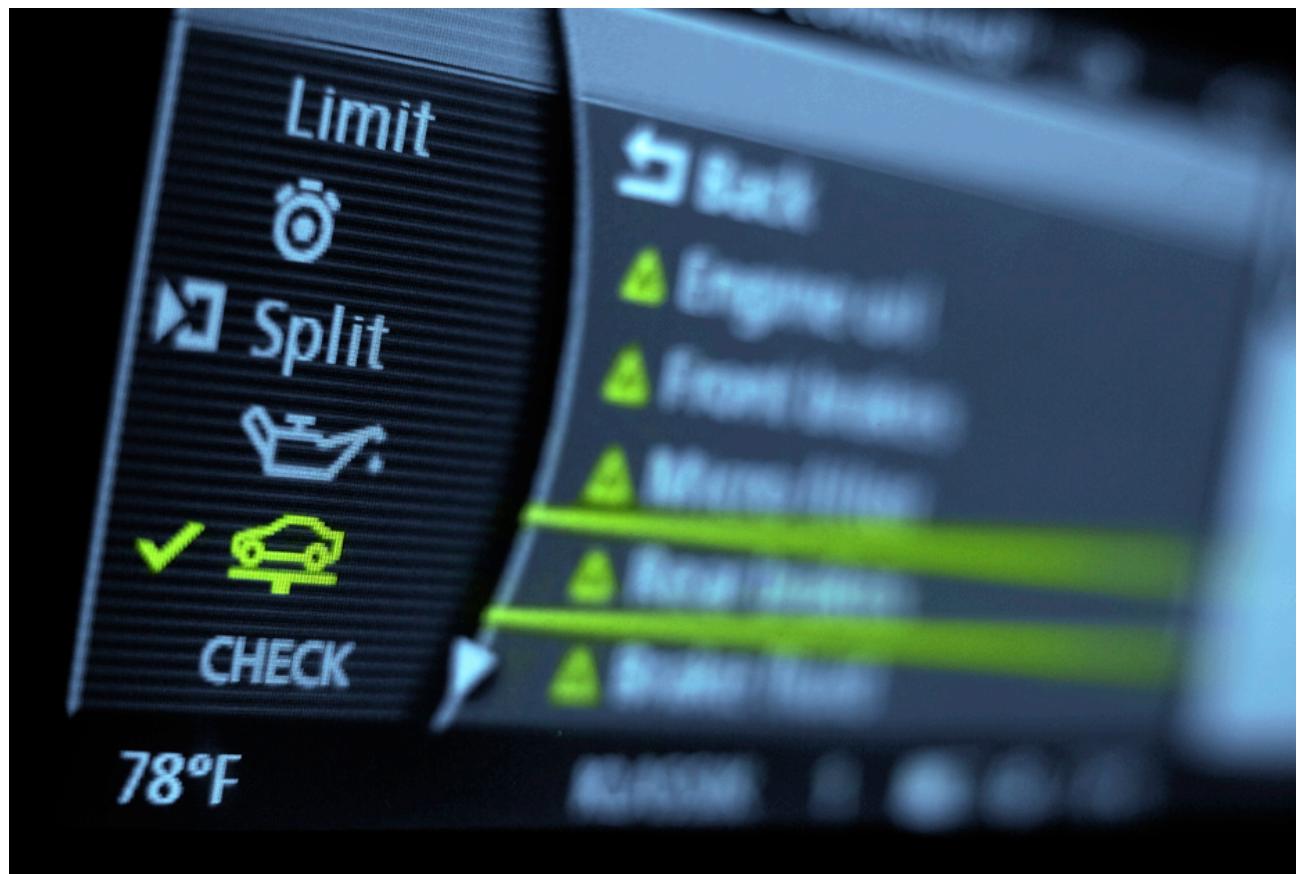




Elektrobit

EB tresos[®] AutoCore Generic 8 Base documentation

product release 8.7.1





Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

<https://www.elektrobit.com/support>

Legal disclaimer

Confidential and proprietary information

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of Elektrobit Automotive GmbH.

All brand names, trademarks and registered trademarks are property of their rightful owners and are used only for description.

Copyright 2019, Elektrobit Automotive GmbH.



Table of Contents

1. Overview of EB tresos AutoCore Generic 8 Base documentation	30
2. Supported features	31
2.1. Overview	31
2.2. Supported MemMap features	31
2.3. Supported Det features	31
3. ACG8 Base release notes	32
3.1. Overview	32
3.2. Scope of the release	32
3.2.1. Configuration tool	32
3.2.2. AUTOSAR modules	32
3.2.3. EB (Elektrobit) modules	33
3.2.4. MCAL modules and EB tresos AutoCore OS	33
3.3. Module release notes	33
3.3.1. ApplTemplates module release notes	34
3.3.1.1. Change log	34
3.3.1.2. New features	36
3.3.1.3. EB-specific enhancements	36
3.3.1.4. Deviations	36
3.3.1.5. Limitations	37
3.3.1.6. Open-source software	37
3.3.2. Atomics module release notes	37
3.3.2.1. Change log	37
3.3.2.2. New features	40
3.3.2.3. EB-specific enhancements	40
3.3.2.4. Deviations	40
3.3.2.5. Limitations	40
3.3.2.6. Open-source software	41
3.3.3. Base module release notes	41
3.3.3.1. Change log	41
3.3.3.2. New features	47
3.3.3.3. EB-specific enhancements	48
3.3.3.4. Deviations	48
3.3.3.5. Limitations	49
3.3.3.6. Open-source software	50
3.3.4. Compiler module release notes	50
3.3.4.1. Change log	50
3.3.4.2. New features	51
3.3.4.3. EB-specific enhancements	51
3.3.4.4. Deviations	52



3.3.4.5. Limitations	52
3.3.4.6. Open-source software	53
3.3.5. Configurators module release notes	53
3.3.5.1. Change log	53
3.3.5.2. New features	79
3.3.5.3. EB-specific enhancements	79
3.3.5.4. Deviations	79
3.3.5.5. Limitations	80
3.3.5.6. Open-source software	83
3.3.6. Det module release notes	83
3.3.6.1. Change log	83
3.3.6.2. New features	86
3.3.6.3. EB-specific enhancements	86
3.3.6.4. Deviations	87
3.3.6.5. Limitations	89
3.3.6.6. Open-source software	89
3.3.7. EcuC module release notes	89
3.3.7.1. Change log	89
3.3.7.2. New features	92
3.3.7.3. EB-specific enhancements	92
3.3.7.4. Deviations	92
3.3.7.5. Limitations	93
3.3.7.6. Open-source software	93
3.3.8. HidWiz module release notes	93
3.3.8.1. Change log	93
3.3.8.2. New features	97
3.3.8.3. EB-specific enhancements	97
3.3.8.4. Deviations	97
3.3.8.5. Limitations	97
3.3.8.6. Open-source software	97
3.3.9. Make module release notes	97
3.3.9.1. Change log	97
3.3.9.2. New features	101
3.3.9.3. EB-specific enhancements	102
3.3.9.4. Deviations	102
3.3.9.5. Limitations	102
3.3.9.6. Open-source software	102
3.3.10. MemMap module release notes	102
3.3.10.1. Change log	102
3.3.10.2. New features	106
3.3.10.3. EB-specific enhancements	107
3.3.10.4. Deviations	107



3.3.10.5. Limitations	107
3.3.10.6. Open-source software	107
3.3.11. PbcfgM module release notes	107
3.3.11.1. Change log	107
3.3.11.2. New features	111
3.3.11.3. EB-specific enhancements	111
3.3.11.4. Deviations	112
3.3.11.5. Limitations	112
3.3.11.6. Open-source software	112
3.3.12. Platforms module release notes	112
3.3.12.1. Change log	112
3.3.12.2. New features	114
3.3.12.3. EB-specific enhancements	114
3.3.12.4. Deviations	114
3.3.12.5. Limitations	114
3.3.12.6. Open-source software	114
3.3.13. SvcAs module release notes	115
3.3.13.1. Change log	115
3.3.13.2. New features	120
3.3.13.3. EB-specific enhancements	120
3.3.13.4. Deviations	120
3.3.13.5. Limitations	120
3.3.13.6. Open-source software	121
3.3.14. Workflows module release notes	121
3.3.14.1. Change log	121
3.3.14.2. New features	124
3.3.14.3. EB-specific enhancements	124
3.3.14.4. Deviations	124
3.3.14.5. Limitations	124
3.3.14.6. Open-source software	124
4. ACG8 Base user's guide	125
4.1. Overview	125
4.2. Det module user guide	125
4.2.1. Overview	125
4.2.2. Background information	125
4.2.2.1. Handling of error reports	126
4.2.2.2. Impact of the Det on production	126
4.2.2.3. Support for error reporting from multiple cores	127
4.2.2.4. Software component description	127
4.2.2.4.1. Data types	128
4.2.2.4.2. Ports	128
4.2.2.4.2.1. DET service according to AUTOSAR 4.0.3	128



4.2.2.4.2.2. DET service according to AUTOSAR 4.3.0 and later	129
4.2.3. Configuring the Det module	129
4.2.3.1. List of module IDs of basic software modules	131
4.3. MemMap module user guide	134
4.3.1. Overview	134
4.3.2. Background information	134
4.3.2.1. Support for MemMap header files generation	134
4.3.2.1.1. Functional description	135
4.3.2.1.2. Configuration options	136
4.3.2.2. Support for multiple BSW-Implementation	137
4.3.2.2.1. Functional description	138
4.3.2.2.2. Configuration options	138
4.3.2.3. Support for AUTOSAR 4.0.2	138
4.3.2.3.1. Functional description	138
4.3.2.3.2. Configuration options	139
4.3.2.4. Support for memory section validation	139
4.3.2.4.1. Functional description	139
4.3.2.4.2. Configuration options	139
4.3.2.5. Support for coreScope	139
4.3.2.5.1. Functional description	140
4.3.2.5.2. Configuration options	140
4.3.2.6. Support for safety level	141
4.3.2.6.1. Functional description	141
4.3.2.6.2. Configuration options	141
4.3.3. Configuring the MemMap module	142
4.3.3.1. Overview	142
4.3.3.2. Configuring MemMap to generate macros compatible to AUTOSAR 4.0.2	143
4.3.3.3. Configuring MemMap to validate memory sections	144
4.3.3.4. Configuring MemMapGenericMapping	145
4.3.3.5. Configuring MemMapSectionSpecificMapping	147
4.4. Atomics module user's guide	149
4.4.1. Background information	149
4.4.1.1. Motivation	150
4.4.1.2. Atomicity of memory accesses	150
4.4.1.3. Consistency of memory accesses	150
4.4.2. Atomics integration notes	151
4.4.3. Migration	152
4.4.3.1. ATOMICS_USE_GENERIC_IMPL	153
4.4.3.2. Backward compatibility with Platforms module	153
4.4.4. Example of use	154
4.5. Application template and application demos	155
4.5.1. Overview	155



4.5.2. Basic template	157
4.5.2.1. Background information	158
4.5.2.2. Applicable workflow	158
4.5.2.3. Content description	158
4.5.2.4. Availability	159
4.5.3. Supplementary files	159
4.5.3.1. Background information	159
4.5.3.2. SWC and system description files	160
4.5.3.3. SWC source files	160
4.5.3.4. OS tasks' execution context	161
4.5.3.5. Availability	161
4.5.4. Simple RTE application demo	161
4.5.4.1. Functional behavior	161
4.5.4.2. Applicable workflow	162
4.5.4.3. Availability	163
4.5.5. Simple CAN application demo	163
4.5.5.1. Functional behavior	163
4.5.5.2. Applicable workflow	164
4.5.5.3. PDU allocation	164
4.5.5.4. Availability	165
4.5.6. Simple FlexRay application demo	165
4.5.6.1. Functional behavior	165
4.5.6.2. Applicable workflow	166
4.5.6.3. PDU allocation	167
4.5.6.4. Availability	167
4.5.7. Simple LIN application demo	167
4.5.7.1. Functional behavior	167
4.5.7.2. Applicable workflow	168
4.5.7.3. PDU allocation	169
4.5.7.4. Availability	169
4.5.8. Simple Ethernet application demo	169
4.5.8.1. Functional behavior	169
4.5.8.2. Applicable workflow	171
4.5.8.3. PDU allocation	171
4.5.8.4. Availability	171
4.5.9. Simple Memory application demo	171
4.5.9.1. Functional behavior	171
4.5.9.2. Applicable workflow	173
4.5.9.3. PDU allocation	173
4.5.9.4. Availability	173
4.5.10. Simple Post-build application demo	174
4.5.10.1. Functional behavior	174



4.5.10.2. Applicable workflow	174
4.5.10.3. Availability	174
4.5.11. Getting an application demo to run	175
4.5.11.1. Prerequisites for starting a new project	175
4.5.11.2. Importing the application demo	175
4.5.11.3. Adapting the build environment	177
4.5.11.3.1. Changing the compiler	178
4.5.11.3.2. Changing the board settings	178
4.5.11.4. Building the application demo	179
4.5.11.5. Taking further steps	179
4.6. Build environment	180
4.6.1. Overview	180
4.6.2. Background information	181
4.6.2.1. Improve build speed with GNU Make command line parameters	183
4.6.2.2. Folder structure of a project	183
4.6.2.3. Basic concepts of the makefiles	184
4.6.2.4. Make plug-ins	185
4.6.2.5. Board support packages	187
4.6.3. Configuring the build process	187
4.6.3.1. Setting the application path	188
4.6.3.2. Setting the compiler path	189
4.6.3.3. Configuring the makefiles	190
4.6.3.3.1. Starting the command shell	191
4.6.3.3.2. Defining the name of the project	192
4.6.3.3.3. Defining C files to be compiled	192
4.6.3.3.4. Defining a list of assembler files	193
4.6.3.3.5. Building libraries of compiled object files	193
4.6.3.3.6. Linking libraries to a project	195
4.6.3.3.7. Linking binary objects to a project	195
4.6.3.3.8. Extending the include path with project-specific path names	195
4.6.3.3.9. Selecting a board support package	196
4.6.3.3.10. Selecting a toolchain/a different compiler	197
4.6.3.3.11. Extending compiler and assembler options	197
4.6.3.3.12. Providing file-specific or library-specific compiler options	198
4.6.3.3.13. Adding preprocessor defines	198
4.6.3.4. Compiling the application	199
4.6.3.4.1. Locating output directories	201
4.6.3.4.2. Other make targets	202
4.7. Service Needs Calculator	203
4.7.1. Overview	203
4.7.2. Background information	203
4.7.3. Using the Service Needs Calculator	205



4.7.3.1. Configuring the Service Needs Calculator	205
4.7.3.2. Running the Service Needs Calculator	207
5. ACG8 Base module references	209
5.1. Overview	209
5.1.1. Notation in EB module references	209
5.1.1.1. Default value of configuration parameters	209
5.1.1.2. Range information of configuration parameters	209
5.2. Atomics	210
5.2.1. Configuration parameters	210
5.2.2. Application programming interface (API)	210
5.2.2.1. Type definitions	210
5.2.2.1.1. Atomic_t	210
5.2.2.1.2. Atomic_value_t	210
5.2.2.1.3. TS_IntStatusType	210
5.2.2.2. Macro constants	211
5.2.2.2.1. ATOMICS_OBJECT_INITIALIZER	211
5.2.2.2.2. ATOMICS_USER_ATOMIC_T	211
5.2.2.2.3. ATOMICS_USER_ATOMIC_VALUE_T	212
5.2.2.2.4. ATOMICS_USER_AUTOSAR_SPINLOCK	212
5.2.2.2.5. ATOMICS_USER_GET_VALUE	212
5.2.2.2.6. ATOMICS_USER_LOCK_OBJECT	212
5.2.2.2.7. ATOMICS_USER_MULTICORE_CASE	213
5.2.2.2.8. ATOMICS_USER_OBJECT_INIT	213
5.2.2.2.9. ATOMICS_USER_OBJECT_INITIALIZER	214
5.2.2.2.10. ATOMICS_USER_SET_VALUE	214
5.2.2.2.11. ATOMICS_USER_THREAD_FENCE	214
5.2.2.2.12. ATOMICS_USER_UNLOCK_OBJECT	214
5.2.2.2.13. ATOMICS_VALUE_MAX	215
5.2.2.2.14. TS_AtomicClearBit	215
5.2.2.2.15. TS_AtomicSetBit	215
5.2.2.3. Functions	216
5.2.2.3.1. Atomics_ClearFlag	216
5.2.2.3.2. Atomics_CompareExchange	216
5.2.2.3.3. Atomics_Exchange	217
5.2.2.3.4. Atomics_FetchAdd	218
5.2.2.3.5. Atomics_FetchAnd	218
5.2.2.3.6. Atomics_FetchOr	218
5.2.2.3.7. Atomics_FetchSub	219
5.2.2.3.8. Atomics_FetchXor	219
5.2.2.3.9. Atomics_Init	220
5.2.2.3.10. Atomics_Load	220
5.2.2.3.11. Atomics_Store	221



5.2.2.3.12. Atomics_TestAndSetFlag	221
5.2.2.3.13. Atomics_ThreadFence	222
5.2.2.3.14. TS_AtomicClearBit_16	222
5.2.2.3.15. TS_AtomicClearBit_32	222
5.2.2.3.16. TS_AtomicClearBit_64	223
5.2.2.3.17. TS_AtomicClearBit_8	223
5.2.2.3.18. TS_AtomicSetBit_16	224
5.2.2.3.19. TS_AtomicSetBit_32	224
5.2.2.3.20. TS_AtomicSetBit_64	224
5.2.2.3.21. TS_AtomicSetBit_8	225
5.2.2.3.22. TS_IntDisable	225
5.2.2.3.23. TS_IntRestore	226
5.2.3. Integration notes	226
5.2.3.1. Exclusive areas	226
5.2.3.2. Production errors	226
5.2.3.3. Memory mapping	226
5.2.3.4. Integration requirements	227
5.3. Base	227
5.3.1. Configuration parameters	227
5.3.1.1. BaseDbg	228
5.3.1.2. PostBuildSelectable	229
5.3.1.3. CustomOverrides	230
5.3.1.4. BaseTypes	233
5.3.1.5. BaseCpuConfig	244
5.3.1.6. GeneralTypes	245
5.3.1.7. BaseEcuConfig	245
5.3.1.8. CommonPublishedInformation	246
5.3.1.9. PublishedInformation	249
5.3.2. Application programming interface (API)	250
5.3.2.1. Type definitions	250
5.3.2.1.1. BufReq_ReturnType	250
5.3.2.1.2. BusTrcvErrorType	250
5.3.2.1.3. CanTrcv_PNActivationType	250
5.3.2.1.4. CanTrcv_TrsvFlagStateType	251
5.3.2.1.5. CanTrcv_TrsvModeType	251
5.3.2.1.6. CanTrcv_TrsvWakeupModeType	251
5.3.2.1.7. CanTrcv_TrsvWakeupReasonType	251
5.3.2.1.8. Can_ControllerStateType	252
5.3.2.1.9. Can_ErrorStateType	252
5.3.2.1.10. Can_HwHandleType	253
5.3.2.1.11. Can_HwType	253
5.3.2.1.12. Can_IdType	253



5.3.2.1.13. Can_PduType	254
5.3.2.1.14. Can_ReturnType	254
5.3.2.1.15. Can_StateTransitionType	254
5.3.2.1.16. ConstVoidPtr	255
5.3.2.1.17. EthIf_MeasurementIdxType	255
5.3.2.1.18. EthIf_SignalQualityResultType	255
5.3.2.1.19. EthIf_SwitchPortGroupIdxType	255
5.3.2.1.20. EthSwt_MacLearningType	255
5.3.2.1.21. EthSwt_MgmtInfoType	256
5.3.2.1.22. EthSwt_PortMirrorCfgType	256
5.3.2.1.23. EthSwt_PortMirrorStateType	256
5.3.2.1.24. EthSwt_StateType	257
5.3.2.1.25. EthTrcv_BaudRateType	257
5.3.2.1.26. EthTrcv_CableDiagResultType	257
5.3.2.1.27. EthTrcv_DuplexModeType	257
5.3.2.1.28. EthTrcv_LinkStateType	257
5.3.2.1.29. EthTrcv_ModeType	257
5.3.2.1.30. EthTrcv_PhysicalLoopbackModeType	258
5.3.2.1.31. EthTrcv_PhysicalTestModeType	258
5.3.2.1.32. EthTrcv_PhysicalTxModeType	258
5.3.2.1.33. EthTrcv_StateType	258
5.3.2.1.34. EthTrcv_WakeupModeType	258
5.3.2.1.35. EthTrcv_WakeupReasonType	258
5.3.2.1.36. Eth_BufListType	258
5.3.2.1.37. Eth_CounterType	259
5.3.2.1.38. Eth_DataType	260
5.3.2.1.39. Eth_EtherStatsType	260
5.3.2.1.40. Eth_FilterActionType	261
5.3.2.1.41. Eth_FrameType	261
5.3.2.1.42. Eth_MacVlanType	262
5.3.2.1.43. Eth_ModeType	262
5.3.2.1.44. Eth_RateRatioType	262
5.3.2.1.45. Eth_RetransmitInfoType	262
5.3.2.1.46. Eth_ReturnType	262
5.3.2.1.47. Eth_RxStatsType	263
5.3.2.1.48. Eth_RxStatusType	263
5.3.2.1.49. Eth_StateType	263
5.3.2.1.50. Eth_TimeIntDiffType	264
5.3.2.1.51. Eth_TimeStampQualType	264
5.3.2.1.52. Eth_TimeStampType	264
5.3.2.1.53. Eth_TxErrorCounterValuesType	264
5.3.2.1.54. Eth_TxStatsType	265



5.3.2.1.55. FrTrcv_TrsvModeType	265
5.3.2.1.56. FrTrcv_TrsvWUReasonType	265
5.3.2.1.57. Fr_ChannelType	265
5.3.2.1.58. Fr_ErrorModeType	266
5.3.2.1.59. Fr_MTSStatusType	266
5.3.2.1.60. Fr_OffsetCorrectionType	266
5.3.2.1.61. Fr_POCStateType	267
5.3.2.1.62. Fr_POCStatusType	267
5.3.2.1.63. Fr_RateCorrectionType	268
5.3.2.1.64. Fr_RxLPduStatusType	268
5.3.2.1.65. Fr_SlotModeType	268
5.3.2.1.66. Fr_StartupStateType	268
5.3.2.1.67. Fr_SyncStateType	269
5.3.2.1.68. Fr_TxLPduStatusType	269
5.3.2.1.69. Fr_WakeupStatusType	269
5.3.2.1.70. IcomConfigIdType	270
5.3.2.1.71. IcomSwitch_ErrorType	270
5.3.2.1.72. LinTrcv_TrsvModeType	270
5.3.2.1.73. LinTrcv_TrsvWakeuModeType	271
5.3.2.1.74. LinTrcv_TrsvWakeuReasonType	271
5.3.2.1.75. Lin_FrameCsModelType	272
5.3.2.1.76. Lin_FrameDIType	272
5.3.2.1.77. Lin_FramePidType	272
5.3.2.1.78. Lin_FrameResponseType	272
5.3.2.1.79. Lin_PduType	272
5.3.2.1.80. Lin_StatusType	273
5.3.2.1.81. NetworkHandleType	274
5.3.2.1.82. NotifResultType	274
5.3.2.1.83. PNCHandleType	274
5.3.2.1.84. PduldtType	275
5.3.2.1.85. PduInfoType	275
5.3.2.1.86. PduLengthType	275
5.3.2.1.87. RetryInfoType	275
5.3.2.1.88. SoAd_MeasurementIdxType	276
5.3.2.1.89. StatusType	276
5.3.2.1.90. Std_ReturnType	276
5.3.2.1.91. Std_VersionInfoType	276
5.3.2.1.92. TPPParameterType	277
5.3.2.1.93. TS_CfgOffsetType	277
5.3.2.1.94. TS_MaxAlignedType	277
5.3.2.1.95. TS_VarOffsetType	277
5.3.2.1.96. Tcplp_MeasurementIdxType	278



5.3.2.1.97. TpDataStateType	278
5.3.2.1.98. VoidPtr	279
5.3.2.1.99. uint32	279
5.3.2.1.100. usize	279
5.3.2.2. Macro constants	279
5.3.2.2.1. ADD_TO_FILTER	279
5.3.2.2.2. BASE_DBG_ENABLE	280
5.3.2.2.3. BASE_PDUR_CONFIG_PTR	280
5.3.2.2.4. BASE_PDUR_ENABLED	280
5.3.2.2.5. BASE_PDUR_HEADER	280
5.3.2.2.6. BOOLEAN_C	280
5.3.2.2.7. BUSTRCV_E_ERROR	281
5.3.2.2.8. BUSTRCV_OK	281
5.3.2.2.9. Base_CustomStdIncludeFiles	281
5.3.2.2.10. COMPILER_AR_RELEASE_MAJOR_VERSION	281
5.3.2.2.11. COMPILER_AR_RELEASE_MINOR_VERSION	281
5.3.2.2.12. COMPILER_AR_RELEASE_REVISION_VERSION	282
5.3.2.2.13. COMPILER_MODULE_ID	282
5.3.2.2.14. COMPILER_SW_MAJOR_VERSION	282
5.3.2.2.15. COMPILER_SW_MINOR_VERSION	282
5.3.2.2.16. COMPILER_SW_PATCH_VERSION	282
5.3.2.2.17. COMPILER_VENDOR_ID	282
5.3.2.2.18. COMSTACK_BF_START	283
5.3.2.2.19. COMSTACK_BYTE_MIRROR16	283
5.3.2.2.20. COMSTACK_BYTE_MIRROR32	283
5.3.2.2.21. COMSTACK_BYTE_REPLICATOR32	283
5.3.2.2.22. COMSTACK_CT_BYTE_MIRROR16	283
5.3.2.2.23. COMSTACK_CT_BYTE_MIRROR32	284
5.3.2.2.24. COMSTACK_CT_HTON_UINT16	284
5.3.2.2.25. COMSTACK_CT_HTON_UINT32	284
5.3.2.2.26. COMSTACK_CT_NTOH_UINT16	284
5.3.2.2.27. COMSTACK_CT_NTOH_UINT32	284
5.3.2.2.28. COMSTACK_GET16	285
5.3.2.2.29. COMSTACK_GET32	285
5.3.2.2.30. COMSTACK_GETCPY32	285
5.3.2.2.31. COMSTACK_HAMMING_WEIGHT32	285
5.3.2.2.32. COMSTACK_HTON_UINT16	286
5.3.2.2.33. COMSTACK_HTON_UINT32	286
5.3.2.2.34. COMSTACK_ISA_WIDTH	286
5.3.2.2.35. COMSTACK_LOWEST_BIT32	286
5.3.2.2.36. COMSTACK_NTOH_UINT16	286
5.3.2.2.37. COMSTACK_NTOH_UINT32	287



5.3.2.2.38. COMSTACK_SET16	287
5.3.2.2.39. COMSTACK_SET32	287
5.3.2.2.40. COMSTACK_SETCPY32	287
5.3.2.2.41. COMTYPE_AR_RELEASE_MAJOR_VERSION	287
5.3.2.2.42. COMTYPE_AR_RELEASE_MINOR_VERSION	288
5.3.2.2.43. COMTYPE_AR_RELEASE_REVISION_VERSION	288
5.3.2.2.44. COMTYPE_MODULE_ID	288
5.3.2.2.45. COMTYPE_SW_MAJOR_VERSION	288
5.3.2.2.46. COMTYPE_SW_MINOR_VERSION	288
5.3.2.2.47. COMTYPE_SW_PATCH_VERSION	288
5.3.2.2.48. COMTYPE_VENDOR_ID	289
5.3.2.2.49. CONSTDEF	289
5.3.2.2.50. CPU_BIT_ORDER	289
5.3.2.2.51. CPU_BYTE_ORDER	289
5.3.2.2.52. CPU_TYPE	290
5.3.2.2.53. CPU_TYPE_16	290
5.3.2.2.54. CPU_TYPE_32	290
5.3.2.2.55. CPU_TYPE_64	290
5.3.2.2.56. CPU_TYPE_8	291
5.3.2.2.57. ETHIF_MEAS_ALL	291
5.3.2.2.58. ETHIF_MEAS_DROP_CRTLIDX	291
5.3.2.2.59. ETHSWT_MACLEARING_HWDISABLED	291
5.3.2.2.60. ETHSWT_MACLEARING_HWENABLED	291
5.3.2.2.61. ETHSWT_MACLEARING_SWENABLED	291
5.3.2.2.62. ETHSWT_STATE_ACTIVE	292
5.3.2.2.63. ETHSWT_STATE_INIT	292
5.3.2.2.64. ETHSWT_STATE_UNINIT	292
5.3.2.2.65. ETHTRCV_BAUD_RATE_1000MBIT	292
5.3.2.2.66. ETHTRCV_BAUD_RATE_100MBIT	292
5.3.2.2.67. ETHTRCV_BAUD_RATE_10MBIT	292
5.3.2.2.68. ETHTRCV_CABLEDIAG_ERROR	292
5.3.2.2.69. ETHTRCV_CABLEDIAG_OK	293
5.3.2.2.70. ETHTRCV_CABLEDIAG_OPEN	293
5.3.2.2.71. ETHTRCV_CABLEDIAG_PENDING	293
5.3.2.2.72. ETHTRCV_CABLEDIAG_SHORT	293
5.3.2.2.73. ETHTRCV_CABLEDIAG_WRONG_POLARITY	293
5.3.2.2.74. ETHTRCV_DUPLEX_MODE_FULL	293
5.3.2.2.75. ETHTRCV_DUPLEX_MODE_HALF	294
5.3.2.2.76. ETHTRCV_LINK_STATE_ACTIVE	294
5.3.2.2.77. ETHTRCV_LINK_STATE_DOWN	294
5.3.2.2.78. ETHTRCV_MODE_ACTIVE	294
5.3.2.2.79. ETHTRCV_MODE_DOWN	294



