected Flash for erase. PFlash: 16 Kbyte DFlash: 4 Kbyte Number of Flash (PFlash or DFlash) sectors to be erased. Note: Number of sectors should lie within single Flash bank. Erase operation across the Flash banks is not supported.
Parameters (out)	-	-
Parameters (in - out)	-	-

FLSLOADER driver
Table 298 Specification for FlsLoader_Erase API (continued)

Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Successful completion. FLSLOADER_E_BUSY: Flash is busy with erase/write operation. FLSLOADER_E_NOT_OK: DET error, Sequence error, Erase verify error, Program verify error (for PFlash), Protection error or Operation error occurred. FLSLOADER_E_LOCKED: Sector is protected (If FlsLoaderEnableLockCheck is enabled).
Description	This function erases the logical sectors of the internal Flash. The completion of this operation is denoted by clearing of busy status flag or error.	
Source	IFX	
Error handling	DET: FLSLOADER_E_PARAM_ADDRESS: API service called with wrong address. FLSLOADER_E_PARAM_LENGTH: API service called with wrong length. FLSLOADER_E_UNINIT: APIs are invoked without initialization of the driver. FLSLOADER_E_BUSY: API service called while driver still busy. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FlsLoaderEnableLockCheck	
User hints	-	

3.3.3.3 FlsLoader_GetVersionInfo
Table 299 Specification for FlsLoader_GetVersionInfo API

Syntax	<pre>void FlsLoader_GetVersionInfo (Std_VersionInfoType * const VersionInfoPtr)</pre>	
Service ID	0x35	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer to where the version information has to be stored
Parameters (in - out)	-	-

FLSLOADER driver
Table 299 Specification for FlsLoader_GetVersionInfo API (continued)

Return	void	-
Description	This function provides the version information of the FLSLOADER driver	
Source	IFX	
Error handling	<p>DET:</p> <p>FLSLOADER_E_PARAM_POINTER: API service called with 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	FlsLoaderVersionInfoApi	
User hints	None	

3.3.3.4 FlsLoader_Init
Table 300 Specification for FlsLoader_Init API

Syntax	<pre>FlsLoader_ReturnType FlsLoader_Init (const FlsLoader_ConfigType * const Address)</pre>	
Service ID	0x2F	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Address	NULL pointer because the driver supports single configuration variant
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	<p>FLSLOADER_E_OK: Successful execution</p> <p>FLSLOADER_E_ROMVERSION: All sectors are protected with OTP or WOP protection</p> <p>FLSLOADER_E_NOT_OK: Development error occurred</p>
Description	This function initializes the Flash module and checks if all the Flash sectors are configured as ROM (OTP or WOP protected)	
Source	IFX	

FLSLOADER driver**Table 300 Specification for FlsLoader_Init API (continued)**

Error handling	<p>DET:</p> <p>FLSLOADER_E_PARAM_IGNORED: API service called with not 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	None

3.3.3.5 FlsLoader_Lock**Table 301 Specification for FlsLoader_Lock API**

Syntax	<pre>FlsLoader_ReturnType FlsLoader_Lock (void)</pre>	
Service ID	0x33	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	<p>FLSLOADER_E_OK: Protections are installed successfully.</p> <p>FLSLOADER_E_BUSY: Flash is busy with erase/write operation.</p> <p>FLSLOADER_E_NOT_OK: DET error or failure (Sequence error, Protection error, Operation error, Program verify error or Erase verify error) reported while installing protections for at least one among DFlash0 read/write or PFlash write or PFlash OTP/WOP protections.</p> <p>FLSLOADER_E_LOCKED: Protections for DFlash0 read/write, PFlash write and PFlash WOP/OTP are already installed.</p>

FLSLOADER driver
Table 301 Specification for FlsLoader_Lock API (continued)

Description	This function locks (protect) the internal PFlash and DFlash0 of micro-controller with pre-configured protections. Following protection configurations are supported by the driver: -DFlash: Read protection, write protection. -DFlash protections are configurable at bank level. -PFlash: Write protection, write once protection (WOP), one time programmable (OTP) protection. -PFlash protections are configurable at sector level. However a bank and its corresponding sectors to be protected shall have same protection configured.
Source	IFX
Error handling	DET: FLSLOADER_E_UNINIT: APIs are invoked without initialization of the driver. FLSLOADER_E_BUSY: API service called while driver still busy. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	FlsLoaderLockUnlockApi
User hints	None

3.3.3.6 FlsLoader_UnLock

Table 302 Specification for FlsLoader_UnLock API

Syntax	<pre>FlsLoader_ReturnType FlsLoader_UnLock (const FlsLoader_AddressType TargetAddress, const FlsLoader_ValueType * const Password)</pre>	
Service ID	0x34	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress Password	UCB address of corresponding Flash to be unlocked. 0xAF402000 - PFlash UCB 0xAF402200 - DFlash UCB Pointer to the 4 double word (256 bit) UCB password of corresponding Flash to be unlocked.
Parameters (out)	-	-

FLSLOADER driver
Table 302 Specification for FlsLoader_UnLock API (continued)

Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Successful completion. FLSLOADER_E_BUSY: Flash is busy with erase/write operation. FLSLOADER_E_NOT_OK: DET error, Operation error, Sequence error or Protection error occurred.
Description	<p>This function is used to unlock the internal PFlash and DFlash0 of the micro-controller from active protection. It temporarily (until next controller reset) disables the current active read or write protection. A wrong password results in protection error.</p> <ul style="list-style-type: none"> - DFlash0 can be unlocked from read and write protections. - PFlash can be unlocked from write protection. WOP and OTP cannot be unlocked. 	
Source	IFX	
Error handling	<p>DET:</p> <p>FLSLOADER_E_UNINIT: APIs are invoked without initialization of the driver.</p> <p>FLSLOADER_E_PARAM_ADDRESS: API service called with wrong address.</p> <p>FLSLOADER_E_BUSY: API service called while driver still busy.</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	FlsLoaderLockUnlockApi	
User hints	None	

3.3.3.7 FlsLoader_Write

Table 303 Specification for FlsLoader_Write API

Syntax	<pre>FlsLoader_ReturnType FlsLoader_Write (const FlsLoader_AddressType TargetAddress, const FlsLoader_LengthType Length, const uint8 * const SourceAddressPtr)</pre>
Service ID	0x31
Sync/Async	Synchronous
ASIL Level	QM
Re-entrancy	Non Reentrant

FLSLOADER driver
Table 303 Specification for FlsLoader_Write API (continued)

Parameters (in)	TargetAddress Length SourceAddressPtr	Target address in Flash memory. It should be aligned to the following page sizes of the selected Flash for write. PFlash: 32 bytes DFlash: 8 bytes Number of bytes to be written. It should be multiple of the following page sizes of the selected Flash for write. PFlash: 32 bytes DFlash: 8 bytes Pointer to source data buffer
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Successful execution. FLSLOADER_E_BUSY: Flash is busy with erase/write operation. FLSLOADER_E_NOT_OK: DET error, Sequence error, Program verify error, Protection error or Operation error occurred. FLSLOADER_E_LOCKED: Sector is protected (If FlsLoaderEnableLockCheck is enabled).
Description	This function is used to program a page of internal Flash. Sectors of PFlash and DFlash can be programmed.	
Source	IFX	
Error handling	DET: FLSLOADER_E_PARAM_ADDRESS: API service called with wrong address. FLSLOADER_E_PARAM_LENGTH: API service called with wrong length. FLSLOADER_E_UNINIT: APIs are invoked without initialization of the driver. FLSLOADER_E_BUSY: API service called while driver still busy. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	FlsLoaderEnableLockCheck	
User hints	None	

3.3.4 Notifications and Callbacks

The driver by itself does not implement any notifications. However, if the call out feature is enabled by user, it provides a call out to application while looping for status during write and erase operations.

FLSLOADER driver

3.3.5 Scheduled functions

This driver does not support any scheduled functions.

3.3.6 Interrupt service routines

This driver does not configure any interrupts for its operation.

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.

Table 304 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
API service called while driver still busy.	IFX	FLSLOADER_E_BUSY=5	FlsLoader_Erase, FlsLoader_Write, FlsLoader_UnLock, FlsLoader_Lock, FlsLoader_DeInit
API service called with wrong address.	IFX	FLSLOADER_E_PARAM_ADDRESS=3	FlsLoader_Erase, FlsLoader_Write, FlsLoader_UnLock
API service called with not a NULL pointer.	IFX	FLSLOADER_E_PARAM_IGNORED=0	FlsLoader_Init
API service called with wrong length.	IFX	FLSLOADER_E_PARAM_LENGTH=2	FlsLoader_Erase, FlsLoader_Write
API service called with NULL pointer.	IFX	FLSLOADER_E_PARAM_POINTER=6	FlsLoader_GetVersionInfo
APIs are invoked without initialization of the driver.	IFX	FLSLOADER_E_UNINIT=4	FlsLoader_DeInit, FlsLoader_Erase, FlsLoader_Write, FlsLoader_UnLock, FlsLoader_Lock

3.3.7.2 Production errors

The driver does not report any production errors.

3.3.7.3 Safety errors

The driver does not report any safety errors.

3.3.7.4 Runtime errors

The driver does not report any runtime errors.

FLSLOADER driver

3.3.8 Deviations and limitations

The section describes the deviations and limitations from software specification.

3.3.8.1 Deviations

Currently there are no deviations for the FLSLOADER driver.

3.3.8.2 Limitations

The section describes the limitations from software specification.

Table 305 Known limitations

Reference	Limitation
PFlash erase	The maximum PFlash sectors that can be erased by erase API is 192. If length is more than 192 sectors, the application should call erase API multiple times as per the erase length.
Write to WOP (Write Once Protection) protected sector	The driver does not support write to the PFlash sectors which are protected with WOP.
Timeout values for write and erase operations	All timeout values used by the FLSLOADER are calculated assuming the FSI operates at 100 MHz.
Increased timeout durations	When the DFlash0/PFlash is accessed by the FlsLoader on the TriCore side and DFlash1/PFlash is accessed simultaneously on the HSM side, 5% additional time is taken for write operations and the duration of erase operations increases by about 15%. Timeout calculations are performed assuming the DFlash0/PFlash and DFlash1/PFlash are accessed simultaneously from the TriCore side and the HSM side. Therefore, if there is no simultaneous access, then the timeouts will be delayed.

3.3.9 Unsupported hardware features

The following hardware features of the DMU are not supported:

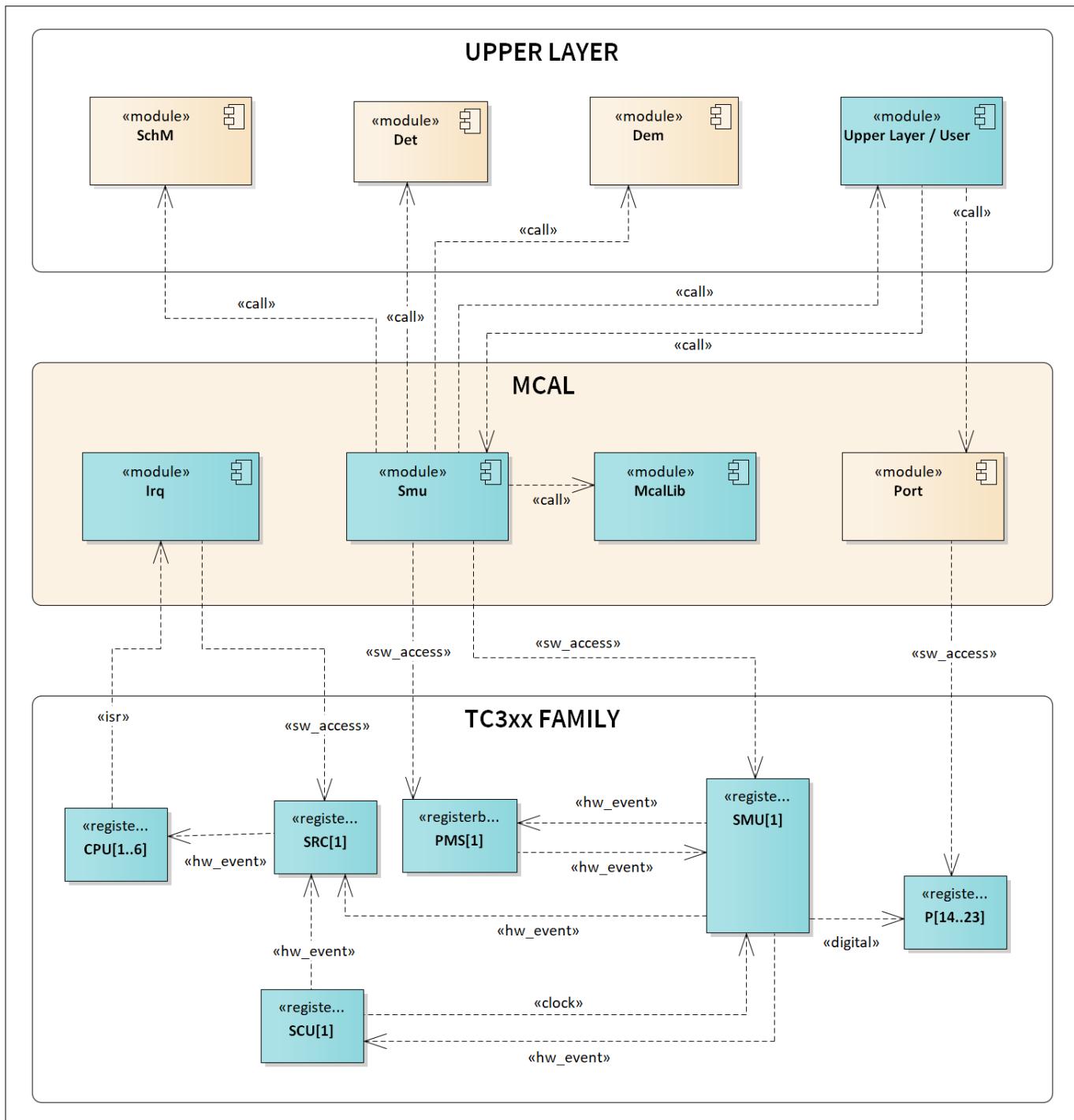
- Compliment sensing mode for DFlash0
- Suspend and resume Flash operations
- ECC error reporting to safety management unit (SMU)
- Interrupt service request

SMU driver**4 SMU driver****4.1 User information****4.1.1 Description**

The SMU driver is an abstraction of the SMU peripheral in the AURIX™ microcontroller family. The SMU peripheral centralizes all the alarm signals related to different hardware safety mechanisms. Each alarm can trigger internal actions and/or notify the presence of faults to the external world through a fault signaling protocol. The SMU driver is active before any of the peripherals are active. Therefore, SMU initialization and de-initialization must be called only on the master core. However, as it encompasses the hardware safety mechanisms distributed across all the cores, the SMU runtime services will be accessible from all cores.

4.1.2 Hardware-software mapping

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

SMU driver**Figure 40** Mapping of hardware-software interfaces**4.1.2.1 PMS: primary hardware peripheral****Hardware functional features**

The SMU driver uses the PMS for SMU_stdby mode operation. The SMU_stdby operates in fBCK frequency provided by the PMS. The SMU_stdby is configurable, that is, it can be put to an idle state. All the alarms in the fBCK domain are forwarded to both SMU_stdby and SMU_core.

The key hardware functional features used by the driver are:

SMU driver

- Alarm status with respect to group 20 and 21 can be queried and cleared
- Monitoring of SMU_core through a runtime service for any alarms

The unsupported feature of the PMS is:

- SMU_stby built-in self-test

Users of the hardware

The PMS is used by the SMU and MCU drivers. The SMU driver exclusively utilizes the SMU_stdby related registers of the PMS. The MCU driver does not utilize the SMU_stdby related registers and hence resource conflict does not happen.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from the PMS are not used by the SMU driver.

4.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The SMU driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the FSPB clock signal for functioning. The driver also depends on the SCU IP for external EMS alarm and watchdog timeout.

The internal alarm reactions resulting from the alarm events are interfaced to the SCU. These interface signals can be of the following types:

- NMI request
- Reset request
- CPU reset request
- Emergency stop request
- Run state request

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

Associated SMU alarms, which can detect any hardware safety mechanism failures, exist.

Hardware events

Hardware events from the SCU are not used by the SMU driver.

4.1.2.3 Port: dependent hardware peripheral

Hardware functional features

The FSP status is routed to the SMU through the port pads. The port pins are driven to the GPIO or SMU mode where the SMU uses the port pins to trigger external reaction mechanism using the FSP.

The SMU driver can also activate the emergency stop feature.

Users of the hardware

The port pads (P33.8 and P33.10) are configured and enabled by the user software through the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

SMU driver

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

4.1.2.4 **SMU: primary hardware peripheral**

Hardware functional features

The SMU driver uses the SMU for providing a generic interface to manage the behavior of the microcontroller under the presence of faults. The SMU centralizes all the alarm signals related to different hardware and software-based safety mechanisms. Each alarm can be individually configured to trigger internal actions and/or to notify externally the presence of faults through a fault signaling protocol.

The SMU provides an abstracted interface to the user to access the SMU peripheral.

The SMU peripheral has two parts: the SMU_core and the SMU_stby. Both operate in different clock and power domains. One of the most important features of SMU_stby is to monitor the correct functioning of SMU_core. The status of any error is triggered as an alarm from SMU_core to SMU_stby.

The key hardware functional features used by the driver are:

- Alarm handling: Setting, querying, clearing of alarms and setup of alarm reactions
- FSP handling: Activation, deactivation of the FSP to request the trigger of safe state
- Port emergency handling: Activation of port emergency stop
- Recovery timer handling: Setup, status query of recovery timers
- SMU state machine: Status query of the SMU state machine and invoking transition between states
- SMU Configuration Protection: Activation of permanent lock mechanism
- Smu_ActivatePES() software command
- Register monitoring feature for safety flip flop

The unsupported features of the SMU are:

None.

Modes or states of the SMU:

The SMU peripheral operates in three states: START, RUN and FAULT states. The application provides the services to invoke the transitions between the states.

The FSP driven by SMU has three states: Power-on, Fault and Fault Free states.

Users of the hardware

The SMU driver exclusively utilizes the SMU IP.

Hardware diagnostic features

The STS register can be read back to ensure the command has been successfully executed after writing the command to the CMD register.

Hardware events

The error status of the hardware and software safety mechanisms is indicated by the alarms. The reaction for these alarms is determined by the configuration parameters.

The SMU is connected to the interrupt router through three service request nodes SRC_SMUx (x = 0 to 2). Each service request can trigger an interrupt on the CPU 0, 1, 2, 3, 4 or 5. The number of CPUs depends on the derivative.

4.1.2.5 **SRC: dependent hardware peripheral**

Hardware functional features

The SMU driver depends on the interrupt router for raising an interrupt to the CPU based on the internal reactions configured for the alarm events. These interrupts are SMU IR request 0, SMU IR request 1 and SMU IR request 2.

SMU driver**Users of the hardware**

The service request nodes SRC_SMUx (x=0 to 2) are exclusively allocated to the SMU peripheral. The service request to be triggered is decided by the SMU peripheral using the interrupt generation set selected by the alarm that is IGSC0, IGSC1 or IGSC2. Each set is a code, which is a combination of the three service requests that need to be triggered; for example, IGSC0 = 011b implies that SRC_SMU0 and SRC_SMU1 service requests are triggered on alarm event.

Hardware diagnostic features

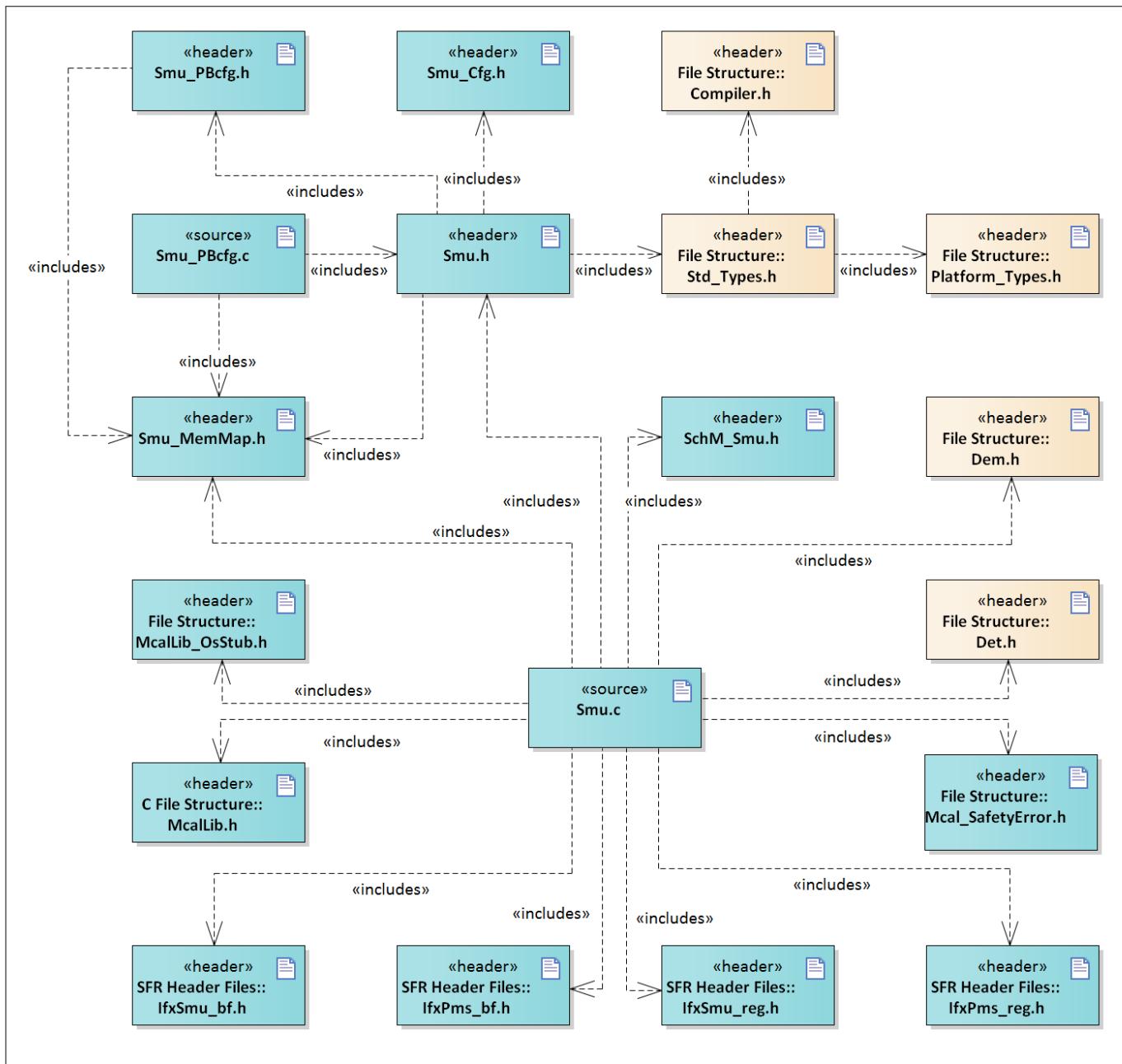
Not applicable.

Hardware events

Hardware events raised by the interrupt router are not used by the SMU driver.

4.1.3 File structure**4.1.3.1 C file structure**

The section provides details of the C files of the SMU driver.

SMU driver

Figure 41 **C file structure**
Table 306 **C file structure**

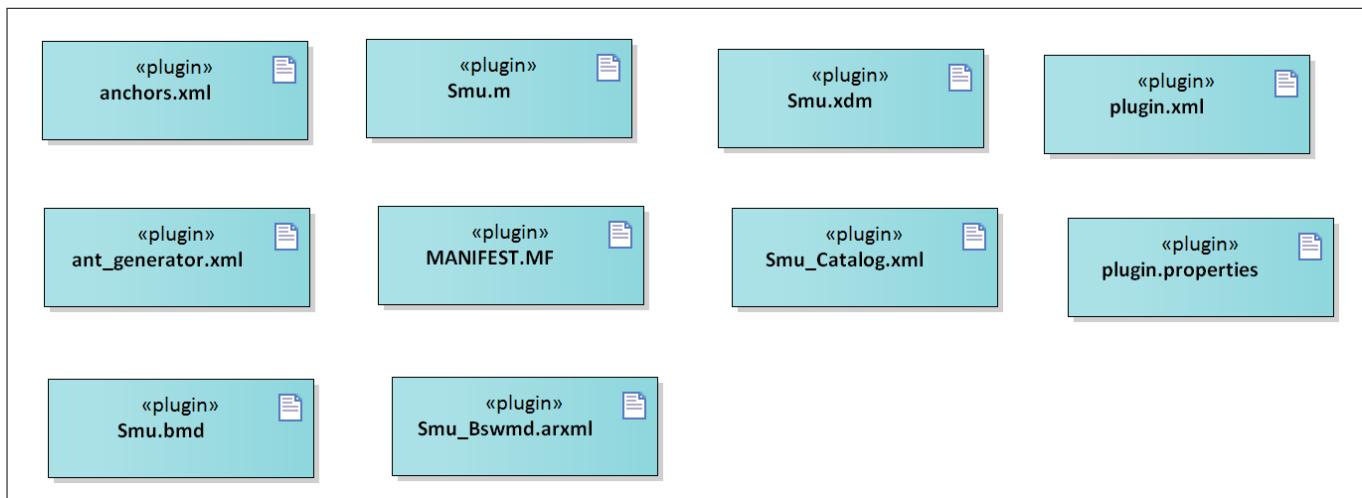
File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Dem.h	Provides the exported interfaces of Diagnostic Event Manager
Det.h	Provides the exported interfaces of Development Error Tracer
IfxPms_bf.h	SFR header file for Pms
IfxPms_reg.h	SFR header file for Pms
IfxSmu_bf.h	SFR header file for SMU
IfxSmu_reg.h	SFR header file for SMU

SMU driver
Table 306 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
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Smu.h	Header file contains the definitions of the SMU critical sections
Smu.c	Contains the source code of the SMU software module
Smu.h	Contains the data types and function prototypes to be exported
Smu_Cfg.h	Contains the pre-compile configuration for the SMU driver. The file implements all pre-processor directives
Smu_MemMap.h	Contains mapping of code and data (variables, constants) to specific memory sections for the SMU driver
Smu_PBcfg.c	Contains the post-build configuration for the SMU driver
Smu_PBcfg.h	Contains generated configuration data of user
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

The section provides details of the plugin files of the SMU driver.


Figure 42 Code generator plugin files
Table 307 Code generator plugin files

File name	Description
MANIFEST.MF	Tresos plugin support file containing the metadata for the SMU driver

SMU driver
Table 307 Code generator plugin files (continued)

File name	Description
Smu.bmd	AUTOSAR format XML data model schema file for the SMU driver
Smu.m	Code template macro file for the SMU driver
Smu.xdm	Tresos format XML data model schema file for the SMU driver
Smu_Bswmd.arxml	AUTOSAR format module description file for the SMU driver
Smu_Catalog.xml	AUTOSAR format catalog file for the SMU driver
anchors.xml	Tresos anchors support file for the SMU 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 SMU driver
plugin.xml	Tresos plugin support file for the SMU driver

4.1.4 Integration hints

The section lists the key points that an integrator or user of the SMU driver must consider.

4.1.4.1 Integration with AUTOSAR stack

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

- **EcuM**

The EcuM module is not required for integrating SMU 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 `Smu_MemMap.h` file.

SMU driver

The `Smu_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:

```

/* User Pragma to be placed here for LMU RAM NC*/
#undef SMU_START_SEC_INIT_VAR_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR
#elif defined SMU_STOP_SEC_INIT_VAR_ASIL_B_GLOBAL_32
/* User Pragma for LMU RAM NC here */
#undef SMU_STOP_SEC_INIT_VAR_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR

/*Configuration data sections-- to be placed in PFx*/
#elif defined SMU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/* User pragmas to be placed here for PF0 */
#undef SMU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR
#elif defined SMU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/* User pragmas to be placed here for PFx */
#undef SMU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

/* Code Section ----- to be placed in PFx*/
#elif defined SMU_START_SEC_CODE_ASIL_B_GLOBAL
/* User Pragma to be placed here */
#undef SMU_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#elif defined SMU_STOP_SEC_CODE_ASIL_B_GLOBAL
/* User Pragma to be placed here */
#undef SMU_STOP_SEC_CODE_ASIL_B_GLOBAL
#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 SMU driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the SMU 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 need 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 SMU driver reports all the production errors to the DEM modules through the `Dem_ReportErrorStatus()` API. The user of the SMU 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 need to be replaced with a complete DEM module during the integration phase.

Note: Reentrancy of the `Smu_ClearAlarmStatus()`, `Smu_SetAlarmStatus()`, `Smu_ReleaseFSP()`, `Smu_ActivateFSP()`, `Smu_RTStop()`, `Smu_ActivateRunState()`, `Smu_ActivatePES()`,

SMU driver

`Smu_CoreAliveTest()` and `Smu_RegisterMonitor()` APIs is dependent on the reentrancy of the `Dem_ReportErrorStatus()` API. As per the design, the APIs of the module are reentrant 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 SMU driver uses the exclusive areas defined in the `SchM_Smu.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the SMU driver are:

- CmdAccess
- DriverAccess

The `SchM_Smu.h` and `SchM_Smu.c` files are provided in the MCAL package as an example code and need to be updated by the integrator. The user must implement the SchM functions defined by the SMU 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*/
#include "IFX_Os.h"
#include "SchM_Smu.h"

void SchM_Enter_CmdAccess(void)
{
/*Suspend all interrupts*/
    SuspendAllInterrupts();
}

void SchM_Exit_CmdAccess(void)
{
/*Resume all interrupts*/
    ResumeAllInterrupts();
}

void SchM_Enter_DriverAccess(void)
{
/*Suspend all interrupts*/
    SuspendAllInterrupts();
}

void SchM_Exit_DriverAccess(void)
{
/*Resume all interrupts*/
    ResumeAllInterrupts();
}
```

- **Safety error**

The SMU 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.

SMU driver

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

- **Notification and callbacks**

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

The SMU driver does not require configuration of any interrupts. The interrupts triggered because of alarm action have to be handled by the user.

4.1.4.2 Multicore and Resource Manager

The SMU driver supports execution of its APIs in parallel from all CPU cores. The following are the key points to be considered with respect to multicore in the SMU driver:

- The runtime services of the SMU driver will be accessible by all cores.
- The hardware and software safety mechanism are associated with specific alarm groups and positions and cannot be reallocated by software or configuration. Hence, the SMU driver does not have any core-specific resource allocation.
- The SMU initialization and de-initialization must be called only from the master core. In case the `Smu_Init()` or `Smu_DeInit()` API is called from any core other than the master core then the API will report an error `E_NOT_OK`. In case DET is enabled, a DET error `SMU_E_CORE_MISMATCH` will also be reported.
- The `Smu_LockConfigRegs`, `Smu_ActivateRunState` and `Smu_ReleaseFSP` APIs shall be invoked from only one core at a time. Invoking from multiple cores simultaneously may lead to inconsistent results or significantly high API execution time due to CPU resource starvation.
- 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). The following should be considered by the user to ensure better performance of the SMU driver:

Code section:

The executable code of the SMU 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.

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.

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.

4.1.4.3 MCU support

The SMU driver does not use any services provided by the MCU driver.

SMU driver**4.1.4.4 Port support**

The PORT driver configures the port pins of the entire microcontroller. The user must configure the port pins 33.8 and 33.10 used by the SMU driver for FSP through the PORT configuration.

4.1.4.5 DMA support

The SMU driver does not use any services provided by the DMA driver.

4.1.4.6 Interrupt connections

The SMU driver does not implement any interrupt handlers. The internal reactions, which are configured because of alarm events, which trigger an interrupt, need to be handled by the user.

SMU driver

4.1.4.7 Example usage

Initializing and de-initializing the SMU driver

The SMU driver is initialized by calling the `Smu_Init()` API. The user must call the `Smu_Init()` API from the master core only. When the `Smu_Init()` API is called from a core other than master core, the API returns an error. The same criteria apply to the `Smu_DeInit()` API. Also in order to check the initialization values, the `Smu_InitCheck()` API can be used. However the `Smu_InitCheck()` API is enabled only when the `SmuInitCheckApi` parameter is enabled. The SMU driver can be initialized and de-initialized as shown follows:

```
#include "Smu_Test.h"
/* Initialize SMU*/
Return = Smu_Init(&(Smu_Config));
/*Check for initialiazation values*/
#if(SMU_INIT_CHECK_API==STD_ON)
Return = Smu_InitCheck(&(Smu_Config));
#endif
/*Call SMU driver functions*/
/*.....*/
/*Deinitialize the driver*/
Return = Smu_DeInit();
```

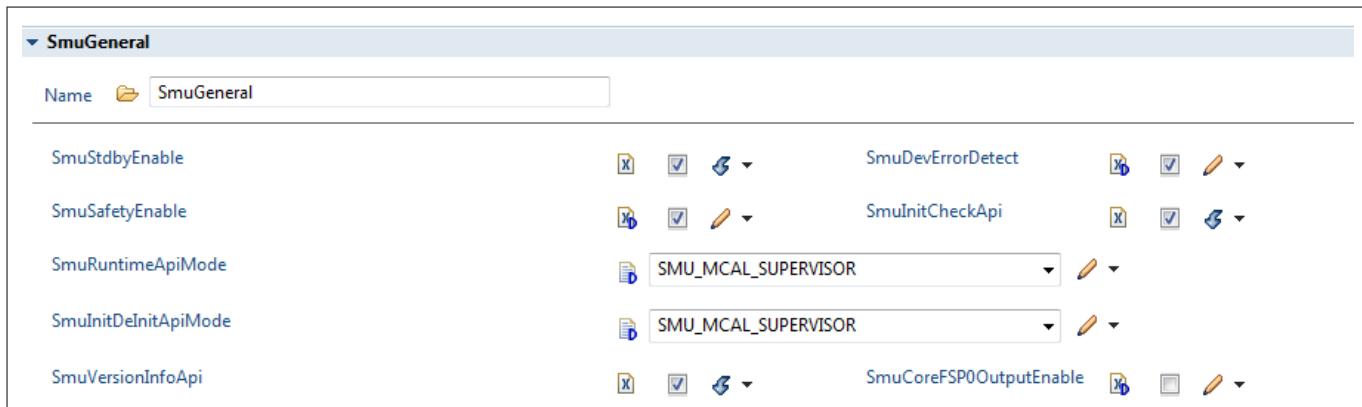


Figure 43 Example configuration for enabling Smu_stdby, InitCheck API, DET, version info API and user mode

In case the `SmuSafetyEnable` parameter is enabled, the user needs to configure the DEM reporting and add the parameter to the `SmuDemEventParameterRefsConf` container.