5.3.2.2.80. ETHTRCV_PHYLOOPBACK_EXTERNAL	294
5.3.2.2.81. ETHTRCV_PHYLOOPBACK_INTERNAL	294
5.3.2.2.82. ETHTRCV_PHYLOOPBACK_NONE	295
5.3.2.2.83. ETHTRCV_PHYLOOPBACK_REMOTE	295
5.3.2.2.84. ETHTRCV_PHYTESTMODE_1	295
5.3.2.2.85. ETHTRCV_PHYTESTMODE_2	295
5.3.2.2.86. ETHTRCV_PHYTESTMODE_3	295
5.3.2.2.87. ETHTRCV_PHYTESTMODE_4	295
5.3.2.2.88. ETHTRCV_PHYTESTMODE_5	296
5.3.2.2.89. ETHTRCV_PHYTESTMODE_NONE	296
5.3.2.2.90. ETHTRCV_PHYTXMODE_NORMAL	296
5.3.2.2.91. ETHTRCV_PHYTXMODE_SCRAMBLER_OFF	296
5.3.2.2.92. ETHTRCV_PHYTXMODE_TX_OFF	296
5.3.2.2.93. ETHTRCV_STATE_ACTIVE	296
5.3.2.2.94. ETHTRCV_STATE_INIT	297
5.3.2.2.95. ETHTRCV_STATE_UNINIT	297
5.3.2.2.96. ETHTRCV_WUM_CLEAR	297
5.3.2.2.97. ETHTRCV_WUM_DISABLE	297
5.3.2.2.98. ETHTRCV_WUM_ENABLE	297
5.3.2.2.99. ETHTRCV_WUR_BUS	297
5.3.2.2.100. ETHTRCV_WUR_GENERAL	298
5.3.2.2.101. ETHTRCV_WUR_INTERNAL	298
5.3.2.2.102. ETHTRCV_WUR_NONE	298
5.3.2.2.103. ETHTRCV_WUR_PIN	298
5.3.2.2.104. ETHTRCV_WUR_POWER_ON	298
5.3.2.2.105. ETHTRCV_WUR_RESET	298
5.3.2.2.106. ETHTRCV_WUR_SYSERR	299
5.3.2.2.107. ETH_ADD_TO_FILTER	299
5.3.2.2.108. ETH_BUFLISTTYPE_DEF	299
5.3.2.2.109. ETH_E_NOT_OK	299
5.3.2.2.110. ETH_E_NO_ACCESS	299
5.3.2.2.111. ETH_INVALID	299
5.3.2.2.112. ETH_MODE_ACTIVE	300
5.3.2.2.113. ETH_MODE_DOWN	300
5.3.2.2.114. ETH_NOT_RECEIVED	300
5.3.2.2.115. ETH_OK	300
5.3.2.2.116. ETH RECEIVED	300
5.3.2.2.117. ETH RECEIVED_FRAMES_LOST	300
5.3.2.2.118. ETH RECEIVED_MORE_DATA_AVAILABLE	300
5.3.2.2.119. ETH REMOVE_FROM_FILTER	301
5.3.2.2.120. ETH RETRANSMITINFOTYPE_DEF	301
5.3.2.2.121. ETH STATE_ACTIVE	301



5.3.2.2.122. ETH_STATE_INIT	301
5.3.2.2.123. ETH_STATE_UNINIT	301
5.3.2.2.124. ETH_UNCERTAIN	301
5.3.2.2.125. ETH_VALID	302
5.3.2.2.126. E_NOT_OK	302
5.3.2.2.127. E_NO_DATA	302
5.3.2.2.128. E_OK	302
5.3.2.2.129. E_SAFETY_HARD_RUNTIMEERROR	302
5.3.2.2.130. E_SAFETY_INIT_ERR	302
5.3.2.2.131. E_SAFETY_INIT_NND	303
5.3.2.2.132. E_SAFETY_INIT_OK	303
5.3.2.2.133. E_SAFETY_INIT REP	303
5.3.2.2.134. E_SAFETY_INIT_SEQ	303
5.3.2.2.135. E_SAFETY_INVALID_ERR	303
5.3.2.2.136. E_SAFETY_INVALID_NND	303
5.3.2.2.137. E_SAFETY_INVALID_OK	304
5.3.2.2.138. E_SAFETY_INVALID REP	304
5.3.2.2.139. E_SAFETY_INVALID_SEQ	304
5.3.2.2.140. E_SAFETY_NODATA_ERR	304
5.3.2.2.141. E_SAFETY_NODATA_NND	304
5.3.2.2.142. E_SAFETY_NODATA_OK	304
5.3.2.2.143. E_SAFETY_NODATA REP	304
5.3.2.2.144. E_SAFETY_NODATA_SEQ	305
5.3.2.2.145. E_SAFETY_SOFT_RUNTIMEERROR	305
5.3.2.2.146. E_SAFETY_VALID_ERR	305
5.3.2.2.147. E_SAFETY_VALID_NND	305
5.3.2.2.148. E_SAFETY_VALID REP	305
5.3.2.2.149. E_SAFETY_VALID_SEQ	305
5.3.2.2.150. E_SEC_NOT_AUTH	306
5.3.2.2.151. E_SEC_NOT_FRESH	306
5.3.2.2.152. E_SER_GENERIC_ERROR	306
5.3.2.2.153. E_SER_MALFORMED_MESSAGE	306
5.3.2.2.154. E_SER_WRONG_INTERFACE_VERSION	306
5.3.2.2.155. E_SER_WRONG_MESSAGE_TYPE	306
5.3.2.2.156. E_SER_WRONG_PROTOCOL_VERSION	307
5.3.2.2.157. FALSE	307
5.3.2.2.158. FR_CIDX_GCOLDSTARTATTEMPTS	307
5.3.2.2.159. FR_CIDX_GCYCLECOUNTMAX	307
5.3.2.2.160. FR_CIDX_GACTIONPOINTOFFSET	307
5.3.2.2.161. FR_CIDX_GDBIT	307
5.3.2.2.162. FR_CIDX_GDCASRXLOWMAX	308
5.3.2.2.163. FR_CIDX_GDCYCLE	308



5.3.2.2.164. FR_CIDX_GDDYNAMICSLOTIDLEPHASE	308
5.3.2.2.165. FR_CIDX_GDIGNOREAFTERTX	308
5.3.2.2.166. FR_CIDX_GDMACROTICK	308
5.3.2.2.167. FR_CIDX_GDMINISLOT	308
5.3.2.2.168. FR_CIDX_GDMINISLOTACTIONPOINTOFFSET	308
5.3.2.2.169. FR_CIDX_GDNIT	309
5.3.2.2.170. FR_CIDX_GDSAMPLECLOCKPERIOD	309
5.3.2.2.171. FR_CIDX_GDSTATICSLOT	309
5.3.2.2.172. FR_CIDX_GDSYMBOLWINDOW	309
5.3.2.2.173. FR_CIDX_GDSYMBOLWINDOWACTIONPOINTOFFSET	309
5.3.2.2.174. FR_CIDX_GDTSSTRANSMITTER	309
5.3.2.2.175. FR_CIDX_GDWAKEUPRXIDLE	310
5.3.2.2.176. FR_CIDX_GDWAKEUPRXLOW	310
5.3.2.2.177. FR_CIDX_GDWAKEUPRXWINDOW	310
5.3.2.2.178. FR_CIDX_GDWAKEUPTXACTIVE	310
5.3.2.2.179. FR_CIDX_GDWAKEUPTXIDLE	310
5.3.2.2.180. FR_CIDX_GLISTENNOISE	310
5.3.2.2.181. FR_CIDX_GMACROPERCYCLE	310
5.3.2.2.182. FR_CIDX_GMAXWITHOUTCLOCKCORRECTFATAL	311
5.3.2.2.183. FR_CIDX_GMAXWITHOUTCLOCKCORRECTPASSIVE	311
5.3.2.2.184. FR_CIDX_GNETWORKMANAGEMENTVECTORLENGTH	311
5.3.2.2.185. FR_CIDX_GNUMBEROFGMINISLOTS	311
5.3.2.2.186. FR_CIDX_GNUMBEROFGSTATICSLOTS	311
5.3.2.2.187. FR_CIDX_GPAYLOADLENGTHSTATIC	311
5.3.2.2.188. FR_CIDX_GSYNCFRAMEIDCOUNTMAX	312
5.3.2.2.189. FR_CIDX_PALLOWHALTDUETOCLOCK	312
5.3.2.2.190. FR_CIDX_PALLOWPASSIVETOACTIVE	312
5.3.2.2.191. FR_CIDX_PCHANNELS	312
5.3.2.2.192. FR_CIDX_PCLUSTERDRIFTDAMPING	312
5.3.2.2.193. FR_CIDX_PDACCEPTEDSTARTUPRANGE	312
5.3.2.2.194. FR_CIDX_PDECODINGCORRECTION	312
5.3.2.2.195. FR_CIDX_PDELAYCOMPENSATIONA	313
5.3.2.2.196. FR_CIDX_PDELAYCOMPENSATIONB	313
5.3.2.2.197. FR_CIDX_PDLISTENTIMEOUT	313
5.3.2.2.198. FR_CIDX_PDMICROTICK	313
5.3.2.2.199. FR_CIDX_PEXTERNALSYNC	313
5.3.2.2.200. FR_CIDX_PFallbackInternal	313
5.3.2.2.201. FR_CIDX_PKEYSLOTID	314
5.3.2.2.202. FR_CIDX_PKEYSLOTONLYENABLED	314
5.3.2.2.203. FR_CIDX_PKEYSLOTUSEDFORSTARTUP	314
5.3.2.2.204. FR_CIDX_PKEYSLOTUSEDFORSYNC	314
5.3.2.2.205. FR_CIDX_PLATESTTX	314



5.3.2.2.206. FR_CIDX_PMACROINITIALOFFSETA	314
5.3.2.2.207. FR_CIDX_PMACROINITIALOFFSETB	314
5.3.2.2.208. FR_CIDX_PMICROINITIALOFFSETA	315
5.3.2.2.209. FR_CIDX_PMICROINITIALOFFSETB	315
5.3.2.2.210. FR_CIDX_PMICROPERCYCLE	315
5.3.2.2.211. FR_CIDX_PNMVECTOREARLYUPDATE	315
5.3.2.2.212. FR_CIDX_POFFSETCORRECTIONOUT	315
5.3.2.2.213. FR_CIDX_POFFSETCORRECTIONSTART	315
5.3.2.2.214. FR_CIDX_PPAYLOADLENGTHDYNMAX	316
5.3.2.2.215. FR_CIDX_PRATECORRECTIONOUT	316
5.3.2.2.216. FR_CIDX_PSAMPLESPERMICROTICK	316
5.3.2.2.217. FR_CIDX_PSECONDKEYSLOTID	316
5.3.2.2.218. FR_CIDX_PTWOKEYSLOTMODE	316
5.3.2.2.219. FR_CIDX_PWAKEUPCHANNEL	316
5.3.2.2.220. FR_CIDX_PWAKEUPPATTERN	316
5.3.2.2.221. FR_SLOTMODE_SINGLE	317
5.3.2.2.222. HIGH_BYTE_FIRST	317
5.3.2.2.223. LOW_BYTE_FIRST	317
5.3.2.2.224. LSB_FIRST	317
5.3.2.2.225. MSB_FIRST	317
5.3.2.2.226. NTFRSLT_E_ABORT	317
5.3.2.2.227. NTFRSLT_E_CANCELATION_NOT_OK	318
5.3.2.2.228. NTFRSLT_E_CANCELATION_OK	318
5.3.2.2.229. NTFRSLT_E_INVALID_FS	318
5.3.2.2.230. NTFRSLT_E_NOT_OK	318
5.3.2.2.231. NTFRSLT_E_NO_BUFFER	318
5.3.2.2.232. NTFRSLT_E_PARAMETER_NOT_OK	319
5.3.2.2.233. NTFRSLT_E_RX_ON	319
5.3.2.2.234. NTFRSLT_E_TIMEOUT_A	319
5.3.2.2.235. NTFRSLT_E_TIMEOUT_BS	319
5.3.2.2.236. NTFRSLT_E_TIMEOUT_CR	320
5.3.2.2.237. NTFRSLT_E_UNEXP_PDU	320
5.3.2.2.238. NTFRSLT_E_VALUE_NOT_OK	320
5.3.2.2.239. NTFRSLT_E_WFT_OVRN	320
5.3.2.2.240. NTFRSLT_E_WRONG_SN	320
5.3.2.2.241. NTFRSLT_OK	321
5.3.2.2.242. NTFRSLT_PARAMETER_OK	321
5.3.2.2.243. NULL_PTR	321
5.3.2.2.244. PLATFORM_AR_RELEASE_MAJOR_VERSION	321
5.3.2.2.245. PLATFORM_AR_RELEASE_MINOR_VERSION	321
5.3.2.2.246. PLATFORM_AR_RELEASE_REVISION_VERSION	322
5.3.2.2.247. PLATFORM_MODULE_ID	322



5.3.2.2.248. PLATFORM_SW_MAJOR_VERSION	322
5.3.2.2.249. PLATFORM_SW_MINOR_VERSION	322
5.3.2.2.250. PLATFORM_SW_PATCH_VERSION	322
5.3.2.2.251. PLATFORM_VENDOR_API_INFIX	322
5.3.2.2.252. PLATFORM_VENDOR_ID	323
5.3.2.2.253. PORT_MIRROR_DISABLED	323
5.3.2.2.254. PORT_MIRROR_ENABLED	323
5.3.2.2.255. REMOVE_FROM_FILTER	323
5.3.2.2.256. SINT16_C	323
5.3.2.2.257. SINT32_C	323
5.3.2.2.258. SINT8_C	324
5.3.2.2.259. SOAD_MEAS_ALL	324
5.3.2.2.260. SOAD_MEAS_DROP_TCP	324
5.3.2.2.261. SOAD_MEAS_DROP_UDP	324
5.3.2.2.262. STATIC	324
5.3.2.2.263. STD_ACTIVE	325
5.3.2.2.264. STD_AR_RELEASE_MAJOR_VERSION	325
5.3.2.2.265. STD_AR_RELEASE_MINOR_VERSION	325
5.3.2.2.266. STD_AR_RELEASE_REVISION_VERSION	325
5.3.2.2.267. STD_HIGH	325
5.3.2.2.268. STD_IDLE	325
5.3.2.2.269. STD_LOW	325
5.3.2.2.270. STD_MODULE_ID	326
5.3.2.2.271. STD_OFF	326
5.3.2.2.272. STD_ON	326
5.3.2.2.273. STD_SW_MAJOR_VERSION	326
5.3.2.2.274. STD_SW_MINOR_VERSION	326
5.3.2.2.275. STD_SW_PATCH_VERSION	326
5.3.2.2.276. STD_VENDOR_ID	327
5.3.2.2.277. STD_VERSION_INFO_TYPE_DEFINED	327
5.3.2.2.278. TCPIP_MEAS_ALL	327
5.3.2.2.279. TCPIP_MEAS_DROP_ARP	327
5.3.2.2.280. TCPIP_MEAS_DROP_ICMPV6	327
5.3.2.2.281. TCPIP_MEAS_DROP_IPV4	327
5.3.2.2.282. TCPIP_MEAS_DROP_IPV6	328
5.3.2.2.283. TCPIP_MEAS_DROP_TCP	328
5.3.2.2.284. TCPIP_MEAS_DROP_UDP	328
5.3.2.2.285. TCPIP_MEAS_REPLACE_ARP	328
5.3.2.2.286. TPPARAMETER_BC	328
5.3.2.2.287. TPPARAMETER_BS	328
5.3.2.2.288. TPPARAMETER_STMIN	329
5.3.2.2.289. TPPARAMTERE_BS	329



5.3.2.2.290. TRUE	329
5.3.2.2.291. TS_ADDITIONAL_PADDING_BYTES_INSERTED	329
5.3.2.2.292. TS_ALIGNMENT_TS_MaxAlignedType	329
5.3.2.2.293. TS_ALIGNMENT_struct	330
5.3.2.2.294. TS_ALIGNMENT_struct_0	330
5.3.2.2.295. TS_ALIGNMENT_struct_NUM_THRESHOLDS	330
5.3.2.2.296. TS_ALIGNMENT_struct_THRESHOLD_0	330
5.3.2.2.297. TS_ALIGNMENT_uint32	330
5.3.2.2.298. TS_ALIGNOF	331
5.3.2.2.299. TS_ARCH_DERIVATE	331
5.3.2.2.300. TS_ARCH_FAMILY	331
5.3.2.2.301. TS_ARM	331
5.3.2.2.302. TS_ARM64	331
5.3.2.2.303. TS_ARRAYALIGNOF	331
5.3.2.2.304. TS_AtomicAssign16	332
5.3.2.2.305. TS_AtomicAssign32	332
5.3.2.2.306. TS_AtomicAssign8	332
5.3.2.2.307. TS_AtomicAssignGeneric	333
5.3.2.2.308. TS_C16X	333
5.3.2.2.309. TS_CAST	333
5.3.2.2.310. TS_CFG_OFFSET_TYPE_MAX	334
5.3.2.2.311. TS_CFG_OFFSET_TYPE_MIN	334
5.3.2.2.312. TS_CHECKEDGETCFG	334
5.3.2.2.313. TS_CHECKEDGETCONSTCFG	335
5.3.2.2.314. TS_CHECKEDGETCONSTVAR	335
5.3.2.2.315. TS_CHECKEDGETVAR	335
5.3.2.2.316. TS_CONSTREF2CFG	335
5.3.2.2.317. TS_CONSTREF2VAR	336
5.3.2.2.318. TS_CORTEXM	336
5.3.2.2.319. TS_DEPRECATED_FUNCTION	336
5.3.2.2.320. TS_DEPRECATED_MACRO	336
5.3.2.2.321. TS_DEPRECATED_TYPEDEF	337
5.3.2.2.322. TS_DEPRECATED_VARIABLE	337
5.3.2.2.323. TS_DOPRAMA	338
5.3.2.2.324. TS_DSPIC33	338
5.3.2.2.325. TS_DefMaxAlignedByteArray	338
5.3.2.2.326. TS_DefMaxAlignedByteArray_HIp	338
5.3.2.2.327. TS_EASYCAN	338
5.3.2.2.328. TS_EBX1XX	339
5.3.2.2.329. TS_EXCALIBUR	339
5.3.2.2.330. TS_F2MC16L	339
5.3.2.2.331. TS_FATBOOL	339



5.3.2.2.332. TS_FATBOOL_LT	339
5.3.2.2.333. TS_FCR4	339
5.3.2.2.334. TS_GetBit	340
5.3.2.2.335. TS_GetBitGroup	340
5.3.2.2.336. TS_H8	340
5.3.2.2.337. TS_HC08	340
5.3.2.2.338. TS_HC12	340
5.3.2.2.339. TS_IsBitSet	341
5.3.2.2.340. TS_LINUX	341
5.3.2.2.341. TS_M32C	341
5.3.2.2.342. TS_M32R	341
5.3.2.2.343. TS_MAKENULLREF2CFG	341
5.3.2.2.344. TS_MAKENULLREF2VAR	342
5.3.2.2.345. TS_MAKEREF2CFG	342
5.3.2.2.346. TS_MAKEREF2VAR	342
5.3.2.2.347. TS_MB91	342
5.3.2.2.348. TS_MB96	342
5.3.2.2.349. TS_MEMBZERO_CUSTOM_OVERRIDE	342
5.3.2.2.350. TS_MEMCMP_CUSTOM_OVERRIDE	343
5.3.2.2.351. TS_MEMCPY_CUSTOM_OVERRIDE	343
5.3.2.2.352. TS_MEMSET_CUSTOM_OVERRIDE	343
5.3.2.2.353. TS_MERGED_COMPILE	344
5.3.2.2.354. TS_MOD_PRIV_DECL	344
5.3.2.2.355. TS_MOD_PRIV_DEFN	344
5.3.2.2.356. TS_MPC551X	344
5.3.2.2.357. TS_MemBZero	344
5.3.2.2.358. TS_MemCmp	345
5.3.2.2.359. TS_MemCopy	345
5.3.2.2.360. TS_MemSet	345
5.3.2.2.361. TS_NEC78K	345
5.3.2.2.362. TS_NIOS2	345
5.3.2.2.363. TS_OFFSETOF	345
5.3.2.2.364. TS_PA	346
5.3.2.2.365. TS_PARAM_UNUSED	346
5.3.2.2.366. TS_PIC24H	346
5.3.2.2.367. TS_PIKEOS	346
5.3.2.2.368. TS_PREVENTEMPTYTRANSLATIONUNIT	346
5.3.2.2.369. TS_PROD_ERR_DISABLE	346
5.3.2.2.370. TS_PROD_ERR REP_TO DEM	347
5.3.2.2.371. TS_PROD_ERR REP_TO DET	347
5.3.2.2.372. TS_R32C	347
5.3.2.2.373. TS_REF2CFG	347



5.3.2.2.374. TS_REF2VAR	347
5.3.2.2.375. TS_RH850	348
5.3.2.2.376. TS_RL78	348
5.3.2.2.377. TS_RZ	348
5.3.2.2.378. TS_S12X	348
5.3.2.2.379. TS_S12Z	348
5.3.2.2.380. TS_SAF7780	348
5.3.2.2.381. TS_SELECT	349
5.3.2.2.382. TS_SH2	349
5.3.2.2.383. TS_SH4	349
5.3.2.2.384. TS_SIG_ARRAY_ALIGNMENT_MASK	349
5.3.2.2.385. TS_SIG_ARRAY_ALIGNMENT_SHIFT	349
5.3.2.2.386. TS_SIG_CHECK_ALIGNMENT_ARRAY	349
5.3.2.2.387. TS_SIG_CHECK_ALIGNMENT_SINT16	350
5.3.2.2.388. TS_SIG_CHECK_ALIGNMENT_SINT32	350
5.3.2.2.389. TS_SIG_CHECK_ALIGNMENT_SINT8	350
5.3.2.2.390. TS_SIG_CHECK_ALIGNMENT_STRUCT	350
5.3.2.2.391. TS_SIG_CHECK_ALIGNMENT_UINT16	351
5.3.2.2.392. TS_SIG_CHECK_ALIGNMENT_UINT32	351
5.3.2.2.393. TS_SIG_CHECK_ALIGNMENT_UINT8	351
5.3.2.2.394. TS_SIG_CHECK_ENDIANESS	351
5.3.2.2.395. TS_SIG_GET_ALIGNMENT	351
5.3.2.2.396. TS_SIG_GET_ALIGNMENT_HLP	351
5.3.2.2.397. TS_SIG_GET_ALIGNMENT_HLP_HLP	352
5.3.2.2.398. TS_SIG_SINT16_ALIGNMENT_MASK	352
5.3.2.2.399. TS_SIG_SINT16_ALIGNMENT_SHIFT	352
5.3.2.2.400. TS_SIG_SINT32_ALIGNMENT_MASK	352
5.3.2.2.401. TS_SIG_SINT32_ALIGNMENT_SHIFT	352
5.3.2.2.402. TS_SIG_SINT8_ALIGNMENT_MASK	352
5.3.2.2.403. TS_SIG_SINT8_ALIGNMENT_SHIFT	353
5.3.2.2.404. TS_SIG_STRUCT_ALIGNMENT_MASK	353
5.3.2.2.405. TS_SIG_STRUCT_ALIGNMENT_SHIFT	353
5.3.2.2.406. TS_SIG_UINT16_ALIGNMENT_MASK	353
5.3.2.2.407. TS_SIG_UINT16_ALIGNMENT_SHIFT	353
5.3.2.2.408. TS_SIG_UINT32_ALIGNMENT_MASK	353
5.3.2.2.409. TS_SIG_UINT32_ALIGNMENT_SHIFT	354
5.3.2.2.410. TS_SIG_UINT8_ALIGNMENT_MASK	354
5.3.2.2.411. TS_SIG_UINT8_ALIGNMENT_SHIFT	354
5.3.2.2.412. TS_SIZE_PdulType	354
5.3.2.2.413. TS_SIZE_PduLengthType	354
5.3.2.2.414. TS_SIZE_TS_MaxAlignedType	354
5.3.2.2.415. TS_SIZE_uint32	355



5.3.2.2.416. TS_ST30	355
5.3.2.2.417. TS_STATIC_ASSERT	355
5.3.2.2.418. TS_STATIC_ASSERT_EVAL	356
5.3.2.2.419. TS_STATIC_ASSERT_JOIN	356
5.3.2.2.420. TS_STM8A	356
5.3.2.2.421. TS_TMS320	356
5.3.2.2.422. TS_TMS470	357
5.3.2.2.423. TS_TRICORE	357
5.3.2.2.424. TS_UNCHECKEDGETCFG	357
5.3.2.2.425. TS_UNCHECKEDGETCONSTCFG	357
5.3.2.2.426. TS_UNCHECKEDGETCONSTVAR	357
5.3.2.2.427. TS_UNCHECKEDGETVAR	358
5.3.2.2.428. TS_V850	358
5.3.2.2.429. TS_VAR_OFFSET_TYPE_MAX	358
5.3.2.2.430. TS_VAR_OFFSET_TYPE_MIN	358
5.3.2.2.431. TS_VENDOR_ID_3SOFT	359
5.3.2.2.432. TS_VENDOR_ID_EB	359
5.3.2.2.433. TS_WINDOWS	359
5.3.2.2.434. TS_XC16X	359
5.3.2.2.435. TS_XC2000	359
5.3.2.2.436. TS_XPC56XX	359
5.3.2.2.437. UINT16_C	360
5.3.2.2.438. UINT32_C	360
5.3.2.2.439. UINT8_C	360
5.3.2.2.440. UNIQUE	360
5.3.2.2.441. UNIQUETYPEDEF	360
5.3.2.2.442. USIZE_C	360
5.3.2.2.443. _STATIC_	360
5.3.2.2.444. include_Atomics_TSPlatforms	361
5.3.2.2.445. include_Platforms_TSPlatforms	361
5.3.2.3. Functions	361
5.3.2.3.1. ComStack_FindNextOne	361
5.3.2.3.2. TS_MemBZero16_NoCheck	361
5.3.2.3.3. TS_MemBZero32_NoCheck	362
5.3.2.3.4. TS_MemBZero64	362
5.3.2.3.5. TS_MemCmp16_NoCheck	362
5.3.2.3.6. TS_MemCmp32_NoCheck	363
5.3.2.3.7. TS_MemCmp64	363
5.3.2.3.8. TS_MemCpy16_NoCheck	364
5.3.2.3.9. TS_MemCpy32_NoCheck	364
5.3.2.3.10. TS_MemCpy64	364
5.3.2.3.11. TS_MemMove	365



5.3.2.3.12. TS_MemSet16_NoCheck	365
5.3.2.3.13. TS_MemSet32_NoCheck	365
5.3.2.3.14. TS_MemSet64	366
5.3.2.3.15. TS_PlatformSigIsValid	366
5.3.3. Integration notes	366
5.3.3.1. Exclusive areas	366
5.3.3.2. Production errors	366
5.3.3.3. Memory mapping	366
5.3.3.4. Integration requirements	367
5.3.3.5. Platform integration	367
5.3.3.5.1. Optimized memory copy function	367
5.3.3.5.2. Optimized memory set function	368
5.3.3.5.3. Optimized memory compare function	368
5.3.3.5.4. Optimized memory zeroing function	369
5.3.3.5.5. Low Level Primitives	369
5.4. Compiler	371
5.4.1. Configuration parameters	371
5.4.2. Application programming interface (API)	371
5.4.2.1. Macro constants	371
5.4.2.1.1. COMPILERCFG_EXTENSION_FILE	371
5.4.2.1.2. COMPILERCFG_EXTENSION_MCAL_FILE	371
5.4.2.1.3. CONST	371
5.4.2.1.4. CONSTP2CONST	372
5.4.2.1.5. CONSTP2FUNC	372
5.4.2.1.6. CONSTP2VAR	372
5.4.2.1.7. FUNC	372
5.4.2.1.8. FUNC_P2CONST	373
5.4.2.1.9. FUNC_P2VAR	373
5.4.2.1.10. LOCAL_INLINE	373
5.4.2.1.11. P2CONST	373
5.4.2.1.12. P2FUNC	373
5.4.2.1.13. P2VAR	374
5.4.2.1.14. VAR	374
5.4.2.1.15. _INLINE_	374
5.4.3. Integration notes	374
5.4.3.1. Exclusive areas	374
5.4.3.2. Production errors	375
5.4.3.3. Memory mapping	375
5.4.3.4. Integration requirements	375
5.5. Det	375
5.5.1. Configuration parameters	375
5.5.1.1. CommonPublishedInformation	376



5.5.1.2. DetGeneral	379
5.5.1.3. DetServiceAPI	382
5.5.1.4. DetNotification	384
5.5.1.5. DetDefensiveProgramming	386
5.5.1.6. SoftwareComponentList	388
5.5.1.7. InstanceIdList	389
5.5.1.8. PublishedInformation	390
5.5.2. Application programming interface (API)	390
5.5.2.1. Macro constants	390
5.5.2.1.1. DET_AR_RELEASE_MAJOR_VERSION	390
5.5.2.1.2. DET_AR_RELEASE_MINOR_VERSION	391
5.5.2.1.3. DET_AR_RELEASE_REVISION_VERSION	391
5.5.2.1.4. DET_E_INVARIANT_ASSERT_FAILED	391
5.5.2.1.5. DET_E_PARAM_POINTER	391
5.5.2.1.6. DET_E_POSTCONDITION_ASSERT_FAILED	391
5.5.2.1.7. DET_E_PRECONDITION_ASSERT_FAILED	391
5.5.2.1.8. DET_E_UNREACHABLE_CODE_ASSERT_FAILED	392
5.5.2.1.9. DET_INTERNAL_API_ID	392
5.5.2.1.10. DET_INVARIANT_ASSERT	392
5.5.2.1.11. DET_INVARIANT_ASSERT_PRINT	392
5.5.2.1.12. DET_MODULE_ID	392
5.5.2.1.13. DET_POSTCONDITION_ASSERT	393
5.5.2.1.14. DET_POSTCONDITION_ASSERT_PRINT	393
5.5.2.1.15. DET_PRECONDITION_ASSERT	393
5.5.2.1.16. DET_PRECONDITION_ASSERT_NO_EVAL	393
5.5.2.1.17. DET_PRECONDITION_ASSERT_PRINT	394
5.5.2.1.18. DET_SID_GET_VERSION_INFO	394
5.5.2.1.19. DET_SID_INIT	394
5.5.2.1.20. DET_SID_REPORT_ERROR	394
5.5.2.1.21. DET_SID_REPORT_RUNTIME_ERROR	394
5.5.2.1.22. DET_SID_REPORT_TRANSIENT_FAULT	395
5.5.2.1.23. DET_SID_START	395
5.5.2.1.24. DET_STATIC_ASSERT	395
5.5.2.1.25. DET_STATIC_ASSERT_JOIN	396
5.5.2.1.26. DET_STATIC_ASSERT_JOIN_HLP	396
5.5.2.1.27. DET_SW_MAJOR_VERSION	396
5.5.2.1.28. DET_SW_MINOR_VERSION	396
5.5.2.1.29. DET_SW_PATCH_VERSION	397
5.5.2.1.30. DET_UNREACHABLE_CODE_ASSERT	397
5.5.2.1.31. DET_UNREACHABLE_CODE_ASSERT_PRINT	397
5.5.2.1.32. DET_VENDOR_ID	397
5.5.2.1.33. Det_ReportError	397



5.5.2.1.34. Det_ReportRuntimeError	398
5.5.2.1.35. Det_ReportTransientFault	398
5.5.2.2. Objects	398
5.5.2.2.1. Det_ErrorIndex	398
5.5.2.2.2. Det_ErrorLost_DevelopmentError	398
5.5.2.2.3. Det_ErrorLost_RuntimeError	399
5.5.2.2.4. Det_FaultLost_TransientFault	399
5.5.2.2.5. Det_UsedSlots_DevelopmentError	399
5.5.2.2.6. Det_UsedSlots_RuntimeError	399
5.5.2.2.7. Det_UsedSlots_TransientFault	400
5.5.2.2.8. Det_WriteIndex_DevelopmentError	400
5.5.2.2.9. Det_WriteIndex_RuntimeError	400
5.5.2.2.10. Det_WriteIndex_TransientFault	400
5.5.2.3. Functions	401
5.5.2.3.1. Det_ASR32_ReportError	401
5.5.2.3.2. Det_ASR40_ReportError	401
5.5.2.3.3. Det_ASR43_ReportRuntimeError	402
5.5.2.3.4. Det_ASR43_ReportTransientFault	402
5.5.2.3.5. Det_GetVersionInfo	403
5.5.2.3.6. Det_Init	404
5.5.2.3.7. Det_InvariantAssertPrint	404
5.5.2.3.8. Det_PostconditionAssertPrint	404
5.5.2.3.9. Det_PreconditionAssertPrint	405
5.5.2.3.10. Det_Start	405
5.5.2.3.11. Det_UnreachableCodeAssertPrint	406
5.5.3. Integration notes	406
5.5.3.1. Exclusive areas	406
5.5.3.1.1. SCHM_DET_EXCLUSIVE_AREA_0	406
5.5.3.2. Production errors	407
5.5.3.3. Memory mapping	407
5.5.3.4. Integration requirements	407
5.5.3.4.1. Det.EB.IntReq.EB_INTREQ_Det_0001	407
5.5.3.4.2. Det.EB.IntReq.EB_INTREQ_Det_0002	407
5.5.3.4.3. Det.EB.IntReq.EB_INTREQ_Det_0003	408
5.5.3.4.4. Det.EB.IntReq.EB_INTREQ_Det_0004	408
5.6. EcuC	408
5.6.1. Configuration parameters	408
5.6.1.1. CommonPublishedInformation	409
5.6.1.2. EcucGeneral	412
5.6.1.3. EcucHardware	413
5.6.1.4. EcucCoreDefinition	413
5.6.1.5. EcucPartitionCollection	414



5.6.1.6. EcucPartition	414
5.6.1.7. EcucPartitionSoftwareComponentInstanceRef	416
5.6.1.8. EcucPduCollection	416
5.6.1.9. MetaDataType	418
5.6.1.10. MetaDataItem	418
5.6.1.11. Pdu	419
5.6.1.12. EcucPduDedicatedPartition	421
5.6.1.13. EcucPostBuildVariants	421
5.6.1.14. EcucVariationResolver	423
5.6.1.15. PublishedInformation	423
5.6.2. Application programming interface (API)	423
5.6.2.1. Macro constants	424
5.6.2.1.1. ECUC_COMPATIBILITY_VERSION_CHECK	424
5.6.2.1.2. ECUC_METADATA_ITEM_ADDRESS_EXTENSION_8	424
5.6.2.1.3. ECUC_METADATA_ITEM_CAN_ID_32	424
5.6.2.1.4. ECUC_METADATA_ITEM_ETHERNET_MAC_64	424
5.6.2.1.5. ECUC_METADATA_ITEM_LIN_NAD_8	424
5.6.2.1.6. ECUC_METADATA_ITEM_PAYLOAD_TYPE_16	424
5.6.2.1.7. ECUC_METADATA_ITEM_PRIORITY_8	425
5.6.2.1.8. ECUC_METADATA_ITEM_SOCKET_CONNECTION_ID_16	425
5.6.2.1.9. ECUC_METADATA_ITEM_SOURCE_ADDRESS_16	425
5.6.2.1.10. ECUC_METADATA_ITEM_TARGET_ADDRESS_16	425
5.6.2.2. Functions	425
5.6.2.2.1. EcuC_CopyRxData	425
5.6.2.2.2. EcuC_CopyRxDataDet	426
5.6.2.2.3. EcuC_CopyTxData	426
5.6.2.2.4. EcuC_GetMetaDataAddrExtension	427
5.6.2.2.5. EcuC_GetMetaDataCanId	428
5.6.2.2.6. EcuC_GetMetaDataEthMac	428
5.6.2.2.7. EcuC_GetMetaDataLinNad	429
5.6.2.2.8. EcuC_GetMetaDataPayloadType	429
5.6.2.2.9. EcuC_GetMetaDataPriority	430
5.6.2.2.10. EcuC_GetMetaDataSoConId	430
5.6.2.2.11. EcuC_GetMetaDataSourceAddr	431
5.6.2.2.12. EcuC_GetMetaDataTargetAddr	431
5.6.2.2.13. EcuC_InitRx	432
5.6.2.2.14. EcuC_InitTx	432
5.6.2.2.15. EcuC_RxBufferIsLocked	432
5.6.2.2.16. EcuC_SetMetaDataAddrExtension	433
5.6.2.2.17. EcuC_SetMetaDataCanId	433
5.6.2.2.18. EcuC_SetMetaDataEthMac	434
5.6.2.2.19. EcuC_SetMetaDataLinNad	434



5.6.2.2.20. EcuC_SetMetaDataType	435
5.6.2.2.21. EcuC_SetMetaPriority	435
5.6.2.2.22. EcuC_SetMetaSoConId	435
5.6.2.2.23. EcuC_SetMetaDataSourceAddr	436
5.6.2.2.24. EcuC_SetMetaDataTargetAddr	436
5.6.2.2.25. EcuC_StartOfReception	437
5.6.2.2.26. EcuC_StartOfReceptionDet	437
5.6.2.2.27. EcuC_TpRxIndication	438
5.6.2.2.28. EcuC_TpTransmit	438
5.6.2.2.29. EcuC_TpTxConfirmation	439
5.6.2.2.30. EcuC_TxBufferIsLocked	439
5.6.3. Integration notes	440
5.6.3.1. Exclusive areas	440
5.6.3.2. Production errors	440
5.6.3.3. Memory mapping	440
5.6.3.4. Integration requirements	441
5.6.3.4.1. EcuC.EB_CALLSEQREQ_EcuC_0001	441
5.6.3.4.2. EcuC.EB_CALLSEQREQ_EcuC_0002	441
5.6.3.4.3. EcuC.EB_CALLSEQREQ_EcuC_0003	441
5.6.3.4.4. EcuC.EB_CALLSEQREQ_EcuC_0004	441
5.6.3.4.5. EcuC.EB_CALLSEQREQ_EcuC_0005	441
5.6.3.4.6. EcuC.EB_CALLSEQREQ_EcuC_0006	441
5.6.3.4.7. EcuC.EB_CALLSEQREQ_EcuC_0007	442
5.6.3.4.8. EcuC.EB_CALLSEQREQ_EcuC_0008	442
5.6.3.4.9. EcuC.EB_CALLSEQREQ_EcuC_0009	442
5.6.3.4.10. EcuC.EB_CALLSEQREQ_EcuC_0010	442
5.6.3.4.11. EcuC.EB_CALLSEQREQ_EcuC_0011	442
5.6.3.4.12. EcuC.EB_CALLSEQREQ_EcuC_0012	442
5.6.3.4.13. EcuC.EB_CALLSEQREQ_EcuC_0013	442
5.6.3.4.14. EcuC.EB_CALLSEQREQ_EcuC_0014	442
5.7. MemMap	443
5.7.1. Configuration parameters	443
5.7.1.1. CommonPublishedInformation	444
5.7.1.2. MemMapAddressingModeSet	447
5.7.1.3. MemMapAddressingMode	450
5.7.1.4. MemMapAllocation	452
5.7.1.5. MemMapGenericMapping	452
5.7.1.6. MemMapSectionSpecificMapping	453
5.7.1.7. MemMapAS40Compatibility	453
5.7.1.8. MemMapGenerateEmptyHeaderFile	454
5.7.1.9. MemMapHeaderFiles	455
5.7.1.10. MemMapValidateMappings	455



5.7.1.11. MemMapValidateSections	456
5.7.1.12. PublishedInformation	459
5.7.2. Application programming interface (API)	459
5.7.3. Integration notes	459
5.7.3.1. Exclusive areas	459
5.7.3.2. Production errors	459
5.7.3.3. Memory mapping	459
5.7.3.4. Integration requirements	460
5.7.3.4.1. lim.MemMap.EB_INTREQ_MemMap_0001	460
5.8. PbcfgM	460
5.8.1. Configuration parameters	460
5.8.1.1. CommonPublishedInformation	461
5.8.1.2. PublishedInformation	464
5.8.1.3. PbcfgMBswModules	464
5.8.1.4. PbcfgMGeneral	465
5.8.2. Application programming interface (API)	467
5.8.2.1. Functions	468
5.8.2.1.1. PbcfgM_GetConfig	468
5.8.2.1.2. PbcfgM_Init	468
5.8.2.1.3. PbcfgM_IsValidConfig	469
5.8.3. Integration notes	469
5.8.3.1. Exclusive areas	469
5.8.3.2. Production errors	469
5.8.3.3. Memory mapping	469
5.8.3.4. Integration requirements	470
6. Bibliography	471



1. Overview of EB tresos AutoCore Generic 8 Base documentation

Welcome to the EB tresos AutoCore Generic 8 Base (ACG8 Base) product documentation.

This document provides:

- ▶ [Chapter 3, “ACG8 Base release notes”](#): release notes for the ACG8 Base modules
- ▶ [Chapter 4, “ACG8 Base user’s guide”](#): containing background information and instructions
- ▶ [Chapter 5, “ACG8 Base module references”](#): information about configuration parameters and the application programming interface



2. Supported features

2.1. Overview

This chapter provides an overview of the ACG8 Base and the features that are currently supported.

[Section 2.2, “Supported MemMap features”](#) contains an overview of `MemMap` features.

2.2. Supported MemMap features

- ▶ **Support for `MemMap` header files generation:** Support for `<SWC>_MemMap.h`, `<BSW>_MemMap.h` and `MemMap.h` header files generation.
- ▶ **Support for multiple BSW-Implementation:** Support for generation of `<BSW>_MemMap.h` for every BSW-Implementation.
- ▶ **Support for AUTOSAR 4.0.2:** Support for generation of macros for `CONFIG_DATA` `MemorySection` compatible with AUTOSAR 4.0.2.
- ▶ **Support for memory section validation:** Support for validating that the memory sections are opened and closed in the right order.
- ▶ **Support for `coreScope`:** Support for validating the `coreScope`.
- ▶ **Support for `safety level`:** Support for validating the `safety level`.

2.3. Supported Det features

- ▶ **Support for reporting run-time errors and transient faults:** Support for reporting errors based on the `Det_ReportRuntimeError()` and `Det_ReportTransientFault()` APIs.



3. ACG8 Base release notes

3.1. Overview

This chapter provides the ACG8 Base product specific release notes. General release notes that are applicable to all products are provided in the EB tresos AutoCore Generic documentation. Refer to the general release notes in addition to the product release notes documented here.

3.2. Scope of the release

3.2.1. Configuration tool

Your release of EB tresos AutoCore is compatible with the release of the EB tresos Studio configuration tool:

- ▶ EB tresos Studio: 26.2.0 b191017-0938

3.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this ACG8 Base release.

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
Base	4.0.3 []	4.3.0 [0000]	5.0.26	Elektrobit Automotive GmbH
Compiler	4.0.3 []	3.2.0 [0000]	1.0.3	Elektrobit Automotive GmbH
Det	4.0.3 []	3.2.0 [3]	6.5.1	Elektrobit Automotive GmbH
EcuC	4.0.3 []	4.0.3 [0000]	5.0.16	Elektrobit Automotive GmbH



Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
MemMap	4.0.3 []	1.4.0 [0]	1.3.4	Elektrobit Automotive GmbH

Table 3.1. Hardware-Independent Modules specified by the AUTOSAR standard

3.2.3. EB (Elektrobit) modules

The following table lists all modules which are part of this release but are not specified by the AUTOSAR standard. These modules include tooling developed by EB or they may hold files shared by all other modules.

Module name	Module version	Supplier
AppTemplates	6.8.0	Elektrobit Automotive GmbH
Atomics	1.0.5	Elektrobit Automotive GmbH
Configurators	2.8.20	Elektrobit Automotive GmbH
HidWiz	1.1.19	Elektrobit Automotive GmbH
Make	4.0.22	Elektrobit Automotive GmbH
PbcfgM	1.2.20	Elektrobit Automotive GmbH
Platforms	3.0.1	Elektrobit Automotive GmbH
SvcAs	1.2.13	Elektrobit Automotive GmbH
Workflows	2.3.1	Elektrobit Automotive GmbH

Table 3.2. Modules not specified by the AUTOSAR standard

3.2.4. MCAL modules and EB tresos AutoCore OS

For information about MCAL modules and OS, refer to the respective documentation, which is available as PDF at `$TRESOS_BASE/doc/3.0_EB_tresos_AutoCore_OS` and `$TRESOS_BASE/doc/5.0_MCAL_modules`¹. It is also available in the online help in EB tresos Studio. Browse to the folders EB tresos AutoCore OS and MCAL modules.

3.3. Module release notes

¹`$TRESOS_BASE` is the location at which you installed EB tresos Studio.



3.3.1. ApplTemplates module release notes

- ▶ Module version: 6.8.0.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 6.8.0

2019-10-30

- ▶ Updated name of EcuM wakeup source macro.

Module version 6.7.0

2019-03-20

- ▶ Updated EcuM callback functions to new EcuM requirements.

Module version 6.6.0

2018-07-11

- ▶ Removed version string from references to BSWMD top level packages in Rte configuration.
- ▶ On WinCore, copy necessary DLLs next to the .exe to assure that they are automatically loaded.

Module version 6.5.0

2015-11-10

- ▶ Changed MemMap header file handling

Module version 6.4.3

2014-05-15



- ▶ Adaptations of initialization sequence to support post-build capable ComM and BswM modules
- ▶ Added support for HighTec gcc compiler

Module version 6.4.2

2013-10-16

- ▶ Added defines for the states used from the SWC_CyclicCounter
- ▶ Replaced EcuM_CalloutLists.c by EcuM_Callout_Stubs.c

Module version 6.4.1

2013-06-26

- ▶ Switched ApplTemplates to MemMap generator.
- ▶ Added support for MicroOs.
- ▶ Replaced file EcuM_Callouts.c by EcuM_Callout_Stubs.c.
- ▶ Added support for softune compiler.

Module version 6.4.0

2013-02-27

- ▶ Moved call of Xcp_Init into BswM_DriverInitThree
- ▶ Incorporated support of EB tresos AutoCore Generic IP Stack modules
- ▶ Removed READONLY from EcuMResetReasonRef references
- ▶ Change ComM macro from COMMConf_ComMChannel_0 to ComMConf_ComMChannel_ComMChannel_0

Module version 6.3.0

2012-10-26

- ▶ Resource module included only for PA targets
- ▶ Bugfix: The initialization function of PbcfgM is not called
- ▶ Removed dependancy to Dcm from basicTemplate
- ▶ Renamed uip package to EBtresosAutoCore_ApplTemplates_*.uip



- ▶ Mcu_InitClock called only when provided by Mcu module
- ▶ Rte module configuration adapted to new package structure of BSWMDs
- ▶ Added possible inclusion of user Merged_Makefile.mak

Module version 6.2.1

2012-06-29

- ▶ Added support for PbcfgM to callouts.
- ▶ Removed Eep code stub.

Module version 6.2.0

2012-03-16

- ▶ The basic template now supports the integration of the memory stack.
- ▶ The basic template now supports shutdown of the ECU.

Module version 6.1.0

2011-09-30

- ▶ Initial AUTOSAR 4.0 version.

3.3.1.2. New features

- ▶ No new features have been added since the last release.

3.3.1.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.1.4. Deviations

This module is not part of the AUTOSAR specification.



3.3.1.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ For this module no limitations are known.

3.3.1.6. Open-source software

Open-source software information is not available for this module.

3.3.2. Atomics module release notes

- ▶ Module version: 1.0.5.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.2.1. Change log

This chapter lists the changes between different versions.

Module version 1.0.5

2019-06-14

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.0.4

2019-03-22

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.0.3

2018-10-26



- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.0.2

2018-09-28

- ▶ Changed default implementation of ATOMICS_USER_LOCK_OBJECT() and ATOMICS_USER_UNLOCK_OBJECT() macros to use Suspend/ResumeAllInterrupts() instead of Disable/EnableAllInterrupts()

Module version 1.0.1

2018-08-24

- ▶ Removed module configuration file Atomics.xdm

Module version 1.0.0

2018-06-22

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 0.10.0

2018-06-14

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 0.9.0

2018-05-25

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 0.8.0

2018-05-22

- ▶ ASCATOMICS-72 Fixed known issue: Default value of ATOMICS_USER_MULTICORE_CASE leads to misbehavior on single-core systems
- ▶ The macro ATOMICS_USER_MULTICORE_CASE has no default value any longer. You must always define it appropriately



- ▶ A compatibility shim was added to support EB tresos AutoCore OS versions 6.0.97 to 6.0.117 (inclusive). There are no compatibility issues from version 6.0.118 onwards
- ▶ The macro ATOMICS_USE_GENERIC_IMPL has no default value any longer. You must always define it to choose between generic or specialized implementation
- ▶ ASCATOMICS-79 Fixed known issue: Compatibility issues with EB tresos AutoCore OS from version 6.-0.97 to 6.0.117

Module version 0.7.0

2018-04-20

- ▶ The extension point dreisoft.tresos.launcher2.plugin.configuration was removed, because the Atomics module has nothing configurable via EB tresos Studio

Module version 0.6.0

2018-03-16

- ▶ Updated BSW module description: the BSW module entry Atomics_DONT_CARE has now the call-type INTERRUPT

Module version 0.5.0

2018-02-16

- ▶ Removed *deprecated* warning from TS_* functions
- ▶ AUTOSAR memory mapping and compiler abstraction are now supported

Module version 0.4.0

2018-01-19

- ▶ Added macro ATOMICS_VALUE_MAX that specifies the maximum value, that you can store in an object of type Atomic_value_t
- ▶ The APIs TS_AtomicSetBit_x(), TS_IntDisable(), and TS_Restore() were C preprocessor macros and are now C language functions

Module version 0.3.0

2017-12-15



- ▶ Added *deprecated* warning to `TS_*` functions

Module version 0.2.0

2017-11-17

- ▶ Implemented atomic functions with strict memory ordering

3.3.2.2. New features

- ▶ The Make variable `ATOMICS_USE_GENERIC_IMPL` is now mandatory. Always select which kind of implementation you want to use, as described in chapter *Migration* of the Atomics module user's guide. If the variable is not set in the build environment provided by EB, the compilation aborts with an error message.
- ▶ The C language macro `ATOMICS_USER_MULTICORE_CASE` is now mandatory. Always select single-core or multi-core case depending on your system characteristics. See also the Atomics module references.
- ▶ This module supports AUTOSAR memory mapping and compiler abstraction.
- ▶ Sequentially consistent atomic functions: There is now a set of functions which enable concurrent manipulation of atomic objects in memory. Complex synchronization techniques, e.g. AUTOSAR spinlocks, are not required when using these functions.

Furthermore, all atomic functions ensure sequential consistency of memory accesses.

3.3.2.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.2.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

You can only use the Atomics module as it is if there is an AUTOSAR-compliant OS in your configuration. You get the highest performance if you use EB tresos AutoCore OS. If you are prepared to accept performance



penalties, you can use any AUTOSAR-compliant OS. If you want to use the Atomics module without an AUTOSAR-compliant OS, you must customize the implementation of the Atomics module according to your requirements.

3.3.2.6. Open-source software

Open-source software information is not available for this module.

3.3.3. Base module release notes

- ▶ Module version: 5.0.26.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.3.1. Change log

This chapter lists the changes between different versions.

Module version 5.0.26

2019-10-30

- ▶ Added the option for customers to implement their own `TS_MemSet`, `TS_MemCmp` and `TS_MemBZero` functions.
- ▶ Added EB specific types for Ethernet plugins to `Eth_GeneralTypes.h`

Module version 5.0.25

2019-07-03

- ▶ Added configuration parameters for defining platform specific settings of AUTOSAR base types to be used in the generated `Platform_Types.h` and `TSAutosar_Cfg.h`
- ▶ Added configuration parameter which enables writing ARXML files with ecu parameter configuration for each module in each generator mode.
- ▶ Removed meaningless type qualifier from macro.
- ▶ Added configuration parameters for defining CPU properties to be used in the generated `Platform_Types.h`

**Module version 5.0.24**

2019-03-06

- ▶ Added new interface types for SoAd and Tcplp to `Eth_GeneralTypes.h`
- ▶ Provided configuration parameter to enable or disable the definition of enum `Can_ControllerState-Type` which is not required to be provided by the Base plugin before ASR 4.3.0
- ▶ Provide `DATA-CONSTRs` for `IMPLEMENTATION-DATA-TYPES` of `BASE-TYPE [s]uint(8|16|32) [-least]` and `boolean`
- ▶ Added support for saved zone option of MemMap for all ASIL level to Base SWCD by introducing additional software addressing methods (SwAddrMethod)
- ▶ ASCBASE-2004 Fixed known issue: Replaced compilation error in generator `Base_Modules.h` when multiple configuration container are used by generation warning, and creating configuration pointer to first configuration set

Module version 5.0.23

2018-11-07

- ▶ Added support for safety level option of MemMap to Base SWCD by introducing additional software addressing methods (SwAddrMethod)
- ▶ ASCBASE-1988 Fixed known issue: Added include of header file `TS_Compiler.h` (that defines macro `TS_CAST`) to header file `ComStack_Helpers.h`
- ▶ Introduce `TpDataStateType` enum value `EB_TP_RETRYINFONULL`

Module version 5.0.22

2018-07-11

- ▶ Added `uint32` to supported range of `PduLengthType`
- ▶ Added header file `Base_Version.h` for the version number of the Base plugin
- ▶ Added support for core scope option of MemMap to Base SWCD by introducing additional software addressing methods (SwAddrMethod)

Module version 5.0.21

2018-03-07

- ▶ Removed bundles `BaseSelectableWizardAsc`, `BaseXGenerator`, `BaseXPathAsc`, and `SW Description Updater` which are now released by the plugin `ToolExtensions`



- ▶ Removed macros `TS_AtomicSetBit` and `TS_AtomicClearBit` as they are now provided by the plugin Atomics
- ▶ ASCBASE-1939 Fixed known issue: Fixed generation error for non-EB complex device drivers
- ▶ Added support for providing custom memory copy function and overriding default implementation

Module version 5.0.20

2017-10-18

- ▶ Added parameters for mapping of AUTOSAR base types to C data types to the Base module configuration to be used by the `typedef` definitions in the generated `Platform_Types.h`
- ▶ Added support for non-EB modules (without `_TS_` infix) in generator `Base_Modules.h` for generation of module specific enable macros
- ▶ ASCBASE-1910 Fixed known issue: Added software addressing method (`SwAddrMethod`) for `VAR_NO_INIT` as specified by AUTOSAR 4.0.3

Module version 5.0.19

2017-04-19

- ▶ Added datatype definitions for `EthTrcv_PhyLoopbackModeType`, `EthTrcv_PhyTxModeType`, `EthTrcv_PhyTestModeType` and `EthTrcv_CableDiagResultType` to `Eth_GeneralTypes.h` according to AUTOSAR 4.3.0
- ▶ Added return value definitions for `E_NO_DATA` to `TransformerTypes.h` according to AUTOSAR 4.3.0
- ▶ Added datatype definitions for `EthSwt_PortMirrorCfgType` and `EthSwt_PortMirrorStateType` to `Eth_GeneralTypes.h` according to AUTOSAR 4.3.0
- ▶ Added macro `TS_CAST` to suppress excessive warnings from some compilers while keeping the code readable and efficient for well-behaving compilers
- ▶ ASCBASE-1893 Fixed known issue: Fixed XPath functions for size and alignment of base types to return correct values for 64bit data types

Module version 5.0.18

2016-11-23

- ▶ Added generator mode `export_asr_config` for writing ARXML files with ecu parameter configuration for each module
- ▶ Added application programming interface (API) to module references
- ▶ Generate definition of data type `Eth_BufIdxType` dependent on `EthIf config` parameter