Configuration register locking

The SMU configuration registers can be protected from unintended access in two ways:

- Temporary lock: The SMU configuration register protection is disabled temporarily to write into the SMU configuration registers and then again enabled.
- Permanent lock: The SMU configuration register protection is enabled to prevent any writing into the SMU configuration register and can be disabled only after application reset.

The `Smu_LockConfigRegs()` API enables the permanent lock on the configuration registers. In case the `Smu_LockConfigRegs()` API fails to turn on the permanent lock and safety error check is enabled, then a DET

SMU driver

is reported to let the user know that the configuration registers could not be permanently locked. An example usage is shown as follows:

```
Std_ReturnType Result = E_NOT_OK;
ResultLockTest = Smu_LockConfigRegs();
```

Alarm status

The SMU driver provides services to set, clear and get the alarm status by using the `Smu_SetAlarmStatus()`, `Smu_ClearAlarmStatus()` and `Smu_GetAlarmStatus()` APIs. An example usage is shown as follows:

```
Smu_SetAlarmStatus is valid only for Smu_core.
ResultAlarmStatus = Smu_ClearAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_0);
if (E_OK == ResultAlarmStatus)
{
    ResultAlarmStatus = Smu_SetAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_0);
    if (E_OK == ResultAlarmStatus)
    {
        ResultAlarmStatus = Smu_GetAlarmStatus(SMU_ALARM_GROUP10, &AlarmStatus);
        if ((E_OK == ResultAlarmStatus) && (0x01U == (AlarmStatus & 0x01)))
        {
            ResultAlarmStatus = Smu_ClearAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_0);
            if (E_OK == ResultAlarmStatus)
            {
                ResultAlarmStatus = Smu_GetAlarmStatus(SMU_ALARM_GROUP10, &AlarmStatus);
            }
        }
    }
}
```

Alarm action

The SMU driver provides services to set and get the alarm actions by using the `Smu_SetAlarmAction()` and `Smu_GetAlarmAction()` APIs.

Two kinds of alarm actions can be configured for SMU_core: internal action and external action. These are configured as follows:

SmuCoreAlarmBehavior		SmuCore...		SmuCoreAlarmIntBeh		SmuCoreAlarmFSP	
Index	Name						
0	SmuCoreAlarmBehavior_0		0	SMU_NA_INT_ACTION		SMU_ALARM_FSP_ENABLED	
1	SmuCoreAlarmBehavior_1		1	SMU_IGCS0_INT_ACTION		SMU_ALARM_FSP_DISABLED	
2	SmuCoreAlarmBehavior_2		2	SMU_NMI_INT_ACTION		SMU_ALARM_FSP_ENABLED	
3	SmuCoreAlarmBehavior_3		3	SMU_NA_INT_ACTION		SMU_ALARM_FSP_DISABLED	

Figure 44 Internal and external action settings for Smu_core

For SMU_stdby, the internal reaction is by default `SMU_NA_INT_ACTION`. Only external action can be configured.

For both SMU_core and SMU_stdby, in case an alarm action is tried to be configured for a reserved alarm group, then the configuration will throw an error as shown in the following figure:

SMU driver

SmuStdbyAlarmBehavior		Index	Name	SmuStdby...	SmuStdbyAlarmFSP
0	SmuStdbyAlarmBehavior_0	0	SMU_ALARM_FSP_ENABLED		
1	SmuStdbyAlarmBehavior_1	1	SMU_ALARM_FSP_DISABLED		
2	SmuStdbyAlarmBehavior_2	2	SMU_ALARM_FSP_DISABLED		
3	SmuStdbyAlarmBehavior_3	3	SMU_ALARM_FSP_DISABLED		
4	SmuStdbyAlarmBehavior_4	4	SMU_ALARM_FSP_ENABLED		

Figure 45 External action setting for Smu_stdby

Register monitoring

The `Smu_RegisterMonitor()` API can be used to enable the SFF tests for the protected registers of particular modules and retrieve the results. The input parameter passed to the API strictly has to follow the sequence of modules as per the bit fields of the RMCTL register. However, the user should ensure that the module for which the SFF test is being requested is present in the derivative being used.

Smu_core alive test

The `Smu_CoreAliveTest()` API provide the means to execute the `SMU_AliveTest` command that checks the `smu_core_alive` signal. For the `smu_core_alive` test to run through the `SMU_core` command sequence, the `SMU_stdby` shall be enabled and the `SMU_core` shall be in `START` state. In case the `SMU_stdby` is not enabled or the `SMU_core` is not in `START` state, the `Smu_CoreAliveTest()` API returns `E_NOT_OK`.

The user shall read the status flags for the `SMU_core alive` alarm (alarm 16 of alarm group 21) using the `Smu_GetAlarmStatus()` API to check the result of the `smu_core_alive` test after execution of the `Smu_CoreAliveTest()` API. The user shall clear the status of the `SMU_core alive` alarm (alarm 16 of alarm group 21) using the `Smu_ClearAlarmStatus()` API, after checking the result of the `smu_core_alive` test so that further alarm detection is possible.

The following API sequence should be followed execute the `SMU_AliveTest` command:

```

/*Execute smu_core_alive test */
ResultCoreAliveTest = Smu_CoreAliveTest();
/*Read status of alarm 16 of alarm group 21 to check the result of the
smu_core_alive test*/
ResultGetAlarmStatus = Smu_GetAlarmStatus(SMU_ALARM_GROUP21, &AlarmStatus16);
/*Clear status of alarm 16 of alarm group 21 after checking the result of
smu_core_alive test so that further alarm detection is possible*/
ResultClearAlarmStatus = Smu_ClearAlarmStatus(SMU_ALARM_GROUP21, SMU_ALARM_16);

```

Alarm execution status

The `Smu_GetAlarmExecutionStatus()` API can be used to retrieve the alarm execution status. Once retrieved, the execution status can be cleared using the `Smu_ClearAlarmExecutionStatus()` API. The alarm reactions are not triggered if the alarm execution status for the particular alarm is not cleared.

SMU driver

An example code usage is shown as follows:

```
Std_ReturnType RetVal = E_NOT_OK;
/*Get the alarm execution status*/
RetVal = Smu_GetAlarmExecutionStatus(ExecReq, &ExecStatus);

if(RetVal == E_OK)
{
/*Clear the alarm execution status of the particular alarms as requested*/
RetVal = Smu_ClearAlarmExecutionStatus(ExecStatusReq);
}
```

FSP handling and Smu_core State Machine

For FSP handling, the glitch filter settings, output pin direction and enable has to set by the user (refer configuration for more details). FSP can be operated by choosing one of the three signaling modes:

- Time switching protocol
- Dual rail protocol
- Bi-stable protocol

FSP output pins can be enabled through configuration for Smu_stdby. PES can be enabled or disabled while using FSP. The prescalar, signaling mode, fault state duration can be selected as per the configuration depicted in the following figure:

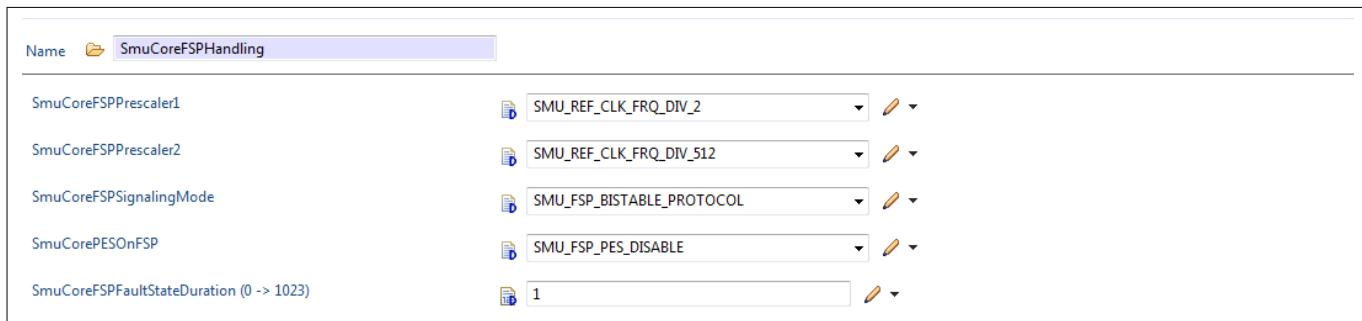


Figure 46 FSP setting for Smu_core

To enable the transition of FAULT to RUN state, the `SmuCoreEnableFaultToRunState` parameter shall be enabled. In addition, the external action with respect to the alarm group and position should be enabled. The internal and external reaction configuration is explained in section **Alarm action**

SMU driver

An example code usage:

```

Smu_CoreStateType SmuState;
/*Setup the error pin in SMU mode*/
ResultFSP = Smu_SetupErrorPin();
if(E_OK == ResultFSP)
{
    /*Set the alarm status of alarm group 10 and position 5*/
    ResultFSP = Smu_SetAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_5);
}
/*Activate FSP to indicate a fault state*/
ResultFSP = Smu_ActivateFSP();
if(E_OK == ResultFSP)
{
    /*Get the SMU state*/
    SmuState = Smu_GetSmuState();
    if(SMU_FAULT_STATE == SmuState)
    {
        /*In case it is FAULT state then release FSP to transition to RUN state*/
        ResultFSP = Smu_ReleaseFSP();
        if(E_OK == ResultFSP)
        {
            /*Release error pin to transition to GPIO mode*/
            ResultFSP = Smu_ReleaseErrorPin();
        }
    }
}
}
}
}

```

Smu_core has three states- START, RUN, FAULT.

Transition from START to RUN: The transition takes place by executing the SMU_core command to activate RUN state by calling the `Smu_ActivateRunState()` API.

Transition from FAULT to RUN: The transition takes place by executing the SMU_core command by calling the Smu_ReleaseFSP() API.

In order to indicate the FAULT state on the error pin, `Smu_ActivateFSP()` API is called.

SMU driver

An example code usage and sequence diagram for state machine and FSP handling is as follows:

```
/*Get the Smu_core state*/
SmuState = Smu_GetSmuState();

switch (SmuState)
{
    case SMU_START_STATE:
    {
        /*Activate the RUN state*/
        Result = Smu_ActivateRunState(SMU_RUN_COMMAND);
        break;
    }
    case SMU_FAULT_STATE:
    {
        /*In case it is FAULT state then release FSP to transition to RUN state*/
        Result = Smu_ReleaseFSP();
        break;
    }
    case SMU_RUN_STATE:
    {
        Result = E_OK;
        break;
    }
    default:
        break;
}
```

SMU driver

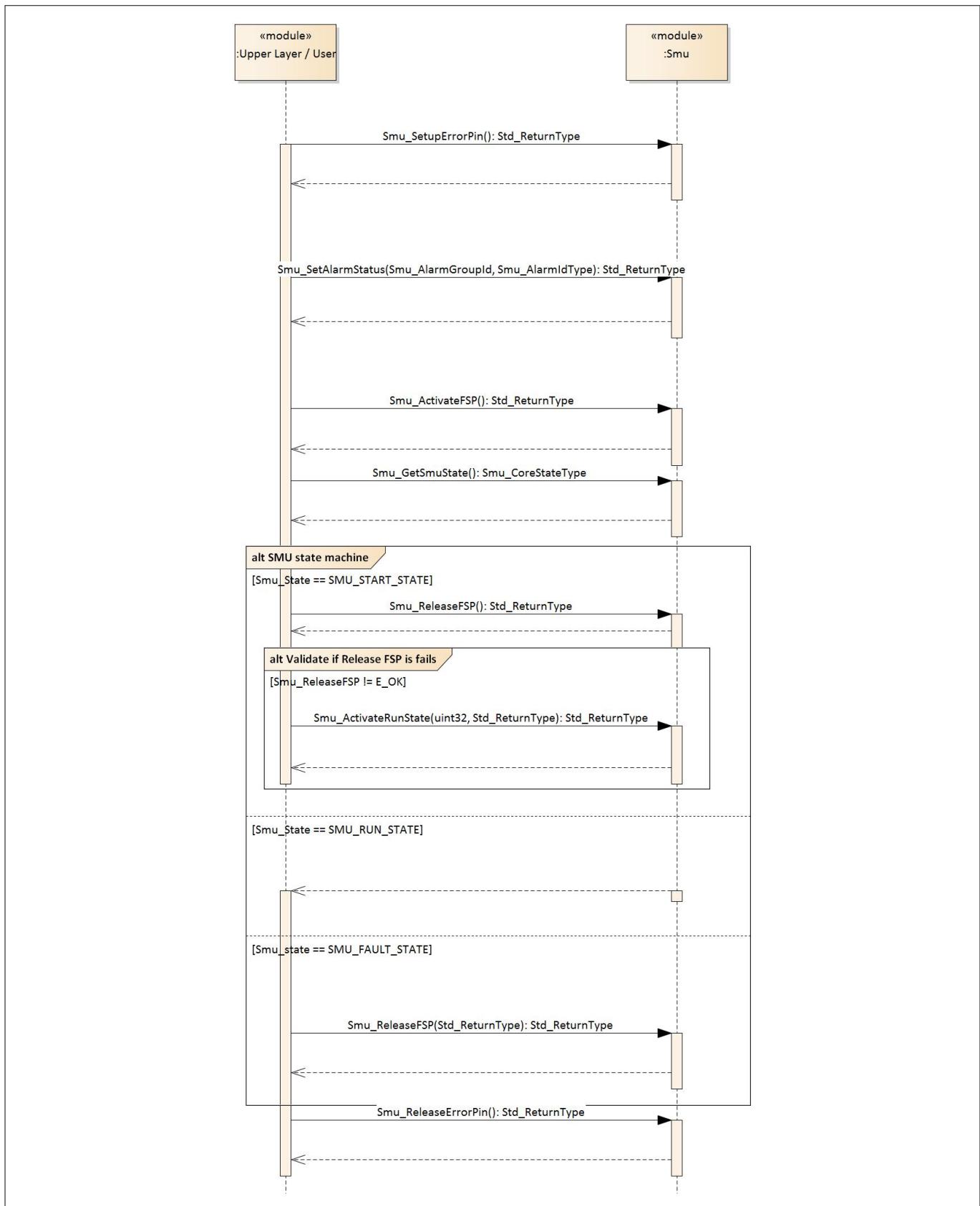


Figure 47 Sequence diagram depicting FSP handling and transition of the states of the Smu_core state machine

Smu_core recovery timer

SMU driver

The recovery timers are configured by first enabling the RT0 and RT1 and setting the RT duration. On enabling, the respective RT group configurations are enabled.

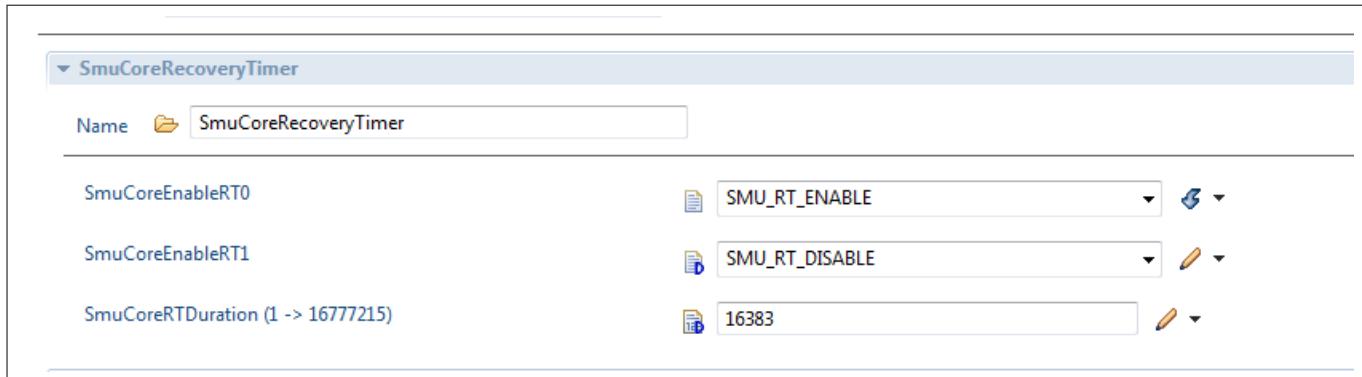


Figure 48 Enable the RT configuration

After the RT group configuration is enabled, the recovery timers can be assigned to the SMU_core alarm groups and positions.

SmuCoreRT0Alarm			
Index	Name	SmuCoreRT0AlarmGroupId	SmuCoreRT0...
0	SmuCoreRT0Alarm_0	SMU_ALARM_GROUP0	0
1	SmuCoreRT0Alarm_1	SMU_ALARM_GROUP1	2
2	SmuCoreRT0Alarm_2	SMU_ALARM_GROUP0	0
3	SmuCoreRT0Alarm_3	SMU_ALARM_GROUP0	0

Figure 49 Assigning alarm groups and positions to RT

In order to configure RT, the respective alarm group and position have to be configured for their internal action. In case it is a reserved alarm position, the user must take care of not assigning the alarm position to RT0 or RT1. The information of reserved alarm positions will be evident when an error is encountered while configuring the internal alarm action for that particular alarm group and position. Therefore, the RT configuration can take reference from the error as discussed.

SMU driver

The SMU driver provides the services to stop the recovery timer and to detect any missed events for the configured RT. An example usage with sequence diagram is shown as follows:

```
volatile uint32 DelayCount = SMU_RT_DELAY;
/*Set the alarm status for alarm group 10 and position 2*/
ResultRecoveryTimer = Smu_SetAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_2);
if (E_OK == ResultRecoveryTimer)
{
    /*Set the alarm status for alarm group 10 and position 3*/
    ResultRecoveryTimer = Smu_SetAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_3);
    if (E_OK == ResultRecoveryTimer)
    {
        /*capture any missed events*/
        ResultRecoveryTimer = Smu_GetRTMissedEvent(SMU_TIMER_NUM, &EventMissed);
        if (0x01 == EventMissed)
        {
            /*Stop the RT is missed events are detected*/
            ResultRecoveryTimer = Smu_RTStop(SMU_TIMER_NUM);
        }
    }
}
```

SMU driver

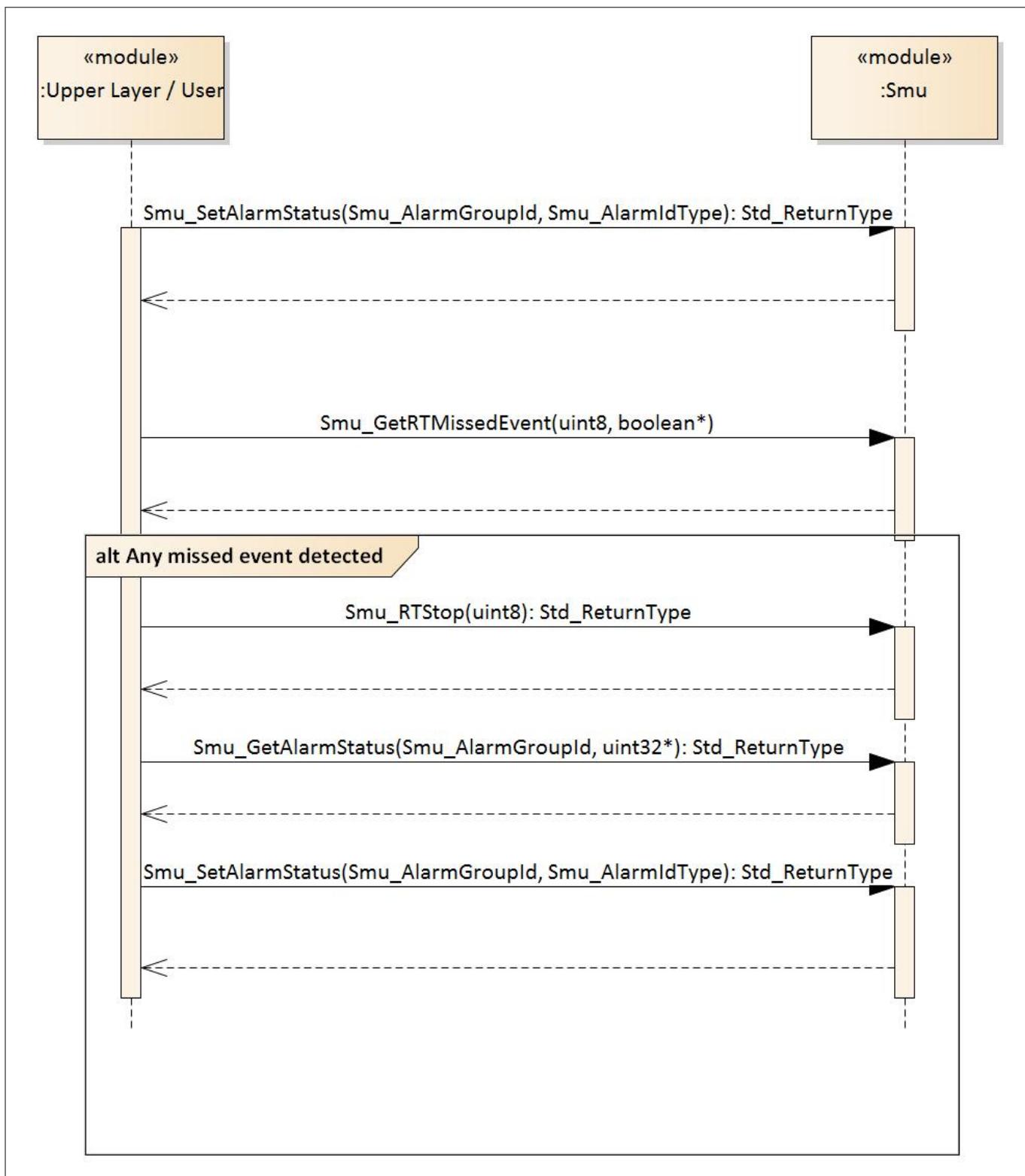


Figure 50 Sequence diagram for recovery timer usage

SMU driver**4.1.5 Key architectural considerations****4.1.5.1 Clearing alarm status during initialization**

During initialization, all the alarm statuses are cleared. Therefore, the user must ensure to keep a track of the alarm status before the `Smu_Init()` API is called.

4.1.5.2 Initialization and de-initialization

The `Smu_Init()` and `Smu_Deinit()` APIs shall be called only from the master cores. In case the `Smu_Init()` or `Smu_Deinit()` API is called from any other core besides the master core, the sequence will return an error. There is no resource distribution across the cores and the SMU driver shall be accessible across all cores except for the `Smu_ActivateRunState` and `Smu_ReleaseFSP` APIs.

4.1.5.3 SMU_core state machine transitions

The SMU_core state machine transitions are not to be verified by the driver. The user must verify the state before using it.

4.1.5.4 Recovery timer handling

While recovery timer is running, a missed alarm mapped to the same recovery timer is logged in the SFR `SMU_STS`. These missed alarms have to be explicitly checked and cleared by the application. The missed alarm can be checked using the `Smu_GetRTMissedEvent()` API.

SMU driver

4.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Clearing of RT missed events**

The user shall explicitly clear the RT missed events detected. Currently there is no service provided by the SMU driver to clear the RT missed events.

[cover parentID SMU={79E76BAC-C9CD-409c-8518-B9778796261D}]

- **Initialization check**

The user shall call the Smu_InitCheck API after initialization but before releasing vehicle safe state. Additionally, the user shall ensure that the Smu_LockConfigRegs API is invoked immediately after the Smu_InitCheck API without any preemption to avoid possible change of the SMU configuration from any other software units. In case the Smu_LockConfigRegs API is called, the APIs Smu_SetAlarmAction, Smu_SetupErrorPin, Smu_ReleaseErrorPin, Smu_RegisterMonitor APIs will raise a DET error indicating that the driver is in the LOCKED state. The command-based APIs will work as per their functionality.

[cover parentID SMU={6779F95B-9003-4e32-A5CA-4E072E2BB8CF}]

- **Non-interference check**

The user shall check that the correct config pointer has been passed and there is no interference to MCAL from other modules.

[cover parentID SMU={B5E820B9-2097-4917-BC6C-60B5A523F429}]

- **SFF test module check**

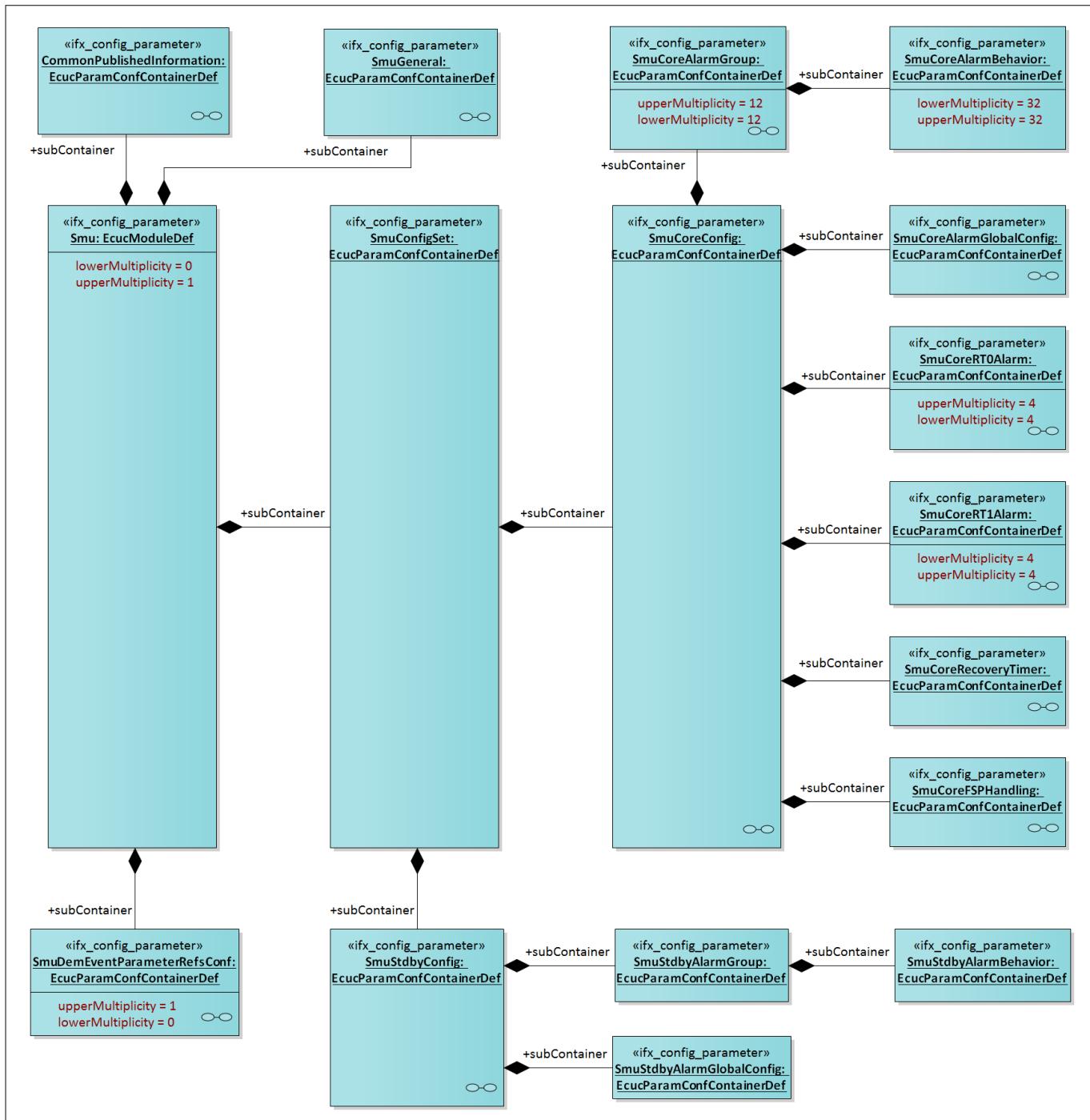
The user shall check if the module for which SFF test has been requested is available or not in the particular device.

[cover parentID SMU={9A7FBC44-B881-4469-9D42-CA6507F9A712}]

- **SMU FSP functionality**

The Smu_Release FSP API is asynchronous and the transition from the FAULT to the RUN state may require several cycles based on the fault state duration configured by the user, control PAD characteristics and/or recurrent fault occurrences. The Smu_Activate FSP API is asynchronous and the transition to the FAULT state may require several cycles based on the control PAD characteristics. Therefore, there is no deterministic time frame within which the state transition can be checked by the driver. The user shall ensure the transition to the intended state has occurred in the SMU_core. The user can check this using the Smu_GetSmuState API.

[cover parentID SMU={D1FB1959-3FB3-4123-92D8-226B551E6129}]

SMU driver**4.3 Reference information****4.3.1 Configuration interfaces****Figure 51 Container hierarchy along with their configuration parameters****4.3.1.1 Container: CommonPublishedInformation**

The container gives the published information for SMU driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

SMU driver**4.3.1.1.1 ArMajorVersion****Table 308 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	-		

4.3.1.1.2 ArMinorVersion**Table 309 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		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.1.3 ArPatchVersion**Table 310 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	This parameter provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 255		

SMU driver**Table 310 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	-		

4.3.1.1.4 ModuleId**Table 311 Specification for ModuleId**

Name	ModuleId		
Description	The configuration parameter defines the module ID of SMU module from module list.		
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	-		

4.3.1.1.5 Release**Table 312 Specification for Release**

Name	Release		
Description	The configuration parameter defines the AURIX 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	-

SMU driver**Table 312 Specification for Release (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.1.6 SwMajorVersion**Table 313 Specification for SwMajorVersion**

Name	SwMajorVersion		
Description	The configuration parameter defines the major version number of the vendor specific implementation of the module. The numbering is vendor specific.		
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	-		

4.3.1.1.7 SwMinorVersion**Table 314 Specification for SwMinorVersion**

Name	SwMinorVersion		
Description	The configuration parameter defines the minor version number of the vendor specific implementation of the module. 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	-		

SMU driver**4.3.1.1.8 SwPatchVersion****Table 315 Specification for SwPatchVersion**

Name	SwPatchVersion		
Description	The configuration parameter defines the patch level 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 version		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Published-Information	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.1.9 VendorId**Table 316 Specification for VendorId**

Name	VendorId		
Description	The configuration parameter defines the vendor ID of the dedicated implementation of the module according to the AUTOSAR vendor list.		
Multiplicity	1..1	Type	EcuclIntegerParamDef
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	-		

4.3.1.2 Container: Smu

The Smu container is the parent container for SMU module.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

SMU driver

4.3.1.2.1 ConfigVariant

Table 317 Specification for ConfigVariant

Name	ConfigVariant		
Description	Selects the config-variant for the SMU module Note: Implementing SMU as post-build is user friendly, hence it is implemented as post-build.		
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	-		

4.3.1.3 Container: SmuConfigSet

The container contains SmuConfigSet configurations for the SMU driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.4 Container: SmuCoreAlarmBehavior

The container contains configuration parameters related to alarm behavior. Each alarm group has thirty two alarm configurations. Both the internal and external behavior can be configured for every alarm.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.4.1 SmuCoreAlarmFSP

Table 318 Specification for SmuCoreAlarmFSP

Name	SmuCoreAlarmFSP		
Description	The configuration parameter defines the value of the FSP configuration. The default value of the parameter is set to the reset value of corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SMU_ALARM_FSP_DISABLED: The configuration parameter literal defines that FSP is disabled. SMU_ALARM_FSP_ENABLED: The configuration parameter literal defines that FSP is enabled.		
Default value	SMU_ALARM_FSP_DISABLED		

SMU driver**Table 318 Specification for SmuCoreAlarmFSP (continued)**

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

4.3.1.4.2 SmuCoreAlarmIntBeh**Table 319 Specification for SmuCoreAlarmIntBeh**

Name	SmuCoreAlarmIntBeh		
Description	The configuration parameter defines the internal behavior of an alarm event. The default value of the parameter is set to the reset value of corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SMU_CPU_RESET_INT_ACTION: The configuration parameter literal defines the internal behavior as sending CPU reset configuration request. SMU_IGCS0_INT_ACTION: The configuration parameter literal defines the internal behavior as sending an interrupt request to the interrupt system according to the interrupt generation configuration set 0. SMU_IGCS1_INT_ACTION: The configuration parameter literal defines the internal behavior as sending an interrupt request to the interrupt system according to the interrupt generation configuration set 1 SMU_IGCS2_INT_ACTION: The configuration parameter literal defines the internal behavior as sending an interrupt request to the interrupt system according to the interrupt generation configuration set 2 SMU_NA_INT_ACTION: The configuration parameter literal defines the internal behavior as no action (default value). SMU_NMI_INT_ACTION: The configuration parameter literal defines the internal behavior as sending NMI request to the SCU. SMU_RESET_INT_ACTION: The configuration parameter literal defines the internal behavior as sending reset request to the SCU.		
Default value	SMU_NA_INT_ACTION		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

SMU driver**4.3.1.4.3 SmuCoreAlmBehaviourId****Table 320 Specification for SmuCoreAlmBehaviourId**

Name	SmuCoreAlmBehaviourId		
Description	The configuration parameter defines the alarm behavior ID corresponding to the particular group. First alarm behavior is selected as the default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 31		
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	-		

4.3.1.5 Container: SmuCoreAlarmGlobalConfig

The container contains the alarm global configuration parameters. The parameters are used for initializing the SMU_AGC register.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.5.1 SmuCoreCpu0ResetRequest**Table 321 Specification for SmuCoreCpu0ResetRequest**

Name	SmuCoreCpu0ResetRequest		
Description	The configuration parameter is a Boolean which denotes whether the reset request to CPU0 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR.		
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	-

SMU driver**Table 321 Specification for SmuCoreCpu0ResetRequest (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.5.2 SmuCoreCpu1ResetRequest**Table 322 Specification for SmuCoreCpu1ResetRequest**

Name	SmuCoreCpu1ResetRequest		
Description	<p>The configuration parameter is a Boolean which denotes whether the reset request to CPU1 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR.</p> <p>Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.</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	-		

4.3.1.5.3 SmuCoreCpu2ResetRequest**Table 323 Specification for SmuCoreCpu2ResetRequest**

Name	SmuCoreCpu2ResetRequest		
Description	<p>The configuration parameter is a Boolean which denotes whether the reset request to CPU2 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR.</p> <p>Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.</p>		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	<p>TRUE</p> <p>FALSE</p>		
Default value	FALSE		

SMU driver**Table 323 Specification for SmuCoreCpu2ResetRequest (continued)**

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

4.3.1.5.4 SmuCoreCpu3ResetRequest**Table 324 Specification for SmuCoreCpu3ResetRequest**

Name	SmuCoreCpu3ResetRequest		
Description	<p>The configuration parameter is a Boolean which denotes whether the reset request to CPU3 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR.</p> <p>Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.</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	-		