- ▶ ASCBASE-1857 Fixed known issue: Fixed value of lower limit and upper limit for AUTOSAR 64bit base types (`uint64, sint64`) in the Base SWCD
- ▶ ASCBASE-1845 Fixed known issue: Added support for module generation output directories to XGenerator plugin

Module version 5.0.17

2016-06-15

- ▶ Added definition of `TS_STATIC_ASSERT` to `TSAutosar.h`
- ▶ Added tooling to calculate checksums on installed plugins and verify against provided checksum

Module version 5.0.16

2016-02-24

- ▶ Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeaderFile`
- ▶ Added definition of `Can_HwType` to `Can_GeneralTypes.h`

Module version 5.0.15

2015-11-25

- ▶ Changed the generator template `Base_Modules.h` to appropriately support complex driver and transformer modules
- ▶ Added support for 64bit CPUs

Module version 5.0.14

2015-08-07

- ▶ Provided type definitions for `VoidPtr` and `ConstVoidPtr`

Module version 5.0.13

2015-07-08

- ▶ Added error codes for E2E Transformer based on proposal from RfC #67553
- ▶ Provided configuration parameter to enable or disable the generation of AUTOSAR 64bit base types (`uint64, sint64`)

**Module version 5.0.12**

2015-03-11

- ▶ Provided configuration parameters for post-build-selectable variants (intermediate solution)
- ▶ Added support for AUTOSAR base types type `uint64` and `sint64`
- ▶ Removed adding of Service Component Prototypes to EcuExtract by SwdUpdater as the prototypes shall be added manually by the user in the composition editor and in the hierarchical model

Module version 5.0.11

2015-01-21

- ▶ Added configuration scheme to enable custom include files to be included via `Platform_Types.h`
- ▶ Support of custom defined `Std_VersionInfoType` via a configuration macro (`STD_VERSION_INFO_TYPE_DEFINED`)
- ▶ ASCBASE-1742 Fixed known issue: Changed the generator template `Base_Modules.h` to appropriately support modules that allow multiple instantiation

Module version 5.0.10

2014-10-02

- ▶ ASCBASE-1683 Fixed known issue: Binary Code generation uses wrong pointer type for non-relocatable configuration
- ▶ ASCBASE-1701 Fixed known issue: Changed the merge algorithm invoked by SwdUpdater in EB tresos Studio to generate a valid service component description
- ▶ Added getting of alignment and size also for derived data types to custom XPath functions

Module version 5.0.9

2014-04-25

- ▶ Added defines for module header file names respecting VendorId and VendorApilnfix from CommonPublishedInformation in `Base_Modules.h`

Module version 5.0.8

2014-02-04

- ▶ ASCBASE-1656 Fixed known issue: Binary Code generation does not support big-endian platforms



- ▶ ASCBASE-1662 Fixed known issue: Binary Code generation does not support relocatable configuration for address offsets different from 0
- ▶ Changed adaptation of ComM's initialization config pointers (`BASE_COMM_CONFIG_PTR`) to support both post-build capable and non post-build capable ComM variants
- ▶ Improved the XGenerator to handle nested structures
- ▶ Added support for signed data types for binary code generation
- ▶ Improved sorting of module specific macros in `Base_Modules.h` in alphabetical order
- ▶ Added support for non-AUTOSAR data types in arrays of post build configurations

Module version 5.0.7

2013-10-21

- ▶ Added explicit cast to `TS_AtomicSet/ClearBit()` in order to prevent compile warnings/errors in case `TS_AtomicSet/ClearBit_x()` are implemented as a functions
- ▶ Added BSW Module Description for Base (including memory section for CODE)
- ▶ ASCBASE-1438 Fixed known issue: Changed the macros to set or get a bit to fix the undefined behavior on 16bit platforms when the index of the bit to get or set is larger than 15
- ▶ Improved the handling of disabled modules in the configuration project, so that these are also disabled in the header file `Base_Modules.h` by defining the macro `BASE_{module}_ENABLED` as `STD_OFF`
- ▶ Implemented support for the AutoCore binary code generation to create a post-generation step to convert XGEN files

Module version 5.0.6

2013-06-14

- ▶ Added developer's guide for XPath functions
- ▶ Changed memory routines to use 32-bit parameter for length parameter
- ▶ Added macro definition checks and optimized structure type definition in header file `Fr_GeneralTypes.h`
- ▶ Added macro to check if a bit set in a variable (without returning its value)

Module version 5.0.5

2013-02-11

- ▶ Added datatype definitions for `CanTrcv_PNActivationType` and `CanTrcv_TrsvFlagStateType` to `Can_GeneralTypes.h`



- ▶ Provided version numbers and module IDs of Platform Types, Compiler Abstraction, Standard Types, and Communication Stack Types

Module version 5.0.4

2012-10-12

- ▶ Adapted top level structure for model elements for which the AUTOSAR specification already defines a standardized name. The package structure has been changed from /AUTOSAR/{module} to /AUTOSAR_{module}

For backward compatibility the old top level structure (/AUTOSAR/{module}) is still supported, but considered deprecated and will be removed in future releases

Module version 5.0.3

2012-06-15

- ▶ Added configuration parameter index macros (FR_CIDX_*) in Fr_GeneralTypes.h according to AUTOSAR R4.0r3
- ▶ Removed type definition Fr_configType from Fr_GeneralTypes.h according to AUTOSAR R4.0r3 FlexRay Driver SWS
- ▶ Added the SW Description Updater to generate and import the service software component descriptions and basic software module descriptions

Module version 5.0.2

2011-09-30

- ▶ Updated Fr_POCStatusType, Fr_RxLPduStatusType, Fr_SlotModeType, Fr_StartupStateType and FrTr-cv_TrcvWUReasonType in Fr_GeneralTypes.h to AUTOSAR R4.0r2

Module version 5.0.1

2011-09-02

- ▶ Initial AUTOSAR 4.0 version

3.3.3.2. New features

- ▶ Base supports the range uint16/uint32 of the AUTOSAR EcuC specification for the EcuC configuration parameter PduLengthType.



The mapping of AUTOSAR base types to C data types can be defined by parameters in the Base module configuration, container `BaseTypes`. The mapping is used to generate the corresponding `typedef` definitions in the generated header file `Platform_Types.h`.

If Platforms also provides mappings, the definitions from Platforms are used as default values for the corresponding parameters of Base. The mappings defined in Base overwrite the mappings of Platforms when `Platform_Types.h` is generated.

- ▶ Base supports 64bit CPUs by providing appropriate macros and functions, especially optimized memory operations (copy, move, compare, set) for 64bit types.
- ▶ The module contains a configuration scheme. The configuration scheme enables a user to configure custom include files which are included via `Platform_Types.h`.

This include mechanism can be used to provide types, macros and function declarations at a central location to all modules in a project.

- ▶ The provision of the type `Std_VersionInfoType` in the file `Std_Types.h` is controlled by a macro. If the macro `STD_VERSION_INFO_TYPE_DEFINED` is set to `STD_ON`, the user must provide a definition of the type `Std_VersionInfoType` in a custom include file.
- ▶ The default implementation of the memory copy function can be overridden by a custom implementation.
- ▶ Base supplies a layer of common operations that are not standardized in C90. Since they are required by several ComStack modules, Base provides the layer of abstraction with provisions to redefine them from Platforms or Compiler packages, effectively overriding them with a more targeted version using either non-standard compiler-known-functions, C or inline assembly. The default implementation is a portable default.
- ▶ All parameters related to C data types (mapping to AUTOSAR base types, size, alignment) are provided as module configuration parameters in the Base module (section `BaseTypes`), instead of by the properties file provided by the Platforms plugin.
- ▶ All parameters related to CPU properties (type resp. bit size, byte order, bit order) are configurable as module configuration parameters in the Base module (section `BaseCpuConfig`). The default values for these parameters are retrieved from the Platforms plugin.
- ▶ The default implementations of the memory set, memory compare and memory zeroing functions can be overridden by a custom implementation.

3.3.3.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.3.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.



- ▶ The type of `PduLengthType` that is defined in header file `ComStack_Cfg.h` is fixed to `uint16`.

Rationale:

Refer to deviations of EcuC.

- ▶ The type of `PduIdType` that is defined in header file `ComStack_Cfg.h` is fixed to `uint16`.

Rationale:

Refer to deviations of EcuC.

- ▶ The header file `Eth_GeneralTypes.h` does not include the header file `Eth_Cfg.h`.

Rationale:

`Eth_Cfg.h` contains the configuration specific parameters for the Ethernet controller driver and is provided by the Eth plugin. The inclusion of the header file introduces a dependency between Base and Eth which is not intended as Base is used by different other plugins.

- ▶ The type of `PduLengthType` is limited to `uint16` and `uint32`.

Rationale:

Refer to deviations of EcuC.

- ▶ The type of `PduIdType` is limited to `uint16`.

Rationale:

Refer to deviations of EcuC.

- ▶ The type of `PduInfoType` is not compliant to AUTOSAR 4.3.0.

Rationale:

No support of Metadata.

3.3.3.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ Dependency to header file `stddef.h` of C standard library

For the compilation of Base, the header file `stddef.h` provided by the C standard library is required. From this header file, only `offsetof()` is used which usually is implemented as macro (ISO-C:90).

Rationale:



`offsetof()` is used by the implementation of the post-build configuration to compute a structure member's offset from the beginning of a structure. The usage of `offsetof()` violates MISRA-C:2012 rule 1.3, but an own implementation for `offsetof()` would create other MISRA-C:2012 violations. As no library from the C standard library is required for linking, MISRA-C:2012 directive 1.1 is not applicable.

3.3.3.6. Open-source software

Open-source software information is not available for this module.

3.3.4. Compiler module release notes

- ▶ Module version: 1.0.3.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.4.1. Change log

This chapter lists the changes between different versions.

Module version 1.0.3

2019-06-03

- ▶ Incremented module version to 1.0.3. Fixed "make debug" rule for all architectures: adapted commandline execution of Lauterbach debugger to latest syntax.
- ▶ Fixed linking issue for Linaro(issue detected in compiler version 6.3.1-2017.05).
- ▶ Added ASM_FILTER_LIST for ARM compiler to support different extensions for assembler files.
- ▶ Replaced the @echo xxx by \$(info xxx) or \$(file xxx) in required places

Module version 1.0.2

2018-11-07

- ▶ Updated assembler support for ARM: armkeil, gnu
- ▶ Added compiler support for ARM64: armkeil, gnu
- ▶ Incremented module version to 1.0.2



- ▶ Added compiler support for IAR Embedded Workbench for CORTEXM
- ▶ A Compiler_Cfg.h file template can be found in the Compiler module in the template folder. It includes all memory and pointer class macros for all ACG modules.
- ▶ Added compiler support for Linaro Embedded Workbench. CORTEXM: linaro
- ▶ Added documentation for compiler abstraction macros and additional EB macros
- ▶ Changed the compiler specific symbol for Power Architecture

Module version 1.0.1

2018-07-11

- ▶ Added compiler abstraction macros for J1939 stack
- ▶ Updated compiler support for RH850: multi
- ▶ Added compiler support for ARM: armkeil, gcc, gnu, multi, tigct
- ▶ Added compiler abstraction macros for EcuC
- ▶ Incremented module version to 1.0.1
- ▶ Fixed Compiler_Cfg.h template by adding missing J1939STACK_DATA macro

Module version 1.0.0

2018-03-07

- ▶ Added Compiler plug-in as new mandatory plug-in for ACG-8.5 to provide compiler-specific header files and makefiles outside of Platforms plug-in. Compiler (version 1.0.0) support for: WINDOWS: bcc, gcc45, gcc62; PA: gcc, multi; TRICORE: gcc, gnu, tasking
- ▶ Added compiler abstraction macros for Atomics module
- ▶ Added compiler support for RH850: multi

3.3.4.2. New features

- ▶ No new features have been added since the last release.

3.3.4.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.



3.3.4.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.4.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ According to the AUTOSAR Specification of Compiler Abstraction, each architecture needs its separate compiler abstraction macro file. Use only one architecture-specific compiler abstraction header file in your project.
- ▶ To configure the architecture-specific compiler abstraction macro file and the compiler in the user build environment, the variables `TARGET` and `TOOLCHAIN` have to be set according to the naming used in the makefiles delivered together with the util directory.
- ▶ The Compiler module includes all currently integrated compilers. This module is tested only with one specific compiler. See the quality statement for information about your release. Other compilers are not tested.
- ▶ Report any errors related to the qualified compiler via <https://www.elektrobit.com/support/>.
- ▶ The compiler may be changed for a project as required. To report any errors related to a non-qualified compiler or to use a new compiler/new architecture, contact sales@elektrobit.com.
- ▶ This Compiler module supports the following `TARGET/TOOLCHAIN` types:
 - ▶ ARM
 - ▶ `armkeil`: ARM keil microcontroller compiler
 - ▶ `dcc`: Wind River Diab Compiler for ARM
 - ▶ `gnu`: `arm-none-eabi-gcc` GNU compiler
 - ▶ `multi`: Greenhills multi compiler for ARM
 - ▶ `tigct`: Texas Instruments ARM C/C++ Compiler
 - ▶ ARM64
 - ▶ `armkeil`: ARM keil microcontroller compiler
 - ▶ `gnu`: `aarch64-elf-gcc` GNU compiler
 - ▶ CORTEXM
 - ▶ `arm5`: Arm Compiler 5
 - ▶ `armkeil`: ARM keil microcontroller compiler
 - ▶ `gnu`: `arm-none-eabi-gcc` GNU compiler



- ▶ iar: IAR Embedded Workbench
- ▶ linaro: arm-eabi-gcc GNU compiler
- ▶ multi: Greenhills multi compiler for ARM
- ▶ tigct: Texas Instruments ARM C/C++ Compiler
- ▶ PA
 - ▶ dcc: Wind River Diab Compiler for PowerPC
 - ▶ htgcc: HighTec compiler suites for Power Architecture
 - ▶ multi: Greenhills multi compiler for Power Architecture
- ▶ RH850
 - ▶ multi: Greenhills multi compiler for RH850
- ▶ TRICORE
 - ▶ dcc: Wind River Diab Compiler for Tricore
 - ▶ gnu: HighTec compiler suites for Tricore
 - ▶ tasking: Altium TASKING compiler
- ▶ WINDOWS
 - ▶ bcc: bounds checking GCC 4.0.4
 - ▶ gcc45: MinGw 4.5 (part of WINDOWS Platforms module)
 - ▶ gcc62: MinGw 6.2 (part of WINDOWS Platforms module)

3.3.4.6. Open-source software

Open-source software information is not available for this module.

3.3.5. Configurers module release notes

- ▶ Module version: 2.8.20.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.5.1. Change log

This chapter lists the changes between different versions.

**Module version 2.8.20**

2019-10-11

- ▶ Extended Transformer for `SomeIpTp` to support `SomeIpTpChannel` entities
- ▶ Extended Transformer for `TcpIp` to configure `TcpIpTcpKeepAliveEnabled`, extended Transformer for `SoAd` to configure `SoAdSocketTcpKeepAlive`
- ▶ Extended Transformer for `StbM` to support parameters `StbMOffsetCorrectionAdaptionInterval`, `StbMOffsetCorrectionJumpThreshold`, `StbMRateCorrectionMeasurementDuration` and `StbMRateCorrectionsPerMeasurementDuration`
- ▶ Extended Transformer for `LinIf` to prevent the configuration of `LinFrame` elements that the configured Ecu Instance sends or receives via `LinSlave` communication controllers

Module version 2.8.19

2019-09-06

- ▶ ASCCONFIGURATORS-1064 Fixed known issue: Transformer for `SoAd` does not create `Pdu Routes` for dynamically configured local IP addresses

Module version 2.8.18

2019-07-12

- ▶ Support array types for the `DcmDspRoutineSignalType`
- ▶ ASCCONFIGURATORS-1055 Fixed known issue: Transformers for `Com`, `ComM` issue a `NullPointerException` if PNC vector is missing

Module version 2.8.17

2019-06-14

- ▶ Remove prefix for names of `DcmDspData` elements
- ▶ Extended Transformer for `ComM` to configure `ComMNmVariant` as `NONE` if the configured Ecu Instance does not send or receive PDUs on the `ComMChannel`
- ▶ Extended Transformers for `EcuC`, `CanNm`, `FrNm`, `UdpNm`, `PduR` and `Com` to configure received Nm User Data PDUs even if the signals contained in the Nm PDUs have no signal port assigned
- ▶ Extended Transformer for `SoAd` to configure `GeneralPurposeIPdu` elements of category `Dlt_TP` and `Dlt_IF`
- ▶ Extended Transformer for `SoAd` to configure `SoAdTxPduCollectionSemantics`



- ▶ ASCCONFIGATORS-1041 Fixed known issue: During the import of the Diagnostic Extract, the creation of a configuration container in Dem throws an exception. This exception is caused by an illegal container short name

Module version 2.8.16

2019-05-17

- ▶ Extended Transformer for `SoAd` to support Service Oriented Communication of System Description .arxml files
- ▶ Extended Transformer for `StbM` to configure `StbMIIsSystemWideGlobalTimeMaster` for abstract Global Time Domains
- ▶ ASCCONFIGATORS-1035 Fixed known issue: Transformer for `LinIf` configures `LinIfCollisionResolvingRef` incorrectly
- ▶ ASCCONFIGATORS-1038 Fixed known issue: Dcm importer wrongly configures the `SessionRefs` and `SecurityLevelRefs` for the `ServiceInstanceId` value of `AccessPermissionValidity`

Module version 2.8.15

2019-04-18

- ▶ Extended Transformer for `Dem` to support the configuration of the `DemOBDSupport` and the `DemOBDSupportKind` parameters
- ▶ Adapted Transformer for `TcpIp` to skip the configuration of `TcpIpStaticIpAddressConfig` containers if the parent `TcpIpLocalAddr` container has been created for a multicast address that is configured at runtime by `Sd`
- ▶ Adapted Transformer documentation for `Dcm` to correctly reflect the information for `DiagnosticComControl` (0x28) service
- ▶ Adapted Transformer for `Dcm` to fill the Data Access Interface by the DEXT importer
- ▶ Adapted Transformer documentation for `Dcm` to write `VARIABLE-SIZE` instead of `VARIABLE-LENGTH`
- ▶ Extended Transformer for `CanNm` to resolve short name collisions in the case that `NmPdu` elements are transmitted in different CAN networks using identical CAN Ids or CAN Id ranges
- ▶ Adapted Transformer for `Dcm` to also take into account `MAPPED-FLAT-SWC-SERVICE-DEPENDENCY` when configuring the `DcmDspRoutineUsePort`
- ▶ Extended Transformer for `Com` to configure `ComSignalDirection` and `ComSignalGroupDirection`
- ▶ Extended Transformers to support the reception of one and the same PDU via different networks

Module version 2.8.13

2019-03-14



- ▶ ASCCONFIGURATORS-1006 Fixed known issue: Transformer for Xcp erroneously adds non-Xcp PDUs

Module version 2.8.12

2019-02-15

- ▶ Extended Transformer for Dcm to configure the DcmDspRequestRoutineResultsIn container inside the Dsp container, only if the REQUESTS tag exists inside the DiagnosticExtract
- ▶ Extended Transformer for IpduM to configure IpduMContainerQueueSize by using ContainerIPdu.minimumRxContainerQueueSize and ContainerIPdu.minimumTxContainerQueueSize
- ▶ Extended Transformer for StbM, EthTSyn, CanTSyn, and FrTSyn to support configuration via AUTOSAR 4.4 GlobalTimeDomain elements
- ▶ Extended Transformer for IpduM to configure IpduMContainerTxTriggerMode of IpduMContainerTxPdu to IPDUM_DIRECT, if the IpduMContainerTxPdu is itself contained in a SecuredIPdu, i.e. a PDU that is processed in SecOC
- ▶ Extended Transformer for ComM to take the PncMapping references to PhysicalChannel elements into account for setting up the ComMPnc lists
- ▶ Extended Transformer for ComM to support the configuration of managed ComMChannel containers

Module version 2.8.11

2019-02-05

- ▶ ASCCONFIGURATORS-996 Fixed known issue: DiagnosticEventHandler issues NullPointerException on Event that is not mapped onto an operation Cycle
- ▶ Extended Transformer for Dem to configure the DiagnosticDemProvidedDataMapping.DataProvider with the standardized values
- ▶ Extended Transformer for IpduM to configure IpduMContainedTxPduPriority

Module version 2.8.10

2019-01-25

- ▶ Extended configuration of CanTSynGlobalTimeFollowUpTimeout in Transformer for CanTSyn
- ▶ Extended Transformer for ComM to configure ComMPncEthIfSwitchPortGroupRef
- ▶ Extended Transformers to allow execution in EB tresos Studio 26 and to support transformation of ASR 4.4 system description files
- ▶ Extended Transformers for CanIf and FrIf to configure PduR as upper layer for routed N-PDUs



- ▶ Extended Transformer for `Dem` to configure the `DemDtcStatusAvailabilityMask` parameter inside the `DemGeneral` container

Module version 2.8.9

2018-12-13

- ▶ ASCCONFIGURATORS-977 Fixed known issue: Transformer for `EcuC` stops with an error if a sent PDU undergoes a tx fanout before `SecOC/IpduM` and before `<Bus>If` at the same time
- ▶ Extended Transformer for `Dcm` to configure the `DcmDspRoutineFixedLength` parameter of a `DcmDspRoutine`
- ▶ Extended Transformer for `Dem` to configure the `DemEventSignificance` parameter inside the `DemEventClass` container
- ▶ ASCCONFIGURATORS-980 Fixed known issue: Transformer for `SoAd` configures `SomeIpTp` N-PDUs as TP API PDUs instead of as IF API PDUs
- ▶ Extended Transformers for `IpduM`, `SecOC`, and `PduR` to support transmission of cryptographic PDUs in container PDUs and routing of cryptographic PDUs
- ▶ Extended Transformer for `Dem` to configure the `DemEventDestination` parameter inside the `DemEventClass` container
- ▶ Extended Transformer for `IpduM` to configure `IpduMContainerQueueSize`

Module version 2.8.8

2018-10-26

- ▶ ASCCONFIGURATORS-964 Fixed known issue: Transformer for `EcuC` does not configure Routing-only Applicative TP SDUs
- ▶ Extended Transformers for `Com` and `LdCom` to support the configuration of `Tp` API for PDUs that are sent or received via `SomeIpTp`
- ▶ Extended Transformer for `ComM` to configure `ComMPncComSignal` entries as ERA if they are associated with a `COMM_GATEWAY_TYPE_PASSIVE` `ComMChannel`
- ▶ ASCCONFIGURATORS-971 Fixed known issue: Transformer for `FrTSyn` does not configure `FrTSynGlobalTimeSyncDataIDList` correctly

Module version 2.8.7

2018-09-28

- ▶ Extended Transformer for `SoAd` to support use of "0.0.0.0" IPV4 addresses as dynamic remote IP addresses



- ▶ Extended Transformer for `IPduM` to support parameter `IpduMUnusedAreasDefault` in `IpduMContainerTxPdus`
- ▶ ASCCONFIGURATORS-957 Fixed known issue: Transformer for `SecOC` does not process cryptographic PDUs

Module version 2.8.6

2018-08-24

- ▶ Removed configuration of `EthSwtPortSpeed` from Transformer for `EthSwt`

Module version 2.8.5

2018-07-31

- ▶ Extended Transformer for `StbM`: Added support for the configuration of `StbMSynchronizedTime-BaseIdentifier`
- ▶ Extended Transformers to ignore Frames, PDUs, and Signals that are sent or received via `LinSlave` elements
- ▶ Extended Transformer for `LdCom` to support the configuration of a received PDU irrespective of its `minimumDelay`, `transmissionModeTrueTiming` and `transmissionModeCondition`
- ▶ Extended Transformer for `Com` to implement OEM specific fallback for configuration of `ComSignalType`
- ▶ Extended Transformer for `EcuC` to exclude secured PDUs and Pdus contained in a secured PDU from the Meta-Data handling
- ▶ Extended Transformer for `Dcm` to configure `DiagnosticDataTransfer`
- ▶ ASCCONFIGURATORS-951 Fixed known issue: Transformer for `IPduM` creates duplicated Contained Rx PDUs

Module version 2.8.4

2018-06-22

- ▶ Removed creation of `IPV4` limited broadcast address entries in Transformer for `TcpIp`
- ▶ Extended Transformer for `PduR` to configure `PduRRoutingGroup` containers
- ▶ Extended Transformer for `SecOC` to support the configuration of distinct Secured and Cryptographic PDU containers
- ▶ Extended Transformer for `EcuC` to configure Meta-Data for Ethernet PDUs that are received or sent by server components



- ▶ Extended Transformer for `EthSwt` to configure modified parameter `EthSwtPortPhysicalLayerType` and new parameter `EthSwtPortMacLayerType` in `EthSwtPort`

Module version 2.8.3

2018-05-25

- ▶ Extended Transformer for `Com` to configure `ISignalIPduGroup` elements that refer to `NmPdu` elements
- ▶ Extended Transformers for `Com`, `SecOC`, `IpduM`, `PduR`, and `EcuC` to support tx fan-out in between these modules
- ▶ Extended Transformers for `Com` and `SomeIP` to support sending and receiving of `ComIPdu` elements via TP API
- ▶ Extended Transformer for `CanIf` to support `CAN_TSYN` as value for `CanIfRxPduUserRxIndicationUL` and `CanIfTxPduUserTxConfirmationUL`
- ▶ Extended Transformer for `Dcm` to configure custom sub-services for `DiagnosticEcuReset`
- ▶ ASCCONFIGURATORS-922 Fixed known issue: `EthIfCtrlMtu` is configured incorrectly
- ▶ Changed Transformer for `EthIf`: Length of VLAN tag is no longer subtracted from `EthIfCtrlMtu`
- ▶ Extended Transformer for `IPduM` to configure the offset and `updateIndicationBitPosition` parameters in `IpduMContained[Rx|Tx]Pdu`, added support for `ContainerIPduHeaderTypeEnum.no-Header`
- ▶ Extended Transformer for `LdCom` to support the configuration of a received PDU irrespective of its transferProperty
- ▶ Extended Transformer for `DoIP` to support `IPv6` during configuration of `VehicleAnnouncement` container
- ▶ Extended Transformer for `PduR` to configure `PduRTpThreshold` only if the `PduRRoutingPath` container holds exactly one `PduRDest` subcontainer
- ▶ Extended Transformer for `FrTSyn` to configure `FrTSynGlobalTimeSequenceCounterJumpWidth` in the context of a `FrTSynGlobalTimeSlave` container
- ▶ ASCCONFIGURATORS-921 Fixed known issue: `CanIfRxPduDlc` is configured incorrectly
- ▶ Extended Transformer for `SoAd` to configure `SoAdSocketTcpInitiate` only in `TcpIp` `SoAdSocketConnectionGroup` containers that contain exactly one `SoAdSocketConnection` sub container

Module version 2.8.2

2018-04-20

- ▶ ASCCONFIGURATORS-904 Fixed known issue: `MultiplexedIPdus` that are sent and received at the same time are configured incorrectly



- ▶ Extended Transformer for `Dcm` to configure references `DcmDspReadMemoryRangeSecurityLevelRef` in `DcmDspReadMemoryRangeInfo` and `DcmDspReadMemoryRangeByLabelInfo`, `DcmDspWriteMemoryRangeSecurityLevelRef` in `DcmDspWriteMemoryRangeInfo` and `DcmDspWriteMemoryRangeByLabelInfo`
- ▶ Extended Transformer for `Dcm` to configure `DiagnosticRequestOnBoardMonitoringTestResults`
- ▶ Extended Transformer for `Com` to configure `ComIPduGroupRef` only for `ComPduGroup` containers whose direction is not opposite to the referencing PDU
- ▶ Extended Transformer for `Dcm` to configure `DiagnosticRequestVehicleInfo`

Module version 2.8.1

2018-03-16

- ▶ ASCCONFIGURATORS-890 Adapted Transformer for `Dcm` to enable the `DcmProcessingConditions` container only in the case when a `DcmModeCondition` or a `DcmModeRule` exists
- ▶ Extended Transformer for `Dcm` to configure the `OBDMode_0x0A` (`DiagnosticRequestEmissionRelatedDTCPermanentStatus`) service
- ▶ Extended Transformer for `ComM` to configure the parameter `ComMPncPrepareSleepTimer`
- ▶ Extended Transformers for `Nm`, `UdpNm`, `CanNm`, and `FrNm` to configure the parameters `NmRepeatMsgIndEnabled`, `NmNodeDetectionEnabled`, `NmNodeIdEnabled` either by using the `NmCluster` data or by using the `NmEcu` data as a fallback
- ▶ Extended Transformer for `SecOC` to configure parameters `SecOCSecured[Rx|Tx]PduOffset` and `SecOCSecured[Rx|Tx]PduLength` in `SecOCRxPduSecuredArea`
- ▶ Extended Transformer for `StbM` to support the configuration of `StbMSynchronizedTimeBase` container for GlobalTimeDomains that are not linked to any network
- ▶ Extended Transformer for `SecOC` to configure parameter `SecOCAuthenticationBuildAttempts` in `SecOCRxPduProcessing` and `SecOCTxPduProcessing`
- ▶ Extended Transformer for `Dcm` to support the configuration of `DcmDspMemory` container for the `DiagnosticWriteMemoryByAddress` and `DiagnosticReadMemoryByAddress` services.
- ▶ Extended Transformer for `ComM` to also take routed PDUs into account for setting up the PNC to ComM-Channel references

Module version 2.8.0

2018-02-16

- ▶ Extended Transformers for `DoIP` and `PduR` to not configure `DoIP` SDUs that are either sent and have their `DiagPduType` field set to `DiagRequest` or that are received and have their `DiagPduType` field set to `DiagResponse`



- ▶ Extended Transformer for FrTSyn to configure FrTSynGlobalTimeOfsDataIDList and FrTSyn-GlobalTimeSyncDataIDList configuration containers
- ▶ Extended Transformer for Dem to configure the DemInternalDataElementClass and DemExternal-CSDaDataElementClass choices of the DemDataElementClass ChoiceContainer
- ▶ ASCCONFIGATORS-878 Fix configuration of DcmDsdSubServiceId for the DiagnosticComControl
- ▶ ASCCONFIGATORS-879 Fix configuration of Data Access Interface in the case of a DiagnosticIOControlService
- ▶ Extended Transformer for Dcm to configure DiagnosticRequestDownload, DiagnosticRequestUpload, DiagnosticRequestTransferExit
- ▶ Extended Transformer for StbM to configure parameters StbMTimeLeapFutureThreshold, StbM-TimeLeapPastThreshold, and StbMClearTimeleapCount
- ▶ Extended Transformer for SecOC to configure parameters SecOCAuthDataFreshnessLen, SecOCAuthDataFreshnessStartPosition, SecOCUseAuthDataFreshness, and SecOCSecuredRxPdu-Verification
- ▶ Added Transformer for SomeIpTp
- ▶ Transformers for EcuC, CanNm, FrNm, UdpNm, PduR, Com, and ComM now configure PNC ERA PDUs and signals for ACTIVE as well as PASSIVE PNC gateways
- ▶ Adapted Transformer for CanTSyn to configure the CanTSynGlobalTimeSequenceCounterJump-Width parameter in CanTSynGlobalTimeSlave

Module version 2.7.11

2018-01-19

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 2.7.10

2017-12-15

- ▶ Extended Transformer for Dcm to support DcmDspPid configuration
- ▶ Extended Transformer for Dcm to configure the OBDMode_0x02 (DiagnosticRequestPowertrain-FreezeFrameData) service
- ▶ Extended Transformer for Dcm to configure the OBDMode_0x01 (DiagnosticRequestCurrentPowertrainDiagnosticData) service
- ▶ Extended Transformer for Dcm to configure the OBDMode_0x04 (DiagnosticClearResetEmission-RelatedInfo) service
- ▶ Extended DcmTransformer for DcmDspComControl to configure the DcmDspComControlSubNode sub-container



- ▶ Extended Transformer for SecOC to configure the SecOCPduType parameter for authentic PDUs
- ▶ ASCCONFIGURATORS-859 Fixed known issue: ComTransferProperty is configured incorrectly for SignalGroups
- ▶ Extended Transformer for CanNm to configure the CanNmNodeIdEnabled parameter for NmEcus
- ▶ ASCCONFIGURATORS-864 Fixed known issue: ComTimeout and ComFirstTimeout are configured incorrectly for SignalGroups

Module version 2.7.9

2017-11-17

- ▶ Extended Transformer for EthSwt to also accept value of CouplingPort.VlanMembership.defaultPriority if all defaultPriority values of a CouplingPort are identical
- ▶ Modified the import of the Variable-size DID signals.
- ▶ Extend the DcmTransformer in order to also configure the DcmDspRequestResultsRoutineSupported and DcmDspStopRoutineSupported
- ▶ Adapted Transformer for IpduM: IpduMContainedRxInContainerPduRef is only configured if RxAcceptContainedIPdu of the associated ContainerIPDU is set to ACCEPT-CONFIGURED
- ▶ Adapted Transformer for SoAd: SoAdSocketRemoteIpAddress is now explicitly configured if Ipv4Configuration/ipv4Address or Ipv6Configuration/ipv6Address is set to ANY
- ▶ Extended Transformer for Dcm to configure the OBDMode_0x08 (RequestControlOfOnBoardDevice) service
- ▶ Support fragmentation of PDU size bigger than MTU in SoAdTransformer and TcpIpTransformer

Module version 2.7.8

2017-10-20

- ▶ Extended Transformer for Dcm to configure the OBDMode_0x03 and OBDMode_0x07 services
- ▶ Improved handling of ConsumedServiceInstance elements linked to ProvidedServiceInstance elements which in turn are not connected to any SocketConnection or SocketConnectionBundle
- ▶ Modified the configuration of DcmDspDidInfo and of its subcontainers: DcmDspDidControl, DcmDspDidRead and DcmDspDidWrite

Module version 2.7.7

2017-09-22



- ▶ Removed workaround in the configuration of `IpduContainerTxTriggerMode` that was required by a `SecOC` restriction which has been resolved in the meantime
- ▶ Added support for the configuration of `ContainerIPdu` elements that are sent and received by the same `EcuInstance`
- ▶ Extended Transformers for `CanTSyn`, `FrTSyn` and `EthTSyn` to support AUTOSAR RFC 75119
- ▶ Extended Transformer for `Com` to configure the `ComTransferProperty` of an `I-SIGNAL-GROUP` by taking the `ComTransferProperty` values of the contained `I-SIGNAL` elements into account
- ▶ Extended Transformer for `CanTSyn` to configure time domain specific data-ID lists
- ▶ Extended Transformer for `SecOC` to reflect the `SecOC` parameter changes in ACG 8.3 and ACG 8.4
- ▶ Adapted Transformers for `CanNm`, `FrNm`, `UdpNm` and `Nm` to reflect that several parameters in the modules have been moved from the general containers to the `Nm` channel related containers
- ▶ ASCCONFIGATORS-818 Fixed known issue: Validation error for `DcmDspRoutineInfo` if no signal is configured
- ▶ Extended `DcmTransformer` for `DcmDspComControl` to configure the `DcmDspComControlAllChannel` and `DcmDspComControlSpecificChannel` subcontainers

Module version 2.7.6

2017-08-24

- ▶ ASCCONFIGATORS-808 Fixed known issue: Incomplete import of Diagnostic configuration for `DcmDspDataType`, `DcmDspDataSize`, and `DcmDspRoutineSignalType`
- ▶ ASCCONFIGATORS-815 Fixed known issue: Received SecuredIPdus with `rxSecurityVerification set to false` are not routed via Gateway
- ▶ ASCCONFIGATORS-814 Fixed known issue: Configuration of one and the same PDU for sending and receiving on the same VLAN does not work

Module version 2.7.4

2017-07-28

- ▶ Extended Transformer for `PduR` to support the fan-out of PDUs on different VLANs of the same Ethernet network
- ▶ Extended Transformer for `Com` to support the configuration of 64-bit signed/unsigned signal datatypes
- ▶ Modified Transformer to correctly create and configure `DcmDspDidInfo` for `DiagnosticIoControlService`
- ▶ Extended Transformer for `DcmProcessingCondition` to configure `DcmModeRule` and `DcmModeCondition`



- ▶ Extended Transformer for Can to make instance suffix creation robust against invalid RxIdentifierRanges
- ▶ ASCCONFIGURATORS-805 Fixed known issue: Received SecuredIPdu in ContainerIPDU with rxSecurityVerification set to false causes error during configuration import
- ▶ ASCCONFIGURATORS-807 Fixed known issue: Importer stops with an error if VLANs of Switch Ports are not connected to Ecu Instance
- ▶ Modified DemTransformer (DiagnosticExtendedDataRecordHandler and DiagnosticEnableConditionGroupHandler) to reduce the number of configured DemExtendedDataClass and DemEnableConditionGroup containers

Module version 2.7.3

2017-06-29

- ▶ Extended Transformer for ComM to configure ComMPncComSignalChannelRef in ERA ComMPnc-ComSignal containers
- ▶ Extended Transformer for EthIf to support MaximumTransmissionUnit from EthernetCommunicationConnector as well as from EthernetCommunicationController
- ▶ Extended Transformer for SecOC to support the configuration of GeneralPurposeIPdu elements in SecOCTxAuthenticPduLayer or SecOCRxAuthenticPduLayer containers
- ▶ Extended Transformer for PduR to support the fanout of sent SecuredIPdu elements
- ▶ Extended Transformer for EthIf and EthSwt to support managed switching of switch ports

Module version 2.7.2

2017-06-02

- ▶ Extended Transformer for Dcm to configure DiagnosticPeriodicID
- ▶ Extended Transformer for PduR to support the configuration of routing-only multiplexed PDUs and the configuration of multiplexed PDUs that contain routing-only demultiplexed PDUs
- ▶ Extended Transformer for PduR, CanIf, FrIf, and LinIf to support the configuration of routed Xcp PDUs
- ▶ Modified DiagnosticDIDHandler, DiagnosticDynamicallyDefineDataIdentifierHandler, and DiagnosticIOControl to reduce the number of configured DcmDspDidInfo containers
- ▶ Extended Transformer for SecOC to support SecureCommunicationFreshnessProps and SecureCommunicationAuthenticationProps for the configuration of SecOC PDUs
- ▶ Extended Transformer for IpduM to support the configuration of IpduMContainerTxTriggerMode
- ▶ Extended Transformers for CanTSyn and FrTSyn to support GlobalTimeDomain.globalTimePduTriggering as well as GlobalTimeDomain.globalTimePdu



Module version 2.7.1

2017-05-05

- ▶ Modified DemTransformer to reduce the number of configured DemFreezeFrameClass and DemFreezeFrameRecNumClass elements
- ▶ Configured DemDTCClass to support DTC of type DiagnosticTroubleCodeObd
- ▶ Changed the way how the DcmDspDataType is configured based on Byte Array Signals
- ▶ Extended Transformer for Dcm to configure DcmDslBuffer
- ▶ Improved the configuration of DiagnosticOperationCycles and DiagnosticAgingCycles
- ▶ Extended DcmTransformer to set the DcmDslProtocolSessionRef reference to every DcmDslProtocolRow

Module version 2.7.0

2017-03-31

- ▶ Extended Transformer for Dcm to configure ClearDiagnosticInformation
- ▶ Extended Transformer for Dcm to configure DiagnosticWriteMemoryByAddress services
- ▶ Extended Transformer for Dcm to configure DiagnosticReadMemoryByAddress services
- ▶ Extended Transformers for CanNm, FrNm, and UdpNm to configure <Bus>NmPnResetTime
- ▶ Extended Transformer for Dem to configure property <Bus>DemAgingCycleRef from the DemEvent-Class
- ▶ Extended Transformers for UdpNm and Nm to support UdpNmChannels associated with EthernetPhysicalChannels/VLANs
- ▶ Extended Transformer for Com to take ISignal.iSignalType into account for the calculation of signal data types
- ▶ Extended Transformer for LinIf to support the configuration of AssignNad, AssignFrameId, and UnassignFrameId frames using AUTOSAR 4.3.0 LinSlaveConfig entities
- ▶ Extended Transformer for Com to support the configuration of ComSignal/ComFirstTimeout
- ▶ Extended Transformers for CanTSyn and FrTSyn to support the configuration of data-ID lists
- ▶ Extended Transformer for EthTSyn to support the configuration of data-ID lists and EthTSynFramePrio
- ▶ Extended Transformers for Com and SoAd to support the IPV4 and IPV6 address value ANY
- ▶ Extended Transformer for UdpNm to support the configuration of UdpNmImmediateNmCycleTime and UdpNmImmediateNmTransmissions
- ▶ Extended Transformers for CanNm, FrNm, UdpNm, and Nm to use NmCluster.nmPncParticipation for the configuration of <Bus>NmComUserDataSupport and <Bus>NmPnEnabled
- ▶ Extended Dcm to support DcmDslProtocolPriority



- ▶ Added Transformer for Fim
- ▶ Set the DiagnosticPeriodicRate category in the DiagnosticPeriodicTransmission field in the DcmDsp container

Module version 2.6.10

2017-03-03

- ▶ Extended Transformer for Dcm to configure DiagnosticIOControl
- ▶ Configured DcmDsdSidTabSubfuncAvail parameter for DiagnosticEcuReset, DiagnosticRoutineControl and DiagnosticReadDTCInformation services
- ▶ Added Transformer for Xcp
- ▶ Modified Dem and Dcm configurators to not create an additional configSet container if one already exists
- ▶ ASCCONFIGATORS-666 Fixed known issue: The DcmRteUsage is configured based on other parameters DcmDspDataUsePort and DcmDspRoutineUsePort
- ▶ Added support for IPV6 IP addresses to Transformers for Tcplp and SoAd
- ▶ Moved the available subfunction for diagnostic service DynamicallyDefineDataIdentifier from the configuration of the DataIdentifier (DID) to the configuration of the diagnostic service
- ▶ ASCCONFIGATORS-652 Fixed known issue: Dcm importer wrongly adds session and security levels for a DiagnosticService
- ▶ ASCCONFIGATORS-719 Fixed known issue: Dem configuration generates an error if DiagnosticDataIdentifier and DiagnosticDataElement have the same ShortName

Module version 2.6.9

2017-02-03

- ▶ ASCCONFIGATORS-651 Fixed known issue: Fixed Dcm error if a DID and one of its aggregated DiagnosticDataElement have the same SHORT-NAME
- ▶ ASCCONFIGATORS-662 The DcmDsdSubServiceSecurityLevelRefs for the SubServices of DiagnosticSecurityAccess will not be configured.
- ▶ Extended Transformer for SecOC to support the configuration of secured PDUs of type DcmIPdu
- ▶ Extended Transformer for SecOC to support the configuration of secured PDUs for which IPdu.Port.rxSecurityVerification is set to false
- ▶ Extended Transformer for ComM to check whether a given PNC ID actually lies within the specified PNC vector
- ▶ Extended Transformer for Tcplp to add limited broadcast address entries if they are needed for DHCP



- ▶ Created ECU parameter configuration in the Bsw module Dcm for UDS service `ReadDataByPeriodicIdentifier` from information contained in the diagnostic extract
- ▶ Extended Dcm to configure the `DcmDspDataUsePort` parameter concerning the cases described by `ServiceNeeds/DiagnosticValueNeeds.processingStyle`
- ▶ ASCCONFIGURATORS-665 Fixed known issue: Fixed Dem configurator to configure the `DemEventPriority` parameter
- ▶ Extended Transformer for Dcm to configure `DynamicallyDefineDataIdentifier`

Module version 2.6.8

2017-01-05

- ▶ ASCCONFIGURATORS-652 Fixed known issue: Fixed Dcm importer wrongly adds session and security levels for a `DiagnosticService`
- ▶ Extended Transformer for Can: Add support for the configuration of `CanObjectId` in multi-CanController scenarios
- ▶ Extended Transformer for EthIf to support `EthIfController`, `EthIfPhysController`, and `EthIfSwitch` configuration containers

Module version 2.6.7

2016-12-02

- ▶ ASCCONFIGURATORS-654 Fixed known issue: Transformer for DoIP does not configure all `DoIPTcp-Connection` containers
- ▶ Extended Transformer for Dcm to configure `ControlDTCSetting`
- ▶ Extended Transformer for IpduM to support multiplexed PDUs and secured PDUs In container PDUs
- ▶ Added Transformer for SecOC

Module version 2.6.6

2016-11-04

- ▶ Added Transformer for FrArTp
- ▶ Extended Transformers for Com and PduR to support bidirectional routing of PDUs and signals
- ▶ Extended Transformer for SoAd: `ProvidedServiceInstance` and `ConsumedEventGroup` elements are considered for the configuration of `SoAdSocketFramePriority`

Module version 2.6.5

2016-10-21



- ▶ ASCCONFIGURATORS-606 Fixed known issue: The parameter DcmDspDataInfoRef is not set and its container DcmDspData is not referenced by DIDs, if no Base Data Type can be determined
- ▶ Extended Transformer for Dcm to configure DiagnosticEcuReset
- ▶ ASCCONFIGURATORS-607 Fixed known issue: The parameter DcmDspRoutineSignalLength is not configured if the DiagnosticDataElement evaluates to a non-array element
- ▶ ASCCONFIGURATORS-608 Fixed known issue: Import ECU configuration erroneously configures SubServices for the Dcm RoutineControl service
- ▶ Extended Transformer for Tcplp to configure TcpIpAssignmentPriority
- ▶ Adapted Transformer for Nm: NmChannelId is not configured any more
- ▶ Adapted Transformer for Tcplp to support AUTOSAR 4.2.2 parameter structure
- ▶ Extended Transformers for PduR and SoAd to support PDU routing between <Bus>Tp and SoAd
- ▶ Extended Transformer for Sd to support AUTOSAR RFC 73286
- ▶ Added Transformer for EthSwt
- ▶ Extended Transformer for Dem to configure DiagnosticExtendedDataRecord

Module version 2.6.4

2016-09-09

- ▶ ASCCONFIGURATORS-614 Fixed known issue: Transformer for Dcm configures DcmDslProtocolRx-TesterSourceAddr incorrectly
- ▶ Extended Transformer for DoIP to support configurations without DiagnosticConnections in the imported system model
- ▶ Added Transformers for CanTSyn and FrTSyn
- ▶ Added Transformer for AUTOSAR 4.0 Dem: Added support for configuring DiagnosticTroubleCode, DiagnosticTroubleCodeGroup, DiagnosticEvent, DiagnosticEnableCondition, DiagnosticEnableConditionGroup and DiagnosticIndicator into Dem
- ▶ Extended Transformer for Dem to configure DiagnosticDataElements and DiagnosticFreeze-FrameS
- ▶ Extended Transformer for Dcm to configure DiagnosticReadDTCInformation
- ▶ Extended Transformer for Dcm to configure /DcmDspRoutine/DcmDspRoutineUsePort

Module version 2.6.3

2016-08-05

- ▶ ASCCONFIGURATORS-604 Fixed known issue: Configuration of vehicle announcement in DoIP assumes incorrect remote IP address



- ▶ ASCCONFIGURATORS-599 Fixed known issue: Default Buffer Assignment does not support CAN 2.0 and CAN FD PDUs with identical CAN Ids

Module version 2.6.2

2016-07-01

- ▶ ASCCONFIGURATORS-594 Fixed known issue: Upper layer of `UserDefinedIPdus` and `GeneralPurposeIPdus` is configured incorrectly in <Bus>If modules
- ▶ Extended Transformer for Tcp to configure at least one `TcpIpAddrAssignment` container for each `TcpIpLocalAddr` entry

Module version 2.6.1

2016-05-25

- ▶ Extended Transformer for SoAd to support `SoAdSocketConnection` elements without `shortLabel` attribute
- ▶ Extended Transformer for Can, Eth, Fr, Lin to support the configuration parameter set defined by the AUTOSAR 4.2.2 standard
- ▶ Extended Transformer for Dcm to configure `DiagnosticRoutineControl`

Module version 2.6.0

2016-04-29

- ▶ ASCCONFIGURATORS-569 Fixed known issue: Upper Layer for UUDT DcmIPdus is configured incorrectly in CanIf, LinIf, and FrIf

Module version 2.5.5

2016-04-01

- ▶ ASCCONFIGURATORS-557 Fixed known issue: Creation of the Dcm configuration from system description does not calculate the value of `DcmDspDataSize` correctly for non-array data instances
- ▶ Extended Transformers for Can and CanIf to create dedicated HOHs for CAN-FD Frames

Module version 2.5.4

2016-03-04



- ▶ ASCCONFIGURATORS-558 Fixed known issue: Transformer for LinIf does not support LinSlaveConfig elements without LinSlaveConfigIdent
- ▶ Extended Transformer for SoAd to configure SoAdSocketTcpKeepAliveProbesMax, SoAdSocketTcpKeepAliveInterval, and SoAdSocketTcpKeepAliveTime
- ▶ Extended Transformer for ComM to configure ComMPNCGatewayType

Module version 2.5.3

2016-02-05

- ▶ Added input data validity checks to Transformer for Dcm

Module version 2.5.2

2016-01-15

- ▶ ASCCONFIGURATORS-533 Fixed known issue: Transformer for Com configures ComSignalDataInvalidValue and ComSignalInitValue for UINT8_N signals incorrectly
- ▶ ASCCONFIGURATORS-534 Fixed known issue: Transformer for Sd creates unnecessary SdConsumedMethods and SdProvidedMethods configuration containers
- ▶ ASCCONFIGURATORS-537 Fixed known issue: Transformer for ComM configures ComMChannelId incorrectly
- ▶ Extended Transformer for LinIf: Add support for LinSlaveConfig elements
- ▶ ASCCONFIGURATORS-506 Fixed known issue: Transformers for PduR and Com do not conform to TPS_SYST_01056
- ▶ Extended Transformers: Resolved naming conflicts for PDUs received with the same CAN ID on different busses
- ▶ ASCCONFIGURATORS-535 Fixed known issue: Transformer for CanIf does not support CAN FD PDU configuration according to Autosar 4.2.2

Module version 2.5.1

2015-11-06

- ▶ ASCCONFIGURATORS-518 Fixed known issue: Configuration import into Com incorrectly reports error on inconsistent endianess in group signals
- ▶ Removed Transformers for ComXf, SomelpXf, E2EXf
- ▶ Adapted Transformer for PduR: Add routing paths for UserDefinedPdus and GeneralPurposePdus
- ▶ ASCCONFIGURATORS-525 Fixed known issue: ComM channel retrieval for Multiplexer PDUs leads to internal assertion failure



- ▶ Adapted Transformer for Dolp: Adaptations according to DolP related changes in AUTOSAR 4.2.2 system model
- ▶ Added Transformer for Dcm

Module version 2.5.0

2015-10-09

- ▶ Adapted Transformer for CanNm, UdpNm: Nm PDU user data byte length is now calculated by subtracting the non user data byte length from the total PDU byte length, removed support for obsolete parameter UdpNmBusLoadReductionEnabled
- ▶ Adapted Transformer for CanIf: Extended support for CAN Id range reception according to AUTOSAR RFC 66324
- ▶ Added support for Com datatype retrieval for DataTypePolicy "transformingISignal"
- ▶ Adapted Transformer for EthTSyn, StbM: Added support for the configuration of EthTSynGlobalTimeTx-Period, StbMOffsetTimeBase
- ▶ Adapted Transformer for CanNm: Implemented parameter mapping of CanNmCarWakeUpFilterEnabled, nmCarWakeUpRxEnabled according to AUTOSAR RFC 65423

Module version 2.4.4

2015-08-14

- ▶ Adapted Transformer for Tcplp: TcplpAddressType is now also configured for TcplpLocalAddr entries that have TcplpStaticIpConfig/TcplpStaticIpAddress set to ANY
- ▶ Adapted Transformer for UdpNm: UdpNmChannelConfig container is only configured for VLANs on which Nm PDUs are sent
- ▶ Adapted Transformer for SoAd: SoAdSocketConnectionGroup/SoAdPduHeaderEnable is set to true if at least one transmitted PDU is associated with a valid header id
- ▶ Added proxy UdpNm Transformer to package dreisoft.tresos.comimporter.api.transformer.asr41 since the extension of the UdpNm module expects the Transformer to reside in that package
- ▶ Adapted Transformer for Sd: SdInstance containers are only created for VLANs in which the configured ECU executes at least one Client Service or Server Service
- ▶ Adapted Transformer for ComXf, SomeIPXf, E2EXf: Transformer technologies that contain either "1" or "1.0.0" as version number are processed now

Module version 2.4.3

2015-06-19



- ▶ Adapted Transformers for SoAd and Sd to support seamless service migration/AUTOSAR RFC 61738

Module version 2.4.2

2015-05-22

- ▶ Update Transformer for Nm according to changes in AUTOSAR RFC 61777
- ▶ ASCCONFIGURATORS-467 Fixed known issue: Transformer for Com issues "zero length BigInteger" error message
- ▶ Added Transformers for EthTSyn, StbM
- ▶ Implemented ComDataType retrieval in Transformer for Com according to AUTOSAR RFC 65384
- ▶ ASCCONFIGURATORS-469 Fixed known issue: Transformer for IpduM configures IpduMSelectorField-Position in MostSignificantByteFirst selector fields incorrectly
- ▶ Added Transformer for E2EXf
- ▶ Added Transformer for ComXf
- ▶ Extended Transformers for Com, IPduM, PduR, CanIf, FrIf, SoAd, EcuC to support Container/Contained PDUs

Module version 2.4.1

2015-02-20

- ▶ Transformer for LinIf now configures LinIfComMNetworkHandleRef
- ▶ Transformer for FrIf now configures FrNm fan-in PDU according to FlexRay slot number, base cycle and cycle repetition parameters
- ▶ Transformers for Com and ComM now configure sent PNC IRA signals
- ▶ Transformers for EcuC, CanNm, FrNm, UdpNm, PduR, Com, ComM now configure PNC ERA PDUs and signals only for ACTIVE PNC gateways
- ▶ Added Transformer for SomelpXf

Module version 2.4.0

2015-01-07

- ▶ ASCCONFIGURATORS-430 Fixed known issue: Transformer for Com configures initial values and invalid values of UINT8_N signals incorrectly
- ▶ ASCCONFIGURATORS-440 Fixed known issue: Transformers stop with an error if direct Tp SDU of FrT-pConnection is not available



- ▶ Added support for 1..* cardinality of CanNmRxPdu containers in Transformer for CanNm
- ▶ Added support for PNC ERA PDU configuration in Transformers for EcuC, CanNm, FrNm, UdpNm, PduR, Com, ComM
- ▶ Added support for PNC Identifier configuration according to AUTOSAR RFC 52483 in Transformer for ComM

Module version 2.3.0

2014-10-03

- ▶ Made Sd Transformer robust against duplicated routing groups
- ▶ ASCCONFIGURATORS-423 Fixed known issue: Transformer for Com generates wrong ComSignalType for signals assigned to "Array" SwBaseType
- ▶ Extended Transformers to support AUTOSAR 4.2.1 system model files as input
- ▶ Added support for dynamic length signals in Com

Module version 2.2.2

2014-09-05

- ▶ Added Transformers for CanSM, FrSM, LinSM and EthSM
- ▶ Added Transformer for DoIP
- ▶ Added Transformer for LdCom

Module version 2.2.1

2014-08-07

- ▶ Adapt configuration of SdClientServiceActivationRef, SdServerServiceActivationRef according to AUTOSAR 4.1.3 Upstream Mapping
- ▶ Adapt Transformer for Eth so that the MAC address is configured according to AUTOSAR
- ▶ Added Transformer for UdpNm
- ▶ Transformer for Sd now configures Sd Control PDUs
- ▶ Transformer for PduR handles routing of Diagnosis PDUs according to AUTOSAR RFC 63555
- ▶ Transformer for Com now supports dataTypePolicy values other than "legacy"
- ▶ ASCCONFIGURATORS-387 Fixed known issue: FrTpTransformer configures FrTpRxPduPoolRef and FrTpTxPduPoolRef incorrectly



- ▶ ASCCONFIGURATORS-393 Fixed known issue: Transformer for CanTp configures identical symbolic names for received N-PDUs used for Data and FC at the same time
- ▶ ASCCONFIGURATORS-394 Fixed known issue: Transformer for FrIf configures incorrect PDU references to EcuC
- ▶ ASCCONFIGURATORS-391 Fixed known issue: PNC ISignalIPduGroups are only considered if the configured ECU instance directly references them
- ▶ Transformer for Sd now supports the configuration of combined Tcp/Udp services
- ▶ ASCCONFIGURATORS-399 Fixed known issue: Transformer for ComM sets ComMNmVariant to "NONE" for LIN Clusters
- ▶ ASCCONFIGURATORS-395 Fixed known issue: Transformers create User Data PDU containers for Nm PDUs containing signals that are not processed by the ECU
- ▶ Transformers for CanNm, FrNm, and UdpNm now configure the links to the associated ComM channels
- ▶ Transformers for ComM now configures ComMChannelId
- ▶ ASCCONFIGURATORS-396 Fixed known issue: Transformer for Nm does not configure NmComUserDataSupport
- ▶ Transformer for Com now configures ComGWMapping entries for group signals