4.3.1.5.5 SmuCoreCpu4ResetRequest**Table 325 Specification for SmuCoreCpu4ResetRequest**

Name	SmuCoreCpu4ResetRequest		
Description	<p>The configuration parameter is a Boolean which denotes whether the reset request to CPU4 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR.</p> <p>Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.</p>		

SMU driver**Table 325 Specification for SmuCoreCpu4ResetRequest (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	-		

4.3.1.5.6 SmuCoreCpu5ResetRequest**Table 326 Specification for SmuCoreCpu5ResetRequest**

Name	SmuCoreCpu5ResetRequest		
Description	<p>The configuration parameter is a Boolean which denotes whether the reset request to CPU5 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR.</p> <p>Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.</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	-		

4.3.1.5.7 SmuCoreCpuResetActivatePES**Table 327 Specification for SmuCoreCpuResetActivatePES**

Name	SmuCoreCpuResetActivatePES
-------------	----------------------------

SMU driver**Table 327 Specification for SmuCoreCpuResetActivatePES (continued)**

Description	The configuration parameter enables the PES on CPU reset. 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	-		

4.3.1.5.8 SmuCoreEnableFaultToRunState**Table 328 Specification for SmuCoreEnableFaultToRunState**

Name	SmuCoreEnableFaultToRunState		
Description	The configuration parameter defines whether the FAULT state to RUN state transition is enabled or disabled. The state transition is possible only when this parameter is defined with SMU_EFRST_ENABLE. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SMU_EFRST_DISABLE: The configuration parameter literal defines that the FAULT state to RUN state transition is disabled. SMU_EFRST_ENABLE: The configuration parameter literal defines that the FAULT state to RUN state transition is enabled.		
Default value	SMU_EFRST_DISABLE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

SMU driver

4.3.1.5.9 SmuCoreIGCS0ActivatePES

Table 329 Specification for SmuCoreIGCS0ActivatePES

Name	SmuCoreIGCS0ActivatePES		
Description	The configuration parameter defines the control of the Port Emergency Stop (PES) feature for IGCS0 internal action. When an IGCS0 internal action is triggered, the hardware triggers automatically the PES, on enabling. 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	-		

4.3.1.5.10 SmuCoreIGCS1ActivatePES

Table 330 Specification for SmuCoreIGCS1ActivatePES

Name	SmuCoreIGCS1ActivatePES		
Description	The configuration parameter defines the control of the Port Emergency Stop (PES) feature for IGCS1 internal action. When an IGCS1 internal action is triggered, the hardware triggers automatically the PES, on enabling. 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	-		

SMU driver

4.3.1.5.11 SmuCoreIGCS2ActivatePES

Table 331 Specification for SmuCoreIGCS2ActivatePES

Name	SmuCoreIGCS2ActivatePES		
Description	The configuration parameter defines the control of the Port Emergency Stop (PES) feature for IGCS2 internal action. When an IGCS2 internal action is triggered, the hardware triggers automatically the PES, on enabling. 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	-		

4.3.1.5.12 SmuCoreInterruptSet0

Table 332 Specification for SmuCoreInterruptSet0

Name	SmuCoreInterruptSet0		
Description	The configuration parameter defines the output value of the interrupt request vector when the alarm configuration flag selects the interrupt configuration set 0. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

SMU driver**Table 332 Specification for SmuCoreInterruptSet0 (continued)**

Range	SMU_SELECT_INT0: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0. SMU_SELECT_INT0_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU1. SMU_SELECT_INT0_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0, SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT0_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU2. SMU_SELECT_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1. SMU_SELECT_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU2. SMU_SELECT_INT_NONE: The configuration parameter literal defines the output value of the interrupt request vector as no interrupt selected.		
Default value	SMU_SELECT_INT_NONE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.5.13 SmuCoreInterruptSet1**Table 333 Specification for SmuCoreInterruptSet1**

Name	SmuCoreInterruptSet1		
Description	The configuration parameter defines the output value of the interrupt request vector when the alarm configuration flag selects the interrupt configuration set 1. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

SMU driver**Table 333 Specification for SmuCoreInterruptSet1 (continued)**

Range	SMU_SELECT_INT0: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0. SMU_SELECT_INT0_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU1. SMU_SELECT_INT0_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0, SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT0_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU2. SMU_SELECT_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1. SMU_SELECT_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU2. SMU_SELECT_INT_NONE: The configuration parameter literal defines the output value of the interrupt request vector as no interrupt selected.		
Default value	SMU_SELECT_INT_NONE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.5.14 SmuCoreInterruptSet2**Table 334 Specification for SmuCoreInterruptSet2**

Name	SmuCoreInterruptSet2		
Description	The configuration parameter defines the output value of the interrupt request vector when the alarm configuration flag selects the interrupt configuration set 2. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

SMU driver**Table 334 Specification for SmuCoreInterruptSet2 (continued)**

Range	SMU_SELECT_INT0: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0. SMU_SELECT_INT0_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU1. SMU_SELECT_INT0_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0, SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT0_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0, SRC_SMU2. SMU_SELECT_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1. SMU_SELECT_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU2. SMU_SELECT_INT_NONE: The configuration parameter literal defines the output value of the interrupt request vector as no interrupt selected.		
Default value	SMU_SELECT_INT_NONE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.5.15 SmuCoreNMIActivatePES**Table 335 Specification for SmuCoreNMIActivatePES**

Name	SmuCoreNMIActivatePES		
Description	The configuration parameter defines the control of the Port Emergency Stop (PES) feature for NMI internal action. When an NMI internal action is triggered, the hardware triggers automatically the PES, on enabling. 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	-

SMU driver**Table 335 Specification for SmuCoreNMIActivatePES (continued)**

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

4.3.1.6 Container: SmuCoreAlarmGroup

The container contains the configuration parameters for SMU_core alarm groups.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.6.1 SmuCoreAlmGrpId**Table 336 Specification for SmuCoreAlmGrpId**

Name	SmuCoreAlmGrpId		
Description	The configuration parameter defines group ID of the SMU alarm group. The value will be assigned to the symbolic name derived from the AlarmGroup container short name. First alarm group is selected as the default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 11		
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	-		

4.3.1.7 Container: SmuCoreConfig

The container contains the configuration parameters related to SMU_core.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.8 Container: SmuCoreFSPHandling

The container contains the configuration parameters related to SMU_core FSP handling.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

SMU driver

4.3.1.8.1 SmuCoreFSPFaultStateDuration

Table 337 Specification for SmuCoreFSPFaultStateDuration

Name	SmuCoreFSPFaultStateDuration		
Description	<p>The configuration parameter enables the maximum fault state duration of FSP signal. The fault duration value is set at bit field, TFSP_HIGH of FSP Register. The configuration parameter is specified as a number of SMU_FS ticks.</p> <p>The default value is the intermediate value of the SFR.</p>		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 1023		
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	-		

4.3.1.8.2 SmuCoreFSPPrescaler1

Table 338 Specification for SmuCoreFSPPrescaler1

Name	SmuCoreFSPPrescaler1		
Description	<p>The configuration parameter defines the dividing factor to apply to the reference clock fBACK. The divided clock is used as reference to generate the timing of the fault signaling protocol fault state. The frequency of the divided clock is F(SMU_FS).</p> <p>The default value of the parameter is set to the reset value of corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef

SMU driver**Table 338 Specification for SmuCoreFSPPrescaler1 (continued)**

Range	SMU_REF_CLK_FRQ_DIV_128: The configuration parameter literal defines that the reference clock frequency is divided by 128. SMU_REF_CLK_FRQ_DIV_16: The configuration parameter literal defines that the reference clock frequency is divided by 16. SMU_REF_CLK_FRQ_DIV_256: The configuration parameter literal defines that the reference clock frequency is divided by 256. SMU_REF_CLK_FRQ_DIV_2: The configuration parameter literal defines that the reference clock frequency is divided by 2. SMU_REF_CLK_FRQ_DIV_32: The configuration parameter literal defines that the reference clock frequency is divided by 32. SMU_REF_CLK_FRQ_DIV_4: The configuration parameter literal defines that the reference clock frequency is divided by 4. SMU_REF_CLK_FRQ_DIV_64: The configuration parameter literal defines that the reference clock frequency is divided by 64. SMU_REF_CLK_FRQ_DIV_8: The configuration parameter literal defines that the reference clock frequency is divided by 8.		
Default value	SMU_REF_CLK_FRQ_DIV_2		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.8.3 SmuCoreFSPPrescaler2**Table 339 Specification for SmuCoreFSPPrescaler2**

Name	SmuCoreFSPPrescaler2		
Description	The configuration parameter defines the dividing factor to apply to the reference clock fBACK. The divided clock is used as reference to generate the timing of the fault free state for the dynamic dual rail and time switching modes of the fault signalling protocol. The frequency of the divided clock is F(SMU_FFS). The default value of the parameter is set to the reset value of corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

SMU driver**Table 339 Specification for SmuCoreFSPPrescaler2 (continued)**

Range	SMU_REF_CLK_FRQ_DIV_1024: The configuration parameter literal defines that the reference clock frequency is divided by 1024. SMU_REF_CLK_FRQ_DIV_2048: The configuration parameter literal defines that the reference clock frequency is divided by 2048. SMU_REF_CLK_FRQ_DIV_4096: The configuration parameter literal defines that the reference clock frequency is divided by 4096. SMU_REF_CLK_FRQ_DIV_512: The configuration parameter literal defines that the reference clock frequency is divided by 512.		
Default value	SMU_REF_CLK_FRQ_DIV_512		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.8.4 SmuCoreFSPSignalingMode**Table 340 Specification for SmuCoreFSPSignalingMode**

Name	SmuCoreFSPSignalingMode		
Description	The configuration parameter defines the type of signal output for FSP on an alarm event. The default value of the parameter is set to the reset value of corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SMU_FSP_BISTABLE_PROTOCOL: The configuration parameter literal defines that bistable protocol is used for FSP handling. SMU_FSP_DUAL_RAIL_PROTOCOL: The configuration parameter literal defines that dual rail protocol is used for FSP handling. SMU_FSP_TIME_SWITCHING_PROTOCOL: The configuration parameter literal defines that time switching protocol is used for FSP handling.		
Default value	SMU_FSP_BISTABLE_PROTOCOL		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

SMU driver**4.3.1.8.5 SmuCorePESOnFSP****Table 341 Specification for SmuCorePESOnFSP**

Name	SmuCorePESOnFSP		
Description	<p>The configuration parameter defines whether the PES is to be automatically requested when an alarm event configured to start the FSP is detected.</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>SMU_FSP_PES_DISABLE: The configuration parameter literal defines that PES feature is disabled.</p> <p>SMU_FSP_PES_ENABLE: The configuration parameter literal defines that PES feature is enabled.</p>		
Default value	SMU_FSP_PES_DISABLE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.9 Container: SmuCoreRecoveryTimer

The container contains the configuration parameters for SMU_core recovery timer.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.9.1 SmuCoreEnableRT0**Table 342 Specification for SmuCoreEnableRT0**

Name	SmuCoreEnableRT0		
Description	<p>The configuration parameter defines whether RT0 is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.</p>		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	<p>SMU_RT_DISABLE: The configuration parameter literal defines that the recovery timer is disabled.</p> <p>Value: 0</p> <p>SMU_RT_ENABLE: The configuration parameter literal defines that the recovery timer is enabled.</p> <p>Value: 1</p>		
Default value	SMU_RT_DISABLE		

SMU driver**Table 342 Specification for SmuCoreEnableRT0 (continued)**

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

4.3.1.9.2 SmuCoreEnableRT1**Table 343 Specification for SmuCoreEnableRT1**

Name	SmuCoreEnableRT1		
Description	The configuration parameter defines whether RT1 is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SMU_RT_DISABLE: The configuration parameter literal defines that the recovery timer is disabled. Value: 0 SMU_RT_ENABLE: The configuration parameter literal defines that the recovery timer is enabled. Value: 1		
Default value	SMU_RT_DISABLE		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.9.3 SmuCoreRTDuration**Table 344 Specification for SmuCoreRTDuration**

Name	SmuCoreRTDuration		
Description	The configuration parameter defines the maximum duration of SMU_core recovery timer. The maximum duration is specified as a number of the F(SMU_FS) clock ticks. The default value is the intermediate value of the tick duration.		
Multiplicity	1..1	Type	EcucIntegerParamDef

SMU driver**Table 344 Specification for SmuCoreRTDuration (continued)**

Range	0 - 0xFFFFFU		
Default value	0x3FFFU		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.10 Container: SmuCoreRT0Alarm

The container enables to select the alarms for RT0. Four alarms can be configured per recovery timer instance.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.10.1 SmuCoreRT0AlarmGroupId**Table 345 Specification for SmuCoreRT0AlarmGroupId**

Name	SmuCoreRT0AlarmGroupId		
Description	The configuration parameter defines the alarm group ID associated with the RT0. First alarm id is selected as the default value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

SMU driver**Table 345 Specification for SmuCoreRT0AlarmGroupId (continued)**

Range	SMU_ALARM_GROUP0: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 0. SMU_ALARM_GROUP10: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 10. SMU_ALARM_GROUP11: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 11. SMU_ALARM_GROUP1: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 1. SMU_ALARM_GROUP2: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 2. SMU_ALARM_GROUP3: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 3. SMU_ALARM_GROUP4: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 4. SMU_ALARM_GROUP5: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 5. SMU_ALARM_GROUP6: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 6. SMU_ALARM_GROUP7: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 7. SMU_ALARM_GROUP8: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 8. SMU_ALARM_GROUP9: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 9.		
Default value	SMU_ALARM_GROUP0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SmuCoreEnableRT0		

4.3.1.10.2 SmuCoreRT0AlarmId**Table 346 Specification for SmuCoreRT0AlarmId**

Name	SmuCoreRT0AlarmId		
Description	The configuration parameter defines the alarm ID associated with the RT0. First alarm id is selected as the default value..		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 31		
Default value	0		

SMU driver**Table 346 Specification for SmuCoreRT0AlarmId (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SmuCoreEnableRT0		

4.3.1.11 Container: SmuCoreRT1Alarm

The container enables to select the alarms for RT1. Four alarms can be configured per recovery timer instance.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.11.1 SmuCoreRT1AlarmGroupId**Table 347 Specification for SmuCoreRT1AlarmGroupId**

Name	SmuCoreRT1AlarmGroupId		
Description	The configuration parameter defines the alarm group ID associated with the RT1. First alarm group is selected as the default value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

SMU driver**Table 347 Specification for SmuCoreRT1AlarmGroupId (continued)**

Range	SMU_ALARM_GROUP0: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 0. SMU_ALARM_GROUP10: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 10. SMU_ALARM_GROUP11: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 11. SMU_ALARM_GROUP1: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 1. SMU_ALARM_GROUP2: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 2. SMU_ALARM_GROUP3: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 3. SMU_ALARM_GROUP4: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 4. SMU_ALARM_GROUP5: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 5. SMU_ALARM_GROUP6: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 6. SMU_ALARM_GROUP7: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 7. SMU_ALARM_GROUP8: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 8. SMU_ALARM_GROUP9: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 9.		
Default value	SMU_ALARM_GROUP0		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SmuCoreEnableRT1		

4.3.1.11.2 SmuCoreRT1AlarmId**Table 348 Specification for SmuCoreRT1AlarmId**

Name	SmuCoreRT1AlarmId		
Description	The configuration parameter defines the alarm ID associated with the RT1. First alarm id is selected as the default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 31		
Default value	0		

SMU driver**Table 348 Specification for SmuCoreRT1AlarmId (continued)**

Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SmuCoreEnableRT1		

4.3.1.12 Container: SmuDemEventParameterRefsConf

The container lists down the production errors supported by the SMU driver.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

4.3.1.12.1 SmuActivateFSPFailureNotification**Table 349 Specification for SmuActivateFSPFailureNotification**

Name	SmuActivateFSPFailureNotification		
Description	The configuration parameter tells whether the notification for DEM related to failure to activate FSP is enabled or not		
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	SmuSafetyEnable		

4.3.1.12.2 SmuActivatePESFailureNotification**Table 350 Specification for SmuActivatePESFailureNotification**

Name	SmuActivatePESFailureNotification		
Description	The configuration parameter tells whether the notification for DEM related to failure of PES is enabled or disabled.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef

SMU driver**Table 350 Specification for SmuActivatePESFailureNotification (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	IFX	Scope	LOCAL
Dependency	SmuSafetyEnable		

4.3.1.12.3 SmuActivateRunStateFailureNotification**Table 351 Specification for SmuActivateRunStateFailureNotification**

Name	SmuActivateRunStateFailureNotification		
Description	The configuration parameter tells whether the notification for DEM related to failure to activate RUN state is enabled or disabled.		
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	SmuSafetyEnable		

4.3.1.12.4 SmuClearAlarmStatusFailureNotification**Table 352 Specification for SmuClearAlarmStatusFailureNotification**

Name	SmuClearAlarmStatusFailureNotification		
Description	The configuration parameter tells whether the notification for DEM related to failure to clear alarm status is enabled or disabled.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		

SMU driver**Table 352 Specification for SmuClearAlarmStatusFailureNotification (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	SmuSafetyEnable		

4.3.1.12.5 SmuCoreAliveFailureNotification**Table 353 Specification for SmuCoreAliveFailureNotification**

Name	SmuCoreAliveFailureNotification		
Description	The configuration parameter tells whether the notification for DEM related to SMU_core_alive test failure is enabled or disabled.		
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	SmuSafetyEnable		

4.3.1.12.6 SmuRTStopFailureNotification**Table 354 Specification for SmuRTStopFailureNotification**

Name	SmuRTStopFailureNotification		
Description	The configuration parameter tells whether the notification for DEM related to failure to stop recovery timer is enabled or disabled.		
Multiplicity	0..1	Type	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	Post-build variant multiplicity	FALSE

SMU driver**Table 354 Specification for SmuRTStopFailureNotification (continued)**

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

4.3.1.12.7 SmuReleaseFSPFailureNotification**Table 355 Specification for SmuReleaseFSPFailureNotification**

Name	SmuReleaseFSPFailureNotification		
Description	The configuration parameter tells whether the notification for DEM related to failure to release FSP is enabled or disabled.		
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	SmuSafetyEnable		

4.3.1.12.8 SmuSetAlarmStatusFailureNotification**Table 356 Specification for SmuSetAlarmStatusFailureNotification**

Name	SmuSetAlarmStatusFailureNotification		
Description	The configuration parameter tells whether the notification for DEM related to failure to set alarm status is enabled or disabled.		
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

SMU driver**Table 356 Specification for SmuSetAlarmStatusFailureNotification (continued)**

Origin	IFX	Scope	LOCAL
Dependency	SmuSafetyEnable		

4.3.1.12.9 SmuSffFailureNotification**Table 357 Specification for SmuSffFailureNotification**

Name	SmuSffFailureNotification		
Description	The configuration parameter tells whether the notification for DEM related to SFF test failure is enabled or disabled.		
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	SmuSafetyEnable		

4.3.1.13 Container: SmuGeneral

The container contains the general configurations of SMU driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.13.1 SmuAGStatusTimeout**Table 358 Specification for SmuAGStatusTimeout**

Name	SmuAGStatusTimeout		
Description	The configuration parameter defines the timeout value to check for the alarm group status within the defined time frame. It corresponds to the maximum number of loop counts for which the software loop continuously checks for the status of the SFR to change. The default value of the timeout is the intermediate value.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	4096 - 65535		
Default value	4096		
Post-build variant value	FALSE	Post-build variant multiplicity	-

SMU driver**Table 358 Specification for SmuAGStatusTimeout (continued)**

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

4.3.1.13.2 SmuCoreFSP0OutputEnable**Table 359 Specification for SmuCoreFSP0OutputEnable**

Name	SmuCoreFSP0OutputEnable		
Description	The configuration parameter sets FSP[0] state to output, if true. The default value is 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	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.13.3 SmuCoreFSP0PortEnable**Table 360 Specification for SmuCoreFSP0PortEnable**

Name	SmuCoreFSP0PortEnable		
Description	The configuration parameter sets FSP[0] PORT enable to true. The default value is 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	FALSE	Post-build variant multiplicity	-

SMU driver**Table 360 Specification for SmuCoreFSP0PortEnable (continued)**

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

4.3.1.13.4 SmuCoreFSP1OutputEnable**Table 361 Specification for SmuCoreFSP1OutputEnable**

Name	SmuCoreFSP1OutputEnable		
Description	The configuration parameter sets FSP[1] state to output, if true. The default value is 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	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.13.5 SmuCoreFSP1PortEnable**Table 362 Specification for SmuCoreFSP1PortEnable**

Name	SmuCoreFSP1PortEnable		
Description	The configuration parameter sets FSP[1] port enable to true. The default value is 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	FALSE	Post-build variant multiplicity	-

SMU driver**Table 362 Specification for SmuCoreFSP1PortEnable (continued)**

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

4.3.1.13.6 SmuCoreGlitchFilterSCU**Table 363 Specification for SmuCoreGlitchFilterSCU**

Name	SmuCoreGlitchFilterSCU		
Description	The configuration parameter sets glitch filter for SCU to enabled state. The default value is 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	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.13.7 SmuCoreGlitchFilterSTS**Table 364 Specification for SmuCoreGlitchFilterSTS**

Name	SmuCoreGlitchFilterSTS		
Description	The configuration parameter sets glitch filter for SMU_STS to be enabled. The default value is 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	FALSE	Post-build variant multiplicity	-

SMU driver**Table 364 Specification for SmuCoreGlitchFilterSTS (continued)**

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

4.3.1.13.8 SmuDevErrorDetect**Table 365 Specification for SmuDevErrorDetect**

Name	SmuDevErrorDetect		
Description	The configuration parameter enables or disables DET checks.		
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	-		

4.3.1.13.9 SmuInitCheckApi**Table 366 Specification for SmuInitCheckApi**

Name	SmuInitCheckApi		
Description	The configuration parameter enables or disables the Smu_InitCheck API. 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	-

SMU driver**Table 366 Specification for SmulnitCheckApi (continued)**

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

4.3.1.13.10 SmulnitDeInitApiMode**Table 367 Specification for SmulnitDeInitApiMode**

Name	SmulnitDeInitApiMode		
Description	The configuration parameter defines the mode in which the Init and Deinit API will be used. Since SMU driver accesses the SFRs, it is more efficient to operate the SMU driver in supervisor mode. Hence, the default mode of operation is supervisor.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SMU_MCAL_SUPERVISOR: The configuration parameter implies that SUPERVISOR mode is used. The parameter takes value 0 when assigned. SMU_MCAL_USER1: The configuration parameter implies that USER1 mode is used. The parameter takes values 1 when assigned.		
Default value	SMU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

4.3.1.13.11 SmuRuntimeApiMode**Table 368 Specification for SmuRuntimeApiMode**

Name	SmuRuntimeApiMode		
Description	The configuration parameter gives the mode in which the runtime API will be used. Since SMU driver accesses the SFRs, it is more efficient to operate the SMU driver in supervisor mode. Hence, the default mode of operation is supervisor. When the parameter is in supervisor mode, then the SmulnitDeIntMode is in supervisor mode.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

SMU driver**Table 368 Specification for SmuRuntimeApiMode (continued)**

Range	SMU_MCAL_SUPERVISOR: The configuration parameter implies that SUPERVISOR mode is used. The parameter takes value 0 when assigned. SMU_MCAL_USER1: The configuration parameter implies that USER1 mode is used. The parameter takes value 1 when assigned.		
Default value	SMU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SmuInitDeInitApiMode		

4.3.1.13.12 SmuSafetyEnable**Table 369 Specification for SmuSafetyEnable**

Name	SmuSafetyEnable		
Description	The configuration parameter defines whether the safety checks mandated by safety standards are enabled or 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	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	-		

4.3.1.13.13 SmuStdbyEnable**Table 370 Specification for SmuStdbyEnable**

Name	SmuStdbyEnable		
Description	The configuration parameter defines whether the SMU_stdby unit is enabled or disabled. The default value is this parameter is set to the reset value of the corresponding SFR.		

SMU driver**Table 370 Specification for SmuStdbyEnable (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	-		

4.3.1.13.14 SmuVersionInfoApi**Table 371 Specification for SmuVersionInfoApi**

Name	SmuVersionInfoApi		
Description	The configuration parameter enables or disables the VersionInfo API. 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	-		

4.3.1.14 Container: SmuStdbyAlarmBehavior

The container contains configuration parameters corresponding to alarm behavior. The behavior type is external if FSP is enabled or no reaction.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

SMU driver

4.3.1.14.1 SmuStdbyAlarmFSP

Table 372 Specification for SmuStdbyAlarmFSP

Name	SmuStdbyAlarmFSP		
Description	The configuration parameter defines whether the FSP is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SMU_ALARM_FSP_DISABLED: The configuration parameter literal defines that FSP is disabled. SMU_ALARM_FSP_ENABLED: The configuration parameter literal defines that the FSP is enabled.		
Default value	SMU_ALARM_FSP_DISABLED		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SmuStdbyEnable		

4.3.1.14.2 SmuStdbyAlmBehaviourId

Table 373 Specification for SmuStdbyAlmBehaviourId

Name	SmuStdbyAlmBehaviourId		
Description	The configuration parameter defines the alarm ID corresponding to the particular group. First alarm behavior id is selected as the default value..		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	0 - 31		
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	SmuStdbyEnable		

4.3.1.15 Container: SmuStdbyAlarmGlobalConfig

The container contains the configuration parameters related to SMU_stdby global configurations.

Post-Build Variant Multiplicity: -

SMU driver

Multiplicity Configuration Class: -

4.3.1.15.1 SmuStdbyEnableFSP0

Table 374 Specification for SmuStdbyEnableFSP0

Name	SmuStdbyEnableFSP0		
Description	The configuration parameter defines whether the use of FSP[0] P33.8 pin is enabled or disabled for FSP handling. The default value of the parameter is the reset value of the SFR.		
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	SmuStdbyEnable		

4.3.1.15.2 SmuStdbyEnableFSP1

Table 375 Specification for SmuStdbyEnableFSP1

Name	SmuStdbyEnableFSP1		
Description	The configuration parameter defines whether the use of FSP[1] P33.10 pin is enabled or disabled for FSP handling. 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	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	SmuStdbyEnable		

SMU driver

4.3.1.16 Container: SmuStdbyAlarmGroup

The container contains the configuration parameters related to SMU_stdby alarm groups.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.1.16.1 SmuStdbyAlmGrpId

Table 376 Specification for SmuStdbyAlmGrpId

Name	SmuStdbyAlmGrpId		
Description	The configuration parameter defines the alarm group ID of the alarm group to be configured. First group id is selected as the default value.		
Multiplicity	1..1	Type	EcclIntegerParamDef
Range	0 - 31		
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	SmuStdbyEnable		

4.3.1.17 Container: SmuStdbyConfig

The container contains the configuration parameters related to SMU_stdby.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

4.3.2 Functions - Type definitions

4.3.2.1 Smu_AlarmGroupId

Table 377 Specification for Smu_AlarmGroupId

Syntax	Smu_AlarmGroupId	
Type	Enumeration	
File	Smu.h	
Range	0 - SMU_GROUP_0	
	1 - SMU_GROUP_1	
	2 - SMU_GROUP_2	
	3 - SMU_GROUP_3	

SMU driver**Table 377 Specification for Smu_AlarmGroupId (continued)**

	4 - SMU_GROUP_4	
	5 - SMU_GROUP_5	
	6 - SMU_GROUP_6	
	7 - SMU_GROUP_7	
	8 - SMU_GROUP_8	
	9 - SMU_GROUP_9	
	10 - SMU_GROUP_10	
	11 - SMU_GROUP_11	
	20 - SMU_GROUP_20	
	21 - SMU_GROUP_21	
Description	Smu_AlarmGroupId enumeration gives the alarm group ID for each group in SMU_core and SMU_stby.	
Source	IFX	

4.3.2.2 Smu_AlarmIdType**Table 378 Specification for Smu_AlarmIdType**

Syntax	Smu_AlarmIdType	
Type	Enumeration	
File	Smu.h	
Range	0 - SMU_ALARM_0	
	1 - SMU_ALARM_1	
	2 - SMU_ALARM_2	
	3 - SMU_ALARM_3	
	4 - SMU_ALARM_4	
	5 - SMU_ALARM_5	
	6 - SMU_ALARM_6	
	7 - SMU_ALARM_7	
	8 - SMU_ALARM_8	
	9 - SMU_ALARM_9	
	10 - SMU_ALARM_10	
	11 - SMU_ALARM_11	
	12 - SMU_ALARM_12	
	13 - SMU_ALARM_13	
	14 - SMU_ALARM_14	

SMU driver**Table 378 Specification for Smu_AlarmIdType (continued)**

	15 - SMU_ALARM_15	
	16 - SMU_ALARM_16	
	17 - SMU_ALARM_17	
	18 - SMU_ALARM_18	
	19 - SMU_ALARM_19	
	20 - SMU_ALARM_20	
	21 - SMU_ALARM_21	
	22 - SMU_ALARM_22	
	23 - SMU_ALARM_23	
	24 - SMU_ALARM_24	
	25 - SMU_ALARM_25	
	26 - SMU_ALARM_26	
	27 - SMU_ALARM_27	
	28 - SMU_ALARM_28	
	29 - SMU_ALARM_29	
	30 - SMU_ALARM_30	
	31 - SMU_ALARM_31	
Description	Smu_AlarmIdType enumeration gives the alarm ID associated with each alarm group.	
Source	IFX	

4.3.2.3 Smu_ConfigType**Table 379 Specification for Smu_ConfigType**

Syntax	Smu_ConfigType	
Type	Structure	
File	Smu.h	
Range	-	
Description	Defines the type of data structure containing the set of configuration parameters required for initializing the SMU driver and the SMU hardware unit.	
Source	IFX	

4.3.2.4 Smu_CoreAlarmActionType**Table 380 Specification for Smu_CoreAlarmActionType**

Syntax	Smu_CoreAlarmActionType
Type	uint8

SMU driver**Table 380 Specification for Smu_CoreAlarmActionType (continued)**

File	Smu.h	
Range	SMU_ALARM_ACTION_NONE	SMU_NA_ALARM_CONFIG implies that no action has to be taken on receiving an alarm.
	SMU_ALARM_ACTION_RSVD	SMU_RSVD_ALARM_CONFIG is reserved and no action is taken. Alarm is disabled.
	SMU_ALARM_ACTION_IGCS0	SMU_IGCS0_ALARM_CONFIG sends an interrupt request to the interrupt system according to the Interrupt Generation Configuration Set 0 from AGC register.
	SMU_ALARM_ACTION_IGCS1	SMU_IGCS1_ALARM_CONFIG sends an interrupt request to the interrupt system according to the Interrupt Generation Configuration Set 1 from AGC register.
	SMU_ALARM_ACTION_IGCS2	SMU_IGCS2_ALARM_CONFIG sends an interrupt request to the interrupt system according to the Interrupt Generation Configuration Set 2 from AGC register.
	SMU_ALARM_ACTION_NMI	SMU_NMI_ALARM_CONFIG sends an NMI request to the SCU.
	SMU_ALARM_ACTION_RESET	SMU_RESET_ALARM_CONFIG sends a RESET request to the SCU. SCU shall be configured to generate an application or system reset.
	SMU_ALARM_ACTION_CPU_RESET	SMU_CPU_RST_ALARM_CONFIG sets a CPU reset using CPU Reset Configuration set from the AGC register.
Description	Smu_CoreAlarmActionType defines the internal action behaviour for the alarms in SMU_Core.	
Source	IFX	

4.3.2.5 Smu_CoreCommandType**Table 381 Specification for Smu_CoreCommandType**

Syntax	Smu_CoreCommandType	
Type	uint8	
File	Smu.h	
Range	SMU_RUN_COMMAND	SMU_RUN_COMMAND makes the SMU_core enter the RUN state.
	SMU_ACTIVATEFSP_COMMAND	SMU_ACTIVATEFSP_COMMAND activates FSP for SMU_core.

SMU driver**Table 381 Specification for Smu_CoreCommandType (continued)**

	SMU_RELEASEFSP_COMMAND	SMU_RELEASEFSP_COMMAND releases FSP for SMU_core.
	SMU_ACTIVATE_PES	SMU_ACTIVATE_PES activates the PES feature for SMU_core.
	SMU_STOPREC_COMMAND	SMU_STOPREC_COMMAND stops the recovery timer for SMU_core.
	SMU_ASCE_COMMAND	SMU_ASCE_COMMAND is alarm status clear enable command for SMU_core. Software shall execute SMU_ASCE_COMMAND prior to clearing of a AG<n> alarm status bit. SMU_ASCE_COMMAND sets the ASCE bit in the STS register.
	SMU_ALARM_COMMAND	SMU_ALARM_COMMAND triggers a software based alarm. ARG specifies the alarm index.
	SMU_ALIVETEST_COMMAND	SMU_ALIVETEST_COMMAND enables the testing of the smu_core_alive signal. Sending SMU_ALIVETEST_COMMAND will forward the smu_core_alive alarm to the SMU_stdby. Argument ARG shall be set to 0x05 to start the test and to 0x0A to end the test.
Description	Smu_CoreCommandType describes the SMU_core command sets.	
Source	IFX	

4.3.2.6 Smu_CoreStateType**Table 382 Specification for Smu_CoreStateType**

Syntax	Smu_CoreStateType	
Type	uint8	
File	Smu.h	
Range	SMU_START_STATE	SMU_START_STATE corresponds to the START state in the SSM. Value:0. The alarms shall be logged in but not processed during START state. Exception to this is the RT and Watchdog timeout alarms. Entry condition: PORST Exit condition: Releasing FSP and activating RUN state through SMU_core command.
	SMU_RUN_STATE	SMU_RUN_STATE corresponds to the RUN state in the SSM. Value:1 The alarms logged are processed. Entry

SMU driver**Table 382 Specification for Smu_CoreStateType (continued)**

		condition: When FSP is released, RUN state is activated through SMU_core commands or FSP fault state timing is expired. Exit condition: When FSP is activated or alarm detected with FSP enabled.
	SMUFAULTSTATE	SMUFAULTSTATE corresponds to the FAULT state in the SSM. Value:2 SMU input alarm events are processed according to their configurations. FSP drives according to the configured reaction and timing. If a new FSP is detected, the FSP fault state timing is restarted. Entry condition: When alarm is detected. Exit condition: When FSP is released or FSP fault state timing is expired.
	SMU_UNDEFINED_STATE	SMU_UNDEFINED_STATE corresponds to the UNDEFINED state in the SSM. Value:3
Description	Smu_CoreStateType defines the various state types of SMU State Machine (SSM).	
Source	IFX	

4.3.2.7 Smu_EnableRunStateType**Table 383 Specification for Smu_EnableRunStateType**

Syntax	Smu_EnableRunStateType	
Type	Enumeration	
File	Smu.h	
Range	0 - SMU_EFRST_DISABLE	The Enable Fault To RUN state is disabled
	1 - SMU_EFRST_ENABLE	The Enable Fault To RUN state is enabled.
Description	Smu_EnableRunStateType enumeration defines whether the fault to run state is enabled or disabled.	
Source	IFX	

4.3.2.8 Smu_FSPActionType**Table 384 Specification for Smu_FSPActionType**

Syntax	Smu_FSPActionType
Type	uint32

SMU driver**Table 384 Specification for Smu_FSPActionType (continued)**

File	Smu.h	
Range	0 - 1	
Description	Smu_FSPActionType defines the FSP action type for the alarm group and ID.	
Source	IFX	

4.3.2.9 Smu_SffTestResType**Table 385 Specification for Smu_SffTestResType**

Syntax	Smu_SffTestResType	
Type	uint8	
File	Smu.h	
Range	0 - 255	
Description	Smu_SffTestResType gives the SFF test results.	
Source	IFX	

4.3.3 Functions - APIs**4.3.3.1 Smu_Init****Table 386 Specification for Smu_Init API**

Syntax	Std_ReturnType Smu_Init (const Smu_ConfigType * const ConfigPtr)	
Service ID	0xA8	
Sync/Async	Synchronous	
ASIL Level	B(D)	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the SMU configuration for initialization
Parameters (out)	-	-
Parameters (in - out)	-	-

SMU driver**Table 386 Specification for Smu_Init API (continued)**

Return	Std_ReturnType	E_OK: Operation successful that is initialization of resources of AURIX SMU peripheral is successful. E_NOT_OK: Operation failed that is initialization of resources of AURIX SMU peripheral is not successful, for example, when driver is already initialized.
Description	The purpose of the API is to setup the SMU peripheral based on the configuration. The SMU driver initializes the resources of the AURIX SMU peripheral, for example the error reaction and the Fault Signaling Protocol (FSP). Initialization should be done only from the master core. During initialization, all the alarm statuses are cleared, hence the user must ensure to keep a track of the alarm status before Smu_Init is called.	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_ALREADY_INITIALIZED: SMU_E_ALREADY_INITIALIZED DET is reported when SMU is already initialized.</p> <p>SMU_E_INIT_FAILED: SMU_E_INIT_FAILED DET is reported when initialization of SMU driver fails due to incorrect configuration parameter.</p> <p>SMU_E_CORE_MISMATCH: SMU_E_CORE_MISMATCH DET is reported when the Init or De-Init is called from any core other than the master core.</p> <p>SMU_E_LOCKED: SMU_E_LOCKED DET is reported when the SMU is already in locked 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	-	
User hints	Smu_Init API is responsible to clear the status of all SMU alarms. However, if the source of an alarm is not cleared or disabled, then the status of that alarm may remain set after execution of Smu_Init API even though the alarm had been cleared in the API.	

4.3.3.2 Smu_DeInit**Table 387 Specification for Smu_DeInit API**

Syntax	Std_ReturnType Smu_DeInit (void)
Service ID	0xAA
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Non Reentrant

SMU driver**Table 387 Specification for Smu_DeInit API (continued)**

Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful: de-initialization of SMU driver by resetting the module registers is successful. E_NOT_OK: Operation failed that is de-initialization of SMU driver by resetting the module registers is not successful, for example when driver is already reset.
Description	The purpose of the API is to de-initialize the SMU driver by resetting the module registers. De-initialization shall be done only from the master core.	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_CORE_MISMATCH: SMU_E_CORE_MISMATCH DET is reported when the Init or De-Init is called from any core other than the master core.</p> <p>SMU_E_LOCKED: SMU_E_LOCKED DET is reported when the SMU is already in locked 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	-	
User hints	Smu_DeInit API is responsible to clear the status of all SMU alarms. However, if the source of an alarm is not cleared or disabled, then the status of that alarm may remain set after execution of Smu_DeInit API even though the alarm had been cleared in the API.	

4.3.3.3 Smu_GetAlarmAction**Table 388 Specification for Smu_GetAlarmAction API**

Syntax	<pre>Std_ReturnType Smu_GetAlarmAction (const Smu_AlarmGroupId AlarmGroup, const Smu_AlarmIdType AlarmPos, Smu_CoreAlarmActionType * const IntAlarmAction, Smu_FSPActionType * const FSPAction)</pre>
Service ID	0xAB

SMU driver**Table 388 Specification for Smu_GetAlarmAction API (continued)**

Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup AlarmPos	Alarm group number (0 - 11, 20, 21) Alarm position within the requested group(0 - 31)
Parameters (out)	IntAlarmAction FSPAction	Alarm action for the requested alarm FSP action for the requested alarm.(0- Disabled , 1- Enabled)
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is retrieval of the internal alarm, FSP action currently configured for the requested alarm is successful. E_NOT_OK: Operation not successful that is invalid alarm action is returned, retrieval of the internal alarm, FSP action currently configured for the requested alarm is not successful, for example due to invalid parameters.
Description	The purpose of the API is to provide the internal alarm, FSP action currently configured for the requested alarm.	
Source	IFX	
Error handling	DET: SMU_E_PARAM_POINTER: SMU_E_PARAM_POINTER DET is reported when the pointer passed as a parameter to an API is a NULL pointer. SMU_E_PARAM_GROUP: SMU_E_PARAM_GROUP DET is reported when the group ID or the alarm position passed as a parameter to an API is not valid. SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state. SMU_E_STDBY_DISABLED: SMU_E_STDBY_DISABLED DET is reported when any alarm action is configured or performed with respect to the standby domain of SMU without enabling the SMU standby mode. 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	

SMU driver**4.3.3.4 Smu_SetAlarmAction****Table 389 Specification for Smu_SetAlarmAction API**

Syntax	<pre>Std_ReturnType Smu_SetAlarmAction (const Smu_AlarmGroupId AlarmGroup, const Smu_AlarmIdType AlarmPos, const Smu_CoreAlarmActionType AlarmAction, const Smu_FSPActionType FSPAction)</pre>	
Service ID	0xAC	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup AlarmPos AlarmAction FSPAction	Alarm group number (0 - 11, 20, 21) Alarm position within the requested group (0-31) The internal alarm action for the requested alarm group and position This parameter gives the FSP action to be set.(0- Disabled, 1- Enabled)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: The alarm action is set. E_NOT_OK: The alarm action could not be set.
Description	The purpose of the API is to set the desired alarm action for the group and position specified.	
Source	IFX	

SMU driver**Table 389 Specification for Smu_SetAlarmAction API (continued)**

Error handling	<p>DET:</p> <p>SMU_E_PARAM_GROUP: SMU_E_PARAM_GROUP DET is reported when the group ID or the alarm position passed as a parameter to an API is not valid.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>SMU_E_LOCKED: SMU_E_LOCKED DET is reported when the SMU is already in locked state.</p> <p>SMU_E_INVALID_ALARM_ACTION: SMU_E_INVALID_ALARM_ACTION DET is reported when the alarm action to be configured is not valid.</p> <p>SMU_E_STDBY_DISABLED: SMU_E_STDBY_DISABLED DET is reported when any alarm action is configured or performed with respect to the standby domain of SMU without enabling the SMU standby 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	None

4.3.3.5 Smu_ClearAlarmStatus**Table 390 Specification for Smu_ClearAlarmStatus API**

Syntax	<pre>Std_ReturnType Smu_ClearAlarmStatus (const Smu_AlarmGroupId AlarmGroup, const Smu_AlarmIdType AlarmPos)</pre>	
Service ID	0xAD	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup AlarmPos	Alarm group number (0 - 11, 20, 21) Alarm position within the requested group(0 - 31)
Parameters (out)	-	-
Parameters (in - out)	-	-

SMU driver**Table 390 Specification for Smu_ClearAlarmStatus API (continued)**

Return	Std_ReturnType	E_OK: The alarm status is cleared successfully E_NOT_OK: The alarm status is not cleared successfully
Description	The purpose of the API is to clear SMU alarm status of the requested alarm.	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_PARAM_GROUP: SMU_E_PARAM_GROUP DET is reported when the group ID or the alarm position passed as a parameter to an API is not valid.</p> <p>SMU_E_STDBY_DISABLED: SMU_E_STDBY_DISABLED DET is reported when any alarm action is configured or performed with respect to the standby domain of SMU without enabling the SMU standby mode.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SMU_E_CLEAR_ALARM_STATUS_FAILURE: SMU_E_CLEAR_ALARM_STATUS_FAILURE DEM is reported when the clearing of alarm status 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	If the source of the requested alarm is not cleared or disabled, then the status of the alarm may remain set after execution of Smu_ClearAlarmStatus API even though the alarm had been cleared in the API.	

4.3.3.6 Smu_GetAlarmStatus**Table 391 Specification for Smu_GetAlarmStatus API**

Syntax	<pre>Std_ReturnType Smu_GetAlarmStatus (const Smu_AlarmGroupId AlarmGroup, uint32 * const AlarmStatus)</pre>	
Service ID	0xAF	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup	Group id of the alarm raised. (0 - 11, 20, 21)

SMU driver**Table 391 Specification for Smu_GetAlarmStatus API (continued)**

Parameters (out)	AlarmStatus	Status of the alarm raised
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is retrieval of SMU alarm status of the requested alarm is successful. E_NOT_OK: Operation unsuccessful that is SMU alarm status of the requested alarm is not retrieved.
Description	The purpose of the API is to provide the alarm status of the requested alarm group.	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_PARAM_POINTER: SMU_E_PARAM_POINTER DET is reported when the pointer passed as a parameter to an API is a NULL pointer.</p> <p>SMU_E_PARAM_GROUP: SMU_E_PARAM_GROUP DET is reported when the group ID or the alarm position passed as a parameter to an API is not valid.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_STDBY_DISABLED: SMU_E_STDBY_DISABLED DET is reported when any alarm action is configured or performed with respect to the standby domain of SMU without enabling the SMU standby 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	None	

4.3.3.7 Smu_SetAlarmStatus**Table 392 Specification for Smu_SetAlarmStatus API**

Syntax	Std_ReturnType Smu_SetAlarmStatus (const Smu_AlarmGroupId AlarmGroup, const Smu_AlarmIdType AlarmPos)
Service ID	0xAE
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant

SMU driver**Table 392 Specification for Smu_SetAlarmStatus API (continued)**

Parameters (in)	AlarmGroup AlarmPos	Alarm group number (0 - 11) Alarm position within the requested group(0-31)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is SMU alarm status of the requested alarm is set successfully. E_NOT_OK: Operation unsuccessful that is SMU alarm status of the requested alarm is not set.
Description	The purpose of the API is to set the requested alarm status. This service can be used by the user software to trigger software SMU alarm. For SMU_core during the START state of the SMU, it shall be possible to set any of the alarms. However, during the RUN state, only the software alarms shall be set. The API is applicable only for SMU_core alarm groups and positions.	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>SMU_E_PARAM_GROUP: SMU_E_PARAM_GROUP DET is reported when the group ID or the alarm position passed as a parameter to an API is not valid.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SMU_E_SET_ALARM_STATUS_FAILURE: SMU_E_SET_ALARM_STATUS_FAILURE DEM is reported when the setting of alarm status 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	None	

4.3.3.8 Smu_GetAlarmDebugStatus**Table 393 Specification for Smu_GetAlarmDebugStatus API**

Syntax	Std_ReturnType Smu_GetAlarmDebugStatus (const Smu_AlarmGroupId AlarmGroup, uint32 * const AlarmStatus)
---------------	--

SMU driver**Table 393 Specification for Smu_GetAlarmDebugStatus API (continued)**

Service ID	0xB0	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup	Alarm group number (0 - 11)
Parameters (out)	AlarmStatus	Alarm Debug Status register value
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is retrieval of alarm status from debug register is successful. E_NOT_OK: Operation unsuccessful that is retrieval of SMU alarm debug status of the requested alarm is not successful.
Description	The purpose of the API is to provide the alarm status for the requested alarm group from the stored debug registers. The debug status is applicable only for SMU_core.	
Source	IFX	
Error handling	DET: SMU_E_PARAM_POINTER: SMU_E_PARAM_POINTER DET is reported when the pointer passed as a parameter to an API is a NULL pointer. SMU_E_PARAM_GROUP: SMU_E_PARAM_GROUP DET is reported when the group ID or the alarm position passed as a parameter to an API is not valid. SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	This is required by the application to know the reason of the malfunction esp. In case the internal reaction was configured to be a reset. This is only for Smu_core.	

4.3.3.9 Smu_LockConfigRegs**Table 394 Specification for Smu_LockConfigRegs API**

Syntax	Std_ReturnType Smu_LockConfigRegs (void)
---------------	---

SMU driver**Table 394 Specification for Smu_LockConfigRegs API (continued)**

Service ID	0xB1	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is SMU configuration registers are locked successfully. E_NOT_OK: Operation not successful that is SMU configuration registers are not locked successfully.
Description	<p>The purpose of the API is to permanently lock the SMU configuration registers to prevent any modification to configuration register content.</p> <p>The API can be called from any core. However, it is recommended to call the API from only one core at any instance of time to ensure consistent behavior.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_LOCKED: SMU_E_LOCKED DET is reported when the SMU is already in locked state.</p> <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <p>SMU_E_SF_CFG_LOCKED: The safety error is reported when the configuration registers do not get locked after applying permanent lock.</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	None	

SMU driver**4.3.3.10 Smu_ReleaseFSP****Table 395 Specification for Smu_ReleaseFSP API**

Syntax	Std_ReturnType Smu_ReleaseFSP (void)	
Service ID	0xB2	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is setting the PCS bit is successful. E_NOT_OK: Operation not successful or SMU is already in RUN state.
Description	<p>The purpose of the API is to switch the SMU peripheral from the FAULT state to the RUN state. This also switches the error pin from the FAULT state to FAULT-FREE state. Additionally, this API can be used to change the FSP state from the power-on state to the Fault-free state. This is essential to setup the error pin to drive the FSP. It is also required for testing of FSP pin. The transitions of states require certain clock cycles to reflect. The API returns before this transition is observed.</p> <p>The API can be called from any core. However, it is recommended to call the API from only one core at any instance of time to ensure consistent behavior.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SMU_E_RELEASE_FSP_FAILURE: SMU_E_RELEASE_FSP_FAILURE DEM is reported when the FSP cannot be released.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	

SMU driver**Table 395 Specification for Smu_ReleaseFSP API (continued)**

User hints	None
-------------------	------

4.3.3.11 Smu_ActivateFSP**Table 396 Specification for Smu_ActivateFSP API**

Syntax	Std_ReturnType Smu_ActivateFSP (void)	
Service ID	0xB3	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is activation of FSP is successful E_NOT_OK: Operation not successful.
Description	<p>The purpose of the API is to activate the FSP to indicate a FAULT state on the error pin to the safe state switching device. When FSP is activated the SMU reaches the fault state. This can be confirmed by reading the SMU state in hardware. Also, In the SMU START state, activation of FSP is only possible using this API as alarms are not processed.</p> <p>Additionally, this is required for the testing of the FSP timing.</p> <p>The transitions of states require certain clock cycles to reflect. The API returns before this transition is observed.</p>	
Source	IFX	
Error handling	<p>DET: SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>Runtime Errors: None</p> <p>DEM: SMU_E_ACTIVATE_FSP_FAILURE: SMU_E_ACTIVATE_FSP_FAILURE DEM is reported when activation of FSP fails.</p> <p>Safety Errors: None</p> <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	

SMU driver**Table 396 Specification for Smu_ActivateFSP API (continued)**

Configuration dependencies	-
User hints	None

4.3.3.12 Smu_SetupErrorPin**Table 397 Specification for Smu_SetupErrorPin API**

Syntax	Std_ReturnType Smu_SetupErrorPin (void)	
Service ID	0xB4	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is switching of the error pin from GPIO mode to SMU mode is successful. E_NOT_OK: Operation not successful or the error pin is already set.
Description	The purpose of the API is to enable the SMU to control the error pin. This API switches the error pin from GPIO mode to SMU mode. Only after switching to the SMU mode, SMU can control the error pin. The transitions of states require certain clock cycles to reflect. The API returns before this transition is observed.	
Source	IFX	

SMU driver**Table 397 Specification for Smu_SetupErrorPin API (continued)**

Error handling	<p>DET:</p> <p>SMU_E_LOCKED: SMU_E_LOCKED DET is reported when the SMU is already in locked state.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</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

4.3.3.13 Smu_ReleaseErrorPin**Table 398 Specification for Smu_ReleaseErrorPin API**

Syntax	Std_ReturnType Smu_ReleaseErrorPin (void)	
Service ID	0xB5	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is control of error pin is successfully released from SMU E_NOT_OK: Operation not successful.
Description	The purpose of the API is to release the control of the error pin. The transitions of modes require certain clock cycles to reflect. The API returns before this transition is observed.	
Source	IFX	

SMU driver**Table 398 Specification for Smu_ReleaseErrorPin API (continued)**

Error handling	<p>DET:</p> <p>SMU_E_LOCKED: SMU_E_LOCKED DET is reported when the SMU is already in locked state.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</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

4.3.3.14 Smu_RTStop**Table 399 Specification for Smu_RTStop API**

Syntax	<pre>Std_ReturnType Smu_RTStop (const uint8 TimerNum)</pre>	
Service ID	0xB6	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	TimerNum	Recovery Timer unit to be stopped.(0,1)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is requested recovery timer is stopped successfully E_NOT_OK: Operation not successful that is requested recovery timer could not be stopped successfully, for example due to invalid parameters
Description	The purpose of the API is to stop the requested recovery timer unit. Possible use case: when a fault occurs, error handler might be triggered. However, this error handler should setup a recovery mechanism or error mitigation mechanism within a finite interval of time to prevent the system from failing.	
Source	IFX	

SMU driver**Table 399 Specification for Smu_RTStop API (continued)**

Error handling	<p>DET:</p> <p>SMU_E_INVALID_TIMER: SMU_E_INVALID_TIMER DET is reported when the timer value passed as a parameter to an API is not valid.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SMU_E_RT_STOP_FAILURE: SMU_E_RT_STOP_FAILURE DEM is reported when the Recovery timer cannot be stopped.</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

4.3.3.15 Smu_GetRTMissedEvent**Table 400 Specification for Smu_GetRTMissedEvent API**

Syntax	<pre>Std_ReturnType Smu_GetRTMissedEvent (const uint8 TimerNum, boolean * const EventMissed)</pre>	
Service ID	0xB7	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	TimerNum	Recovery Timer unit for which the status has to be procured (0,1).
Parameters (out)	EventMissed	<p>EventMissed :</p> <p>TRUE: Event has been missed</p> <p>FALSE: Event has NOT been missed</p>
Parameters (in - out)	-	-
Return	Std_ReturnType	<p>E_OK: Operation successful that is check for missed events successful.</p> <p>E_NOT_OK: Operation not successful that is check for missed events not successful, for example due to invalid parameters</p>

SMU driver**Table 400 Specification for Smu_GetRTMissedEvent API (continued)**

Description	The purpose of the API is to know if any alarms requiring the requested recovery timer were set while the recovery timer was running.
Source	IFX
Error handling	<p>DET:</p> <p>SMU_E_INVALID_TIMER: SMU_E_INVALID_TIMER DET is reported when the timer value passed as a parameter to an API is not valid.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>SMU_E_PARAM_POINTER: SMU_E_PARAM_POINTER DET is reported when the pointer passed as a parameter to an API 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	-
User hints	None

4.3.3.16 Smu_ActivatePES**Table 401 Specification for Smu_ActivatePES API**

Syntax	Std_ReturnType Smu_ActivatePES (void)	
Service ID	0xB8	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is activation of the Port Emergency Stop (PES) is successful. E_NOT_OK: Operation not successful.

SMU driver**Table 401 Specification for Smu_ActivatePES API (continued)**

Description	The purpose of this API is to trigger the activation of the Port Emergency Stop (PES). The PES is also directly controlled by the SMU_core when entering the FAULT state.
Source	IFX
Error handling	<p>DET:</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SMU_E_ACTIVATE_PES_FAILURE: SMU_E_ACTIVATE_PES_FAILURE DEM is reported when activation of the PES feature 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	None

4.3.3.17 Smu_RegisterMonitor**Table 402 Specification for Smu_RegisterMonitor API**

Syntax	<pre>Std_ReturnType Smu_RegisterMonitor (const uint16 * const RegMonPtr, Smu_SffTestResType * const RegMonResult)</pre>	
Service ID	0xB9	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	RegMonPtr	Pointer to the array, which holds the modules which have SFF test enabled. The total number of elements is the total number of modules, which can undergo SFF tests. The elements have to be as per the bits specified in RMCTL register. Additionally, the module needs to be present in the specific device.
Parameters (out)	RegMonResult	Pointer to array for the SFF test results for the modules. In case a module test was not enabled but has an error recorded, it will indicate that failure as well.
Parameters (in - out)	-	-

SMU driver**Table 402 Specification for Smu_RegisterMonitor API (continued)**

Return	Std_ReturnType	E_OK: Operation successful that is the SFF test execution was completed successfully, irrespective of the SFF test result E_NOT_OK: Operation not successful that is the SFF test execution was not completed successfully, irrespective of the SFF test result
Description	<p>The purpose of the API is to provide the initialization, execution and termination of the safety flip-flop tests to be executed for different modules as enabled in the RegMonPtr parameter. The user shall take care of the prerequisites for safety flip-flop test as mentioned in the HW UM before invoking the API.</p> <p>The API returns whether the safety flip-flop test execution has been successfully completed or not, irrespective of the safety flip-flop test results. The result of the safety flip-flop tests can be obtained through the RegMonResult parameter. The RegMonResult parameter needs to be checked only when the API returns E_OK, which implies that the safety flip-flop test execution has been successfully completed.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_LOCKED: SMU_E_LOCKED DET is reported when the SMU is already in locked state.</p> <p>SMU_E_PARAM_POINTER: SMU_E_PARAM_POINTER DET is reported when the pointer passed as a parameter to an API is a NULL pointer.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SMU_E_SFF_TEST_FAILURE: SMU_E_SFF_TEST_FAILURE DEM is reported when timeout occurs before SFF test status is reflected.</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	

4.3.3.18 Smu_GetSmuState**Table 403 Specification for Smu_GetSmuState API**

Syntax	Smu_CoreStateType Smu_GetSmuState (void)
Service ID	0xBA
Sync/Async	Synchronous

SMU driver**Table 403 Specification for Smu_GetSmuState API (continued)**

ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Smu_CoreStateType	State of SMU core state machine
Description	The purpose of the API is to provide the current state of the SMU core. This is referred to as the safety status of the system as all critical faults will cause SMU to go to the FAIL state.	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in 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	-	
User hints	None	

4.3.3.19 Smu_ActivateRunState**Table 404 Specification for Smu_ActivateRunState API**

Syntax	<pre>Std_ReturnType Smu_ActivateRunState (const uint32 Cmd)</pre>	
Service ID	0xBB	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	Cmd	Command to switch the SMU to the RUN state
Parameters (out)	-	-

SMU driver**Table 404 Specification for Smu_ActivateRunState API (continued)**

Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is activation of fault free RUN state is successful. E_NOT_OK: Operation not successful that is activation of fault free RUN state is not successful, for example SMU is not initially in START state.
Description	<p>The purpose of the API is to allow switching the SMU peripheral into the RUN fault-free state as requested by the caller. The SMU validates the request based on integrity checks of SMU (that is check of the command value).</p> <p>The API can be called from any core. However, it is recommended to call the API from only one core at any instance of time to ensure consistent behavior.</p>	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SMU_E_ACTIVATE_RUN_STATE_FAILURE: SMU_E_ACTIVATE_RUN_STATE_FAILURE DEM is reported when the activation of RUN state 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	-	

4.3.3.20 Smu_GetVersionInfo**Table 405 Specification for Smu_GetVersionInfo API**

Syntax	void Smu_GetVersionInfo (Std_VersionInfoType * const VersionInfoPtr)
Service ID	0xBC
Sync/Async	Synchronous
ASIL Level	B
Re-entrancy	Reentrant

SMU driver**Table 405 Specification for Smu_GetVersionInfo API (continued)**

Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer to store information about the module
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to return the version information of the SMU driver. The version information includes Module ID, Vendor ID and vendor specific version numbers. This function is available only if the SMU_VERSION_INFO_API is set ON.	
Source	IFX	
Error handling	DET: SMU_E_PARAM_POINTER: SMU_E_PARAM_POINTER DET is reported when the pointer passed as a parameter to an API is 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	SmuVersionInfoApi	
User hints	-	

4.3.3.21 Smu_CoreAliveTest**Table 406 Specification for Smu_CoreAliveTest API**

Syntax	<pre>Std_ReturnType Smu_CoreAliveTest (void)</pre>	
Service ID	0xBD	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-

SMU driver**Table 406 Specification for Smu_CoreAliveTest API (continued)**

Return	Std_ReturnType	E_OK: The SMU_AliveTest start command is executed successfully. E_NOT_OK: The SMU_AliveTest start command is not executed successfully
Description	The purpose of the API is to provide the means to execute the SMU_AliveTest command that checks the smu_core_alive signal. The API returns whether the SMU_AliveTest command to start the test has been successfully executed or not. It does not return the result of the smu_core_alive test. The result of the smu_core_alive test can be obtained by reading the status flag for the SMU_core alive alarm (alarm 16 of alarm group 21) by means of the Smu_GetAlarmStatus API. The Smu_CoreAliveTest API also executes the SMU_AliveTest command to stop the test, which provides the user with flexibility to call the API cyclically during runtime. The SMU_stby has to remain enabled and the SMU_core has to be in the START state to execute this command.	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_STDBY_DISABLED: SMU_E_STDBY_DISABLED DET is reported when any alarm action is configured or performed with respect to the standby domain of SMU without enabling the SMU standby mode.</p> <p>Runtime Errors: None</p> <p>DEM:</p> <p>SMU_E_CORE_ALIVE_FAILURE: SMU_E_CORE_ALIVE_FAILURE DEM is reported when the core alive test of SMU core 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	The user shall read the status flags for SMU_core alive alarm (alarm 16 of alarm group 21) using the Smu_GetAlarmStatus() API to check the result of the smu_core_alive test after execution of the Smu_CoreAliveTest() API. The user shall clear the status of SMU_core alive alarm (alarm 16 of alarm group 21) using the Smu_ClearAlarmStatus() API, after checking the result of smu_core_alive test so that further alarm detection is possible.	

4.3.3.22 Smu_InitCheck**Table 407 Specification for Smu_InitCheck API**

Syntax	Std_ReturnType Smu_InitCheck (const Smu_ConfigType * const ConfigPtr)
Service ID	0xA9

SMU driver**Table 407 Specification for Smu_InitCheck API (continued)**

Sync/Async	Synchronous	
ASIL Level	B(D)	
Re-entrancy	Reentrant	
Parameters (in)	ConfigPtr	Pointer to the SMU configuration for initialization
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : Initialization comparison successful E_NOT_OK : Initialization comparison failed
Description	The purpose of the API is to check the initialization values after SMU is initialized. The API should be called after the SMU driver is initialized to check the initialization 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	SmulInitCheckApi	
User hints	None	

4.3.3.23 Smu_GetAlarmExecutionStatus**Table 408 Specification for Smu_GetAlarmExecutionStatus API**

Syntax	<pre>Std_ReturnType Smu_GetAlarmExecutionStatus (const uint32 AlarmExecStatusReq, uint32 * const AlarmExecStatus)</pre>	
Service ID	0xBE	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	AlarmExecStatusReq	Alarm reaction for which execution status is requested.
Parameters (out)	AlarmExecStatus	Pointer that stores the alarm execution status result.

SMU driver**Table 408 Specification for Smu_GetAlarmExecutionStatus API (continued)**

Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is retrieval of the status of the alarm reactions executed is successful. E_NOT_OK: Operation not successful that is retrieval of the status of the alarm reactions executed is not successful, for example due to invalid parameters
Description	This API gives the status of the alarm reactions executed.	
Source	IFX	
Error handling	<p>DET:</p> <p>SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.</p> <p>SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.</p> <p>SMU_E_INVALID_EXECUTION_STATUS: SMU_E_INVALID_EXECUTION_STATUS DET is reported when an invalid error mechanism is requested for alarm execution status.</p> <p>SMU_E_PARAM_POINTER: SMU_E_PARAM_POINTER DET is reported when the pointer passed as a parameter to an API 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	-	
User hints	None	

4.3.3.24 Smu_ClearAlarmExecutionStatus**Table 409 Specification for Smu_ClearAlarmExecutionStatus API**

Syntax	Std_ReturnType Smu_ClearAlarmExecutionStatus (const uint32 AlarmExecStatusReq)	
Service ID	0xBF	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	AlarmExecStatusReq	Alarm reactions for which the alarm execution status clear has been requested

SMU driver
Table 409 Specification for Smu_ClearAlarmExecutionStatus API (continued)

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is the status of the alarm reactions executed is cleared successfully E_NOT_OK: Operation not successful that is the status of the alarm reactions executed is not cleared successfully, for example due to invalid parameters
Description	The purpose of the API is to clear the alarm reaction execution status for the requested alarm reaction.	
Source	IFX	
Error handling	DET: SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state. SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED. SMU_E_INVALID_EXECUTION_STATUS: SMU_E_INVALID_EXECUTION_STATUS DET is reported when an invalid error mechanism is requested for alarm execution status. 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	

4.3.4 Notifications and Callbacks

SMU driver does not have any notifications or callbacks.

4.3.5 Scheduled functions

SMU driver does not have any scheduled functions.

4.3.6 Interrupt service routines

Interrupt handlers are not applicable for SMU driver.

4.3.7 Error codes classification

This section explains various error types and their corresponding source APIs.

SMU driver

4.3.7.1 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 410 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.	IFX	SMU_E_UNINIT=0x01	Smu_ClearAlarmExecutionStatus, Smu_GetAlarmDebugStatus, Smu_GetAlarmExecutionsStatus, Smu_GetRTMissedEvent, Smu_RTStop, Smu_SetAlarmStatus, Smu_GetAlarmStatus, Smu_ClearAlarmStatus, Smu_CoreAliveTest, Smu_ActivateRunState, Smu_GetSmuState, Smu_RegisterMonitor, Smu_ActivatePES, Smu_ReleaseErrorPin, Smu_SetupErrorPin, Smu_ActivateFSP, Smu_ReleaseFSP, Smu_LockConfigRegs, Smu_GetAlarmAction, Smu_SetAlarmAction, Smu_DelInit
SMU_E_ALREADY_INITIALIZED DET is reported when SMU is already initialized.	IFX	SMU_E_ALREADY_INITIALIZED=0x02	Smu_Init
SMU_E_INIT_FAILED DET is reported when initialization of SMU driver fails due to incorrect configuration parameter.	IFX	SMU_E_INIT_FAILED=0x03	Smu_Init
SMU_E_PARAM_POINTER DET is reported when the pointer passed as a parameter to an API is a NULL pointer.	IFX	SMU_E_PARAM_POINTER=0x04	Smu_GetRTMissedEvent, Smu_GetVersionInfo, Smu_RegisterMonitor, Smu_GetAlarmExecutionsStatus, Smu_GetAlarmDebugStatus, Smu_GetAlarmStatus, Smu_GetAlarmAction

SMU driver**Table 410 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
SMU_E_PARAM_GROUP DET is reported when the group ID or the alarm position passed as a parameter to an API is not valid.	IFX	SMU_E_PARAM_GROUP=0x05	Smu_SetAlarmStatus, Smu_ClearAlarmStatus, Smu_SetAlarmAction, Smu_GetAlarmDebugStatus, Smu_GetAlarmStatus, Smu_GetAlarmAction
SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.	IFX	SMU_E_INVALID_DRIVER_STATE=0x06	Smu_ClearAlarmExecutionStatus, Smu_GetAlarmExecutionsStatus, Smu_GetRTMissedEvent, Smu_RTStop, Smu_ClearAlarmStatus, Smu_CoreAliveTest, Smu_ActivateRunState, Smu_RegisterMonitor, Smu_ActivatePES, Smu_ReleaseErrorPin, Smu_ActivateFSP, Smu_ReleaseFSP, Smu_SetupErrorPin, Smu_LockConfigRegs, Smu_SetAlarmAction, Smu_SetAlarmStatus
SMU_E_INVALID_TIMER DET is reported when the timer value passed as a parameter to an API is not valid.	IFX	SMU_E_INVALID_TIMER=0x07	Smu_RTStop, Smu_GetRTMissedEvent
SMU_E_STDBY_DISABLED DET is reported when any alarm action is configured or performed with respect to the standby domain of SMU without enabling the SMU standby mode.	IFX	SMU_E_STDBY_DISABLED=0x08	Smu_GetAlarmStatus, Smu_SetAlarmAction, Smu_CoreAliveTest, Smu_GetAlarmAction, Smu_ClearAlarmStatus
SMU_E_LOCKED DET is reported when the SMU is already in locked state.	IFX	SMU_E_LOCKED=0x09	Smu_Delnit, Smu_Init, Smu_SetAlarmAction, Smu_RegisterMonitor, Smu_ReleaseErrorPin, Smu_SetupErrorPin, Smu_LockConfigRegs
SMU_E_INVALID_ALARM_ACTION DET is reported when the alarm action to be configured is not valid.	IFX	SMU_E_INVALID_ALARM_ACTION=0x0A	Smu_SetAlarmAction

SMU driver**Table 410 Description of development errors reported (continued)**

Description	Source	Error code and value	Applicable APIs
SMU_E_INVALID_EXECUTION_STATUS DET is reported when an invalid error mechanism is requested for alarm execution status.	IFX	SMU_E_INVALID_EXECUTION_STATUS=0x0B	Smu_ClearAlarmExecutionStatus, Smu_GetAlarmExecutionStatus
SMU_E_CORE_MISMATCH DET is reported when the Init or De-Init is called from any core other than the master core.	IFX	SMU_E_CORE_MISMATCH=0x68	Smu_Delnit, Smu_Init

4.3.7.2 Production errors

The following table lists all the production errors reported by the driver.