Module version 2.2.0

2014-04-25

- ▶ ASCCONFIGURATORS-349 Fixed known issue: Transformer for PduR sets up incorrect routing paths for reversed FlexRay Tp SDUs
- ▶ ASCCONFIGURATORS-355 Fixed known issue: Transformer for Com does not create gateway mapping entries for signal groups
- ▶ ASCCONFIGURATORS-346 Fixed known issue: Transformer for LinIf configures AssignNad frame incorrectly
- ▶ ASCCONFIGURATORS-354 Fixed known issue: Transformer for LinIf reports an error if it encounters AssignFrameIdRange schedule table entries without PID
- ▶ Extended Transformers to support AUTOSAR 4.1.2 system model files as input; added Transformer for Sd

Module version 2.1.10

2014-01-17

- ▶ Integrated handling for CanTp N-PDUs assigned to multiple CanTp connections
- ▶ Integrated AUTOSAR 4.0 ComM Transformer

**Module version 2.1.9**

2013-10-11

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 2.1.8

2013-09-13

- ▶ Removed configuration of PN ERA PDUs which are not yet supported by CanNm and FrNm

Module version 2.1.7

2013-06-14

- ▶ ASCCONFIGURATORS-296 Fixed known issue: The configuration of one and the same CanNm PDU for sending and receiving results in a naming conflict
- ▶ Added support for CONTAINED-I-SIGNAL-I-PDU-GROUP-REF to detect PNC-enabled NmClusters
- ▶ Added support for routed-only Tp-SDUs in PduR
- ▶ Added configuration of CanNmPnFilterMaskByte in CanNm
- ▶ Added support for configuration of ComTxModeTrue for sent PDUs without PDU Timing in Com

Module version 2.1.6

2013-05-10

- ▶ ASCCONFIGURATORS-277 Fixed known issue: Com Transformer restricts signals to be contained at most once per FlexRay frame
- ▶ ASCCONFIGURATORS-279 Fixed known issue: CanNm Transformer issues exception if handed over Can Network contains Rx Nm PDUs but no CAN Frame Triggerings
- ▶ ASCCONFIGURATORS-276 Fixed known issue: If a PDU is sent and received at the same time and fan-out takes place for the sent PDU, the received PDU instance is configured incorrectly
- ▶ ASCCONFIGURATORS-283 Fixed known issue: LinTp Transformer produces name clashes when importing from AUTOSAR 3.x LinTp configurations
- ▶ ASCCONFIGURATORS-286 Fixed known issue: CanTp Transformer cannot handle CAN-TP-CONNECTIONS without TP-SDU-REF

Module version 2.1.5

2013-02-08



- ▶ ASCCONFIGURATORS-230 Fixed known issue: Transformer for AUTOSAR 4.0 Com configures signal invalid values and signal init values > 2^53-1 incorrectly
- ▶ ASCCONFIGURATORS-250 Fixed known issue: ComTxModeNumberOfRepetition is not configured correctly for AUTOSAR 4.0.3 Com configurations
- ▶ ASCCONFIGURATORS-254 Fixed known issue: LinIf Transformer uses wrong set of PDUs for retrieval of TxPdu fan-out information
- ▶ ASCCONFIGURATORS-256 Fixed known issue: PduR Transformer cannot handle <any bus>-PDU to Ethernet-PDU gateway mappings
- ▶ ASCCONFIGURATORS-258 Fixed known issue: FrIf Transformer does not configure FrIfFluster.gdBit according to upstream mapping
- ▶ Implemented support for TP-SDUs referenced by multiple PDU triggerings
- ▶ Removed configuration of <Net>NmPnFilterMaskByte due to unclear upstream mapping rule
- ▶ Added pure gateway routing support

Module version 2.1.4

2012-10-12

- ▶ AUTOSAR 4.0 FrIf/FrNm: Added support for FrNm Rx PDU fan-in
- ▶ ASCCONFIGURATORS-230 Fixed known issue: Transformer for Com does not create GroupSignal if SignalTriggering is missing

Module version 2.1.3

2012-09-14

- ▶ AUTOSAR 4.0 Com: Added support for Com Rx Signal DataFilters
- ▶ AUTOSAR 4.0 Com: Adapted ComSignalType configuration: Special handling of type BOOLEAN

Module version 2.1.2

2012-08-17

- ▶ AUTOSAR 4.0 Com: User-defined prefix is added to group signal containers
- ▶ AUTOSAR 4.0 SoAd: Adapted according to configuration changes
- ▶ AUTOSAR 4.0 Nm: Adapted according to configuration changes
- ▶ AUTOSAR 4.0 Com: Adapted configuration of parameter ComSignalType according to AUTOSAR 4.0 Rev 3 System Template



Module version 2.1.1

2012-06-15

- ▶ AUTOSAR 4.0 SoAd: Implemented configuration of `SoAdSocketConnectionGroup` containers
- ▶ AUTOSAR 4.0 Transformers: PDU Router fan-out support has been added

Module version 2.1.0

2012-05-16

- ▶ AUTOSAR 4.0 CanIf, CanTp: Adapted parameter configuration according to configuration changes in AUTOSAR 4.0 Rev 3

Module version 2.0.6

2012-04-13

- ▶ AUTOSAR 4.0 SoAd: Adaptation due to removed/obsolete parameter `SoAdPduHeaderEnable`

Module version 2.0.5

2012-03-27

- ▶ Added support for the configuration of partial networks for AUTOSAR 4.0 FrNm
- ▶ Integrated AUTOSAR 4.0 Transformers for modules SoAd, EthIf, Eth, TcpIp
- ▶ ASCCONFIGURATORS-187 Fixed known issue: AUTOSAR 4.0 Com: Fixed signal data type calculation algorithm

Module version 2.0.4

2012-02-17

- ▶ Removed obsolete AUTOSAR 3.x Transformers
- ▶ Added support for the configuration of partial networks for AUTOSAR 4.0 CanNm

Module version 2.0.3

2012-01-20

- ▶ Integrated AUTOSAR 4.0 Transformers for modules Lin, LinIf, and LinTp



- ▶ Added User Data Nm PDU support to the involved AUTOSAR 4.0 Transformers EcuC, PduR, Com, and CanNm, FrNm

Module version 2.0.2

2011-10-12

- ▶ Integrated AUTOSAR 4.0 Transformers for modules Fr, FrIf, CanTp, CanNm, FrTp, FrNm, Nm
- ▶ ASCCONFIGURATORS-138 Fixed known issue: FrNm: Fixed Rx/Tx Nm PDU handling

Module version 2.0.0

2011-09-02

- ▶ ASCCONFIGURATORS-86 Fixed known issue: IpduM: Transformer for AUTOSAR 3.x IpduM configures IPduMTxSelectorValue
- ▶ ASCCONFIGURATORS-85 Fixed known issue: IpduM, EcuC: Containers for Demultiplexed PDUs are only created if the Com module actually processes them
- ▶ Integrated AUTOSAR 4.0 Transformers for modules EcuC, Can, CanIf, IpduM, PduR, and Com

Module version 1.1.2

2011-04-08

- ▶ ASCCONFIGURATORS-51 Fixed known issue: LinIf: Fixed references to EcuC PDU collection
- ▶ Improvement CanTp: Support configuration of multicast CanTp connection channels

Module version 1.1.1

2011-03-11

- ▶ Improvement CanTp, CanIf, EcuC: Duplication of CanTp Tx N-PDUs has been introduced
- ▶ Improvement LinIf: Transformer configures LinIfFramePriority if provided via LDF importer
- ▶ Transformer for generic Nm was added
- ▶ Improvement Com: Support for zero bitsize signals has been added

Module version 1.1.0

2011-02-03



- ▶ Requirements tracing: Resolve unmapped tests, unmapped requirements
- ▶ Improvement CanNm, FrNm, CanIf, LinIf, Com: Transformers use new Meta-Model 6/AUTOSAR 3.1.4 System parameters for module configuration
- ▶ Improvement CanIf, Can: Transformers use better naming schema for configuring HOH containers

Module version 1.0.2

2010-11-18

- ▶ ASCCONFIGURATORS-28 Fixed known issue: AUTOSAR 2.1/3.x Com: Com data types calculated correctly from integral system model data types with open ranges

Module version 1.0.1

2010-10-08

- ▶ Improvement AUTOSAR 3.x LinIf: Conditional frames obtain LinIfInternalPdu as LinIfPduDirection during com imports
- ▶ ASCCONFIGURATORS-22 Fixed known issue: AUTOSAR 2.1/3.x Com: Bit offsets and length parameters of signal groups are correctly exported

Module version 1.0.0

2010-09-10

- ▶ First implementation of Configurators

3.3.5.2. New features

- ▶ No new features have been added since the last release.

3.3.5.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.5.4. Deviations

This module is not part of the AUTOSAR specification.



3.3.5.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ `ComSignalInitValue/ComSignalDataInvalidValue` when importing from FIBEX

Description:

When importing from FIBEX, initial and invalid values that have been defined as bit pattern that represent `FLOAT` values are interpreted as integral values and written to `ComSignalInitValue/ComSignalDataInvalidValue` as such.

Rationale:

Initial and invalid values are defined as `INTERNAL-CONSTRS` in FIBEX, which requires these values to be integral values.

- ▶ `CanTpChannel/CanTpChannelMode` when importing from FIBEX

Description:

When importing from FIBEX, only elements of `CanTpChannel` that contain a single `CanTpRxNSdu/CanTpTxNSdu` are created. Also, the `CanTpChannelMode` of these elements `CanTpChannel` is always set to `CANTP_MODE_HALF_DUPLEX`.

Rationale:

FIBEX 3.x only supports a single TP-CONNECTION per TP-CHANNEL.

- ▶ `ISignal.dataTypePolicy:networkRepresentationFromComSpec` requires either a `SenderReceiverToSignalMapping` or a `SenderReceiverToSignalGroupMapping`

Description:

An `ISignal` that has `dataTypePolicy` set to `networkRepresentationFromComSpec` requires a `SenderReceiverToSignalMapping` or a `SenderReceiverToSignalGroupMapping`. These mapping elements reference a `PortPrototype` which contains a `PortComSpec` that defines the NetworkRepresentation. A DataMapping other than `SenderReceiverToSignalMapping` or `SenderReceiverToSignalGroupMapping` is not supported.

- ▶ `ISignal.dataTypePolicy: Data type retrieval via networkRepresentationFromComSpec and ImplementationDataType` is only supported for plain `ISignal` elements

Description:

In the AUTOSAR System Template 4.2 Rev 2, [TPS_SYST_02006] and [TPS_SYST_02079] describe the data type retrieval via `ImplementationDataType` in the case that no `SenderComSpec` or `Receiver-`



ComSpec is available. This is currently only implemented for plain signals, not for ISignals that represent group signals.

- ▶ ComIPdu/ComIPduCounter is not configured

Description:

ComIPdu/ComIPduCounter configuration containers are not configured even if the imported AUTOSAR System Description file contains SIGNAL-I-PDU-COUNTER elements.

Rationale:

The ACG Com module does not support ComIPdu/ComIPduCounter configuration containers.

- ▶ ComIPdu/ComIPduReplication is not configured

Description:

ComIPdu/ComIPduReplication configuration containers are not configured even if the imported AUTOSAR System Description file contains SIGNAL-I-PDU-REPLICATION elements.

Rationale:

The ACG Com module does not support ComIPdu/ComIPduReplication configuration containers.

- ▶ ComIPdu containers are not configured for TP SDUs

Description:

ComIPdu configuration containers are only configured for SIGNAL-I-PDU elements which are either related to FRAME elements or to SOCKET-CONNECTION elements. If an SIGNAL-I-PDU element is only related to TP-CONNECTION elements, it won't be configured in Com.

Rationale:

The ACG Com module does not support the processing of TP SDUs.

- ▶ ComSignal/ComDataInvalidAction and ComSignalGroup/ComDataInvalidAction are not configured

Description:

ComSignal/ComDataInvalidAction and ComSignalGroup/ComDataInvalidAction are not configured even if the imported system model contains INVALIDATION-POLICY elements.

Rationale:

The ACG Com module does not require ComSignal/ComDataInvalidAction or ComSignalGroup/ComDataInvalidAction for configuration code generation.



- ▶ ComSignal/ComRxDataTimeoutAction **and** ComSignalGroup/ComRxDataTimeoutAction **are** not configured

Description:

ComSignal/ComRxDataTimeoutAction **and** ComSignalGroup/ComRxDataTimeoutAction **are** not configured even if the imported system model contains HANDLE-TIMEOUT-TYPE elements.

Rationale:

The imported system model can contain two or more HANDLE-TIMEOUT-TYPE elements that are related to one and the same ComSignal/ComRxDataTimeoutAction **or** ComSignalGroup/ComRxDataTimeoutAction parameter. Since these HANDLE-TIMEOUT-TYPE elements may contain different values, a configuration is not possible.

- ▶ FrController/FrFiFo is not configured

Description:

FrController/FrFiFo configuration containers are not configured even if the imported AUTOSAR System Description file contains FLEXRAY-FIFO-CONFIGURATION elements.

Rationale:

Not all types of FlexRay communication controllers can be configured using the parameters provided in FrController/FrFiFo. It is therefore required to configure FrController/FrFiFo by hand, while taking into account the type of FlexRay communication controller that is in use.

- ▶ No support for tx fan-out of PDUs that are contained in CONTAINER-I-PDU elements

Description:

While the AUTOSAR System Template 4.2 Rev 1 indicates that PDUs contained in CONTAINER-I-PDU elements can be subject to a tx fan-out in the PduR module, the Transformer for PduR currently does not support this feature.

Rationale:

The tx fan-out support for contained PDUs requires a dedicated extension of the Transformer for PduR which has not yet been implemented.

- ▶ AssignNAD frames referencing Lin slave nodes which are not declared in the imported LDF file lead to import errors

Description:

During LDF imports, it is required that all AssignNAD frames in the LDF file reference Lin slave nodes which are also present in the file. If this is not the case, an error is issued.



Rationale:

The AUTOSAR System Model, which serves as internal data storage during the import, does not provide the possibility to store an AssignNAD frame without referencing the related slave node at the same time. Since the EB tresos Studio importer framework is built on top of the AUTOSAR System Model, this parameter cannot be configured.

3.3.5.6. Open-source software

Configurators does not use open-source software.

3.3.6. Det module release notes

- ▶ AUTOSAR R4.0 Rev 3
- ▶ AUTOSAR SWS document version: 3.2.0
- ▶ Module version: 6.5.1.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.6.1. Change log

This chapter lists the changes between different versions.

Module version 6.5.1

2019-10-11

- ▶ Internal module improvement. This module version update does not affect module functionality.

Module version 6.5.0

2019-06-14

- ▶ Implemented Det_ReportRuntimeError() and Det_ReportTransientFault() according to ASR4.3.1

Module version 6.4.9

2018-10-26

- ▶ Internal module improvement. This module version update does not affect module functionality.



Module version 6.4.8

2018-06-22

- ▶ Added memory sections for the Det software component description
- ▶ Removed the function name from defensive programming assertion error messages

Module version 6.4.7

2018-02-16

- ▶ Added support for error reporting from multiple cores

Module version 6.4.6

2017-09-22

- ▶ Comply to MISRA-C:2012

Module version 6.4.5

2017-07-28

- ▶ Internal module improvement. This module version update does not affect module functionality.

Module version 6.4.4

2017-05-05

- ▶ Internal module improvement. This module version update does not affect module functionality.

Module version 6.4.3

2016-11-04

- ▶ Created a new defensive macro for precondition checking without evaluating the condition

Module version 6.4.2

2016-10-07

- ▶ Internal module improvement. This module version update does not affect module functionality.

Module version 6.4.1

2016-04-29



- ▶ Internal module improvement. This module version update does not affect module functionality.

Module version 6.4.0

2016-02-05

- ▶ Internal module improvement. This module version update does not affect module functionality.

Module version 6.3.3

2015-07-21

- ▶ ASCDET-348 Fixed known issue: Wrong definition of C/S DETService - ReportError interface (numerical value of E_OK) leads to errors during Rte verification

Module version 6.3.2

2015-06-19

- ▶ Reworked information about "Influence of the Det on production" in documentation (chapter "Background information")

Module version 6.3.1

2015-03-20

- ▶ The Det documentation is now delivered as a standalone document. This is no longer part of the diagnostic stack user's guide.
- ▶ Added support for optional generation of Service APIs according to AUTOSAR 4.2.1. Revised arxml templates for generation process.

Module version 6.3.0

2013-06-14

- ▶ Changed the top-level structure of the software-component description in the ARXML files from /AUTOSAR/Det to /AUTOSAR_Det.
- ▶ Added support for optional generation of Service APIs according to AUTOSAR 3.2

Module version 6.2.1

2012-06-18



- ▶ Updated to AUTOSAR R4.0.3

Module version 6.2.0

2012-03-16

- ▶ Added support for BSWMD generation
- ▶ Used standardized path and package structure in Det SWCD

Module version 6.1.0

2011-09-06

- ▶ Initial AUTOSAR 4.0 version

3.3.6.2. New features

- ▶ No new features have been added since the last release.

3.3.6.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- ▶ Defensive programming

Description:

The Det offers macros and functions for defensive programming including checking of pre- and post-conditions, invariants, unreachable code, and compile time assertions.

Rationale:

This extension enables defensive programming techniques to be used within BSW modules and is seamlessly integrated within the regular development error tracing as defined by AUTOSAR.

- ▶ Reported errors buffer

Description:

To support debugging without implementing project-specific logging facilities the module provides a buffer that stores reported errors. The buffer is available to a debugger. The buffer can be enabled and disabled via the ECU configuration. The buffer can be configured to either store the last or the first logged errors. The buffer facility is tailored only for debugger usage. A failsafe (reentrant) facility to access the buffer from a running system is not supported.



3.3.6.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- ▶ No interface to Dlt

Description:

The Det does not forward error reports to the Dlt module.

Requirements:

DET042, DET034

- ▶ AUTOSAR Debugging is not supported

Requirements:

DET030, DET032

- ▶ DET module's source code shall not include the file MemMap.h

Requirements:

DET006

- ▶ Det_Init() function parameter not used

Description:

The current functionality of Det_Init() is not using the ConfigPtr parameter.

Rationale:

The current implementation of the Det module is not using any configuration sets, so the ConfigPtr parameter does not have a purpose.

Requirements:

SWS_Det_00008

- ▶ Det_ConfigType not used

Description:

The current Det implementation does not use a configuration set.

Rationale:

Det module has only global variables initialized in Det_Init(), and none are set according to a Det config parameter.



Requirements:

SWS_Det_00210

- ▶ DETService does not implement a single service

Description:

DETService is implemented according to ASR-4.3.1 and it uses multiple services: Det_ReportError, Det_ReportRuntimeError.

Requirements:

DET201

- ▶ Different parent container for the DetReportRuntimeErrorCallout and DetReportTransientFaultCallout configuration parameters.

Description:

The containers for parameter definition DetReportRuntimeErrorCallout and DetReportTransientFaultCallout are moved to DetNotification from DetGeneral.

Rationale:

This deviation is required to achieve backward compatibility conforming to AUTOSAR release 4.2

Requirements:

ECUC_Det_00010, ECUC_Det_00011

- ▶ DetConfigSet and its child containers not available

Description:

The EB-provided SoftwareComponentList container is used instead of DetConfigSet. The same situation applies for the DetModuleId and DetInstanceId parameters, which have been replaced with ModuleId and InstanceId in the current implementation. DetModule and DetModuleInstance no longer exist.

Rationale:

This deviation is required to achieve backwards compatibility.

Requirements:

ECUC_Det_00012, ECUC_Det_00009, ECUC_Det_00008, ECUC_Det_00013, ECUC_Det_00007



3.3.6.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ No limitations are reported.

3.3.6.6. Open-source software

Det does not use open-source software.

3.3.7. EcuC module release notes

- ▶ AUTOSAR R4.0 Rev 3
- ▶ AUTOSAR SWS document version: 4.0.3
- ▶ Module version: 5.0.16.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.7.1. Change log

This chapter lists the changes between different versions.

Module version 5.0.16

2019-10-11

- ▶ Added support for meta data item PAYLOAD_TYPE_16

Module version 5.0.15

2019-04-18

- ▶ Added post build selectable support for Pdu container and its PduLength parameter

Module version 5.0.14

2019-01-25

- ▶ ASCECUC-185 Fixed known issue: Compilation error due to inconsistent use of memory section



Module version 5.0.13

2018-10-26

- ▶ Updated configuration allowing a reduction of inter-core calls
- ▶ ASCECUC-178 Fixed known issue: EcuC might not generate when EcucMetaHandlingEnabled is enabled

Module version 5.0.12

2018-08-24

- ▶ ASCECUC-162 Fixed known issue: Postbuild loadable and selectable cannot be configured

Module version 5.0.11

2018-06-22

- ▶ Added version compatibility check macro for EcuC library

Module version 5.0.10

2018-05-25

- ▶ Added support for `PduLengthTypeEnum` in the range `uint16, uint32`
- ▶ Added support for meta data handling
- ▶ Added generic upper layer TP-API library

Module version 5.0.9

2018-02-16

- ▶ Updated configuration parameters for multi-core and BSW distribution support

Module version 5.0.8

2017-09-22

- ▶ Added configuration parameters for multi-core and BSW distribution support

Module version 5.0.7

2016-05-25



- ▶ Added configuration parameter for post-build variants

Module version 5.0.6

2015-01-07

- ▶ Changed maximum length of Pdu (PduLength) to 65535 Bytes

Module version 5.0.5

2013-10-11

- ▶ Added `postBuildChangeable` attribute and reworked config variant to `PostBuild`

Module version 5.0.4

2013-06-14

- ▶ Implemented tresosDB-style entity names for `REF` values

Module version 5.0.3

2012-10-12

- ▶ Updated to AUTOSAR 4.0 Rev 3

Module version 5.0.2

2012-06-15

- ▶ Implemented switches for optional containers on same tabs as lists

Module version 5.0.1

2012-01-20

- ▶ Limited range of `PduLengthTypeEnum` and `PduIdTypeEnum` to `uint16`

Module version 5.0.0

2011-09-02



- ▶ Initial AUTOSAR 4.0 version

3.3.7.2. New features

- ▶ No new features have been added since the last release.

3.3.7.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- ▶ This module provides no EB-specific enhancements.

3.3.7.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- ▶ Configuration parameter `PduIdTypeEnum` is fixed to value `uint16`

Description:

The configuration parameter `PduIdTypeEnum` is fixed to value `uint16` and cannot be changed by users.

- ▶ Configuration parameter `PduLengthTypeEnum` has a limited configuration range

Description:

The configuration parameter `PduLengthTypeEnum` is only configurable in the range `uint16,uint32`.

- ▶ Non-compliant deviations in vendor-specific module definition file

Description:

The vendor-specific module definition file (VSMD) has non-compliant deviations to the AUTOSAR specification:

Violations against Rule EcucSws_2132: The attribute `postBuildChangeable` shall be only set to `true` if the corresponding container is a descendant subcontainer of the container that has the `multipleConfigurationContainer` attribute set to `true`.

- ▶ **StMD-Node:** /AUTOSAR/EcuC/EcucPduCollection/Pdu

Rationale: It shall be possible to change the Length of a Pdu at post build time, but the used AUTOSAR version has not foreseen a multiple Configuration Container.

Violations against Rule EcucSws_1007: For integer and float parameters the `MIN` values must be \geq and the `MAX` values \leq as in the StMD.



- ▶ StMD-Node: /AUTOSAR/EcuC/EcucPduCollection/Pdu/PduLength

Rationale: Due to new developments in AUTOSAR the maximum length of Pdus has been increased. The changes are implemented according to http://www.autosar.org/bugzilla/show_bug.cgi?id=57764 respectively ASR 4.2.1.

3.3.7.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ For this module no limitations are known.

3.3.7.6. Open-source software

Open-source software information is not available for this module.

3.3.8. HidWiz module release notes

- ▶ Module version: 1.1.19.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.8.1. Change log

This chapter lists the changes between different versions.

Module version 1.1.19

2019-06-14

- ▶ Extended ComTxPduHandleIdPushOperation to support SecOC *cryptographic* PDUs
- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.18

2018-01-19



- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.17

2017-12-15

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.16

2017-10-20

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.15

2017-03-03

- ▶ Extended `AdjacentLayerHandleIdPolicy` to support n-to-one routing paths in the PduR module

Module version 1.1.14

2017-01-05

- ▶ Added Handle ID policy for copying Handle ID values from other Handle ID parameters

Module version 1.1.13

2016-07-01

- ▶ Added Handle ID policy for all NM-PDUs which the UdpNm module is receiving
- ▶ Added Handle ID policy for CAN HOHs.

Module version 1.1.12

2016-04-29

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.11

2016-04-01



- ▶ Added Handle ID policy for all NM-PDUs which the CanNm module is receiving

Module version 1.1.10

2015-06-19

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.9

2015-02-20

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.8

2014-10-03

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.7

2014-04-25

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.6

2013-10-11

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.5

2013-06-13

- ▶ Added Handle ID policy for all N-PDUs which the CanTp module is receiving

Module version 1.1.4

2013-05-08



- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.1.3

2013-02-08

- ▶ Adapted `AdjacentLayerHandleIdPolicy` to use XPath function for `ZeroCost` Handle ID calculation

Module version 1.1.2

2013-01-11

- ▶ Introduced Handle ID policy for subgrouping of Handle IDs

Module version 1.1.1

2012-10-12

- ▶ Adapted `AdjacentLayerHandleIdPolicy` according to requirements of SoAd module

Module version 1.1.0

2012-09-14

- ▶ Added first version of generic `AdjacentLayerHandleIdPolicy`

Module version 1.0.2

2012-06-15

- ▶ Introduced Handle ID policy for grouping via XPath references

Module version 1.0.1

2012-03-16

- ▶ Implemented warning in the Handle ID policy that is reported if an EcuC PDU is referenced by multiple modules

Module version 1.0.0

2012-02-17



- ▶ Initial version of Handle ID Wizard. It provides a Handle ID policy for the Handle ID configuration of PDUs in the PduR module

3.3.8.2. New features

- ▶ No new features have been added since the last release.

3.3.8.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.8.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.8.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ For this module no limitations are known.

3.3.8.6. Open-source software

HidWiz does not use open-source software.

3.3.9. Make module release notes

- ▶ Module version: 4.0.22.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.9.1. Change log

This chapter lists the changes between different versions.



Module version 4.0.22

2019-10-30

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 4.0.21

2019-07-03

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 4.0.20

2019-03-06

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 4.0.19

2018-11-07

- ▶ ASCMAKE-625 Fixed known issue: fixed generation of plugin core path in generator `Make_cfg.mak` for MCAL plugins with variant suffix

Module version 4.0.18

2018-07-11

- ▶ Improved configuration of calls for tresos studio batch invocation from makefile
- ▶ Adapted generator `Make_cfg.mak` to also generate the location of each plugin through the make variables `<module>_CORE_PATH` to support multiple plugin locations
- ▶ ASCMAKE-619 Fixed known issue: Fixed order of inclusion of plugin specific `*_defs.mak` makefile to resolve actual values for file suffixes correctly
- ▶ ASCMAKE-622 Fixed known issue: fixed writing of compiler options containing double minus within the value to options file

Module version 4.0.17

2018-03-07

- ▶ ASCMAKE-614 Fixed known issue: fixed writing of compiler options containing blanks to options file

**Module version 4.0.16**

2017-10-18

- ▶ Added support for non-EB modules (without `_TS_` infix) in generator `Make_cfg.mak` for generation of list with enabled modules

Module version 4.0.15

2017-04-19

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 4.0.14

2016-11-23

- ▶ Fixed macro `unique` in `global.mak` to handle large lists of files properly
- ▶ Added support for Linux to the Make plugin
- ▶ Added make target `flat_src` to copy all source files into a flat directory
- ▶ Added make target `check_dups` to print warnings for duplicated header files
- ▶ Added Perl script `cppDeps.pl` to determine header file dependencies without calling the GNU cpp
- ▶ Added support for module generation output directories

Module version 4.0.13

2016-06-15

- ▶ Added support for object files that are provided and only linked to make build environment
- ▶ Added support for global libs that are built only once to make build environment which is disabled per default

Module version 4.0.12

2016-02-24

- ▶ Added support for file or library specific compiler option file to make build environment

Module version 4.0.11

2015-11-25



- ▶ Added support for C source files or assembler files with different file extension to make build environment

Module version 4.0.10

2015-07-08

- ▶ Fixed make build environment by re-adding make targets to generate preprocessed files

Module version 4.0.9

2015-03-11

- ▶ Fixed make build environment by re-adding post-build binary support

Module version 4.0.8

2015-01-21

- ▶ ASCMAKE-496 Fixed known issue: the build environment includes the source files of the CanTrcv plugin by fixing the extraction of the module name from the plugin name
- ▶ Improved make build environment to be more lean and faster by refactoring of `global.mak`, additionally added make targets to build a single file or a single library

Module version 4.0.7

2014-10-02

- ▶ Adapted formatting of release notes

Module version 4.0.6

2014-04-25

- ▶ Implemented sorting of module-specific variables in `Make_cfg.mak` in alphabetical order
- ▶ Added binary of GNU Make 4.0 to tools
- ▶ Updated ts5 test behavior to work with EB tresos Studio 15 in case .arxml files are merged to a single .-tdb file

Module version 4.0.5

2013-10-14



- ▶ Improved Make rules for parallel execution

Module version 4.0.4

2013-06-14

- ▶ Implemented a separate workspace for the generation step of each test in EB tresos Studio
- ▶ Added binary of GNU Make 3.82 to tools
- ▶ Changed log output of make to increase performance

Module version 4.0.3

2013-02-11

- ▶ Fixed usage of variable SOFTWARE_MODULES to remove errors on recursive variable references
- ▶ Added support for generation of AUTOSAR 3.1 SWCD files

Module version 4.0.2

2012-10-12

- ▶ Upgraded preprocessor for dependency generation from GNU CPP V4.6.2 to V4.7.0
- ▶ Added make target for creating post-build binary files
- ▶ Added support for generation of AUTOSAR 3.2 SWCD files

Module version 4.0.1

2012-03-16

- ▶ Upgraded preprocessor for dependency generation from GNU CPP V3.4.5 to V4.6.2

Module version 4.0.0

2011-09-02

- ▶ Initial AUTOSAR 4.0 version

3.3.9.2. New features

- ▶ No new features have been added since the last release.



3.3.9.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.9.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.9.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ For this module no limitations are known.

3.3.9.6. Open-source software

Open-source software information is not available for this module.

3.3.10. MemMap module release notes

- ▶ AUTOSAR R4.0 Rev 3
- ▶ AUTOSAR SWS document version: 1.4.0
- ▶ Module version: 1.3.4.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.10.1. Change log

This chapter lists the changes between different versions.

Module version 1.3.4

2019-10-11

- ▶ Internal module improvement. This module version update does not affect module functionality.

**Module version 1.3.3**

2019-06-14

- ▶ ASCMEMMAP-251 Fixed known issue: MemMap generation will fail with NullPointerException when the short-name of a MemMapAllocation is a substring of another MemMapAllocation.
- ▶ ASCMEMMAP-252 Fixed known issue: MemMap does not generate if SwAddrMethod and MemorySection short names are identical in different MemMapAllocations.
- ▶ Validation parameters can now be configured in MemMap to enable/disable coreScope, safety and mapping validations.

Module version 1.3.2

2019-02-15

- ▶ ASCMEMMAP-232 Fixed known issue: CoreScope validation fails if multiple options are set.

Module version 1.3.1

2018-10-26

- ▶ Warnings are reported when the safety level is invalid.
- ▶ ASCMEMMAP-214 Fixed known issue: MemMap generation fails at coreLocal validation.
- ▶ ASCMEMMAP-220 Fixed known issue: MemMap generation fails if RteImplementationRef is disabled.
- ▶ ASCMEMMAP-221 Fixed known issue: MemMap reports invalid warnings when `MemMapGenerateEmptyHeaderFile` is disabled.
- ▶ ASCMEMMAP-223 Fixed known issue: MemMap removes logical and arithmetic OR (|, ||) from pragma definitions.

Module version 1.3.0

2018-06-22

- ▶ ASCMEMMAP-140 Fixed known issue: MemMapHeaderFiles values are not exported into an EPC file.
- ▶ Error logged for overlapping MemMapAlignmentSelector.
- ▶ ASCMEMMAP-139 Fixed known issue: Error detection when <mod>_MemMap.h is used.
- ▶ One memory mapping header is created for BSW and SWC if their names are identical.
- ▶ The MemMapHeaderFiles and header files for each BSW Module are alphabetically ordered in MemMap.h. The generated memory sections macros are alphabetically ordered in MemMap header files.
- ▶ Warnings are reported when MemMapSwAddressMethodRef references a SwAddrMethod that has a different package than the same SwAddrMethod referenced by the memory sections.



- ▶ Warnings are reported when coreScope is invalid.
- ▶ Common_MemMap.h is no longer generated. The set of #pragma are generated under the start and stop of a memory section if a valid MemMap mapping exists.
- ▶ Warnings are reported when MemMapAddressingModeStart and MemMapAddressingModeStop are not configured, to inform the user of inactive mappings.
- ▶ ASCMEMMAP-201 Fixed known issue: Sections <Prefix>_(START/STOP)_CONFIG_DATA are never validated.
- ▶ The <SWC>_MemMap.h file for a software component which has only one implementation is now generated even if a corresponding RteSwComponentType is not found in the Rte configuration.

Module version 1.2.7

2018-02-16

- ▶ Internal module improvement. This module version update does not affect module functionality.

Module version 1.2.6

2017-09-22

- ▶ Memory sections with and without SEC will be generated in order to ensure compatibility to Autosar 4.0.2.
- ▶ ASCMEMMAP-131 Fixed known issue: Prefix <snp> generated from MemorySection shortName instead of symbol.

Module version 1.2.5

2017-03-31

- ▶ Generate for each module a MemMap file compose from the Module ShortName and MemMap

Module version 1.2.4

2016-08-05

- ▶ ASCMEMMAP-60 Improved: Improve operation status messages.

Module version 1.2.3

2016-05-25

- ▶ Internal module improvement. This module version update does not affect module functionality.

**Module version 1.2.2**

2016-03-29

- ▶ ASCMEMMAP-105 Fixed known issue: The mapping between the SwAddrMethod and the MemMapAddressingModeSet is not performed when their options match.
- ▶ ASCMEMMAP-101 Improved: A Null-Pointer-Exception is thrown if a BSW-Implementation or a SWC-Implementation has no ResourceConsumption.

Module version 1.2.1

2016-02-05

- ▶ ASCMEMMAP-55 Fixed known issue: Validate correct usage of section macros (no START preceded by a START, no STOP without the corresponding START).
- ▶ ASCMEMMAP-90 Fixed known issue: Empty <SWC>_Memmap.h files are still created.
- ▶ ASCMEMMAP-93 Fixed known issue: The M4 macros are not replaced in MemMap.h file.

Module version 1.2.0

2015-11-12

- ▶ ASCMEMMAP-72 Fixed known issue: The MemMap module does support MemMapSpecificMappings for MemorySections defined for software components.
- ▶ ASCMEMMAP-78 Fixed known issue: The MemMap module does support multiple BswImplementations per BswModuleDescription.
- ▶ Empty <SWC>_Memmap.h and MemMap.h files can be prevented from being generated through the 'MemMapGenerateEmptyHeaderFile' configuration parameter.

Module version 1.1.4

2015-07-08

- ▶ ASCMEMMAP-70 Fixed known issue: The MemMap module does not always report an error if the alignment selector is invalid.
- ▶ Fixed C style violation by adding newline at end of header template file.

Module version 1.1.3

2015-03-11

- ▶ ASCMEMMAP-64 Fixed known issue: Allow mapping of memory sections without alignment attribute.



- ▶ Improved generation of MemMap header files by creating an empty output header file if a SWC does not contain any memory sections.

Module version 1.1.2

2014-10-02

- ▶ Improved Java code by removing compiler warnings.

Module version 1.1.1

2014-04-25

- ▶ ASCMEMMAP-46 Fixed known issue: Missing MEMMAP_ERROR definition in the generated memmap file for SWC.
- ▶ ASCMEMMAP-50 Fixed known issue: The MemMap generator does not generate memory mapping macros if the configured alignment selector does not match the alignment attribute of the corresponding SwAddrMethod.
- ▶ ASCMEMMAP-49 Fixed known issue: The MemMap generator does not filter correctly if a constraint is defined but no elements to be filtered exist.

Module version 1.1.0

2013-10-11

- ▶ ASCMEMMAP-33 Fixed known issue: The MemMap generator does not generate the memory sections correctly if a MemMapGenericMapping is used.
- ▶ ASCMEMMAP-35 Fixed known issue: The MemMap generator does not create the vendor API infix correctly.
- ▶ ASCMEMMAP-34 Fixed known issue: The MemMap generator does not verify the MemMapAlignmentS-elector correctly.

Module version 1.0.0

2013-06-21

- ▶ Initial release of the MemMap module.

3.3.10.2. New features

- ▶ No new features have been added since the last release.



3.3.10.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- ▶ This module does not provide any EB-specific enhancements.

3.3.10.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- ▶ No support for symbol check

Description:

The memory mapping header files does not check if they have been included with a valid memory mapping symbol.

Requirements:

EB_Custom_Valid_MemoryMapping_Symbol

3.3.10.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ For this module no limitations are known.

3.3.10.6. Open-source software

MemMap does not use open-source software.

3.3.11. PbcfgM module release notes

- ▶ Module version: 1.2.20.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.11.1. Change log

This chapter lists the changes between different versions.



Module version 1.2.20

2019-10-11

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.19

2019-06-14

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.18

2019-02-15

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.17

2018-10-26

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.16

2018-06-22

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.15

2018-02-16

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.14

2017-09-22

- ▶ Removed license checks for binary post-build feature

Module version 1.2.13

2017-07-28



- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.12

2017-03-03

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.11

2016-11-04

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.10

2016-10-07

- ▶ Implemented non-functional code improvements to avoid compiler warning

Module version 1.2.9

2016-05-25

- ▶ Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeader-File`

Module version 1.2.8

2016-01-15

- ▶ Extended license checks for binary post-build feature to cover new license string `EB_BASE_PB_GEN-BINARY` as well.

Module version 1.2.7

2015-11-06

- ▶ Implemented non-functional code improvements to avoid compiler warning

Module version 1.2.6

2015-06-19



- ▶ Implemented non-functional code improvements to avoid compiler warning

Module version 1.2.5

2015-02-20

- ▶ Corrected naming of outdated memory sections
- ▶ Added support for variant specific configuration generation enabling selectable post-build configurations

Module version 1.2.4

2015-01-07

- ▶ Added support for configurable mapping of <Mod>_IsValidConfig function to dedicate memory section

Module version 1.2.3

2014-10-03

- ▶ ASCPBCFGM-100 Fixed known issue: PbcfgM configuration generation fails for modules with multiple instances

Module version 1.2.2

2014-04-25

- ▶ ASCPBCFGM-75 Fixed known issue: PbcfgM module generates files that cause a compiler error on case sensitive file systems
- ▶ Added support for EB MCAL modules
- ▶ Changed config class of parameter `PbcfgMBinarySupportEnable` from `PreCompile` to `PostBuild`
- ▶ ASCPBCFGM-87 Fixed known issue: PbcfgM module does not compile if function call tracing using Dbg is enabled for the API function `PbcfgM_GetConfig()`
- ▶ Updated list of modules to those which actually support initialization via PbcfgM

Module version 1.2.1

2013-10-30

- ▶ Changed the configuration class of the `PbcfgMBswModuleRef` parameter to pre-compile
- ▶ Added check enforcing an implementation config variant of post-build for modules referenced by `PbcfgMBswModuleRef`



Module version 1.2.0

2013-10-11

- ▶ Added support of binary code generation
- ▶ Added configuration parameter `PbcfgMConstCfgAddress`

Module version 1.1.0

2013-06-14

- ▶ Added checking of published information signature to prevent loading of incompatible post-build configuration

Module version 1.0.2

2013-02-08

- ▶ Implemented the custom platform signature check

Module version 1.0.1

2012-10-19

- ▶ ASCPBCFGM-15 Fixed known issue: A linker error occurs if modules are referenced that are not configured to be relocatable

Module version 1.0.0

2012-06-26

- ▶ Initial version of EB Post-Build Configuration Manager `PbcfgM`

3.3.11.2. New features

- ▶ No new features have been added since the last release.

3.3.11.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.11.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.11.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ For this module no limitations are known.

3.3.11.6. Open-source software

Open-source software information is not available for this module.

3.3.12. Platforms module release notes

- ▶ Module version: 3.0.1.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.12.1. Change log

This chapter lists the changes between different versions.

Module version 3.0.1

2018-07-11

- ▶ Updated compiler options:
 - ▶ Added `-fno-ident`
 - ▶ Added `-Wpedantic`
 - ▶ Added `-Wno-unused-local-typedefs`
 - ▶ Removed `-pipe`
- ▶ Updated compiler options: Replaced `-Wstack-usage=200` by `-fstack-usage` to avoid unnecessary warnings and get stack usage information from all files

**Module version 3.0.0**

2017-09-22

- ▶ Added rule for manifest file to prevent User Access Control warning
- ▶ Added support for GNU C Compiler version 6.2.0
- ▶ Added CIF library
- ▶ Added Compiler abstractions for the asc_Crypto and asc_Crylf modules
- ▶ removed compiler specific files and compiler settings, they are moved to the new Compiler module update module version 3.0.0

Module version 2.1.0

2015-07-08

- ▶ Add templates for MemMap.h and Compiler_Cfg.h.
- ▶ Use public OS API for concurrent access locks
- ▶ Add support for GNU C Compiler version 4
- ▶ Make Compiler.h compatible to AUTOSAR release 4.0 [R001]
- ▶ Shortened output messages of make calls
- ▶ Use .i suffix for preprocessor files when using the gcc toolchain
- ▶ Added support for CONSTP2FUNC compiler abstraction macro.
- ▶ Improved make build environment to be more lean and faster by updating makefiles to refactoring of global.mak of Make plugin
- ▶ Added compiler abstraction for module SecOC.
- ▶ Added support for GNU C Compiler version 4.8.1
- ▶ Removed support for older GNU C Compilers (before version 4.5.1)
- ▶ Added support for Dlt module

Module version 2.0.0

2010-02-11

Module version 1.0.2

2010-10-08

- ▶ Make Compiler.h compatible to AUTOSAR release 4.0 [R001]



Module version 1.0.1

2010-04-30

- ▶ Add templates for MemMap.h and Compiler_Cfg.h.
- ▶ Use public OS API for concurrent access locks
- ▶ Add support for GNU C Compiler version 4

Module version 1.0.0

2010-01-05

- ▶ Initial setup of Platform module for WIN32X86.

3.3.12.2. New features

- ▶ No new features have been added since the last release.

3.3.12.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.12.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.12.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ For this module no limitations are known.

3.3.12.6. Open-source software

Open-source software information is not available for this module.



3.3.13. SvcAs module release notes

- ▶ Module version: 1.2.13.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.13.1. Change log

This chapter lists the changes between different versions.

Module version 1.2.13

2019-10-11

- ▶ ASCSVCAS-383 Fixed NullPointerException which occured when Calculate Service Needs wizard is exceeded for EcuM with name different than EcuM_Config_0.

Module version 1.2.12

2019-06-14

- ▶ The SvcAs shall provide support for Preconditions
- ▶ The SvcAs will print more detailed information when there will be problem with processing. Change severity of problem with processing from warning to error.

Module version 1.2.11

2019-02-15

- ▶ The SvcAs shall provide AccessingApplications for Oslsr.IsrEntry and IsrNames for OsResource.ResourceEntry.
- ▶ ASCSVCAS-340 Fixed known issue: SvcAs does not correctly delete obsolete containers with name part from another name of a requested container.
- ▶ ASCSVCAS-342 Fixed known issue: SvcAs does not correctly delete obsolete refs when their container is still requested.
- ▶ ASCSVCAS-326 The SvcAs shall provide a detailed status for parameters that are disabled from schema but they have requested values.

Module version 1.2.10

2018-10-26

**Module version 1.2.9**

2018-09-13

- ▶ ASCSVCAS-306 Fixed OsAppScheduleTableRef references that were not cleaned up correctly
- ▶ ASCSVCAS-311 Fixed os AccessingApplication references that were not cleaned up correctly.
- ▶ ASCSVCAS-310 Fixed known issue: Service Needs wizard deletes manually edited and imported Com-Notification.
- ▶ Changed SvcAs to not delete com containers anymore.
- ▶ Changed SvcAs to not set two times XfrmVariableDataPrototypeInstanceRef in single run.

Module version 1.2.8

2018-06-22

- ▶ Included in the PushOperation the TP callback functions for LdCom. Now, the names for these functions will be set by SvcAs if they are requested.
- ▶ Added NvMUserHeader name to the Svc calculation.
- ▶ ASCSVCAS-296 Fixed known issue: Service Needs wizard deletes manually edited EcuMFlexModule-ConfigurationRef references.

Module version 1.2.7

2018-02-16

- ▶ ASCSVCAS-245 Fixed known issue: XfrmVariableDataPrototypeInstanceRef for XfrmMappings is enabled but with no values set.
- ▶ Improved run time execution for service needs assistant. The process of writing the status for service needs assistant was slowed down by an implementation of Tresos itself. The problem was solved by using another approach for adding the status.
- ▶ Changed the value set for DemOperationCycleRef in order to take the same name as DemOperationCycle. Until now, Service Needs calculated a default value (hardcoded) for this param every time, no matter what.
- ▶ ASCSVCAS-260 Fixed known issue: Service Needs wizard toggles between enable/disable and doesn't correctly clean up obsolete elements.

Module version 1.2.6

2017-08-09

- ▶ Added osSpinlockMethod to OsSpinlock configuration.



- ▶ ASCSVCAS-234 Fixed known issue: The extra data from Importer Info is missing when calculating Service Needs.

Module version 1.2.5

2017-03-03

- ▶ The SvcAs shall push init/invalid values for ComGroupSignals and ComRxDataTimeoutAction for Com-GroupSignals/ComSignals

Module version 1.2.4

2016-11-04

- ▶ RTE schedule tables are created only if the received requests contain less than 5000 distinct expiry points. The threshold value of 5000 is sometimes exceeded because of duplicate expiry points and therefore the schedule tables were not created. Now any two expiry points are considered distinct only if they have different names, otherwise one of them is ignored
- ▶ Updated the NvM push operation so that the value of `blockUseSyncMechanism` decides if `readRamBlockFromNvCallback` and `writeRamBlockToNvCallback` are enabled. Also, if `blockUseSyncMechanism` is missing, it is set to true if the other two are available, and false if at least one is missing
- ▶ Disabled the `OsIoc` container if there are no `OsIocDataTypeIncludeHeader` and `OsIocCommunication`

Module version 1.2.3

2016-05-02

- ▶ Internal module improvement. This module version update does not affect module functionality

Module version 1.2.2

2016-04-29

- ▶ Updated the Os IOC channel request (`os.iocChannels_1.xsd`) so that it contains new configuration parameters: `senderApiIsTrapping`, `dataTypeType/variableLength`, `intraCoreLockType`, `useInterCoreLock` which allow to set the respective parameters for each `OsIoc`

Module version 1.2.1

2016-04-01



- ▶ Added support to configure parameter `XfrmOsApplicationRef` for each Xfrm implementation mapping to the Service Needs Calculator
- ▶ Fixed the discovery mechanism for affected nodes of Calculate Service Needs wizard

Module version 1.2.0

2016-02-08

- ▶ Updated the Calculate Service Needs wizard to store the list of affected nodes in importer information. This information is used by cleanup for future runs
- ▶ Added support to configure LdCom PDUs to the Service Needs Calculator

Module version 1.1.7

2015-11-06

- ▶ Updated the NvM block request (`nvm.blocks_1.xsd`) to contain the optional parameters `initBlockCallback`, `readRamBlockFromNvCallback`, and `writeRamBlockToNvCallback` which allow to set the parameters `NvMInitBlockCallback`, `NvMReadRamBlockFromNvCallback`, and `NvMWriteRamBlockToNvCallback` respectively for each `NvMBlockDescriptor`
- ▶ Added support to configure Xfrm implementation mappings to the Service Needs Calculator
- ▶ Updated the NvM block request (`nvm.blocks_1.xsd`) so that all values are optional. Added parameter `blockUseSyncMechanism` to set `NvMBlockUseSyncMechanism`

Module version 1.1.6

2015-02-27

- ▶ Added support to configure Os spinlocks to the Service Needs Calculator

Module version 1.1.5

2014-04-25

- ▶ ASCSVCAS-135 Fixed known issue: The SvcAs calculates the offset for expiry points in picoseconds ($1*10^{12}$). Smaller values are truncated

Module version 1.1.4

2013-05-08



- ▶ Updated module to be ready for mass production

Module version 1.1.3

2013-02-08

- ▶ Extended the resolution of `OsSecondsPerTick` by the factor 1000

Module version 1.1.2

2012-10-12

- ▶ Updated the NvM block request (`nvm.blocks_1.xsd`) to contain the optional parameter `singleBlockCallback` which allows to set the parameter `NvMSingleBlockCallback` for each `NvMBlockDescriptor`

Module version 1.1.1

2012-06-14

- ▶ Updated the Service Needs Calculator to properly handle the parameters `NvMBlockCrcType`, `NvMRamBlockDataAddress`, `NvMRomBlockDataAddress`, `NvMSelectBlockForReadAll` and `NvMSelectBlockForWriteAll` of module NvM as optional parameters
- ▶ Changed the Service Needs Calculator to not warn if the parameters `NvMUUserProvidesSpaceForBlockAndCrc` and `NvMExtraBlockChecks` of module NvM are not existent
- ▶ Updated the Service Needs Calculator to set a configuration parameter properly if the parameter is optional and disabled

Module version 1.1.0

2012-03-16

- ▶ Changed the Service Needs Calculator to not override imported data
- ▶ Changed the Service Needs Calculator to no longer configure exclusive areas and main functions in the SchM module since the SchM module is now part of the Rte

Module version 1.0.1

2011-09-30

- ▶ Changed the Service Needs Calculator to not calculate the parameter `NvMNvBlockBaseNumber` since it is now independent of the `NvMNvramBlockIdentifier`



- ▶ Adapted NvM service provider to schema changes of the NvM module

Module version 1.0.0

2011-09-02

- ▶ Initial AUTOSAR 4.0 version

3.3.13.2. New features

- ▶ No new features have been added since the last release.

3.3.13.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.13.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.13.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ The Service Needs Calculator has the following limitations:
 - ▶ Not all init functions are automatically allocated.

Description:

In AUTOSAR 4.0, only the modules Det, Mcu, Port, Dio, Gpt, Wdg, WdgM, Adc, Icu, and Pwm are initialized by the EcuMFlex. The other modules are initialized by the BswM module. The Service Needs Calculator cannot configure the init functions. The init functions have to be called in C-callout functions.

- ▶ The Service Needs Calculator does not configure the right size for NvM blocks.

Description:



When NvM blocks are allocated, the Service Needs Calculator is not able to determine the correct block size in all cases. During compilation, the NvM module checks if the configured block size equals the required size by using the `sizeof` operator. If the values do not match, an error occurs. You must update the block size in the NvM configuration manually.

Rationale:

The block size depends on the configuration of the compiler.

- ▶ The Service Needs Calculator does not configure the right size for NvM blocks.

Description:

When NvM blocks are allocated, the Service Needs Calculator is not able to determine the correct block size in all cases. During compilation, the NvM checks if the configured size equals the required size by using the `sizeof` operator. If the values do not match, there will be an error. Then you must manually update the block size in the NvM configuration.

Rationale:

The block size is configuration and compiler-dependent.

3.3.13.6. Open-source software

Open-source software information is not available for this module.

3.3.14. Workflows module release notes

- ▶ Module version: 2.3.1.B271942
- ▶ Supplier: Elektrobit Automotive GmbH

3.3.14.1. Change log

This chapter lists the changes between different versions.

Module version 2.3.1

2018-10-26

- ▶ Fixed incorrect entry in SoftwareComponents.arxml



Module version 2.3.0

2018-07-11

- ▶ Reworked memory mapping sections of software components and updated workflows

Module version 2.2.8

2018-05-15

- ▶ Aligned post-build workflows with changes in user build environment (Make)
- ▶ Fixed array out-of-bounds access

Module version 2.2.7

2015-11-25

- ▶ Add IP Stack workflow

Module version 2.2.6

2015-06-19

- ▶ Removed Dbg workflow
- ▶ Corrected symbols of BswM user callouts

Module version 2.2.5

2015-03-17

- ▶ Added workflows for memory protection demos

Module version 2.2.4

2014-05-08

- ▶ Updated period time for CyclicEvent of SWC CyclicCounter
- ▶ Added system description files for the post-build workflows

Module version 2.2.3

2013-10-24

- ▶ Finalized post-build workflow

**Module version 2.2.2**

2013-06-26

- ▶ Added workflow for Lin
- ▶ Added workflow for Debug & Trace

Module version 2.2.1

2013-02-27

- ▶ Added Ethernet system description to the supplements
- ▶ Changed name of *AutoCore Workflow Basic* to *AutoCore Workflow Rte*
- ▶ Added instruction to *AutoCore Workflow Memory Stack* that also `NvMMultiBlockCallback` shall be enabled in NvM configuration
- ▶ Made clear linkage among the workflows

Module version 2.2.0

2012-10-26

- ▶ Improved and corrected workflow for FlexRay stack
- ▶ Removed SWC source code-style warning violations
- ▶ Changed the implementation so that the `counterValue` variable is incremented only each 10th cycle call for derivatives other than `_X86`
- ▶ Added LIN system description to the supplements
- ▶ Added a workflow for the configuration of the Lin stack (DRAFT)
- ▶ Added recommendation within Can, FlexRay, Lin workflows that `ComMDcmUsage=false`
- ▶ Added a workflow for post-build configuration (DRAFT)

Module version 2.1.1

2012-06-29

- ▶ Added a workflow for the configuration of the FlexRay stack

Module version 2.1.0

2012-03-16

- ▶ Added memory stack workflow



- ▶ Added a BswM EPC fragment for configuring the BswM in a way that it communicates the current mode to the SWCs
- ▶ Updated the system description file to schema version 4.0.3 and reworked package structure
- ▶ Added shutdown support to the demo software components
- ▶ Added storage of counter value in NVRAM to the demo software components
- ▶ Added a DBC file for the COM stack configuration, modified CAN stack workflow to support DBC and FIBEX files

Module version 2.0.0

2011-09-30

- ▶ Initial AUTOSAR 4.0 version

3.3.14.2. New features

- ▶ No new features have been added since the last release.

3.3.14.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.14.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.14.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ For this module no limitations are known.

3.3.14.6. Open-source software

Open-source software information is not available for this module.



4. ACG8 Base user's guide

4.1. Overview

The ACG8 Base user's guide provides information about common modules that are always available as part of the EB tresos AutoCore Generic product.

- ▶ [Section 4.2, “Det module user guide”](#) describes the concept of the Development Error Tracer module and how to configure it.
- ▶ [Section 4.3, “MemMap module user guide”](#) describes the concept of the Memory Mapping module and how to configure it.
- ▶ [Section 4.4, “Atomics module user's guide”](#) describes the concept of the Atomics module and how to use it.
- ▶ [Section 4.5, “Application template and application demos”](#) describes an application template and example programs (application demos).
- ▶ [Section 4.6, “Build environment”](#) describes the build environment and how to adapt this for your project needs.
- ▶ [Section 4.7, “Service Needs Calculator”](#) describes an integration tool to assist in the configuration of module dependencies known as Service Needs in AUTOSAR.

4.2. Det module user guide

4.2.1. Overview

This user guide provides you with Det-specific information:

- ▶ [Section 4.2.2, “Background information”](#) explains the concepts of the Det.
- ▶ [Section 4.2.3, “Configuring the Det module”](#) provides instructions on how to configure the Det.

For a list of module IDs of the basic software modules, see [Section 4.2.3.1, “List of module IDs of basic software modules”](#).

4.2.2. Background information

The Default Error Tracer Det is used for reporting development and run-time errors and also transient faults. Errors and faults are identified by all basic software modules and reported to the Det.



4.2.2.1. Handling of error reports

The `Det` handles the error reports depending on its configuration in one of the following ways:

▶ *Internal logging mode*

In internal logging mode, errors are written in a buffer of configured size if the functions `Det_ReportError()`, `Det_ReportRuntimeError()`, or `Det_ReportTransientFault()` are called.

▶ *Breakpoint logging mode*

In breakpoint logging mode, the system is halted during the call of `Det_ReportError()`, `Det_ReportRuntimeError()`, and `Det_ReportTransientFault()`. The `Det` cannot automatically activate breakpoints. You must manually set a breakpoint in the functions `Det_ReportError()`, `Det_ReportRuntimeError()`, or `Det_ReportTransientFault()`. Consult the specific manual of the used debugger/emulator equipment for further details.

Breakpoint handling depends on the architecture and the toolchain and may not be supported on every architecture.