Table 411 Description of production errors reported

Description	Source	Error code and value	Applicable APIs
SMU_E_ACTIVATE_FSP_FAILURE DEM is reported when activation of FSP fails.	IFX	SMU_E_ACTIVATE_FSP_FAILURE=Assigned by DEM	Smu_ActivateFSP
SMU_E_ACTIVATE_PES_FAILURE DEM is reported when activation of the PES feature fails.	IFX	SMU_E_ACTIVATE_PES_FAILURE=Assigned by DEM	Smu_ActivatePES
SMU_E_ACTIVATE_RUN_STATE_FAILURE DEM is reported when the activation of RUN state fails.	IFX	SMU_E_ACTIVATE_RUN_STATE_FAILURE=Assigned by DEM	Smu_ActivateRunState
SMU_E_CLEAR_ALARM_STATUS_FAILURE DEM is reported when the clearing of alarm status fails.	IFX	SMU_E_CLEAR_ALARM_STATUS_FAILURE=Assigned by DEM	Smu_ClearAlarmStatus
SMU_E_RELEASE_FSP_FAILURE DEM is reported when the FSP cannot be released.	IFX	SMU_E_RELEASE_FSP_FAILURE=Assigned by DEM	Smu_ReleaseFSP
SMU_E_RT_STOP_FAILURE DEM is reported when the Recovery timer cannot be stopped.	IFX	SMU_E_RT_STOP_FAILURE=Assigned by DEM	Smu_RTStop

SMU driver
Table 411 Description of production errors reported (continued)

Description	Source	Error code and value	Applicable APIs
SMU_E_SET_ALARM_STAT US_FAILURE DEM is reported when the setting of alarm status fails.	IFX	SMU_E_SET_ALARM_STATUS_FAILURE=Assigned by DEM	Smu_SetAlarmStatus
SMU_E_SFF_TEST_FAILURE DEM is reported when an SFF test fails.	IFX	SMU_E_SFF_TEST_FAILURE=Assigned by DEM	Smu_RegisterMonitor
SMU_E_CORE_ALIVE_FAILURE DEM is reported when the core alive test of SMU core fails.	IFX	SMU_E_CORE_ALIVE_FAILURE=Assigned by DEM	Smu_CoreAliveTest

4.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 412 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
The safety error is reported when the configuration registers do not get locked after applying permanent lock.	IFX	SMU_E_SF_CFG_LOCKED=0xC8	Smu_LockConfigRegs

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

There are no deviations reported for the SMU driver.

4.3.8.2 Limitations

There are no known limitations for the SMU driver.

4.3.9 Unsupported hardware features

The following hardware feature is not currently supported by the SMU driver: MONBISTCTRL feature for Smu_stdby

UART driver

5 **UART driver**

5.1 **User information**

5.1.1 **Description**

The UART driver is responsible for providing communication services as per the UART protocol. The ASCLIN module provides hardware support for asynchronous communication to realize the UART protocol. The UART driver provides functionality for configuration, initialization, data transmission, reception and also provides optional features such as abort transmission and abort reception.

5.1.2 **Hardware-software mapping**

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

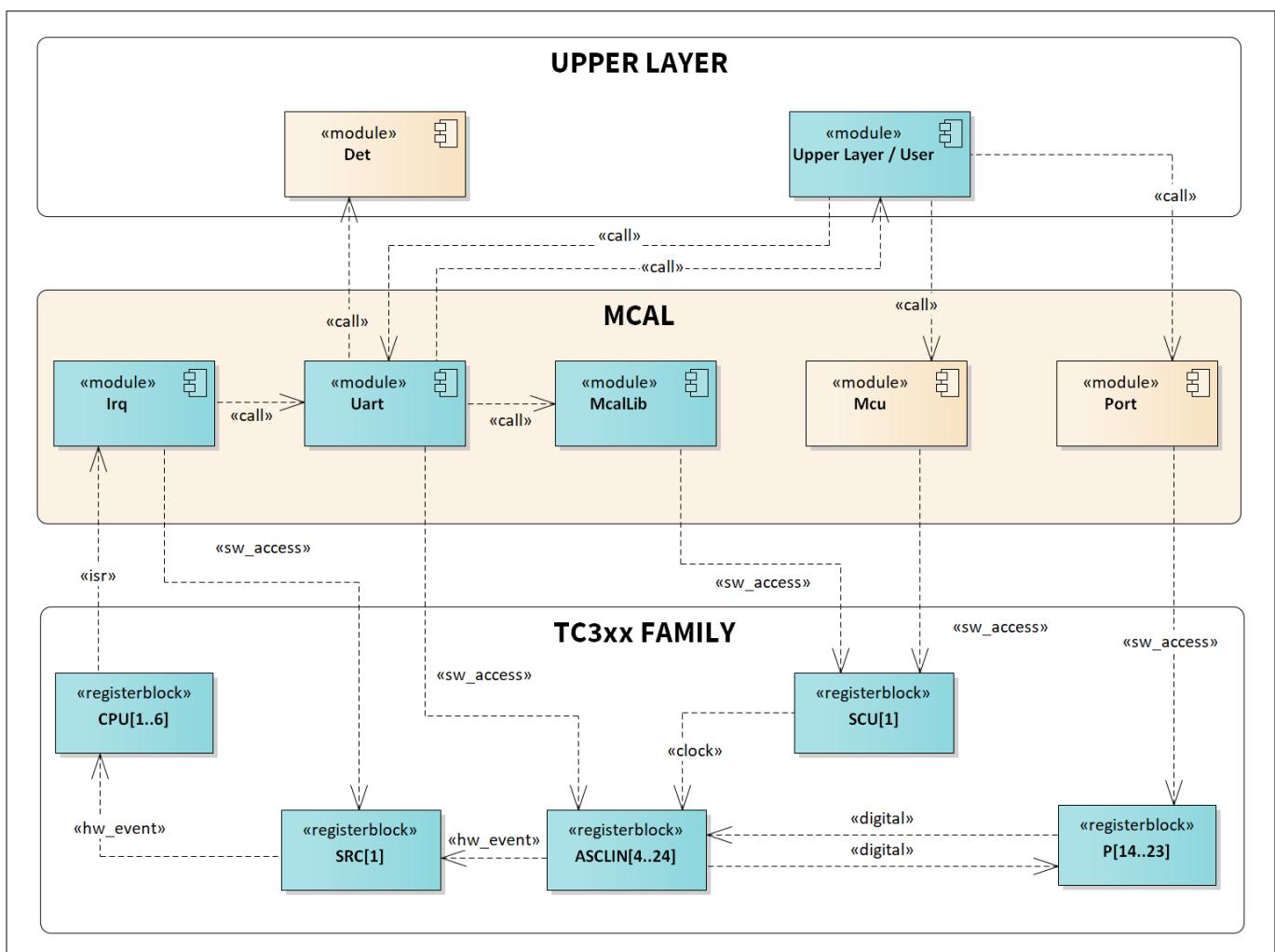


Figure 52 Mapping of hardware-software interfaces

5.1.2.1 **ASCLIN: primary hardware peripheral**

Hardware functional features

The UART driver uses the ASCLIN for transmission and reception of data. The key hardware functional features used by the driver are:

UART driver

- Full-duplex asynchronous operating modes
- Supports half duplex operating mode
- 16 bytes TXFIFO
- 16 bytes RXFIFO
- 2 to 16 bits data frames
- Parity-bit generation/checking
- One or two stop bits
- Baud rate configuration
- Optional RTS / CTS handshaking
- Programmable over-sampling 4 to 16 times per bit
- Programmable sample point position
- Interrupt generation
- Interrupt signals capable of triggering a CPU
- Programmable digital glitch filter and median filter for incoming bit stream
- Pack / unpack capabilities of the Tx and Rx FIFO
- Shift direction LSB first for ASC

The unsupported feature of the ASCLIN is:

- Internal loop-back mode

Users of the hardware

The LIN and UART drivers utilizes the ASCLIN IP. The allocation of ASLIN 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 UART driver.

Hardware events

The UART driver uses the following hardware events from the ASCLIN IP:

- Configurable transmit FIFO level interrupt
- Configurable receive FIFO level interrupt
- Error condition parity error, frame error, RXFIFO overflow error

5.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The UART driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB, fASCLINF and fASCLINS 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 UART driver.

Hardware events

Hardware events from the SCU are not used by the UART driver.

UART driver**5.1.2.3 Port: dependent hardware peripheral****Hardware functional features**

The ARX, ATX, CTS and RTS signals are routed to the ASCLIN through the port pads. These signals 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 UART driver.

5.1.2.4 SRC: dependent hardware peripheral**Hardware functional features**

The UART driver depends on the interrupt router for raising an interrupt to the CPU based on the transmit and receive events, which indicates successful data transmission and reception respectively.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software. No functional block of the interrupt router is administered by the UART driver.

Hardware diagnostic features

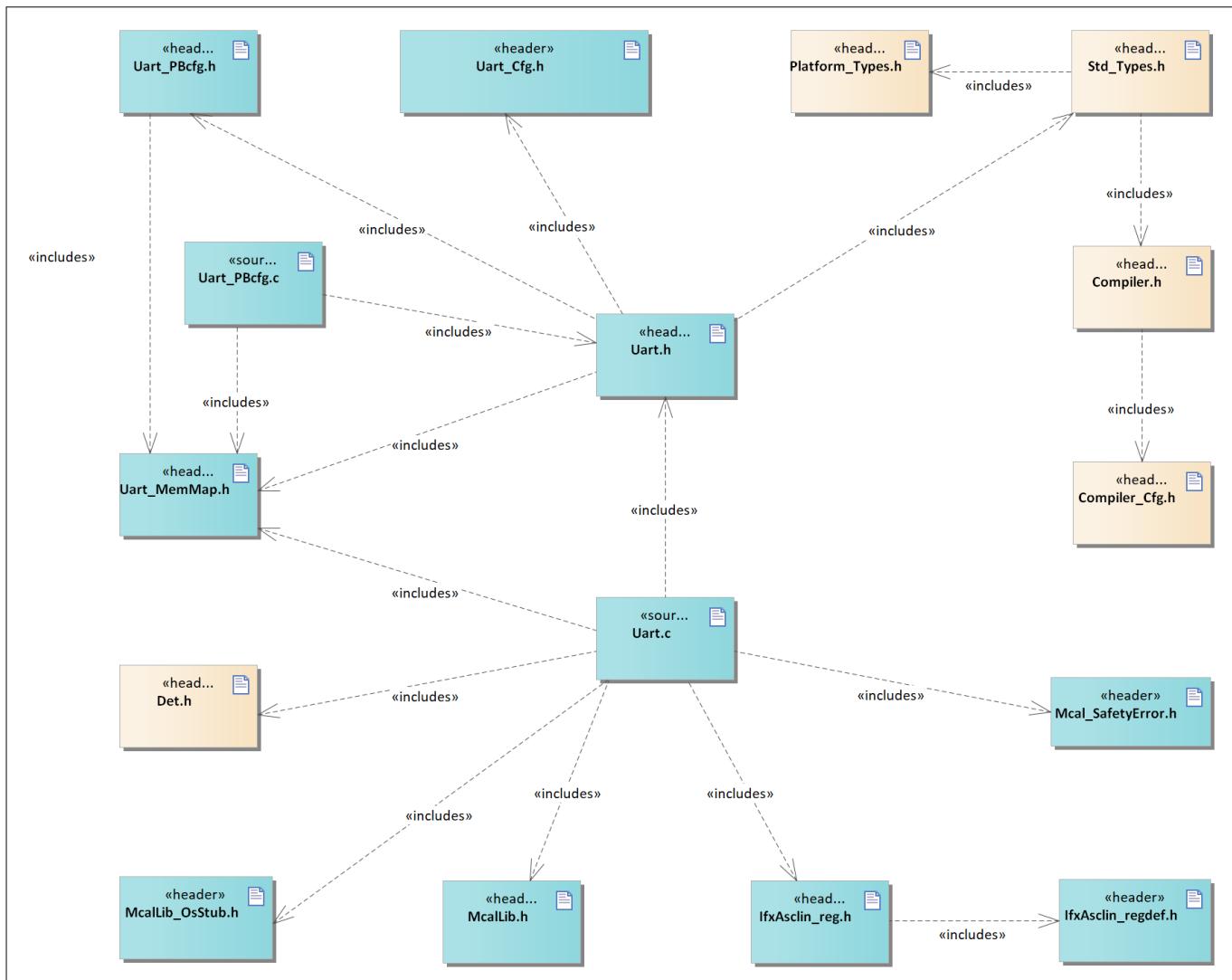
The SMU alarms configured for interrupt router are not monitored by the UART driver.

Hardware events

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

5.1.3 File structure**5.1.3.1 C file structure**

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

UART driver**Figure 53 C file structure****Table 413 C file structure**

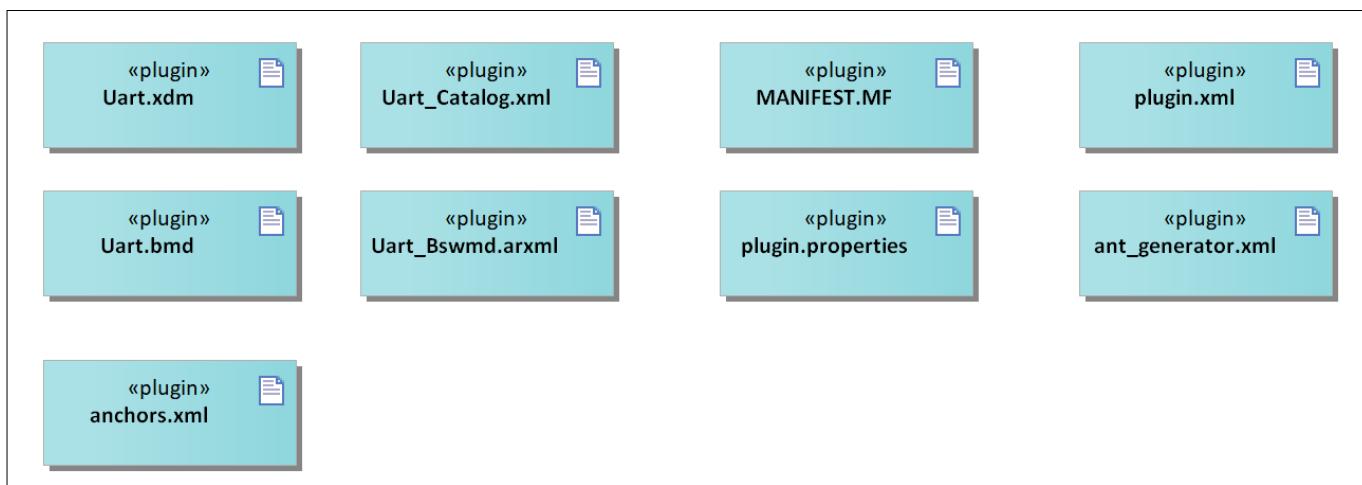
File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Compiler_Cfg.h	Configuration header file for compiler abstraction
Det.h	Provides the exported interfaces of Development Error Tracer
IfxAsclin_Reg.h	SFR header file for ASCLIN
IfxAsclin_Regdef.h	SFR header file for ASCLIN
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 the 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
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR

UART driver
Table 413 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.
Uart.c	File (static) containing implementation of APIs and ISRs
Uart.h	Header file (Static) defining prototypes of data structures, APIs and Interrupt handlers
Uart_Cfg.h	Contains UART driver pre-compile configuration parameters
Uart_MemMap.h	File containing the memory section definitions used by the UART driver
Uart_PBcfg.c	File (generated) containing objects to data structures
Uart_PBcfg.h	Post-build header file for the UART driver

5.1.3.2 Code generator plugin files

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

**Figure 54 Code generator plugin files****Table 414 Code generator plugin files**

File name	Description
MANIFEST.MF	Tresos plugin support file containing the metadata for UART driver
Uart.bmd	AUTOSAR format XML data model schema file (for each device)
Uart.xdm	Tresos format XML data model schema file
Uart_Bswmd.arxml	AUTOSAR format module description file
Uart_Catalog.xml	AUTOSAR format catalog file
anchors.xml	Tresos anchors support file for the UART 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 UART driver

UART driver
Table 414 Code generator plugin files (continued)

File name	Description
plugin.xml	Tresos plugin support file for the UART driver

5.1.4 Integration hints

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

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 UART 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 `Uart_MemMap.h` file.

The `Uart_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.

UART driver

```

/*To be used for all global or static variables.*/
#if defined UART_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_8
/* User Pragma here */
#define UART_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_8
#define MEMMAP_ERROR
#elif defined UART_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_8
/* User Pragma here */
#define UART_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_8
#define MEMMAP_ERROR
#elif defined UART_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
/* User Pragma here */
#define UART_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
#define MEMMAP_ERROR
#elif defined UART_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
#define UART_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
#define MEMMAP_ERROR
#elif defined UART_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
/* User Pragma here */
#define UART_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined UART_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
/* User Pragma here */
#define UART_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
/*To be used for global or static constants.*/
#elif defined UART_START_SEC_CONST_ASIL_B_LOCAL_32
/* User Pragma here */
#define UART_START_SEC_CONST_ASIL_B_LOCAL_32
#define MEMMAP_ERROR
#elif defined UART_STOP_SEC_CONST_ASIL_B_LOCAL_32
/* User Pragma here */
#define UART_STOP_SEC_CONST_ASIL_B_LOCAL_32
#define MEMMAP_ERROR
/* UART module configuration data */
#elif defined UART_START_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
/* User Pragma here */
#define UART_START_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
#elif defined UART_STOP_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
/* User Pragma here */
#define UART_STOP_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
#define MEMMAP_ERROR
/* Code section */
#elif defined UART_START_SEC_CODE_ASIL_B_LOCAL
/* User Pragma here */
#define UART_START_SEC_CODE_ASIL_B_LOCAL
#define MEMMAP_ERROR
#elif defined UART_STOP_SEC_CODE_ASIL_B_LOCAL
/* User Pragma here */
#define UART_STOP_SEC_CODE_ASIL_B_LOCAL
#define MEMMAP_ERROR
#endif

```

UART driver

```
#if defined MEMMAP_ERROR
#error "Uart_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 UART driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the UART driver must process all the errors reported to the DET module through the API `Det_ReportError()`.

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 UART driver.

- **SchM**

The SchM is not required for integrating the UART driver.

- **Safety error**

The UART 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 UART driver does not implement any notifications. However, the UART driver reports the completion of a transmit, receive, abort transmit and abort receive operation through notification functions. These notification functions can be configured by the user in EB tresos for each UART channel separately.

- **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.

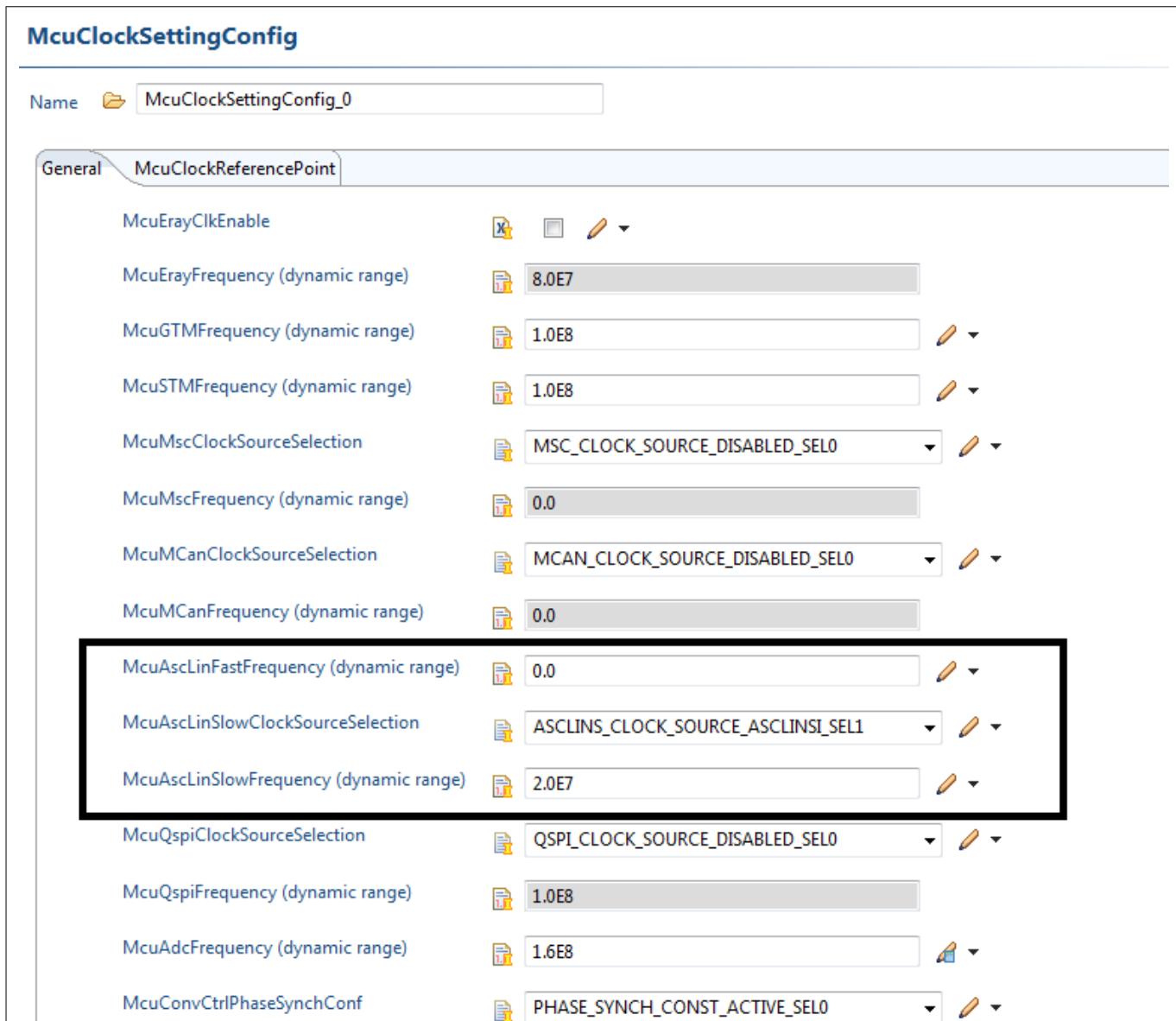
5.1.4.2 Multicore and Resource Manager

The UART driver does not support execution on multiple cores simultaneously.

5.1.4.3 MCU support

The UART driver is dependent on the MCU driver for clock configuration and channel allocation services. The initialization of the UART driver must be started only after completing the MCU initialization. The following must be considered while configuring the MCU driver in tresos:

- 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:

UART driver

Figure 55 UART fast or slow clock configuration

- The ASCLIN hardware IP is shared between UART and LIN drivers. The resource allocation is configured in the MCU driver. Following is the example of allocation of the ASCLIN0 hardware resource to the UART driver. Index ID for the node represents the ASCLIN Hw resource ID.

UART driver

McuHardwareResourceAllocationConf

Name	McuHardwareResourceAllocationConf_0																																																																		
McuGtmAllocationConf																																																																			
McuAscLinAllocationC																																																																			
McuCcu6ModuleAllocat																																																																			
McuGpt12ModuleAlloca																																																																			
McuEruAllocationConf																																																																			
>1																																																																			
<table border="1"> <thead> <tr> <th>Index</th> <th>Name</th> <th>McuAscLinChannelAllocationConf</th> </tr> </thead> <tbody> <tr><td>0</td><td>McuAscLinAllocationConf_0</td><td>ASCLIN_CH_USED_BY_UART_DRIVER</td></tr> <tr><td>1</td><td>McuAscLinAllocationConf_1</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>2</td><td>McuAscLinAllocationConf_10</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>3</td><td>McuAscLinAllocationConf_11</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>4</td><td>McuAscLinAllocationConf_12</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>5</td><td>McuAscLinAllocationConf_13</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>6</td><td>McuAscLinAllocationConf_14</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>7</td><td>McuAscLinAllocationConf_15</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>8</td><td>McuAscLinAllocationConf_16</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>9</td><td>McuAscLinAllocationConf_17</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>10</td><td>McuAscLinAllocationConf_18</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>11</td><td>McuAscLinAllocationConf_19</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>12</td><td>McuAscLinAllocationConf_2</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>13</td><td>McuAscLinAllocationConf_20</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>14</td><td>McuAscLinAllocationConf_21</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>15</td><td>McuAscLinAllocationConf_22</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>16</td><td>McuAscLinAllocationConf_23</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>17</td><td>McuAscLinAllocationConf_3</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>18</td><td>McuAscLinAllocationConf_4</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>19</td><td>McuAscLinAllocationConf_5</td><td>ASCLIN_CH_NOT_USED</td></tr> <tr><td>20</td><td>McuAscLinAllocationConf_6</td><td>ASCLIN_CH_NOT_USED</td></tr> </tbody> </table>		Index	Name	McuAscLinChannelAllocationConf	0	McuAscLinAllocationConf_0	ASCLIN_CH_USED_BY_UART_DRIVER	1	McuAscLinAllocationConf_1	ASCLIN_CH_NOT_USED	2	McuAscLinAllocationConf_10	ASCLIN_CH_NOT_USED	3	McuAscLinAllocationConf_11	ASCLIN_CH_NOT_USED	4	McuAscLinAllocationConf_12	ASCLIN_CH_NOT_USED	5	McuAscLinAllocationConf_13	ASCLIN_CH_NOT_USED	6	McuAscLinAllocationConf_14	ASCLIN_CH_NOT_USED	7	McuAscLinAllocationConf_15	ASCLIN_CH_NOT_USED	8	McuAscLinAllocationConf_16	ASCLIN_CH_NOT_USED	9	McuAscLinAllocationConf_17	ASCLIN_CH_NOT_USED	10	McuAscLinAllocationConf_18	ASCLIN_CH_NOT_USED	11	McuAscLinAllocationConf_19	ASCLIN_CH_NOT_USED	12	McuAscLinAllocationConf_2	ASCLIN_CH_NOT_USED	13	McuAscLinAllocationConf_20	ASCLIN_CH_NOT_USED	14	McuAscLinAllocationConf_21	ASCLIN_CH_NOT_USED	15	McuAscLinAllocationConf_22	ASCLIN_CH_NOT_USED	16	McuAscLinAllocationConf_23	ASCLIN_CH_NOT_USED	17	McuAscLinAllocationConf_3	ASCLIN_CH_NOT_USED	18	McuAscLinAllocationConf_4	ASCLIN_CH_NOT_USED	19	McuAscLinAllocationConf_5	ASCLIN_CH_NOT_USED	20	McuAscLinAllocationConf_6	ASCLIN_CH_NOT_USED
Index	Name	McuAscLinChannelAllocationConf																																																																	
0	McuAscLinAllocationConf_0	ASCLIN_CH_USED_BY_UART_DRIVER																																																																	
1	McuAscLinAllocationConf_1	ASCLIN_CH_NOT_USED																																																																	
2	McuAscLinAllocationConf_10	ASCLIN_CH_NOT_USED																																																																	
3	McuAscLinAllocationConf_11	ASCLIN_CH_NOT_USED																																																																	
4	McuAscLinAllocationConf_12	ASCLIN_CH_NOT_USED																																																																	
5	McuAscLinAllocationConf_13	ASCLIN_CH_NOT_USED																																																																	
6	McuAscLinAllocationConf_14	ASCLIN_CH_NOT_USED																																																																	
7	McuAscLinAllocationConf_15	ASCLIN_CH_NOT_USED																																																																	
8	McuAscLinAllocationConf_16	ASCLIN_CH_NOT_USED																																																																	
9	McuAscLinAllocationConf_17	ASCLIN_CH_NOT_USED																																																																	
10	McuAscLinAllocationConf_18	ASCLIN_CH_NOT_USED																																																																	
11	McuAscLinAllocationConf_19	ASCLIN_CH_NOT_USED																																																																	
12	McuAscLinAllocationConf_2	ASCLIN_CH_NOT_USED																																																																	
13	McuAscLinAllocationConf_20	ASCLIN_CH_NOT_USED																																																																	
14	McuAscLinAllocationConf_21	ASCLIN_CH_NOT_USED																																																																	
15	McuAscLinAllocationConf_22	ASCLIN_CH_NOT_USED																																																																	
16	McuAscLinAllocationConf_23	ASCLIN_CH_NOT_USED																																																																	
17	McuAscLinAllocationConf_3	ASCLIN_CH_NOT_USED																																																																	
18	McuAscLinAllocationConf_4	ASCLIN_CH_NOT_USED																																																																	
19	McuAscLinAllocationConf_5	ASCLIN_CH_NOT_USED																																																																	
20	McuAscLinAllocationConf_6	ASCLIN_CH_NOT_USED																																																																	

Figure 56 **UART channel allocation in MCU driver**

For the devices where ASCII INs are non-consecutive last Index ID indicates the last ASCII IN Hw resource

For example if max number of the ASCLINs are 5 i.e. ASCLIN0, ASCLIN1, ASCLIN2, ASCLIN3 and ASCLIN5 then mapping for below configuration is

Index – 0 for ASCII IN 0

Index = 1 for ASCII IN 1

Index = 2 for ASCII IN 2

Index - 3 for ASCLIN 3

Index – 4 for ASCII IN 5

If no answer is given, then

The below configuration is:

Index - 0 for ASCII IN 0

Index - 1 for ASCII LIN 1

Index - 2 for ASCII IN 2

Index 3 for ASCIIIN 3

Index 4 for ASCLIN 9

WILSON - 1997-1998

UART driver

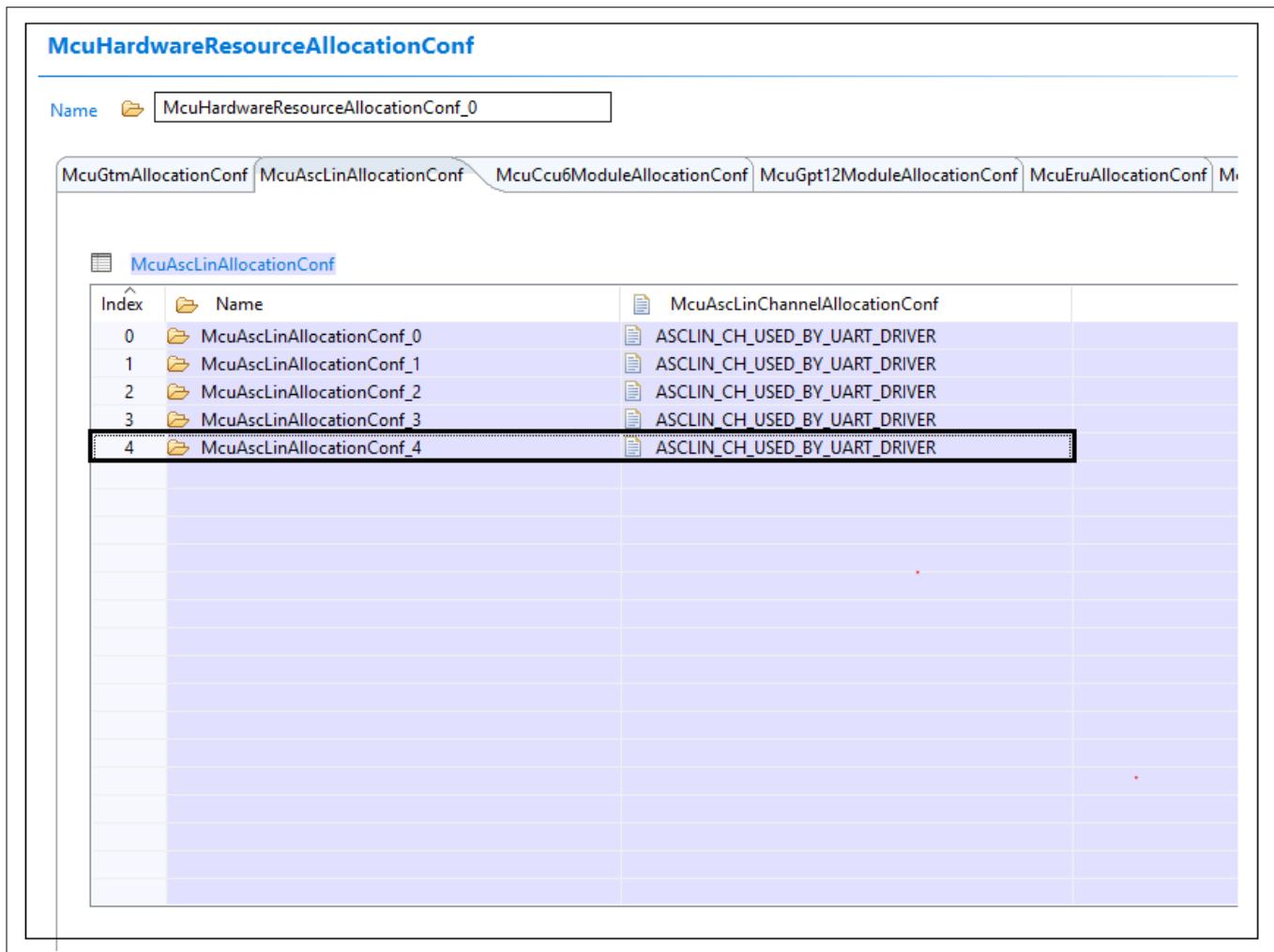


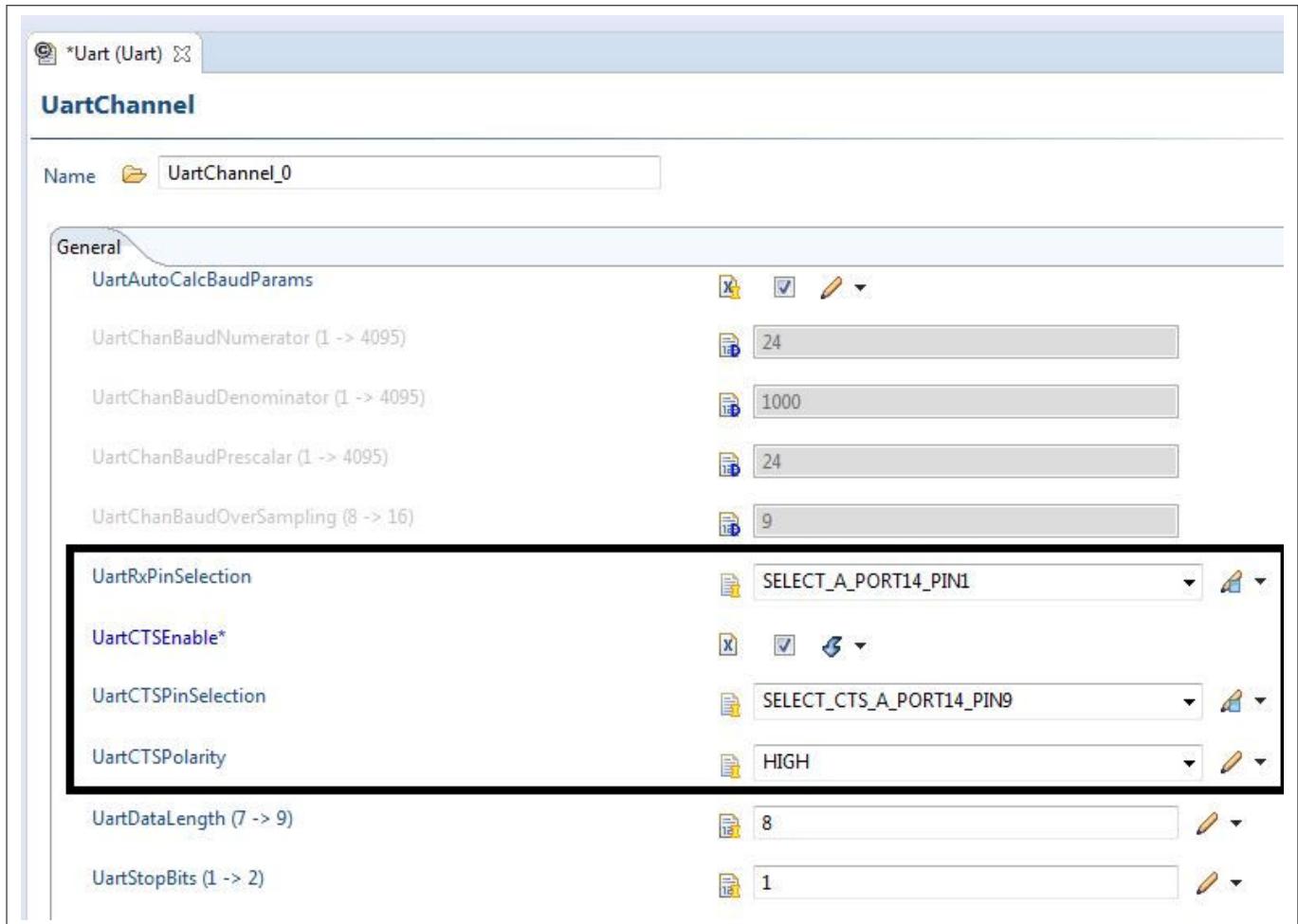
Figure 57 UART channel allocation in MCU driver for non-consecutive ASCLIN devices

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 UART driver through the PORT configuration and initialize the port pins prior to invoking of the UART initialization. The following must be considered while configuring PORT driver in the EB tresos tool:

- Configure all port pins that are used in the UART driver for RX, TX, CTS and RTS. That is, parameters such as `PortPinDirection` (input or output), `PortPinInitialMode` (as GPIO for input pin or corresponding ALT option for output pins) and so on.
- For all output pins used by the UART driver, the `PortPinControllerSelect` parameter shall be selected as ENABLE.

Refer to the following sample configurations for the PORT driver:

UART driver**Figure 58 Alternative RX and CTS pin selection for UART channel**

UART driver

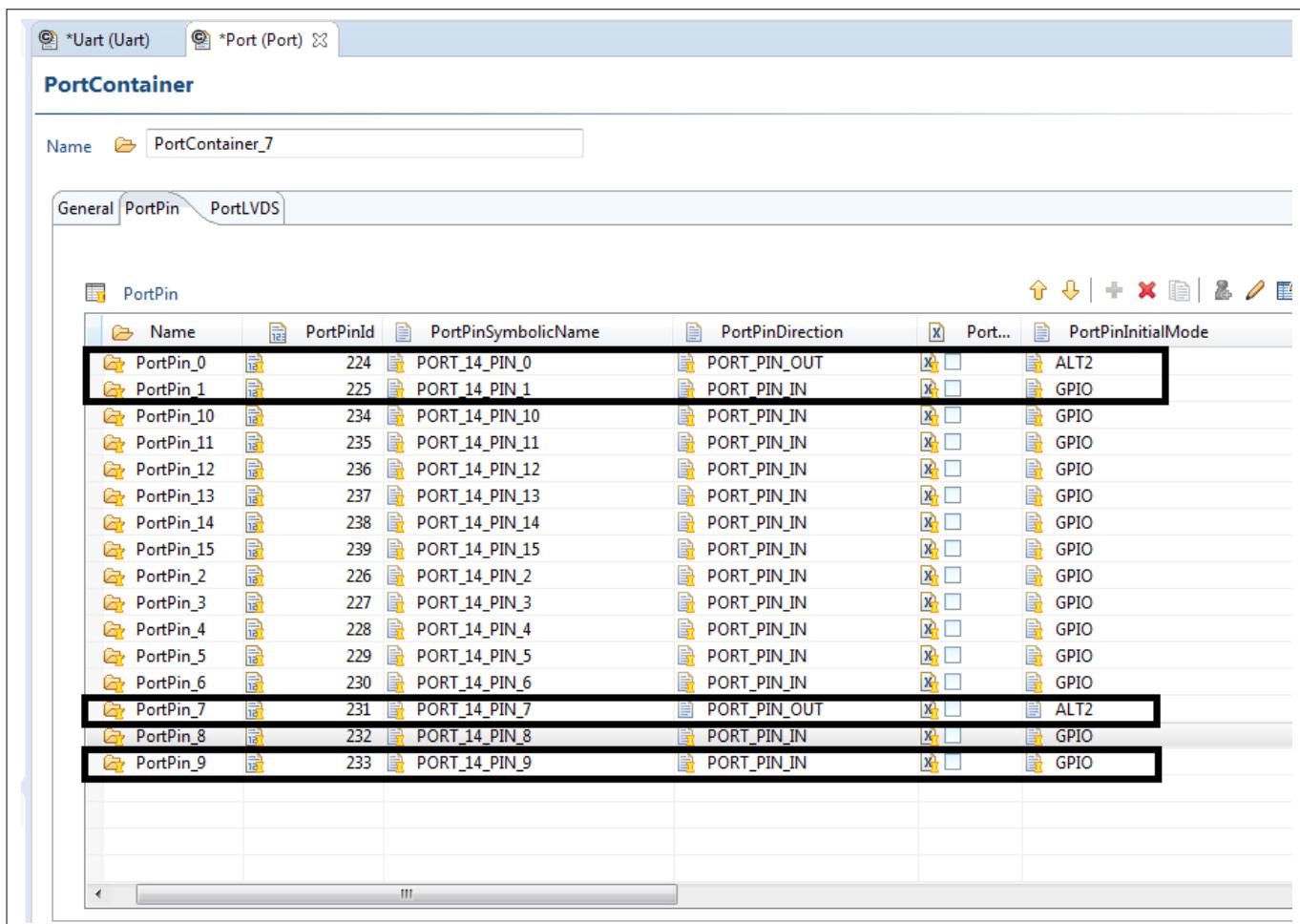


Figure 59 Port pin direction and ALT configuration for UART

5.1.4.5 DMA support

The UART driver does not use any services provided by the DMA driver.

5.1.4.6 Interrupt connections

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

- TFL: TXFIFO level interrupt

UART driver

This interrupt is triggered when TXFIFO transmission is completed successfully. A sample invocation for TFL interrupt handler is shown as follows:

```
#include "Uart.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 Uart Interrupt function*/
    Uart_IsrTransmit(0U);
}
```

- RFL: RXFIFO level interrupt

This interrupt is triggered when RXFIFO filled up to configured level with received data. A sample invocation for RFL interrupt handler is shown as follows:

```
#include "Uart.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 Uart Interrupt function*/
    Uart_IsrReceive(0U);
}
```

- ERR/Transmit complete: Receive error or transmit complete

UART driver

This interrupt is triggered when receive error (Parity, Frame, RXFIFO overflow) event or transmit complete (last stop bit transmitted out from ASCLIN kernel) event occurs. A sample invocation for error interrupt handler is shown as follows:

```
#include "Uart.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 Uart Interrupt function*/
    Uart_IsrError(0U);
}
```

Configuration of interrupt category and priority shall be configured in the IRQ driver. The following examples show interrupt configurations for ASCLINO:

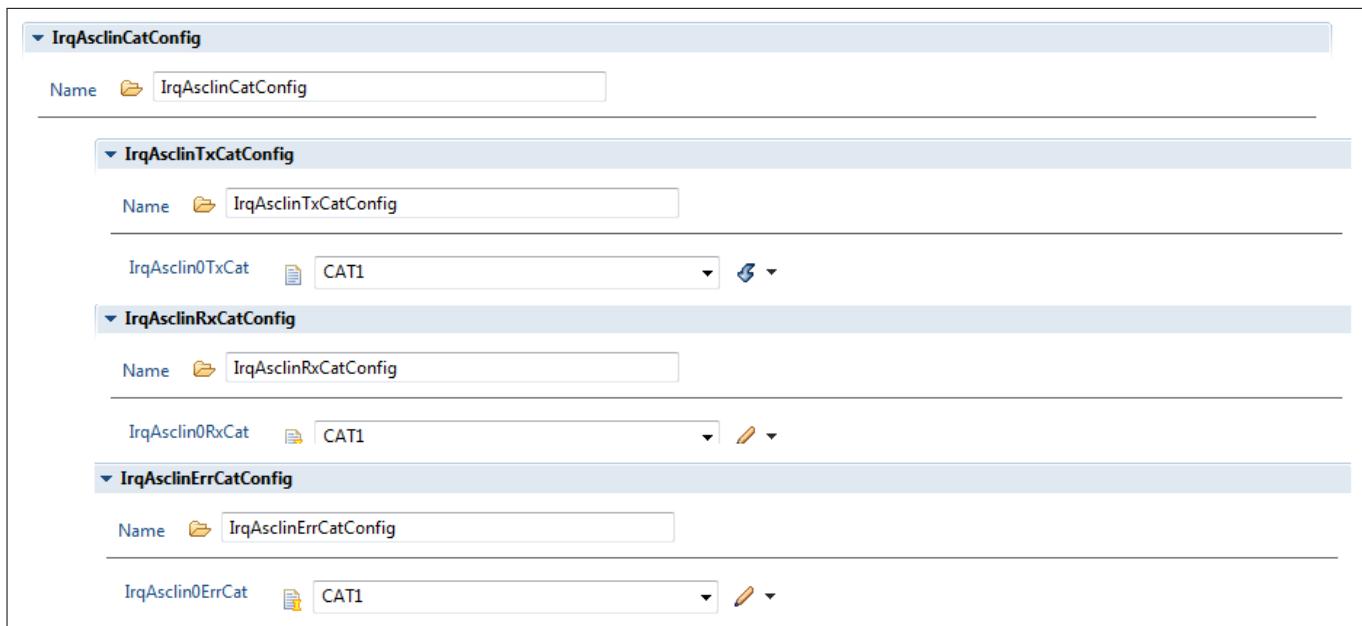
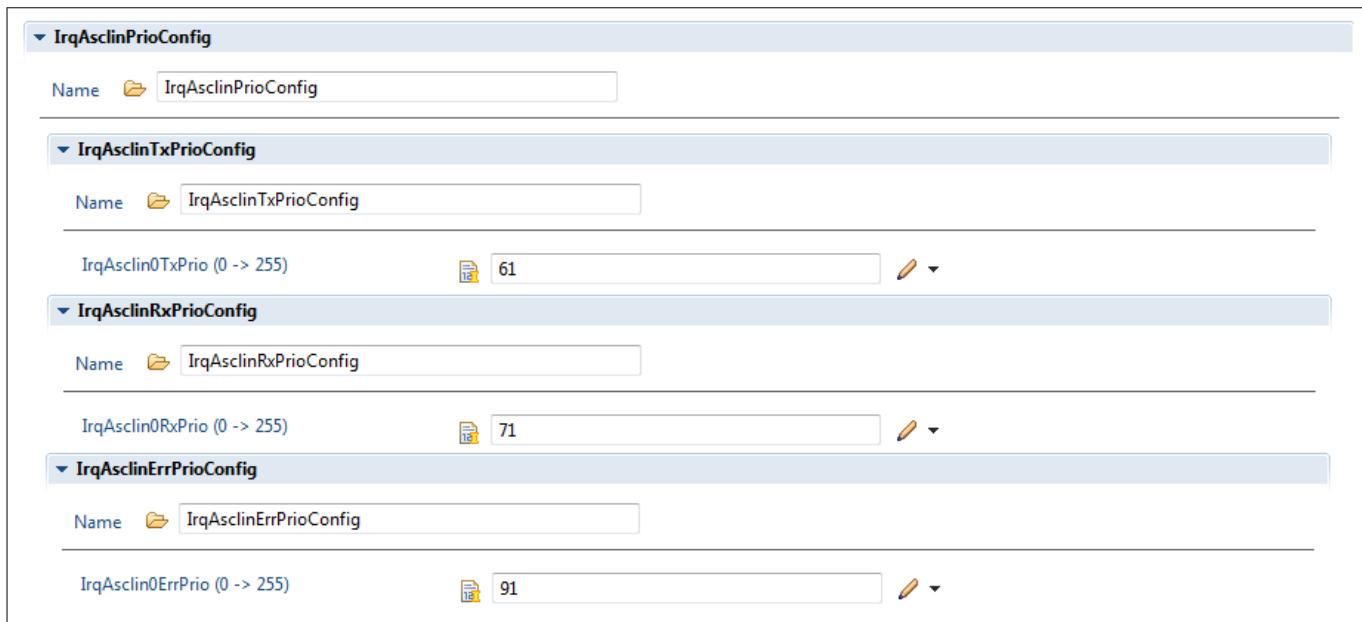


Figure 60 Interrupt category configuration for ASCLINO

UART driver**Figure 61 Interrupt priority configuration for ASCLIN0**

UART driver**5.1.4.7 Example usage**

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

Configuration of UART and other modules

- Configuration of system clock: Before using the UART 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, RXD, optional CTS, RTS (for the relevant RX and CTS pin select) pins of the UART channels should be configured using the PORT driver.
- Configuration of the UART interrupts: For UART drivers with interrupt mode enabled, configure the interrupt priority, type of service and interrupt type in the IRQ driver.
- Configuration of the UART driver: Select the required API configuration and choose channel dependent parameters like Baud-rate, Tx and Rx mode (polling or interrupt) Stop bits, Parity, Optional CTS selection etc.

UART driver initialization

Refer to the Integration hints section and add all the dependent modules. Follow the sequence in the application code:

1. Initialize the MCU driver and the clock using the `Mcu_Init` API.
2. Initialize the PORT driver using the `Port_Init` API.
3. Initialize the IRQ to enable the interrupt generation.
4. Initialize the UART driver using the `Uart_Init` API.

The sample code for the UART driver initialization is shown as follows:

UART driver

```
#include "Mcu.h"
#include "Uart.h"
#include "Port.h"
#include "Irq.h"

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

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

/* Irq initialization */
IrqAsclin_Init();

/* Uart driver initialization */
Uart_Init(&Uart_Config);

/* Check Uart initialization */
RetVal = Uart_InitCheck(&Uart_Config);
if(RetVal == E_NOT_OK)
{
    /* Uart initialization fail */
}

/* Uart driver de-initialization */
Uart_DeInit();
```

UART transmit operation in interrupt mode

The sequence diagram for the UART transmit operation in the interrupt mode is shown as follows:

UART driver

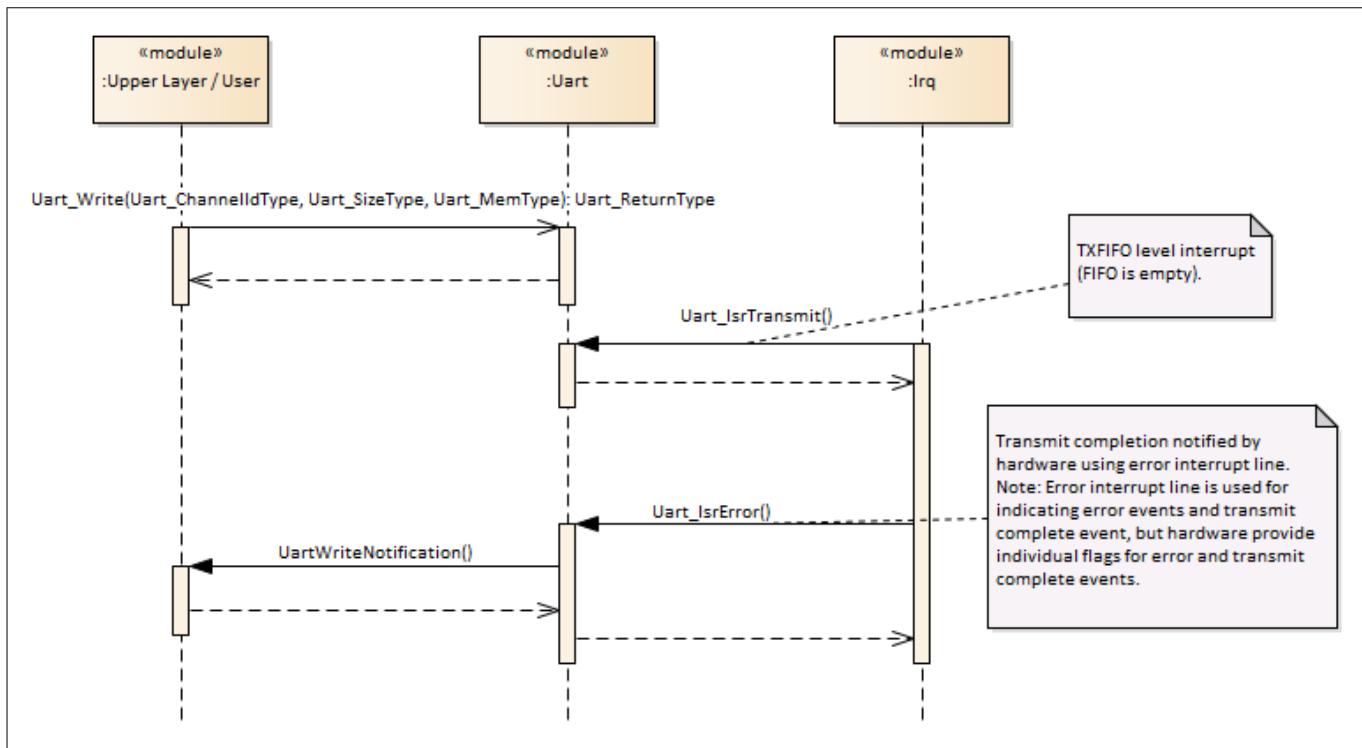
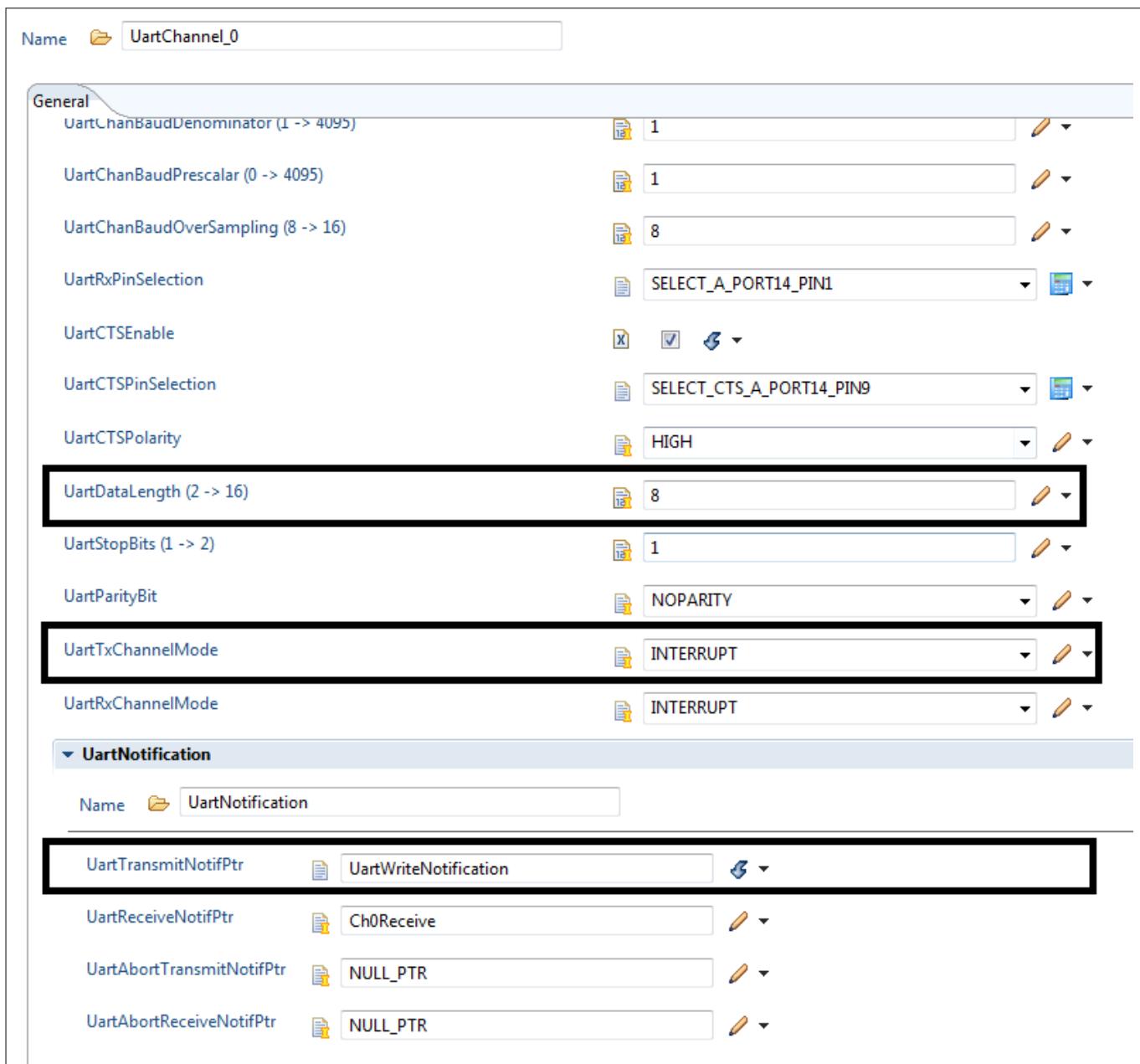


Figure 62 **UART transmit operation in interrupt mode**

The sample configuration for the UART transmit in the interrupt mode with 8-bit frame length is shown as follows:

UART driver**Figure 63**

Configuration: Frame length 8 bits, transmit in interrupt mode, transmit Notification function `UartWriteNotification`

UART driver

A sample code for transmitting 20 frames with 8-bit frame length in the interrupt mode is as follows:

```

/* Transmit buffer */
uint8 TxBuffer[20] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};

/* Write notification function */
void UartWriteNotification(Uart_ErrorIdType ErrorId)
{
    if(ErrorId == UART_NO_ERR)
    {
        /* 20 frames transmitted successfully */
    }
}

/* Uart write */
Uart_Write(0,&TxBuffer[0],20);

```

UART transmit operation in polling mode

The sequence diagram for the UART transmit operation in the polling mode is shown as follows:

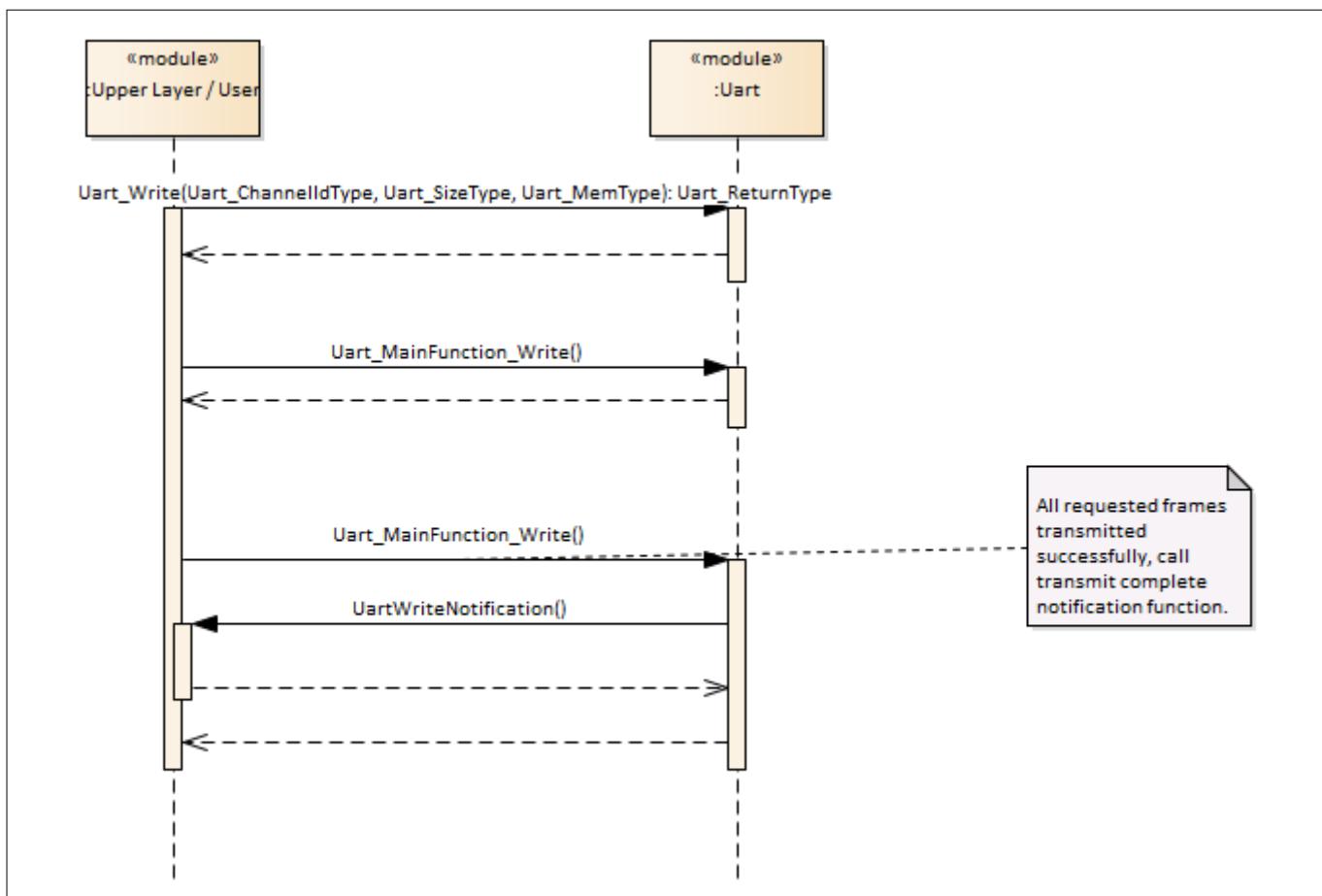
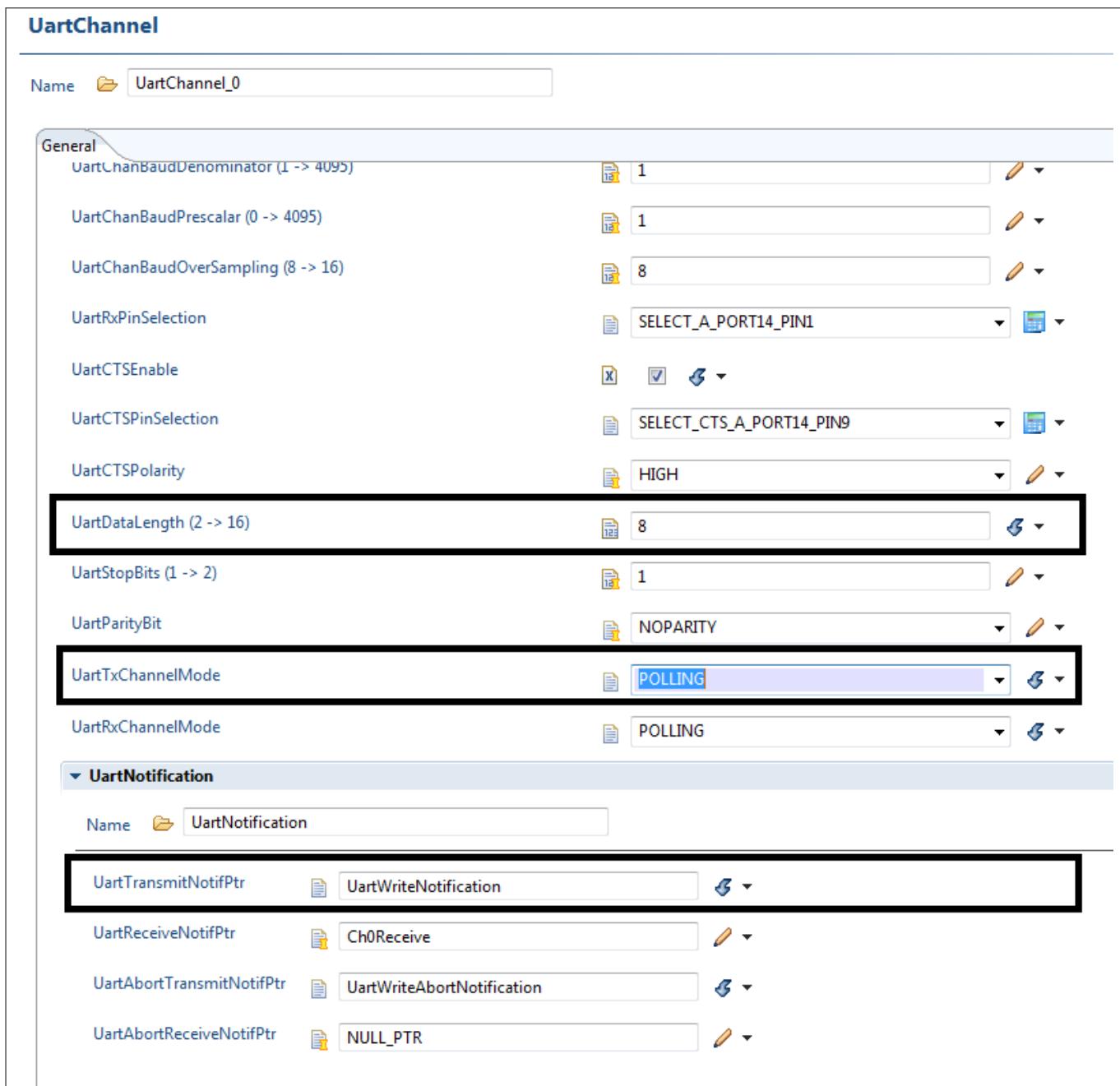


Figure 64 UART transmit operation in polling mode

The sample configuration for transmitting 8-bit frame in the polling mode is shown as follows:

UART driver**Figure 65**

Configuration: Frame length 8 bits, transmit in polling mode, transmit Notification function UartWriteNotification

UART driver

A sample code for transmitting 20 frames with 8-bit frame size in the polling mode is as follows:

```
/* Buffer use for transmission */
uint8 TxBuffer[20] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};

/* Write notification function */
void UartWriteNotification(Uart_ErrorIdType ErrorId)
{
    if(ErrorId == UART_NO_ERR)
    {
        /* Write operation completed without error */
    }
}

/* Uart write */
Uart_Write(0,&TxBuffer[0],20);
/* Poll till transmission is completed */
while(RetVal == UART_BUSY_TRANSMIT)
{
    /* Function to poll data transmission and give notification once transmission
    is finished */
    Uart_MainFunction_Write();
    /* Get channel 0 status */
    RetVal = Uart_GetStatus(0);
}
```

UART transmit abort operation

The sequence diagram of UART transmit abort operation is shown as follows:

UART driver

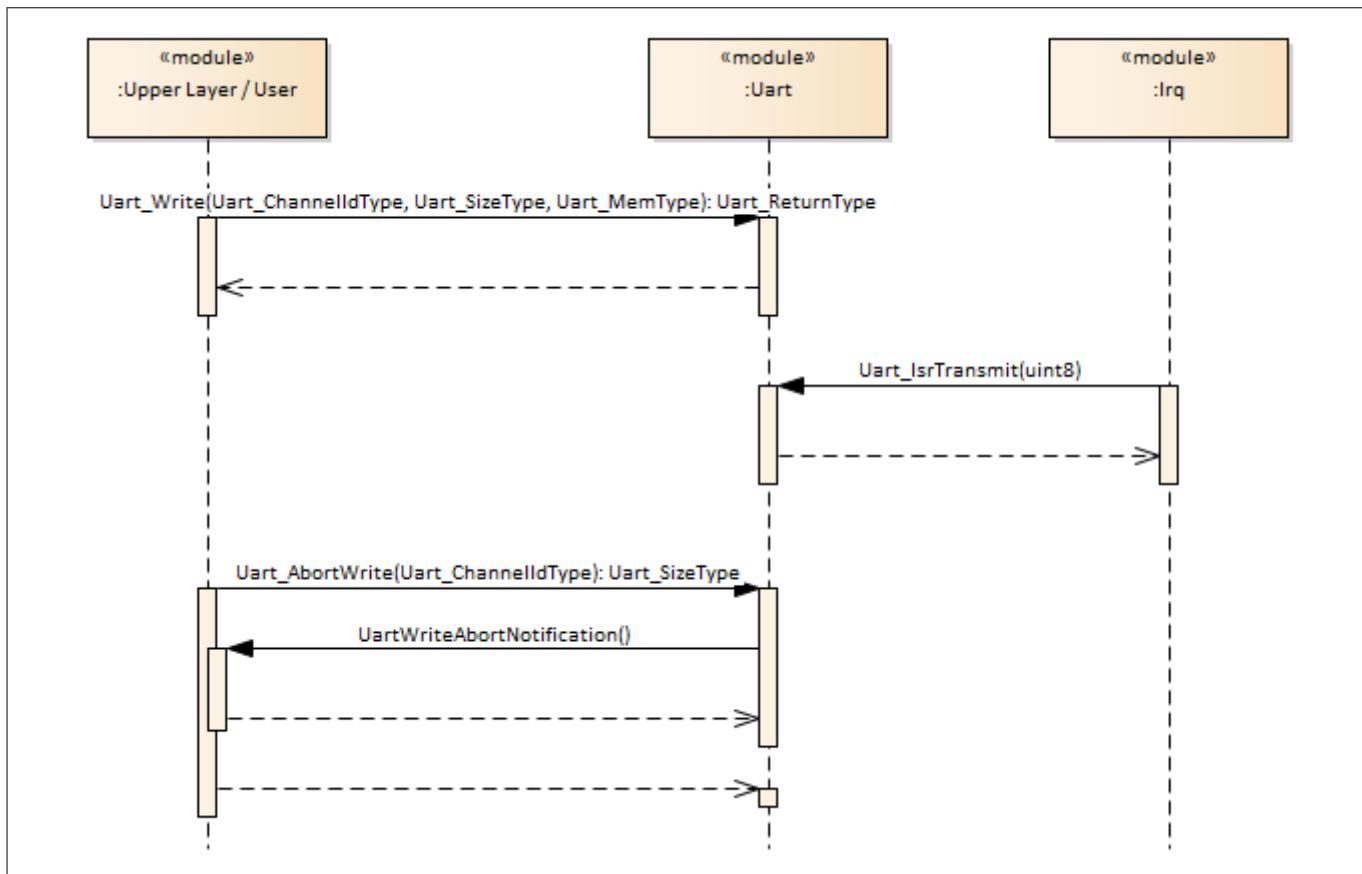


Figure 66 Transmit abort operation

UART driver

A sample code for the abort transmit operation is as follows:

```

/* Buffer use for transmission */
uint8 TxBuffer[128];
uint16 NumberOfBytesTransmited;

/* Write notification function */
void UartWriteAbortNotification(Uart_ErrorIdType ErrorId)
{
    if(ErrorId == UART_NO_ERR)
    {
        /* transmit operation aborted successfully */
    }
}

/* Initialize TxBuffer */
for(Counter = 0; Counter < 128; Counter++)
{
    TxBuffer[Counter] = Counter;
}

/* Uart write */
Uart_Write(0,&TxBuffer[0],128);

/* Abort write operation on channel 0 */
NumberOfBytesTransmited = Uart_AbortWrite(0);

```

UART receive operation in interrupt mode

The sequence diagram for the UART receive operation in the interrupt mode is shown as follows:

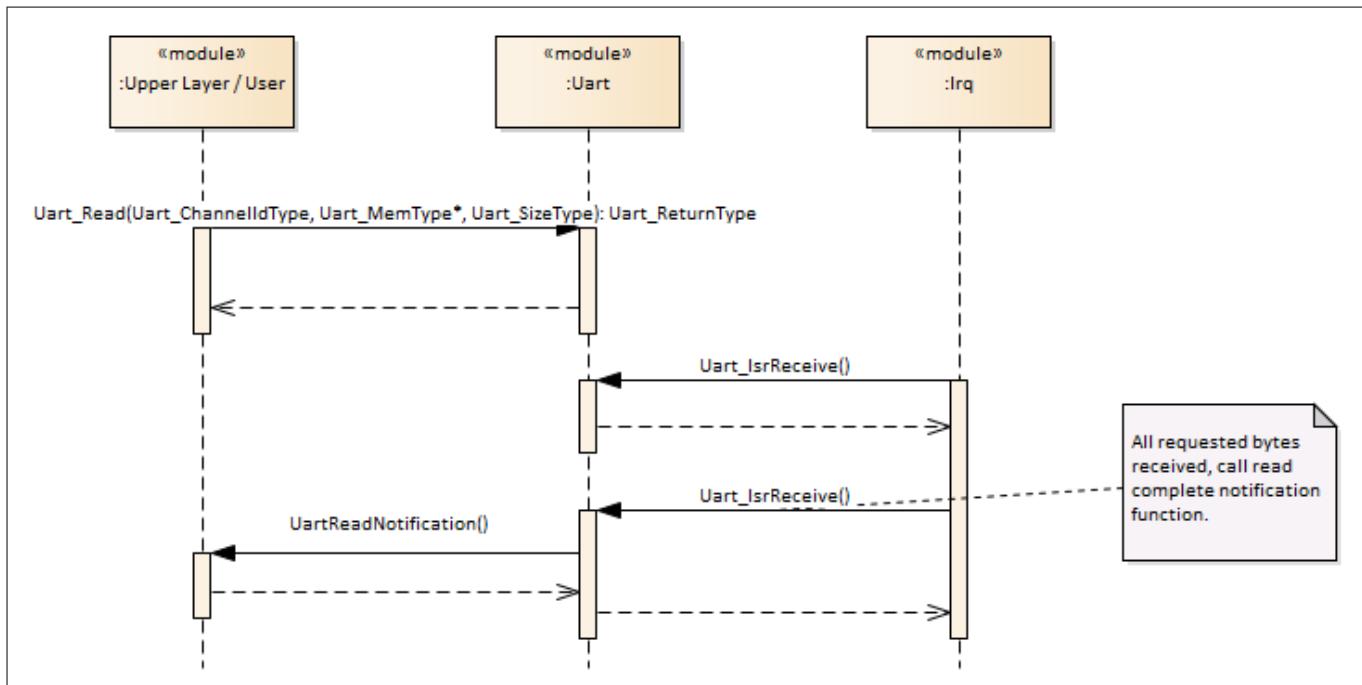


Figure 67 UART receive operation in interrupt mode

UART driver

The sample configuration for receive 8-bit frame in the interrupt mode is as follows:

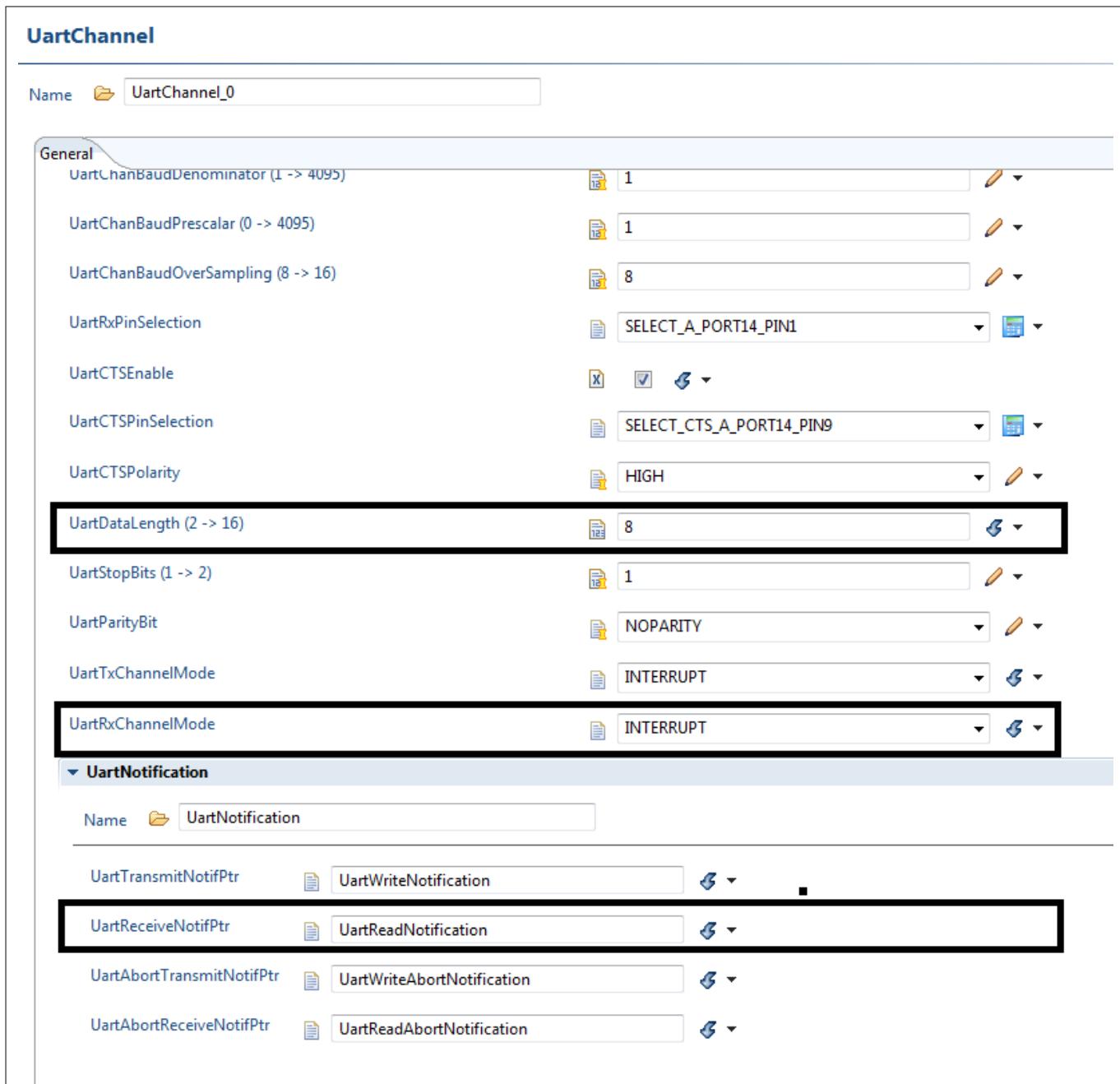


Figure 68

Configuration: Frame length 8 bits, receive in interrupt mode, receive notification function UartReadNotification

UART driver

A sample code for receiving 20 frames with 8-bit frame size in the interrupt mode is as follows:

```

/* Receive buffer */
uint8 RxBuffer[20];

/* Read notification function */
void UartReadNotification(Uart_ErrorIdType ErrorId)
{
    if(ErrorId == UART_NO_ERR)
    {
        /* 20 frames received without error start process received data */
    }
}

/* Uart read */
Uart_Read(0,&RxBuffer[0],20);

```

UART receive operation in polling mode

The sequence diagram for the UART receive operation in the polling mode is shown as follows:

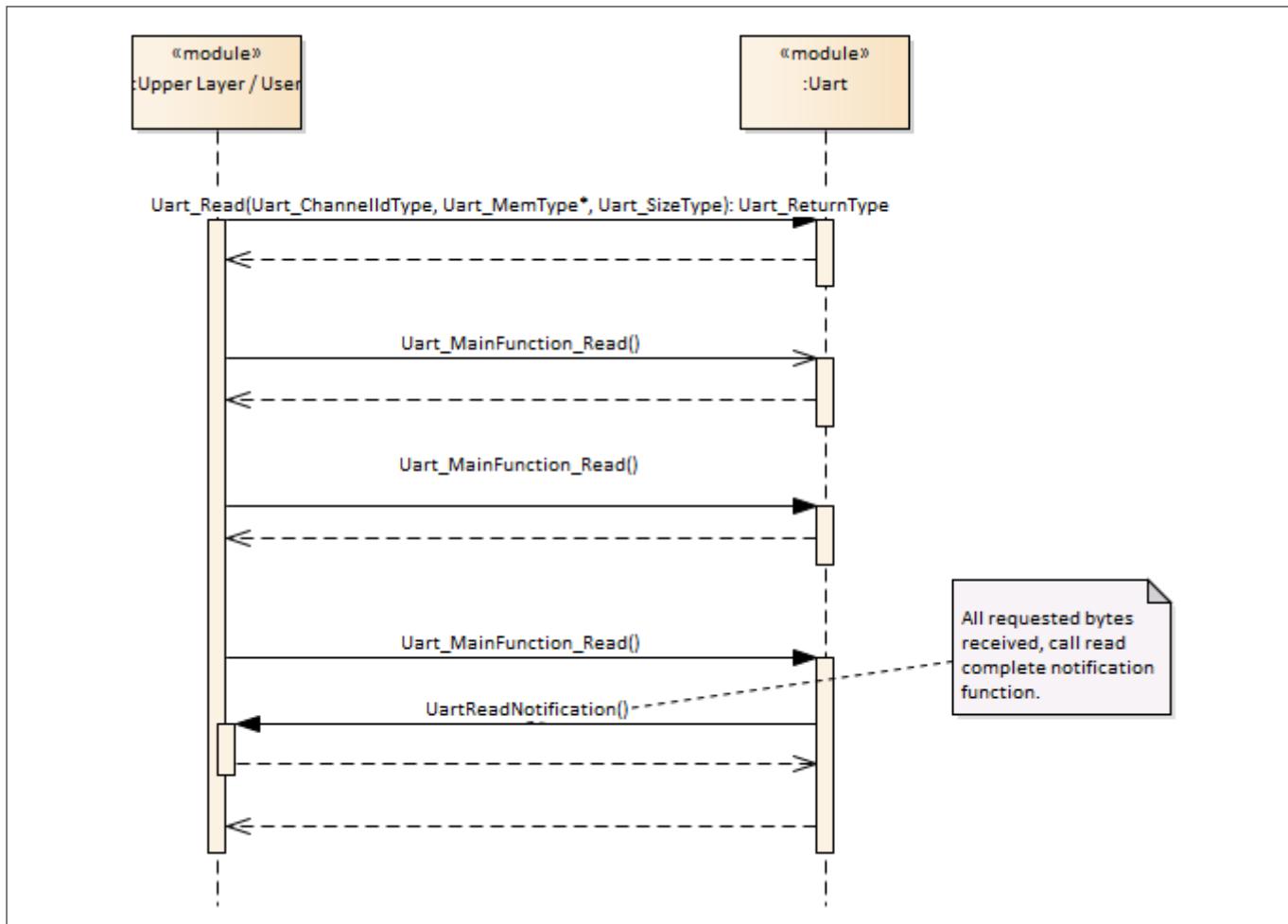


Figure 69

UART receive operation in polling mode

UART driver

A sample code for receiving 20 frames with 8-bit frame size in the polling mode is as follows:

```
/* Receive buffer */
uint8 RxBuffer[20];

/* Read notification function */
void UartReadNotification(Uart_ErrorIdType ErrorId)
{
    if(ErrorId == UART_NO_ERR)
    {
        /* 20 frames received without error start process received data */
    }
}

/* Uart read */
Uart_Read(0,&RxBuffer[0],20);
/* Poll till transmission is completed */
while(RetVal == UART_BUSY_RECEIVE)
{
    /* Function to poll data receive and give notification once receivce
    operation is finished */
    Uart_MainFunction_Read();
    /* Get channel 0 status */
    RetVal = Uart_GetStatus(0);
}
```

UART abort read operation

The sequence diagram for the UART receive abort operation in the interrupt mode is shown as follows:

UART driver

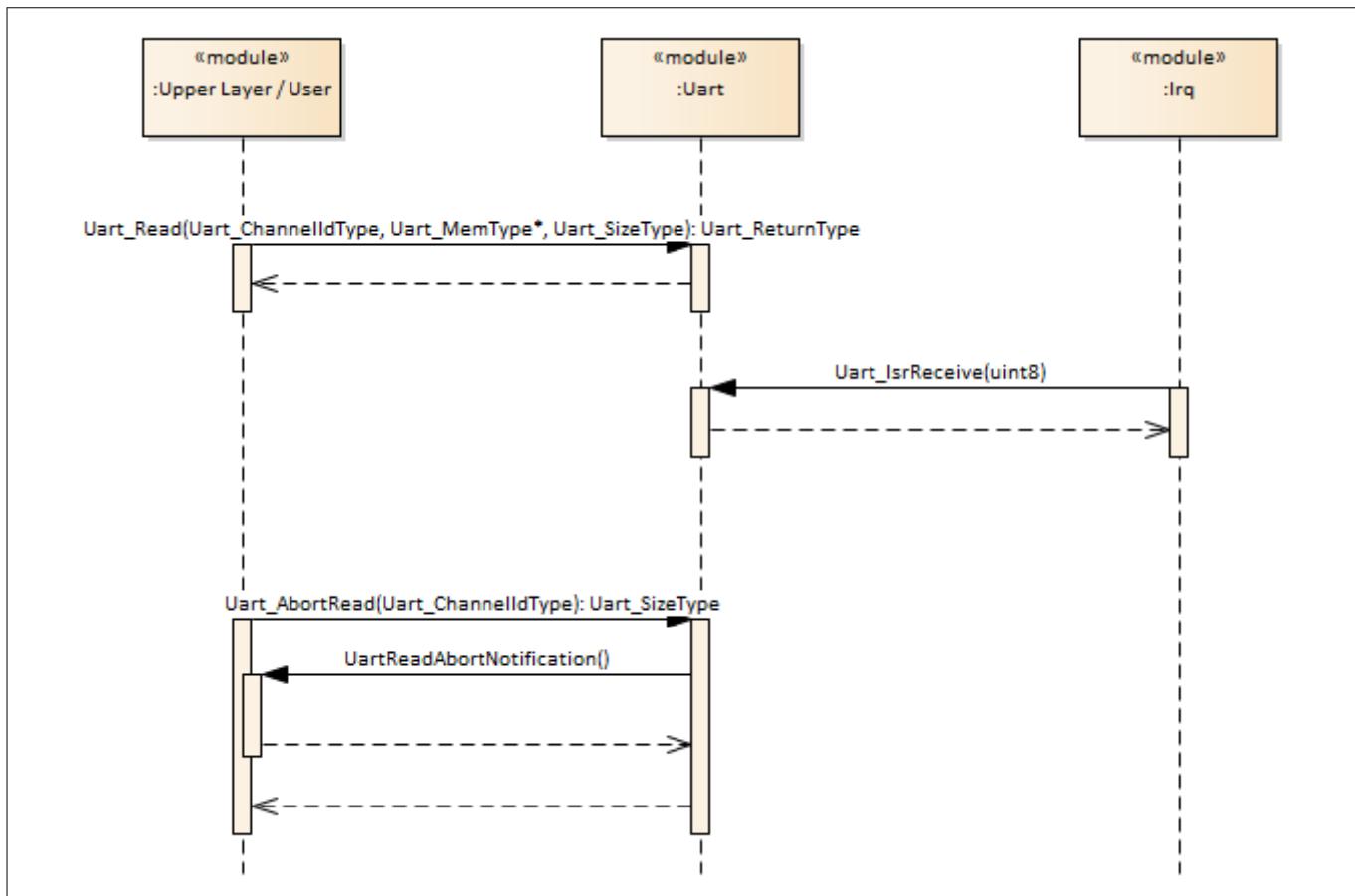


Figure 70 UART abort receive operation

A sample code for the abort receive operation is as follows:

```

/* Receive buffer */
uint8 RxBuffer[128];
uint16 NumberOfBytesReceived;

/* Read abort notification function */
void UartReadAbortNotification(Uart_ErrorIdType ErrorId)
{
    if(ErrorId == UART_NO_ERR)
    {
        /* Read operation aborted successfully */
    }
}

/* Uart read */
Uart_Read(0,&RxBuffer[0],128);

/* Abort read operation on channel 0 */
NumberOfBytesReceived = Uart_AbortRead(0);

```

5.1.5 Key architectural considerations

UART driver

5.2 Assumptions of Use (AoUs)

The AoUs for the driver are as follows:

- **Configuration check**

Integrator shall check that configuration code generated is correct for all the configured UART channels.

[cover parentID UART={B3D0ECC3-553D-4743-AC7A-8C6A81DEF4C0}]

- **Address check**

Integrator shall pass valid buffer pointer to transmit/receive data.

[cover parentID UART={E601C70E-216F-42eb-A2E4-DCDC329C91F1}]

- **Freedom from interference for MCAL data**

Integrator shall ensure that there is no interference to the MCAL from other modules. Rationale: Variables/SFRs can be corrupted by the QM software.

[cover parentID UART={ADFDA904-FOCE-431e-AC65-E1D42A402D37}]

- **Frequency check**

Baudrate parameters of ASCLIN are calculated using configured frequency. Therefore, user shall ensure that the UART driver is invoked only when the operating frequency of ASCLIN is same as the configured frequency. In case of a mismatch, the ASCLIN operates with a different baudrate than the configured value (UartBaudRate).

[cover parentID UART={4C54A9DC-9200-4126-B41F-C8E733371CEF}]

- **Receiver check**

ASCLIN cannot detect errors when data is being shifted from the shift register to the UART pins. Therefore the receiver device shall ensure to have an error detection mechanisms in place.

[cover parentID UART={85AB9E29-DE6F-49fd-B058-BE1A307BC8E5}]

- **Transmission complete notification**

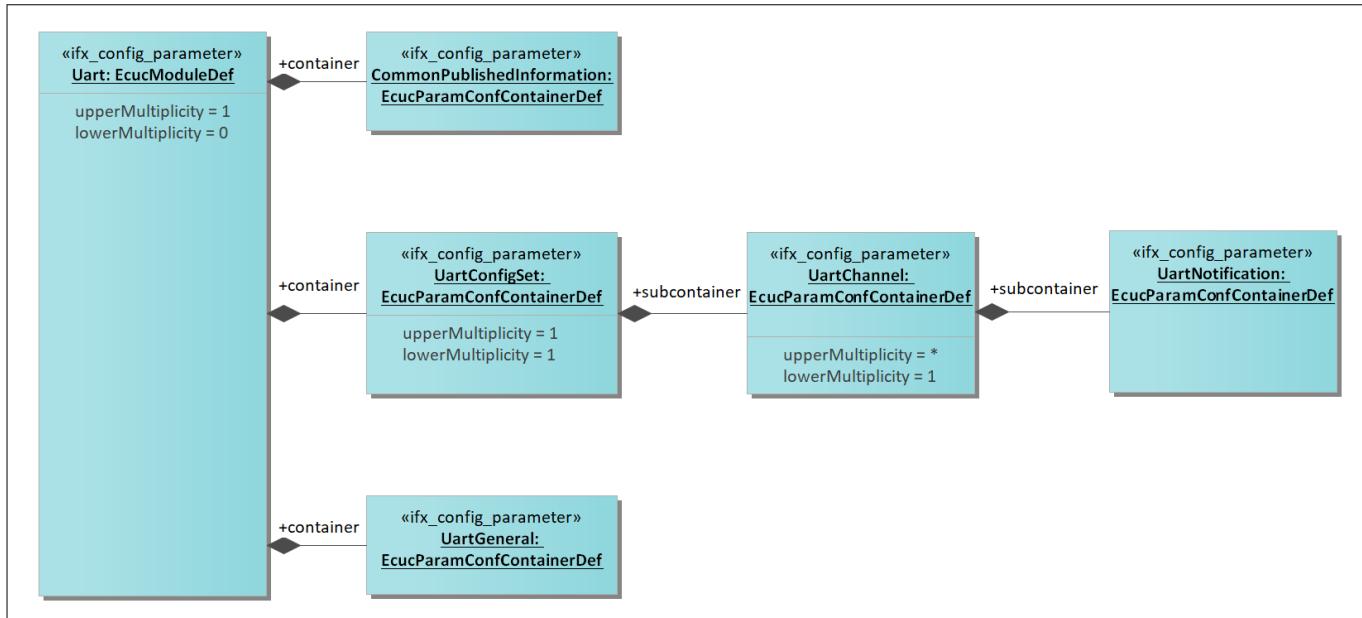
Hardware triggers the interrupt when the last frame is shifted from TXFIFO to the shift register. Hence last frame transmission is not completed when the interrupt is triggered which will call transmit complete notification. The next frame transmission can be initiated by the user using the Uart_Write API, which will fill the TXFIFO without disturbing the current frame transmission. However if the peripheral has to be de-initialized the user shall wait for 1 frame duration else the last frame transmission may be stopped.

[cover parentID UART={51BE3A04-5FA3-420d-AC83-D5378ABB7003}]

- **UART driver usage mode**

It is assumed that the user of the UART driver is aware of the number of bytes to be received from the external device as the Uart_Read() API has Size as a parameter. Also it is assumed that the user knows the instance of time when the data is expected to be received from external device and accordingly the Uart_Read() API is invoked. That is driver shall be operated in master configuration and not as a slave or peer device.

[cover parentID UART={D6E91D13-C314-435a-AAB5-716892AA30EF}]

UART driver**5.3 Reference information****5.3.1 Configuration interfaces****Figure 71 Container hierarchy along with their configuration parameters****5.3.1.1 Container: CommonPublishedInformation**

Publish information about module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

5.3.1.1.1 ArMajorVersion**Table 415 Specification for ArMajorVersion**

Name	ArMajorVersion		
Description	Major version number of 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	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

UART driver**5.3.1.1.2 ArMinorVersion****Table 416 Specification for ArMinorVersion**

Name	ArMinorVersion		
Description	Minor version of 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	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.1.3 ArPatchVersion**Table 417 Specification for ArPatchVersion**

Name	ArPatchVersion		
Description	Patch level version number of 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	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.1.4 ModuleId**Table 418 Specification for ModuleId**

Name	ModuleId		
Description	Module identifier of UART driver from module list.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 65535		

UART driver**Table 418 Specification for ModuleId (continued)**

Default value	255		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.1.5 Release**Table 419 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 UART driver.		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.1.6 SWMajorVersion**Table 420 Specification for SWMajorVersion**

Name	SWMajorVersion		
Description	Major version number of the implementation of the module.		
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	Pre-Compile	Multiplicity configuration class	-

UART driver**Table 420 Specification for SWMajorVersion (continued)**

Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.1.7 SWMinorVersion**Table 421 Specification for SWMinorVersion**

Name	SWMinorVersion		
Description	Minor version number of implementation of the module.		
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	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.1.8 SWPatchVersion**Table 422 Specification for SWPatchVersion**

Name	SWPatchVersion		
Description	Patch level version number of implementation of the module.		
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	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

UART driver**5.3.1.1.9 VendorId****Table 423 Specification for VendorId**

Name	VendorId		
Description	Vendor identifier of dedicated implementation of UART driver 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	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.2 Container: UartChannel

This container contains the configuration parameters of UART channel. Maximum number of UART channels varies as per device variant.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

5.3.1.2.1 UartAutoCalcBaudParams**Table 424 Specification for UartAutoCalcBaudParams**

Name	UartAutoCalcBaudParams		
Description	Enable or disable automatic calculation of baud rate parameters (Numerator, Denominator, Pre-scalar and Over sampling) based on the configuration of parameter UartBaudRate. User can disable the feature and manually enter the values for baud rate parameters. TRUE: Automatic calculation of baudrate parameters are enabled. FALSE: Automatic calculation of baudrate parameters are disabled.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

UART driver**Table 424 Specification for UartAutoCalcBaudParams (continued)**

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

5.3.1.2.2 UartBaudRate**Table 425 Specification for UartBaudRate**

Name	UartBaudRate		
Description	UART channel transmit and receive baud rate in bits per second. Parameter is applicable if UartAutoCalcBaudParams is enabled. Note: Default value set to 9600 bits per second as UART standard baud rate.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1000 - 6250000		
Default value	9600		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	UartAutoCalcBaudParams		

5.3.1.2.3 UartCTSEnable**Table 426 Specification for UartCTSEnable**

Name	UartCTSEnable		
Description	Enable/disable CTS for UART channel. CTS (clear to transmit) used to notify sender that receiver is ready to receive data. TRUE: CTS is enabled. FALSE: CTS is disabled. Note: Default CTS is disabled to save hardware resource (port pin) for basic communication.		
Multiplicity	1..1	Type	EcuBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

UART driver**Table 426 Specification for UartCTSEnable (continued)**

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

5.3.1.2.4 UartCTSPinSelection**Table 427 Specification for UartCTSPinSelection**

Name	UartCTSPinSelection		
Description	Parameter decides the selection of the CTS pin. Note: Default value of parameter set with CTS pin of ASCLIN0.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SELECT_CTS_A_PORTY_PINZ: UART channel CTS select line A. Where 'Y' is port number and 'Z' is pin number. Values of Y, Z will be extracted from property file. SELECT_CTS_X_PORTY_PINZ: Select alternate pin selection of CTS. X is varies from A to D as per device variant. 'Y' is port number and 'Z' is pin number. Values of X, Y, Z will be extracted from property file.		
Default value	SELECT_CTS_A_PORTY_PINZ		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	UartCTSEnable		

5.3.1.2.5 UartCTSPolarity**Table 428 Specification for UartCTSPolarity**

Name	UartCTSPolarity		
Description	Parameter decides active polarity of CTS pin. Parameter applicable if UartCTSEnable is enabled. Note: Default polarity set with HIGH.		
Multiplicity	1..1	Type	EcucEnumerationParamDef

UART driver**Table 428 Specification for UartCTSPolarity (continued)**

Range	HIGH: CTS is considered to be active when the signal is HIGH. LOW: CTS is considered to be active when the signal is LOW.		
Default value	HIGH		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	UartCTSEnable		

5.3.1.2.6 UartChanBaudDenominator**Table 429 Specification for UartChanBaudDenominator**

Name	UartChanBaudDenominator		
Description	<p>Specifies the BRG register denominator value used for Baudrate calculation. The value configured in this parameter will be written to the BRG.DENOMINATOR register field.</p> <p>Baud rate is derived based on the below formula.</p> $fPD = fA / (\text{BITCON.PRESCALER} + 1)$ $fOVS = fPD * \text{BRG.NUMERATOR} / \text{BRG.DENOMINATOR}$ $fSHIFT = fOVS / (\text{BITCON.OVERSAMPLING} + 1)$ <p>fASCLINF or fASCLINS is used as input clock frequency (fA).</p> <p>Note: If UartAutoCalcBaudParams is enabled then value of this parameter calculated internally. Default value set 1000 to achieve baud rates 9600 bits per second (20 MHz input frequency).</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 4095		
Default value	1000		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	UartAutoCalcBaudParams		

UART driver**5.3.1.2.7 UartChanBaudNumerator****Table 430 Specification for UartChanBaudNumerator**

Name	UartChanBaudNumerator		
Description	<p>Specifies the BRG register numerator value used for Baudrate calculation. The value configured in this parameter will be written to the BRG.NUMERATOR register field.</p> <p>Baud rate is derived based on the below formula.</p> $f_{PD} = f_A / (\text{BITCON.PRESCALER} + 1)$ $f_{OVS} = f_{PD} * \text{BRG.NUMERATOR} / \text{BRG.DENOMINATOR}$ $f_{SHIFT} = f_{OVS} / (\text{BITCON.OVERSAMPLING} + 1)$ <p>fASCLINF or fASCLINS is used as input clock frequency (fA).</p> <p>Note: If UartAutoCalcBaudParams is enabled then value of this parameter calculated internally. Default value set 24 to achieve baud rates 9600 bits per second (20 MHz input frequency).</p>		
Multiplicity	1..1	Type	EcuclIntegerParamDef
Range	1 - 4095		
Default value	24		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	UartAutoCalcBaudParams		

5.3.1.2.8 UartChanBaudOverSampling**Table 431 Specification for UartChanBaudOverSampling**

Name	UartChanBaudOverSampling		
Description	<p>Specifies the BITCON register over sampling value used for Baudrate calculation. The value configured in this parameter will be written to the BITCON.OVERSAMPLING register field.</p> <p>Baud rate is derived based on the below formula.</p> $f_{PD} = f_A / (\text{BITCON.PRESCALER} + 1)$ $f_{OVS} = f_{PD} * \text{BRG.NUMERATOR} / \text{BRG.DENOMINATOR}$ $f_{SHIFT} = f_{OVS} / (\text{BITCON.OVERSAMPLING} + 1).$ <p>fASCLINF or fASCLINS is used as input clock frequency (fA).</p> <p>Note: If UartAutoCalcBaudParams enabled then value of parameter calculated internally. Default value set 9 to achieve baud rates 9600 bits per second (20 MHz input frequency).</p>		
Multiplicity	1..1	Type	EcuclIntegerParamDef
Range	3 - 15		

UART driver**Table 431 Specification for UartChanBaudOverSampling (continued)**

Default value	9		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	UartAutoCalcBaudParams		

5.3.1.2.9 UartChanBaudPrescalar**Table 432 Specification for UartChanBaudPrescalar**

Name	UartChanBaudPrescalar		
Description	<p>Specifies the BITCON register prescalar value used for Baudrate calculation. The value configured in this parameter will be written to the BITCON.PRESCALAR register field.</p> <p>Baud rate is derived based on the below formula.</p> $fPD = fA / (\text{BITCON.PRESCALER} + 1)$ $fOVS = fPD * \text{BRG.NUMERATOR} / \text{BRG.DENOMINATOR}$ $fSHIFT = fOVS / (\text{BITCON.OVERSAMPLING} + 1).$ <p>fASCLINF or fASCLINS is used as input clock frequency (fA).</p> <p>Note: If UartAutoCalcBaudParams is enabled then value of this parameter calculated internally. Default value set 4 to achieve baud rates 9600 bits per second (20 MHz input frequency).</p>		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - 4095		
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	UartAutoCalcBaudParams		

5.3.1.2.10 UartChannelId**Table 433 Specification for UartChannelId**

Name	UartChannelId
-------------	---------------

UART driver**Table 433 Specification for UartChannelId (continued)**

Description	UART channel logical identifier. Upper limit of the channel identifier varies as per device variant. Note: Minimum value of the parameter set as default value.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	0 - *		
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	-		

5.3.1.2.11 UartDataLength**Table 434 Specification for UartDataLength**

Name	UartDataLength		
Description	Parameter decides the frame length of UART channel. Note: Default frame size set as 8 because commonly used.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	2 - 16		
Default value	8		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.2.12 UartHwUnit**Table 435 Specification for UartHwUnit**

Name	UartHwUnit		
Description	Parameter specify ASCLIN hardware channel configured for logical channel. Maximum number of ASCLIN channel depends on device variant. Note: Default value is set with parameter minimum value.		

UART driver**Table 435 Specification for UartHwUnit (continued)**

Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	ASCLIN0: Hardware channel ASCLIN0. ASCLINx: Hardware channel varies as per device variant from ASCLIN1 to ASCLINx where x is maximum number of ASCLIN channel supported by the device.		
Default value	ASCLIN0		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.2.13 UartParityBit**Table 436 Specification for UartParityBit**

Name	UartParityBit		
Description	Parameter decides type of parity in data frame. Note: Default type set with no parity to reduce frame size.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	EVEN_PARITY: Parity bit set with 1 when even number of 1's in data frame. NO_PARITY: Parity bit not present in data frame. ODD_PARITY: Parity bit set with 1 when odd number of 1's present in data frame.		
Default value	NO_PARITY		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.2.14 UartRxChannelMode**Table 437 Specification for UartRxChannelMode**

Name	UartRxChannelMode
-------------	-------------------

UART driver**Table 437 Specification for UartRxChannelMode (continued)**

Description	UART channel receive operation configuration mode. Note: Default set in interrupt mode to disable optional interface (schedule function will enable in case any channel configured in polling mode).		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	INTERRUPT: UART channel receive operation in interrupt mode. POLLING: UART channel receive operation in polling mode.		
Default value	INTERRUPT		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.2.15 UartRxPinSelection**Table 438 Specification for UartRxPinSelection**

Name	UartRxPinSelection		
Description	Parameter decides the selection of the receiver pin. Note: The first available data line for configured ASCLIN HW unit is selected as default value.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	SELECT_A_PORTY_PINZ: UART data receive select line A. Where 'Y' is port value available as per ASCLIN channel, 'Z' is pin number as per ASCLIN channel. Values of Y, Z will be extracted from property file. SELECT_X_PORTY_PINZ: Select receive pin where value of 'X' varies from B to G(B,C,D,E,F,G), 'Y' is port value available as per ASCLIN channel, 'Z' is pin number as per ASCLIN channel. Values of X, Y, Z will be extracted from property file.		
Default value	SELECT_A_PORTY_PINZ		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

UART driver**5.3.1.2.16 UartStopBits****Table 439 Specification for UartStopBits**

Name	UartStopBits		
Description	This parameter is used for selecting the number of stop bits configuration in data frame. Note: Default value set with 1 bit to reduce frame size.		
Multiplicity	1..1	Type	EcuIntegerParamDef
Range	1 - 2		
Default value	1		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	
Dependency	-		

5.3.1.2.17 UartTxChannelMode**Table 440 Specification for UartTxChannelMode**

Name	UartTxChannelMode		
Description	UART channel transmit operation mode. Note: Default set in interrupt mode to disable optional interface (schedule function will enable in case any channel configured in polling mode).		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	INTERRUPT: UART channel transmit operation in interrupt mode. POLLING: UART channel transmit operation in polling mode.		
Default value	INTERRUPT		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.3 Container: UartConfigSet

This container contains the channel configuration of the UART driver. This container is a multiple configuration container. This container and its sub-containers exist once per configuration set.

UART driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

5.3.1.4 Container: UartGeneral

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

5.3.1.4.1 UartAbortReadApi**Table 441 Specification for UartAbortReadApi**

Name	UartAbortReadApi		
Description	Switch to enable/disable abort read feature. 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	IFX	Scope	LOCAL
Dependency	-		

5.3.1.4.2 UartAbortWriteApi**Table 442 Specification for UartAbortWriteApi**

Name	UartAbortWriteApi		
Description	Switch to enable/disable abort write feature. 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	-

UART driver**Table 442 Specification for UartAbortWriteApi (continued)**

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

5.3.1.4.3 UartClockRef**Table 443 Specification for UartClockRef**

Name	UartClockRef		
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	EcuReferenceDef
Range	Reference to Node: McuAscLinChannelAllocationConf, 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	-		

5.3.1.4.4 UartCsrClksel**Table 444 Specification for UartCsrClksel**

Name	UartCsrClksel		
Description	<p>This parameter selects the baud rate logic clock for the UART driver.</p> <p>Note: Default value set with fast mode.</p>		
Multiplicity	1..1	Type	EcuEnumerationParamDef
Range	ASCLINF: McuAscLinFastFrequency is selected as input frequency of ASCLIN. ASCLINS: McuAscLinSlowFrequency is selected as input frequency of ASCLIN.		
Default value	ASCLINF		
Post-build variant value	FALSE	Post-build variant multiplicity	-

UART driver**Table 444 Specification for UartCsrClksel (continued)**

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

5.3.1.4.5 UartDeInitApi**Table 445 Specification for UartDeInitApi**

Name	UartDeInitApi		
Description	Switch to enable/disable UART driver de-init feature. 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	IFX	Scope	LOCAL
Dependency	-		

5.3.1.4.6 UartDevErrorDetect**Table 446 Specification for UartDevErrorDetect**

Name	UartDevErrorDetect		
Description	Switch to enable/disable development error detection. Note: Default value is enabled to detect development time error.		
Multiplicity	1..1	Type	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	Post-build variant multiplicity	-

UART driver**Table 446 Specification for UartDevErrorDetect (continued)**

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

5.3.1.4.7 UartIndex**Table 447 Specification for UartIndex**

Name	UartIndex		
Description	Specifies the instance identifier of this module instance. In case single instance is present value should be 0. Note: Default value set minimum because single instance is present.		
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	-		

5.3.1.4.8 UartInitCheckApi**Table 448 Specification for UartInitCheckApi**

Name	UartInitCheckApi		
Description	Parameter adds or removes the Uart_InitCheck() 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	-

UART driver**Table 448 Specification for UartInitCheckApi (continued)**

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

5.3.1.4.9 UartInitDeInitApiMode**Table 449 Specification for UartInitDeInitApiMode**

Name	UartInitDeInitApiMode		
Description	Configuration parameter defines the privilege mode in which the initialization and de-initialization API's operate. Note: Since UART driver accesses the SFRs, it is more efficient to operate the UART driver in supervisor mode. Hence, the default mode of operation is supervisor.		
Multiplicity	1..1	Type	EcucEnumerationParamDef
Range	UART_MCAL_SUPERVISOR: Init and De-init APIs operate in supervisory mode. UART_MCAL_USER1: Init and De-init APIs operate in USER1 mode.		
Default value	UART_MCAL_SUPERVISOR		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.4.10 UartMainFunctionReadPeriod**Table 450 Specification for UartMainFunctionReadPeriod**

Name	UartMainFunctionReadPeriod		
Description	Specifies the period of main function Uart_MainFunction_Read in seconds. UART driver does not require this information but the BSW schedule will use this information.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0 - 10.0		
Default value	0.005		
Post-build variant value	FALSE	Post-build variant multiplicity	-

UART driver**Table 450 Specification for UartMainFunctionReadPeriod (continued)**

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

5.3.1.4.11 UartMainFunctionWritePeriod**Table 451 Specification for UartMainFunctionWritePeriod**

Name	UartMainFunctionWritePeriod		
Description	Specifies the period of main function Uart_MainFunction_Write in seconds. UART driver does not require this information but the BSW schedule will use this information.		
Multiplicity	1..1	Type	EcucFloatParamDef
Range	0 - 10.0		
Default value	0.005		
Post-build variant value	FALSE	Post-build variant multiplicity	-
Value configuration class	Pre-Compile	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.4.12 UartRunTimeErrorDetect**Table 452 Specification for UartRunTimeErrorDetect**

Name	UartRunTimeErrorDetect		
Description	The activation of runtime errors is configurable (ON / OFF) at pre-compile time. Note: The detection of runtime related errors is enabled by default to ensure that runtime 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	-

UART driver**Table 452 Specification for UartRunTimeErrorDetect (continued)**

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

5.3.1.4.13 UartSafetyEnable**Table 453 Specification for UartSafetyEnable**

Name	UartSafetyEnable		
Description	Switch to enable/disable the safety check. TRUE: Enable safety check FALSE: Disable safety check. 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	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	-		

5.3.1.4.14 UartSleepEnable**Table 454 Specification for UartSleepEnable**

Name	UartSleepEnable		
Description	Switch enable/disable the ASCLIN module sleep request handling by setting EDIS bit in CLC register. MCU API can request for sleep mode. Refer MCU design specification for more details. TRUE: EDIS bit is set to 1 in CLC register, sleep mode request can be recognized by ASCLIN module and enter in sleep mode. FALSE: EDIS is set to 0, a sleep mode request is ignore and module continues its operation. Note: The optional feature is disabled by default.		

UART driver**Table 454 Specification for UartSleepEnable (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	-		

5.3.1.4.15 UartTimeoutCount**Table 455 Specification for UartTimeoutCount**

Name	UartTimeoutCount		
Description	Specifies the maximum time in nanoseconds to wait for hardware timeout errors. Note: Maximum value is kept as default value for this parameter.		
Multiplicity	1..1	Type	EcucIntegerParamDef
Range	100 - 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	-		

5.3.1.4.16 UartVersionInfoApi**Table 456 Specification for UartVersionInfoApi**

Name	UartVersionInfoApi		
Description	Switch to enable/disable get version information API. Note: The optional APIs are disabled by default to minimize the executable code size.		
Multiplicity	1..1	Type	EcucBooleanParamDef

UART driver**Table 456 Specification for UartVersionInfoApi (continued)**

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	-		

5.3.1.5 Container: UartNotification

Container to hold notification(read, write, read abort, write abort) function address.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

5.3.1.5.1 UartAbortReceiveNotifPtr**Table 457 Specification for UartAbortReceiveNotifPtr**

Name	UartAbortReceiveNotifPtr		
Description	Parameter which holds receive abort notification function address. Definition of function present in application. If the user not required notification then parameter shall be configured to NULL_PTR. Note: Optional interface so default value set NULL_PTR. The UART driver does not validate the configured function name or address, User should configure valid address or function.		
Multiplicity	1..1	Type	Uart_NotificationPtrT ype
Range			
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

UART driver**5.3.1.5.2 UartAbortTransmitNotifPtr****Table 458 Specification for UartAbortTransmitNotifPtr**

Name	UartAbortTransmitNotifPtr		
Description	<p>Parameter holds transmit abort notification function address. Definition of function present in application. If the user not required notification then parameter shall be configured to NULL_PTR.</p> <p>Note: Optional interface so default value set NULL_PTR. The UART driver does not validate the configured function name or address, User should configure valid address or function.</p>		
Multiplicity	1..1	Type	Uart_NotificationPtrT ype
Range			
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.5.3 UartReceiveNotifPtr**Table 459 Specification for UartReceiveNotifPtr**

Name	UartReceiveNotifPtr		
Description	<p>Parameter holds receive complete notification function address. Definition of function present in application. If the user not required notification then parameter shall be configured with NULL_PTR.</p> <p>Note: Optional interface so default value set with NULL_PTR. The UART driver does not validate the configured function name or address, User should configure valid address or function.</p>		
Multiplicity	1..1	Type	Uart_NotificationPtrT ype
Range			
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL

UART driver**Table 459 Specification for UartReceiveNotifPtr (continued)**

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

5.3.1.5.4 UartTransmitNotifPtr**Table 460 Specification for UartTransmitNotifPtr**

Name	UartTransmitNotifPtr		
Description	Parameter holds transmit complete notification function address. Definition of function present in application. If the user not required notification then parameter should be configured with NULL_PTR. Note: Optional interface so default value set NULL_PTR. The UART driver does not validate the configured function name or address, User should configure valid address or function.		
Multiplicity	1..1	Type	Uart_NotificationPtrType
Range			
Default value	NULL_PTR		
Post-build variant value	TRUE	Post-build variant multiplicity	-
Value configuration class	Post-Build	Multiplicity configuration class	-
Origin	IFX	Scope	LOCAL
Dependency	-		

5.3.1.6 Container: Uart

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

5.3.2 Functions - Type definitions**5.3.2.1 Uart_ChannelIdType****Table 461 Specification for Uart_ChannelIdType**

Syntax	Uart_ChannelIdType	
Type	uint8	
File	Uart.h	
Range	0-255	
Description	Data type used to specifies logical channel identifier of UART.	
Source	IFX	

UART driver**5.3.2.2 Uart_ConfigType****Table 462 Specification for Uart_ConfigType**

Syntax	Uart_ConfigType	
Type	Structure	
File	Uart.h	
Range	--	The elements of the data structure are specific to the microcontroller.
Description	Data type used to specify UART driver configuration.	
Source	IFX	

5.3.2.3 Uart_ErrorIdType**Table 463 Specification for Uart_ErrorIdType**

Syntax	Uart_ErrorIdType	
Type	Enumeration	
File	Uart.h	
Range	0 - UART_E_NO_ERR	No error.
	1 - UART_E_PARITY_ERR	Parity error.
	2 - UART_E_FRAME_ERR	Frame error.
	3 - UART_E_RXOVERFLOW_ERR	RXFIFO overflow error.
Description	Data type specifies the error occurred during the data transmission or reception.	
Source	IFX	

5.3.2.4 Uart_MemType**Table 464 Specification for Uart_MemType**

Syntax	Uart_MemType	
Type	uint8	
File	Uart.h	
Range	0-255	
Description	Data type of the buffer used in read and writes operation.	
Source	IFX	

5.3.2.5 Uart_NotificationPtrType**Table 465 Specification for Uart_NotificationPtrType**

Syntax	Uart_NotificationPtrType	
---------------	--------------------------	--

UART driver**Table 465 Specification for Uart_NotificationPtrType (continued)**

Type	Pointer to a function of type void Function_Name (const Uart_ErrorIdType ErrorId)
File	Uart.h
Description	Data type to specify function pointer declaration of UART call back.
Source	IFX

5.3.2.6 Uart_ReturnType**Table 466 Specification for Uart_ReturnType**

Syntax	Uart_ReturnType	
Type	Enumeration	
File	Uart.h	
Range	0 - UART_E_OK	API successful completed.
	1 - UART_E_NOT_OK	API reported development error.
	2 - UART_E_BUSY	UART channel is busy in same operation which is requested by API.
Description	Data type used specifies return value of UART driver API.	
Source	IFX	

5.3.2.7 Uart_SizeType**Table 467 Specification for Uart_SizeType**

Syntax	Uart_SizeType	
Type	uint16	
File	Uart.h	
Range	0-65535	
Description	Data type used to specify the number of bytes to be transmit or receive.	
Source	IFX	

5.3.2.8 Uart_StatusType**Table 468 Specification for Uart_StatusType**

Syntax	Uart_StatusType	
Type	Enumeration	
File	Uart.h	
Range	0 - UART_IDLE	Idle state (no transmits or receives operation in progress).

UART driver**Table 468 Specification for Uart_StatusType (continued)**

	1 - UART_BUSY_TRANSMIT	UART channel busy in transmit operation.
	2 - UART_BUSY_RECEIVE	UART channel busy in receive operation.
	3 - UART_BUSY_TRANSMIT_RECEIVE	UART channel busy in receive and transmit operation.
Description	Data type used to specify UART channel status.	
Source	IFX	

5.3.2.9 UartNotificationCallback**Table 469 Specification for UartNotificationCallback**

Syntax	UartNotificationCallback
File	Uart.h
Source	IFX

5.3.3 Functions - APIs

This section lists all the APIs of the UART driver.

5.3.3.1 Uart_InitCheck**Table 470 Specification for Uart_InitCheck API**

Syntax	Std_ReturnType Uart_InitCheck (const Uart_ConfigType * const ConfigPtr)	
Service ID	0xD8	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Address of UART driver configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Initialization check passed. E_NOT_OK: Initialization check failed.

UART driver**Table 470 Specification for Uart_InitCheck API (continued)**

Description	API returns the status of the modules initialization. API (optional API) is available only when the parameter UartInitCheckApi is enabled. Note: Init check should be performed in the following sequence: 1. Call Uart_Init. 2. Call Uart_InitCheck.
Source	IFX
Error handling	DET: UART_E_PARAM_POINTER: API service used with NULL pointer. UART_E_UNINIT: API service used without UART driver initialization. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	UartInitCheckApi
User hints	-

5.3.3.2 Uart_Init**Table 471 Specification for Uart_Init API**

Syntax	void Uart_Init (const Uart_ConfigType * const ConfigPtr)	
Service ID	0xD7	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Address of UART driver configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	API to initialize all configured ASCLIN hardware units with the values referenced by the parameter ConfigPtr.	
Source	IFX	

UART driver**Table 471 Specification for `Uart_Init` API (continued)**

Error handling	<p>DET:</p> <p>UART_E_ALREADY_INITIALIZED: UART driver is already initialized.</p> <p>UART_E_INIT_FAILED: UART driver initialization fails.</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	-

5.3.3.3 Uart_Read**Table 472 Specification for `Uart_Read` API**

Syntax	<pre>Uart_ReturnType Uart_Read (const Uart_ChannelIdType Channel, Uart_MemType * const MemPtr, const Uart_SizeType Size)</pre>	
Service ID	0xD9	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel (Not for the same channel)	
Parameters (in)	Channel Size	UART channel id. Number of bytes to be read. Note: If channel frame length configured with greater than 8 bit then number of bytes should be multiple of 2.
Parameters (out)	MemPtr	Application buffer address.
Parameters (in - out)	-	-
Return	Uart_ReturnType	UART_E_OK - Receive operation initiated successfully. UART_E_NOT_OK - Receive operation couldn't be initiated due to development errors. UART_E_BUSY - UART channel is busy in receive operation. If DET and Safety is disabled API will return UART_E_OK and UART_E_BUSY.
Description	API to read data from an UART channel, with specified size and the memory location.	

UART driver**Table 472 Specification for Uart_Read API (continued)**

Source	IFX
Error handling	<p>DET:</p> <p>UART_E_UNINIT: API service used without UART driver initialization.</p> <p>UART_E_INVALID_SIZE: API Service called with invalid data length parameter.</p> <p>UART_E_STATE_BUSY: API service called when channel is in busy state.</p> <p>UART_E_INVALID_CHANNEL: API service used with an invalid channel identifier.</p> <p>UART_E_PARAM_POINTER: API service used with 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	-

5.3.3.4 Uart_Write**Table 473 Specification for Uart_Write API**

Syntax	<pre>Uart_ReturnType Uart_Write (const Uart_ChannelIdType Channel, const Uart_MemType * const MemPtr, const Uart_SizeType Size)</pre>	
Service ID	0xDA	
Sync/Async	Asynchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel (Not for the same channel)	
Parameters (in)	Channel MemPtr Size	UART channel id. Application memory address from where data to be transmit. Number of data bytes to be transmitted. Note: If channel frame length configured with greater than 8 bits then number of bytes should be multiple of 2.
Parameters (out)	-	-
Parameters (in - out)	-	-

UART driver**Table 473 Specification for Uart_Write API (continued)**

Return	Uart_ReturnType	UART_E_OK - Transmit operation initiated successfully. UART_E_NOT_OK - Transmit operation couldn't be initiated due to development errors. UART_E_BUSY - UART channel is busy in transmit operation. If DET and Safety is disabled API will return UART_E_OK and UART_E_BUSY.
Description	API to write data to an Uart channel, with specified size and the memory location.	
Source	IFX	
Error handling	<p>DET:</p> <ul style="list-style-type: none"> UART_E_STATE_BUSY: API service called when channel is in busy state. UART_E_INVALID_SIZE: API Service called with invalid data length parameter. UART_E_UNINIT: API service used without UART driver initialization. UART_E_INVALID_CHANNEL: API service used with an invalid channel identifier. UART_E_PARAM_POINTER: API service used with NULL pointer. <p>Runtime Errors: None</p> <p>DEM: None</p> <p>Safety Errors:</p> <ul style="list-style-type: none"> UART_E_TXFIFO_FILL_ERR: TXFIFO fill error, Fill level not matched with number of bytes filled in TXFIFO. <p><i>Note: All DET IDs are also reported as safety errors.</i></p>	
Configuration dependencies	-	
User hints	-	

5.3.3.5 Uart_AbortRead**Table 474 Specification for Uart_AbortRead API**

Syntax	Uart_SizeType Uart_AbortRead (const Uart_ChannelIdType Channel)	
Service ID	0xDC	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel (Not for the same channel)	
Parameters (in)	Channel	UART channel id.
Parameters (out)	-	-

UART driver**Table 474 Specification for Uart_AbortRead API (continued)**

Parameters (in - out)	-	-
Return	Uart_SizeType	Number of bytes successfully received and stored to the application memory location before the read operation was aborted.
Description	API to abort read operation on given channel. Note: API (optional API) is available only when the parameter UartAbortReadApi is enabled. Abort read notification is called when receive operation on given UART channel is in progress state. (Not in preparation or idle state).	
Source	IFX	
Error handling	DET: UART_E_UNINIT: API service used without UART driver initialization. UART_E_INVALID_CHANNEL: API service used with an invalid channel identifier. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	UartAbortReadApi	
User hints	-	

5.3.3.6 Uart_AbortWrite**Table 475 Specification for Uart_AbortWrite API**

Syntax	Uart_SizeType Uart_AbortWrite (const Uart_ChannelIdType Channel)	
Service ID	0xDB	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel (Not for the same channel)	
Parameters (in)	Channel	UART channel id.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Uart_SizeType	Number of bytes that have been successfully transmitted before the write operation was aborted.

UART driver**Table 475 Specification for `Uart_AbortWrite` API (continued)**

Description	API to abort data transmission on given channel. Note: API(optional API) is available only when the parameter UartAbortWriteApi is enabled. Abort write notification is called when transmit operation on given UART channel is in progress state. (Not in preparation or idle state).
Source	IFX
Error handling	DET: UART_E_UNINIT: API service used without UART driver initialization. UART_E_INVALID_CHANNEL: API service used with an invalid channel identifier. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	UartAbortWriteApi
User hints	-

5.3.3.7 `Uart_GetStatus`**Table 476 Specification for `Uart_GetStatus` API**

Syntax	Uart_StatusType <code>Uart_GetStatus</code> (const Uart_ChannelIdType Channel)	
Service ID	0xDD	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant for different channel (Not for the same channel)	
Parameters (in)	Channel	UART channel id.
Parameters (out)	-	-
Parameters (in - out)	-	-

UART driver**Table 476 Specification for Uart_GetStatus API (continued)**

Return	Uart_StatusType	UART_IDLE: Idle state (no transmit or receive operation in progress). UART_BUSY_TRANSMIT: UART channel busy in transmit operation. UART_BUSY_RECEIVE: UART channel busy in receive operation. UART_BUSY_TRANSMIT_RECEIVE: UART channel busy in transmit and receive operation.
Description	API to read an UART channels status.	
Source	IFX	
Error handling	DET: UART_E_UNINIT: API service used without UART driver initialization. UART_E_INVALID_CHANNEL: API service used with an invalid channel identifier. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>	
Configuration dependencies	-	
User hints	-	

5.3.3.8 Uart_DeInit**Table 477 Specification for Uart_DeInit API**

Syntax	void Uart_DeInit (void)	
Service ID	0xDE	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

UART driver**Table 477 Specification for `Uart_DeInit` API (continued)**

Description	UART driver de-initialization function. Note: API (optional API) is available only if parameter <code>UartDeInitApi</code> is enabled.
Source	IFX
Error handling	DET: UART_E_UNINIT: API service used without UART driver initialization. Runtime Errors: None DEM: None Safety Errors: None <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	<code>UartDeInitApi</code>
User hints	-

5.3.3.9 `Uart_GetVersionInfo`**Table 478 Specification for `Uart_GetVersionInfo` API**

Syntax	void <code>Uart_GetVersionInfo</code> (<code>Std_VersionInfoType</code> * const <code>VersionInfoPtr</code>)	
Service ID	0xDF	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	<code>VersionInfoPtr</code>	Address on which version information to be stored.
Parameters (in - out)	-	-
Return	<code>void</code>	-
Description	API to get the version information of UART driver. Note: API (optional API) is available only if parameter <code>UartVersionInfoApi</code> is enabled.	
Source	IFX	

UART driver**Table 478 Specification for `Uart_GetVersionInfo` API (continued)**

Error handling	<p>DET:</p> <p>UART_E_PARAM_POINTER: API service used with 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	UartVersionInfoApi
User hints	-

5.3.4 Notifications and Callbacks

The driver does not implement any notification and callback function. The UART driver give a notification whenever UART transmit, UART receive, UART receive abort or UART transmit abort operation is completed successfully. The application is responsible for the implementing the notification function.

5.3.5 Scheduled functions

This section lists all the scheduled functions of the UART driver.

5.3.5.1 `Uart_MainFunction_Read`**Table 479 Specification for `Uart_MainFunction_Read` API**

Syntax	void Uart_MainFunction_Read (void)	
Service ID	0xE0	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Schedule function to handle receives operation in polling mode.</p> <p><i>Note: Function will be available if any of channels receive operation configured in polling mode.</i></p>	

UART driver**Table 479 Specification for Uart_MainFunction_Read API (continued)**

Source	IFX
Error handling	<p>DET: None</p> <p>Runtime Errors:</p> <ul style="list-style-type: none"> UART_E_FRAME_ERROR: This runtime error is reported when the frame check fail. UART_E_RXFIFO_OVERFLOW: This runtime error is reported when the RXFIFO overflow error set. UART_E_PARITY_ERROR: This runtime error is reported when the parity check fail. <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	UartRxChannelMode
User hints	-

5.3.5.2 Uart_MainFunction_Write**Table 480 Specification for Uart_MainFunction_Write API**

Syntax	void Uart_MainFunction_Write (void)	
Service ID	0xE1	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	<p>Schedule function to handle transmits operation in polling mode.</p> <p><i>Note: Function will be available if any of channels transmit operation configured in polling mode.</i></p>	
Source	IFX	

UART driver**Table 480 Specification for Uart_MainFunction_Write API (continued)**

Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: UART_E_TXFIFO_FILL_ERR: TXFIFO fill error, Fill level not matched with number of bytes filled in TXFIFO. <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	UartTxChannelMode
User hints	-

5.3.6 Interrupt service routines

This section lists all the interrupt handlers of the driver.

5.3.6.1 Uart_IsrError**Table 481 Specification for Uart_IsrError API**

Syntax	void Uart_IsrError (const uint8 HwUnit)	
Service ID	0xE2	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant (Not for the same HW Unit).	
Parameters (in)	HwUnit	ASCLIN channel number.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	IRQ handler for error occurred during reception and handle transmit complete interrupt.	
Source	IFX	

UART driver**Table 481 Specification for `Uart_IsrError` API (continued)**

Error handling	<p>DET: None</p> <p>Runtime Errors:</p> <ul style="list-style-type: none"> UART_E_RXFIFO_OVERFLOW: This runtime error is reported when the RXFIFO overflow error set. UART_E_PARITY_ERROR: This runtime error is reported when the parity check fail. UART_E_FRAME_ERROR: This runtime error is reported when the frame check fail. <p>DEM: None</p> <p>Safety Errors:</p> <ul style="list-style-type: none"> UART_E_INVALID_HW_UNIT: IRQ handler called with invalid hardware unit identifier. UART_E_SPURIOUS_INTERRUPT: Spurious interrupt detected. <p><i>Note: All DET IDs are also reported as safety errors.</i></p>
Configuration dependencies	UartTxChannelMode, UartRxChannelMode
User hints	-

5.3.6.2 `Uart_IsrReceive`**Table 482 Specification for `Uart_IsrReceive` API**

Syntax	<pre>void Uart_IsrReceive (const uint8 HwUnit)</pre>	
Service ID	0xE3	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant (Not for the same HW Unit).	
Parameters (in)	HwUnit	ASCLIN channel number.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	IRQ handler for RXFIFO level interrupts.	
Source	IFX	

UART driver**Table 482 Specification for Uart_IsrReceive API (continued)**

Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: UART_E_INVALID_HW_UNIT: IRQ handler called with invalid hardware unit identifier. UART_E_SPURIOUS_INTERRUPT: Spurious interrupt detected. <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	UartRxChannelMode
User hints	-

5.3.6.3 Uart_IsrTransmit**Table 483 Specification for Uart_IsrTransmit API**

Syntax	void Uart_IsrTransmit (const uint8 HwUnit)	
Service ID	0xE4	
Sync/Async	Synchronous	
ASIL Level	B	
Re-entrancy	Reentrant (Not for the same HW Unit).	
Parameters (in)	HwUnit	ASCLIN channel number.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	IRQ handler for TXFIFO level interrupts.	
Source	IFX	

UART driver**Table 483 Specification for `Uart_IsrTransmit` API (continued)**

Error handling	DET: None Runtime Errors: None DEM: None Safety Errors: UART_E_SPURIOUS_INTERRUPT: Spurious interrupt detected. UART_E_INVALID_HW_UNIT: IRQ handler called with invalid hardware unit identifier. UART_E_TXFIFO_FILL_ERR: TXFIFO fill error, Fill level not matched with number of bytes filled in TXFIFO. <i>Note: All DET IDs are also reported as safety errors.</i>
Configuration dependencies	UartTxChannelMode
User hints	-

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.

Note: The following error IDs are also reported as safety errors.

Table 484 Description of development errors reported

Description	Source	Error code and value	Applicable APIs
API service used without UART driver initialization.	IFX	UART_E_UNINIT=0x00	Uart_InitCheck, Uart_GetStatus, Uart_AbortWrite, Uart_AbortRead, Uart_Write, Uart_Read, Uart_Delnit
API service used with an invalid channel identifier.	IFX	UART_E_INVALID_CHANNEL=0x01	Uart_GetStatus, Uart_AbortRead, Uart_AbortWrite, Uart_Write, Uart_Read
API service used with NULL pointer.	IFX	UART_E_PARAM_POINTER=0x02	Uart_GetVersionInfo, Uart_Write, Uart_Read, Uart_InitCheck
API service called when channel is in busy state.	IFX	UART_E_STATE_BUSY=0x03	Uart_Write, Uart_Read
UART driver initialization fails.	IFX	UART_E_INIT_FAILED=0x04	Uart_Init

UART driver
Table 484 Description of development errors reported (continued)

Description	Source	Error code and value	Applicable APIs
API Service called with invalid data length parameter.	IFX	UART_E_INVALID_SIZE=0x05	Uart_Write, Uart_Read
UART driver is already initialized.	IFX	UART_E_ALREADY_INITIALIZED=0x6	Uart_Init

5.3.7.2 Production errors

The driver does not report any production errors.

5.3.7.3 Safety errors

The following table lists all the safety errors reported by the driver.

Table 485 Description of safety errors reported

Description	Source	Error code and value	Applicable APIs
Spurious interrupt detected.	IFX	UART_E_SPURIOUS_INTERRUPT=0x7	Uart_IsrError, Uart_IsrReceive, Uart_IsrTransmit
IRQ handler called with invalid hardware unit identifier.	IFX	UART_E_INVALID_HW_UNIT=0x8	Uart_IsrError, Uart_IsrTransmit, Uart_IsrReceive
TXFIFO fill error, Fill level not matched with number of bytes filled in TXFIFO.	IFX	UART_E_TXFIFO_FILL_ERR=0x9	Uart_MainFunction_Write, Uart_IsrTransmit, Uart_Write

5.3.7.4 Runtime errors

The following table lists all the runtime errors reported by the driver.

Table 486 Description of runtime errors reported

Description	Source	Error code and value	Applicable APIs
This runtime error is reported when the frame check fail.	IFX	UART_E_FRAME_ERROR=0x02	Uart_MainFunction_Read, Uart_IsrError
This runtime error is reported when the parity check fail.	IFX	UART_E_PARITY_ERROR=0x01	Uart_MainFunction_Read, Uart_IsrError
This runtime error is reported when the RXFIFO overflow error set.	IFX	UART_E_RXFIFO_OVERFLOW=0x03	Uart_MainFunction_Read, Uart_IsrError

UART driver**5.3.8 Deviations and limitations**

The section describes the deviations and limitations from software specification.

5.3.8.1 Deviations

The driver does not report any deviations.

5.3.8.2 Limitations

The section describes the limitations from software specification.

Table 487 Known limitations

Reference	Limitation
Uart transmit complete notification.	<p>It is observed that the UART channel transmit complete notification is triggered before transmission of the last frame from the ASCLIN kernel.</p> <p>If application has any use case of calling <code>Uart_DeInit</code> API immediately after receiving the transmit completion notification, it is being observed that the receiver is unable to receive the last frame. This behavior is observed in both interrupt and polling mode.</p> <p>Workaround: Provide one frame delay (based on the baudrate used) before calling the <code>Uart_DeInit</code> API. Example: In 9600 kbps baud rate configuration, if the application software is waiting for transmit complete notification to invoke <code>Uart_DeInit</code> API, there should be a delay of 1.04167 milliseconds between the application received transmit complete notification and <code>Uart_DeInit</code> API invocation.</p>

5.3.9 Unsupported hardware features

The following features of the ASCLIN are not supported:

- RXFIFO underflow error detection
- TXFIFO overflow error detection
- DMA base data transfer

Revision history

Revision history

Major changes since the last revision

Date	Version	Description
2020-05-22	1.40.0_1 3.0	<ul style="list-style-type: none"> • FLSLOADER <ul style="list-style-type: none"> - Added file FlsLdr_ExclArea.h for FLSLOADER exclusive area, updated the file structure diagram. - Added the example usage for FLSLOADER exclusive area during PFlash write and erase operations. - Added limitation for increased timeout values during write and erase operation considering parallel access to DFlash0/PFlash and DFlash1/PFlash by Tricore and HSM respectively. • Dma <ul style="list-style-type: none"> - Error handling updated for the Dma_MEStatusClear API. - Limitations section updated for the handling of multiple RP error interrupts and usage of SPI and DMA drivers together. - Interrupt connections section updated with DMA interrupt priority information.
2020-03-18	1.40.0_1 2.0	<ul style="list-style-type: none"> • Dma <ul style="list-style-type: none"> - Description updated for the following APIs and configuration parameters, to improve clarity. No functional impact. - APIs: Dma_GetEvents ,Dma_MEStatusClear,Dma_GetCurrentTimeStamp,Dma_IsC hannelInitDone ,Dma_GetCrcValue Configuration parameters: DmaTcsReferenceDataCrc,DmaTcsReferenceAddressCrc • Smu <ul style="list-style-type: none"> - Description and Re-entrancy fields updated for the Smu_LockConfigRegs API - Multicore and Resource Manager updated with regards to multicore capability and re-entrancy of the Smu_LockConfigRegs API. • Uart <ul style="list-style-type: none"> - MCU integration hints updated for the devices where ASCLINs are non-consecutive • Reference to the BASIC User Manual is updated.

Revision history

Date	Version	Description
2020-01-28	1.40.0_1 1.0	<ul style="list-style-type: none"> • Smu <ul style="list-style-type: none"> - Limitations section updated. • Dsadc <ul style="list-style-type: none"> - Configuration container DsadcCommonPublishedInformation renamed to CommonPublishedInformation • Dma <ul style="list-style-type: none"> - Dma_ChInit API description updated - DmaTcsSwapDataCRCByteOrder parameter added - Parameters DmaTcsSourceAddress, DmaTcsDestinationAddress, DmaTcsDoubleBuffer, DmaTcsReferenceDataCrc updated • Uart <ul style="list-style-type: none"> - Configuration dependencies added in Uart_IsrError, Uart_IsrTransmit and Uart_IsrReceive API's • Mapping of hardware-software interface image is updated
2019-12-12	1.31.0_1 0.0	Reference to the BASIC User Manual is updated.
2019-10-10	1.30.0_9. 0	<ul style="list-style-type: none"> • FlsLoader <ul style="list-style-type: none"> - Example usage section is updated. • Dsadc <ul style="list-style-type: none"> - Configuration dependency for Dsadc_DelInit, DsadcReadStreamResults, Dsadc_ReadResult, Dsadc_SetupResultBuffer, Dsadc_TimerIsr and Dsadc_Isr APIs is updated. • Smu <ul style="list-style-type: none"> - API Description, Return value description and Error handling description for Smu_RegisterMonitor API is updated. - Updated the Deviations and limitations section.
2019-08-05	8.0	Reference to the BASIC User Manual is updated.
2019-07-26	7.0	<ul style="list-style-type: none"> • Dsadc <ul style="list-style-type: none"> - Updated the Deviations and limitations section - Updated user hints field in the Dsadc_Isr API

Revision history

Date	Version	Description
2019-07-23	6.0	<ul style="list-style-type: none"> • Dma <ul style="list-style-type: none"> - Dma_ChIsStatusAsserted, Dma_GetChannelEvent and Dma_GetMoveEngineEvent APIs are removed. The Dma_GetEvents API is added. The DmaTcsSourceAddress, DmaTcsDestinationAddress and DmaTcsDoubleBuffer configuration parameters are updated. The DmaUserHeaderFileWithExternDeclarations configuration parameter is added. - Limitations and deviations section is updated. • Dsadc <ul style="list-style-type: none"> - Example usage for trigger mode window for GTM with PWM driver added. - Interrupt connection in Integration hints section is updated. - Key architectural considerations section is updated. - AoUs are for the PWM driver usage along with DSADC driver is added. - Dsadc_GetVersionInfo API and the DsadcVersionInfoApi configuration parameter are added. • FlsLoader <ul style="list-style-type: none"> - Limitations and deviations section is updated. • Smu <ul style="list-style-type: none"> - User hints added in Smu_Init, Smu_DelInit and Smu_ClearAlarmStatus APIs. - Description updated and User hints added for the Smu_CoreAliveTest API. - Example usage updated for the Smu_CoreAliveTest API. • Uart <ul style="list-style-type: none"> - UartChanBaudOverSampling configuration parameter 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 DSADC driver.
2018-02-05	3.0	Updated formatting in the <i>Key architectural considerations</i> section.
2019-02-04	2.0	<ul style="list-style-type: none"> • Updated <i>Integration hints</i> and <i>Reference information</i> for all modules. • Updated module-specific AoUs.
2018-10-12	1.0	Initial version.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-05-22

Published by

**Infineon Technologies AG
81726 Munich, Germany**

**© 2020 Infineon Technologies AG
All Rights Reserved.**

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

**Document reference
IFX-lxn1559808687965**

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.