▶ *Callback function*

Additionally to the logging modes, a callback function can be enabled and configured. There are three types of callback functions:

- ▶ `Det_ReportError()` calls first the function configured in the **DetErrorHook** list.
- ▶ `Det_ReportRuntimeError()` calls first the function configured in **DetReportRuntimeErrorCallout** list.
- ▶ `Det_ReportTransientFault()` calls first the function configured in **DetReportTransientFaultCallout** list.

The prototype of each callback function type is the same as its corresponding function: `Det_ReportError()`, `Det_ReportRuntimeError()` or `Det_ReportTransientFault()`. Thus, the arguments are just passed on.

You must declare the callback function in a header file that you provide. You must provide the name of the header file in the configuration.

4.2.2.2. Impact of the `Det` on production

Before production starts, you should be sufficiently sure that no more development errors are reported.

During production, the `Det` can:

- ▶ *stay switched on*. In this case, you should define a procedure for each error type that could occur. As development errors, run-time errors, and transient faults could result in a data overwrite or other undefined behavior, a reset could be an appropriate behavior.



- ▶ *be switched off* for optimization of ROM size and run-time. However, after any software change, the `Det` should be switched on again temporarily to check if development errors, run-time errors or transient faults occur.

The EB tresos AutoCore is designed in a way that, if there are no development errors, the functional behavior with the `Det` switched on is the same as with the `Det` switched off, except that some verification code is removed that is only relevant for development errors.

4.2.2.3. Support for error reporting from multiple cores

The `Det` supports error reporting from multiple cores.

The global indexes `Det_WriteIndex_DevelopmentError`, `Det_UsedSlots_DevelopmentError`, `Det_ErrorLost_DevelopmentError`, and the `Det_ErrorBuffer` are atomically accessed and are protected against any possible data corruption resulting from concurrently calling `Det_ReportError`.

The global indexes `Det_WriteIndex_RuntimeError`, `Det_UsedSlots_RuntimeError`, `Det_ErrorLost_RuntimeError`, and the `Det_ErrorBuffer` are atomically accessed and are protected against any possible data corruption resulting from concurrently calling `Det_ReportRuntimeError`.

The global indexes `Det_WriteIndex_TransientFault`, `Det_UsedSlots_TransientFault`, `Det_FaultLost_TransientFault`, and the `Det_ErrorBuffer` are atomically accessed and are protected against any possible data corruption resulting from concurrently calling `Det_ReportTransientFault`.

The multi-core capable `Det` module is implemented as one instance with a common buffer. This `Det` instance can be mapped to any `Os` application or core.

The multi-core support of `Det_ReportError`, `Det_ReportRuntimeError` or `Det_ReportTransientFault` is only enabled if more than one `Os` core is configured.

4.2.2.4. Software component description

This section describes the data types and interfaces provided by the software component description of the `Det`. For detailed configuration parameter and API descriptions, see [Chapter 5, “ACG8 Base module references”](#).

Exactly one `Det` service port is assigned to each software component.

It is also possible to specify multiple module IDs for one software component. This results in the generation of multiple `Det` service ports.

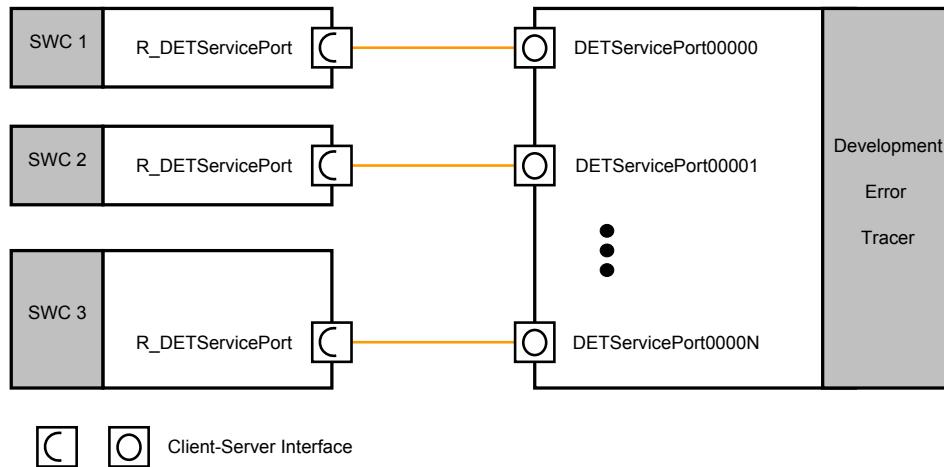


Figure 4.1. Overview of the software component description of the Det

4.2.2.4.1. Data types

Element	IntegerType
Type	UInt16
Range	256 - 65535
Description	This type defines the module ID of the software component description. The module ID is modeled as port-defined argument value.

Table 4.1. ModuleId

Element	IntegerType
Type	UInt8
Range	0 - 255
Description	This type defines the instance ID of the software component description. The instance ID is modeled as port-defined argument value.

Table 4.2. InstanceId

4.2.2.4.2. Ports

4.2.2.4.2.1. DET service according to AUTOSAR 4.0.3

Interface type	Client-server
----------------	---------------



Interface name	DETService
Operation prototypes	ReportError(Uint8 Instanceld, Uint8 Apild, Uint8 ErrorId)
Description	This port allows software components to report development errors to the Det. For detailed information, see the description of the API function <code>Det_ReportError()</code> .

Table 4.3. AUTOSAR 4.0.3 DETServicePort

4.2.2.4.2.2. DET service according to AUTOSAR 4.3.0 and later

Interface type	Client-server
Interface name	DETService
Operation prototypes	ReportError(Uint8 Apild, Uint8 ErrorId)
Description	This port allows software components to report development errors to the Det. For detailed information, see the description of the API function <code>Det_ReportError()</code> .

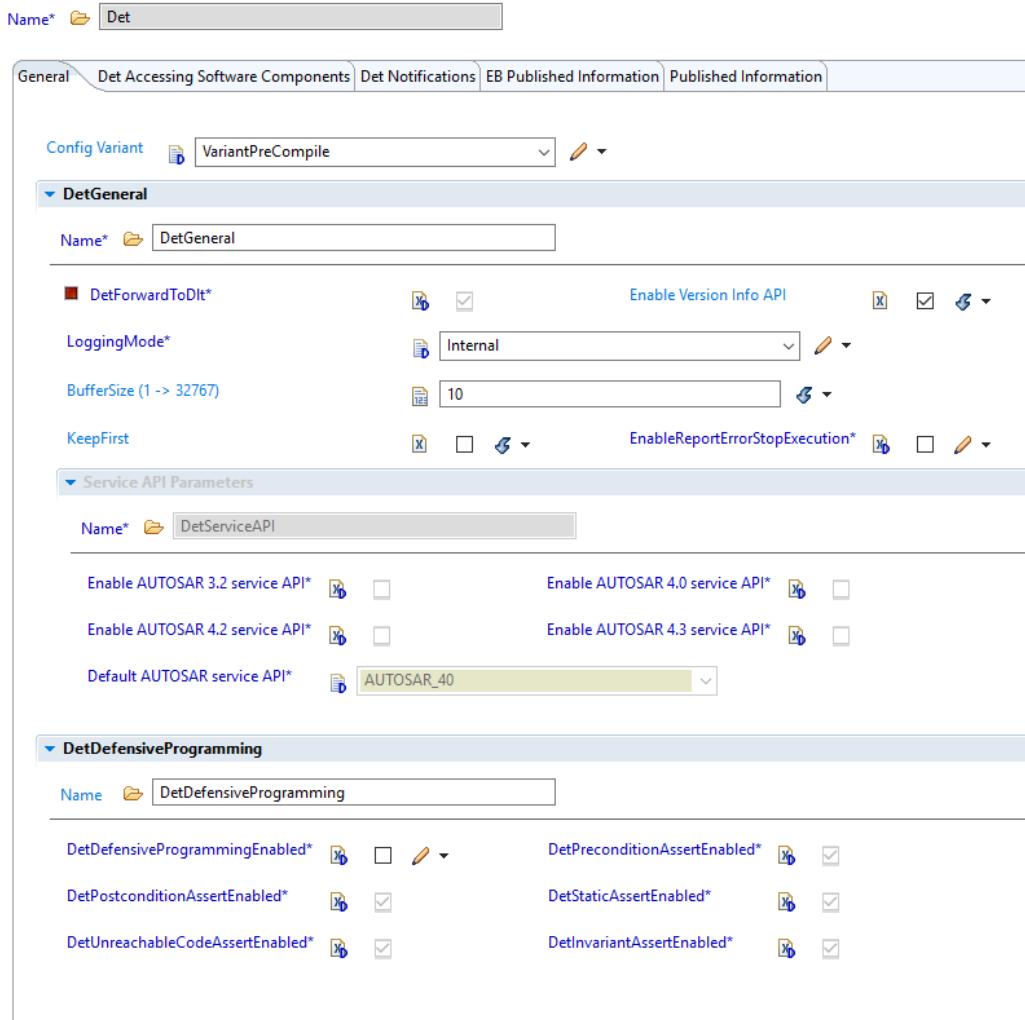
Table 4.4. Development error DETServicePort in AUTOSAR 4.3.0 and later

Interface type	Client-server
Interface name	DETService
Operation prototypes	ReportRuntimeError(Uint8 Apild, Uint8 ErrorId)
Description	This port allows software components to report run-time errors to the Det. For detailed information, see the description of the API function <code>Det_ReportRuntimeError()</code> .

Table 4.5. Run-time error DETServicePort in AUTOSAR 4.3.0 and later

4.2.3. Configuring the Det module

After opening the Det editor, you see the **General** tab:

Figure 4.2. The **General** tab

Configure the logging mode as described in [Section 4.2.2.1, “Handling of error reports”](#).

The parameter **BufferSize** is only relevant for the internal logging mode. It defines the number of entries that can be recorded in internal logging mode for each error type (development or run-time errors, or transient faults).

The parameter **KeepFirst** is only relevant for the internal logging mode. It defines the `Det` behavior if a buffer overflow occurs:

- ▶ **true**: The first entry is kept.
- ▶ **false**: The last entry is kept.

The parameter **EnableReportErrorStopExecution** is only relevant for the internal logging mode. It defines the `Det` behavior if a development error is reported:

- ▶ **true**: `Det` stops the execution of the whole process.
- ▶ **false**: `Det` does not stop the execution and just stores the reported errors.



Switch to the tab **Det Accessing Software Components**:

Index	Name	SoftwareComponentAutosarVersion43	ModuleId
0	SWC_0	<input checked="" type="checkbox"/>	256
1	SWC_1	<input checked="" type="checkbox"/>	257
2	SWC_2	<input type="checkbox"/>	258
3	SWC_3	<input checked="" type="checkbox"/>	259

Figure 4.3. The **Det Accessing Software Components** tab

Click the button for each software component that uses the `Det` service port. Assign a unique **ModuleId** to each entry. For non-basic software modules, the IDs range from 256 to 65535. The IDs from 0 to 255 are reserved for the basic software modules (see [Section 4.2.3.1, “List of module IDs of basic software modules”](#)).

The parameter **SoftwareComponentAutosarVersion43** defines the use of the software component according to AUTOSAR 4.3 or later.

- ▶ *true*: Both `ModuleId` and `Instanceld` are used to identify the component and component instance by using port-defined argument values.
- ▶ *false*: Only the `ModuleId` is used to identify the component by using port-defined argument values.

The functionality of the **SoftwareComponentAutosarVersion43** parameter is enabled only if you enabled the AUTOSAR 4.3 service API in the **DetServiceAPI** container.

4.2.3.1. List of module IDs of basic software modules

The following table lists all module IDs and the associated basic software module. The list is sorted in numerical order, starting with the lowest module ID number. This information enables you to find the caller of the `Det_ReportError()` function. You need this information for debugging:

Module ID	Module name
1	OS
2	Rte
10	EcuM
11	Fim
12	ComM
13	WdgM
15	Det



Module ID	Module name
20	NvM
21	Fee
22	MemIf
29	Nm
31	CanNm
32	FrNm
33	UdpNm
35	CanTp
36	FrTp
40	Ea
42	BswM
43	WdgIf
50	Com
51	PduR
52	IpduM
53	Dcm
54	Dem
56	SoAd
57	Dbg
60	CanIf
61	FrIf
62	LinIf
63	LinNm
64	LinTrcv
65	EthIf
70	CanTrcv
71	FrTrcv
73	EthTrcv
80	Can
81	Fr
82	Lin



Module ID	Module name
83	Spi
88	Eth
90	Eep
92	Fls
93	RamTst
100	Gpt
101	Mcu
102	Wdg
103	CoreTst
104	FlsTst
110	Csm
120	Dio
121	Pwm
122	Icu
123	Adc
124	Port
130	SchM
140	CanSM
141	LinSM
142	FrSM
160	StbM
201	CrC
206	Cal
207	E2E
212	Xcp
254	IoHwAb
255	CDD

Table 4.6. List of module IDs of basic software modules



4.3. MemMap module user guide

4.3.1. Overview

This user guide provides you with MemMap-specific information:

- ▶ [Section 4.3.2, “Background information”](#) explains the concepts of the MemMap module.
- ▶ [Section 4.3.3, “Configuring the MemMap module”](#) provides instructions on how to configure the MemMap module.

For MemMap integration notes, see [Section 5.7.3, “Integration notes”](#).

4.3.2. Background information

The Memory Mapping (MemMap) module is used to map code and data to specific memory sections via memory mapping files. MemMap generates specific header files which contain memory mapping preprocessor defines for `MemorySection` elements and compiler-specific instructions.

4.3.2.1. Support for MemMap header files generation

The MemMap generates three types of memory section header files:

- ▶ The `<SWC>_MemMap.h` header file contains the memory allocation keywords for the `MemorySection` elements as defined in the Software Component Description. `<SWC>_MemMap.h` contains additional compiler-specific instructions if valid generic or specific mappings are created. One header file is generated for each instantiated software component type. `<SWC>` is the `shortName` of the software component type.
- ▶ The `<BSW>_MemMap.h` header file contains the memory allocation keywords for the `MemorySection` elements as defined in the Basic Software Module Description. `<BSW>_MemMap.h` contains additional compiler-specific instructions if valid generic or specific mappings are created. If the Basic Software Module Description contains only one BSW-Implementation, a single `<BSW>_MemMap.h` header file is generated. `<BSW>` is the `shortName` of the `BswModuleDescription`.
- ▶ The `MemMap.h` header file includes the `MemMapHeaderFiles` and the generated `<BSW>_MemMap.h` header files. With configuration parameter `MemMapHeaderFiles`, additional header files can be included in the `MemMap.h` header file by adding the file names to the list provided.



NOTE**Common MemMap header file**

The memory allocation keywords for the `MemorySection` elements defined within the Basic Software Module Description and the Software Component Description are generated in the same header file if MemMap generates the same header file name for both the BSW-Implementation and the SWC-Implementation.

4.3.2.1.1. Functional description

The generated MemMap header files are intended to be included after a memory allocation keyword is defined in the code. This macro is then checked if it is defined in the included MemMap header file. If it is defined, the macro is then undefined and compiler-specific instructions are set if a mapping was created for the `MemorySection` for which this macro was generated. If the memory allocation keyword is not defined, an error is thrown. The error informs that the `MemorySection`, as referenced by the memory allocation keyword, is not defined in the BSW-Implementation or the SWC-Implementation.

The pattern for the memory allocation keyword is `<PREFIX>_[START|STOP]_SEC_<NAME>` with the following parts:

`<PREFIX>`

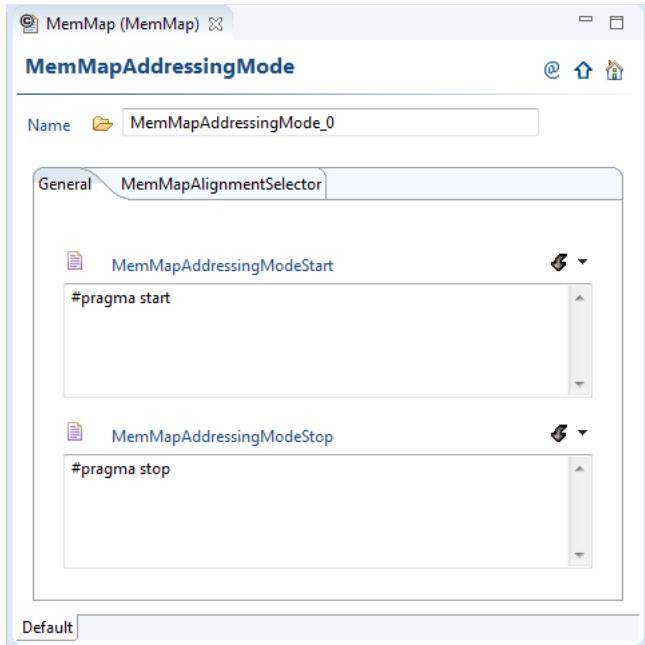
For the BSW modules, this is composed of the `sectionNamePrefix` `shortName`, the `vendorId` and the `vendorApiInfix`, appended by underscores. If the `sectionNamePrefix` attribute is not defined, it is composed of the `BswModuleDescription` `shortName`, the `vendorId` and the `vendorApiInfix`, appended by underscores. If no `vendorApiInfix` is defined, the `_vendorId_vendorApiInfix` is omitted from the name of the file.

For the software components, it is the `shortName` of the software component type.

`<NAME>`

It is the `symbol` of the `MemorySection` if the `symbol` attribute is defined. If `symbol` is not defined, it is the `shortName` of the `MemorySection`.

The pragma commands can be defined to instruct the compiler about the location of code or data. These pragma commands are specific to toolchain and platform. Refer to the AUTOSAR specification of the MemMap module [5] for information about using pragma commands. These compiler-specific instructions are created with the help of specific and generic mappings.

Figure 4.4. The **MemMapAddressingMode**

The pragmas are generated only if one or more `MemMapGenericMapping` and `MemMapSectionSpecificMapping` are created. See [Section 4.3.3, “Configuring the MemMap module”](#) for more details on how to configure the generic and specific mappings. When using the `MemMapGenericMapping`, the same set of pragmas are defined for all the `MemorySections` that reference the same `SwAddrMethod` used in the mapping configuration.

NOTE

MemMapSectionSpecificMapping has priority over MemMapGenericMapping



If a `MemMapGenericMapping` and a `MemMapSectionSpecificMapping` are configured for the same `MemorySection`, only the compiler-specific pragmas mapped via the `MemMapSectionSpecificMapping` are mapped to the `MemorySection`.

4.3.2.1.2. Configuration options

To enable the `MemMap` module to generate header files with checks for memory mapping preprocessor defines, the configuration should contain `MemorySection` elements as defined in the Basic Software Module Description or the Software Component Description.

In the Basic Software Module Description, define one or more `BSW-Implementation` in which its `ResourceConsumption` contains the `MemorySection` elements.

In the Software Component Description, define one or more `SWC-Implementation` in which its `ResourceConsumption` contains the `MemorySection` elements. One or more `ApplicationSwComponentType`



should also be defined. In addition, configure the `Rte` module. Create one or more `RteSwComponentType` that reference the `ApplicationSwComponentType` and the `SWC-Implementation` for which you want a `<SWC>_MemMap.h` header file to be generated.

Make sure that you also have `SwAddrMethod` elements defined for your configuration.

Example of a `MemorySection`:

```
<!-- 8bit section -->
<MEMORY-SECTION>
  <SHORT-NAME>VAR_INIT</SHORT-NAME>
  <ALIGNMENT>8</ALIGNMENT>
  <OPTIONS>
    <OPTION>safetyAsila</OPTION>
    <OPTION>coreLocal</OPTION>
  </OPTIONS>
  <SW-ADDRMETHOD-REF DEST="SW-ADDR-METHOD">
    /AUTOSAR_MemMap/SwAddrMethods/VAR_INIT_ASIL_A_LOCAL
  </SW-ADDRMETHOD-REF>
  <SYMBOL>VAR_INIT_ASIL_A_LOCAL_8</SYMBOL>
</MEMORY-SECTION>
```

Example of a `SwAddrMethod`:

```
<SW-ADDR-METHOD>
  <SHORT-NAME>VAR_INIT_ASIL_A_LOCAL</SHORT-NAME>
  <MEMORY-ALLOCATION-KEYWORD-POLICY>
    ADDR-METHOD-SHORT-NAME-AND-ALIGNMENT
  </MEMORY-ALLOCATION-KEYWORD-POLICY>
  <OPTIONS>
    <OPTION>safetyAsila</OPTION>
    <OPTION>coreLocal</OPTION>
  </OPTIONS>
  <SECTION-INITIALIZATION-POLICY>INIT</SECTION-INITIALIZATION-POLICY>
  <SECTION-TYPE>VAR</SECTION-TYPE>
</SW-ADDR-METHOD>
```

4.3.2.2. Support for multiple BSW-Implementation

In a Basic Software Module Description, multiple BSW-Implementation instances can be defined. The `MemMap` module generates header files for every BSW-Implementation instance. The generated header files are included in the `MemMap.h` header file.



4.3.2.2.1. Functional description

A `<BSW>_MemMap.h` is generated for a BSW-Implementation. The name of the file is composed of the `BswModuleDescription shortName`, **the vendorId** and **the vendorApiInfix** of the BSW-Implementation, appended by underscores. If no `vendorApiInfix` is defined for the BSW module, the `_vendorId_vendorApiInfix` is omitted from the name of the file.

4.3.2.2.2. Configuration options

For a Basic Software Description, you should set different `vendorId` and `vendorApiInfix` attributes for the multiple BSW-Implementation instances. If there is only one BSW-Implementation, the attribute `vendorApiInfix` can be omitted. `vendorApiInfix` is mandatory in case of multiple BSW-Implementation instances.

NOTE

Invalid multiple BSW-Implementation

If you have multiple BSW-Implementation instances, you must use a different `vendorId` and `vendorApiInfix` for each BSW-Implementation. If you omit the `vendorApiInfix` or use the same `vendorId` and `vendorApiInfix` for more than one BSW-Implementation, the configuration is considered invalid.

The `MemMap` module does not report an error but it considers only the last BSW-Implementation from the invalid configuration.

4.3.2.3. Support for AUTOSAR 4.0.2

For `MemorySection CONFIG_DATA`, the syntax of the memory allocation keyword differs from one AUTOSAR version to another. This feature ensures compatibility between versions.

4.3.2.3.1. Functional description

The syntax for the `MemorySection CONFIG_DATA` is as follows:

- ▶ for AUTOSAR 4.0.2 and older: `<PREFIX> [START|STOP]<NAME>`
- ▶ for AUTOSAR 4.0.3 and newer: `<PREFIX>_ [START|STOP]_SEC_<NAME>`

To configure the `MemMap` module to generate the memory allocation keywords for the `MemorySection CONFIG_DATA` compatible to AUTOSAR 4.0.2, enable the `MemMapAS40Compatibility` parameter.

The feature is disabled by default.



4.3.2.3.2. Configuration options

The `MemMapAS40Compatibility` parameter can be:

- ▶ enabled: The memory allocation keywords for `MemorySection CONFIG_DATA` are generated using the syntaxes `<PREFIX>_[START|STOP]_<NAME>` for AUTOSAR 4.0.2 and also `<PREFIX>[START|STOP]_SEC_<NAME>` for AUTOSAR 4.0.3 and newer.
- ▶ disabled: The memory allocation keywords for `MemorySection CONFIG_DATA` are generated using the syntax `<PREFIX>[START|STOP]_SEC_<NAME>` for AUTOSAR 4.0.3 and newer.

4.3.2.4. Support for memory section validation

Due to programming errors, memory sections might be nested or opened memory sections might not be closed correctly. With this feature, you are informed about any programming errors. This feature is implemented as a preprocessor check.

4.3.2.4.1. Functional description

In case of nested memory sections, an error message informs you that the memory section you try to open is nested in an opened section. In case of closing a memory section without prior opening it, an error message informs you that you are closing a memory section without opening it first.

You can configure the `MemMap` module to validate the memory sections by enabling the `MemMapValidateSections` parameter, located in the container `MemMapValidateSections`.

The feature is disabled by default.

4.3.2.4.2. Configuration options

The `MemMapValidateSections` parameter can be:

- ▶ enabled: The memory sections are checked that they are opened and closed in the correct order, that no START is preceded by a START, and no STOP is set without the corresponding START. If the order is not correct, an error message informs you that a section is open in an already open section or that a section is closed without prior opening it.
- ▶ disabled: The memory sections are not checked. If sections are not opened and closed in the correct order, no errors are reported.

4.3.2.5. Support for coreScope

In case of multi-core ECUs, the `coreScope` keywords, `LOCAL` and `GLOBAL`, indicate the restriction and qualifications of code and data. The feature checks whether the `coreScope` keywords are set correctly.



4.3.2.5.1. Functional description

MemMap reports errors if the `coreScope` is invalid, e.g.:

- ▶ `coreScope` is set multiple times
- ▶ `coreLocal` is not set with the correct `SwAddrMethod sectionInitializationPolicy`
- ▶ `coreLocal` is not present in both the name and options of the `MemorySection`
- ▶ `coreLocal` is not present in both the name and the options of the referenced `SwAddrMethod`

If the `coreScope` keyword is missing, the default `coreScope` option shall be treated as `GLOBAL`. Because of this, MemMap only checks that the `coreGlobal` option is not set multiple times in the `MemorySection` options and the options of the referenced `SwAddrMethod`.

You can configure the MemMap module to validate the `coreScope` by enabling the `MemMapValidateCoreScope` parameter, located in the container `MemMapValidateSections`.

The feature is enabled by default.

For more information on `coreScope`, refer to the AUTOSAR specification of the MemMap module [\[6\]](#).

4.3.2.5.2. Configuration options

The `MemMapValidateCoreScope` parameter can be:

- ▶ enabled: The memory sections are checked that they contain the correct `coreScope` keyword and option.
- ▶ disabled: The memory sections are not checked. If the `coreScope` keyword and option are not used correctly, no errors are reported. The invalid memory sections are taken into account at generation.

For a valid `coreScope LOCAL`, ensure that the `MemorySection` and `SwAddrMethod` are defined as follows:

- ▶ If the `MemorySection` has no `symbol` attribute defined, the word `LOCAL` is set in the `MemorySection shortName`.
- ▶ If the `MemorySection` has the `symbol` attribute defined, the word `LOCAL` is set in the `MemorySection symbol`.
- ▶ The option `coreLocal` is set as one of the `MemorySection` options.
- ▶ The option `coreLocal` is set as one of the `SwAddrMethod` options referenced by the `MemorySection`.
- ▶ The `SwAddrMethod sectionInitializationPolicy` is set to `CLEARED` or `INIT`.

You should set the `coreScope` option only once in the `MemorySection` and the referenced `SwAddrMethod`.



4.3.2.6. Support for safety level

The safety level keywords `QM`, `ASIL_A`, `ASIL_B`, `ASIL_C` and `ASIL_D` indicate the restriction and qualifications of code and data. The feature checks whether the safety level keywords are set correctly.

4.3.2.6.1. Functional description

`MemMap` reports errors if the safety level is invalid, e.g.:

- ▶ safety level is set multiple times
- ▶ safety level is not present in both the name and options of the `MemorySection`
- ▶ safety level is not present in both the name and the options of the referenced `SwAddrMethod`

If the safety level keyword is missing, the default safety level option shall be treated as `QM`.

You can configure the `MemMap` module to validate the safety level by enabling the `MemMapValidateSafety` parameter, located in the container `MemMapValidateSections`.

The feature is enabled by default.

For more information on safety level, refer to the AUTOSAR specification of the `MemMap` module [6].

4.3.2.6.2. Configuration options

The `MemMapValidateSafety` parameter can be:

- ▶ enabled: The memory sections are checked that they contain the correct safety level keyword and option.
- ▶ disabled: The memory sections are not checked. If the safety level keyword and option are not used correctly, no errors are reported. The invalid memory sections are taken into account at generation.

For a valid safety level `safetyAsilA`, ensure that the `MemorySection` and `SwAddrMethod` are defined as follows:

- ▶ If the `MemorySection` has no `symbol` attribute defined, the word `ASIL_A` is set in the `MemorySection` `shortName`.
- ▶ If the `MemorySection` has the `symbol` attribute defined, the word `ASIL_A` is set in the `MemorySection` `symbol`.
- ▶ The option `safetyAsilA` is set as one of the `MemorySection` options.
- ▶ The option `safetyAsilA` is set as one of the `SwAddrMethod` options referenced by the `MemorySection`.



The safety levels `safetyAsilB`, `safetyAsilC` and `safetyAsilD` should be set in the same way as in the `safetyAsilA` example.

You should set the `safety` level option only once in the `MemorySection` and the referenced `SwAdrMethod`.

4.3.3. Configuring the MemMap module

This chapter provides you with information on how to configure specific components of the `MemMap` module. Depending on the status of your project, you may not need to configure all components that are described in this chapter.

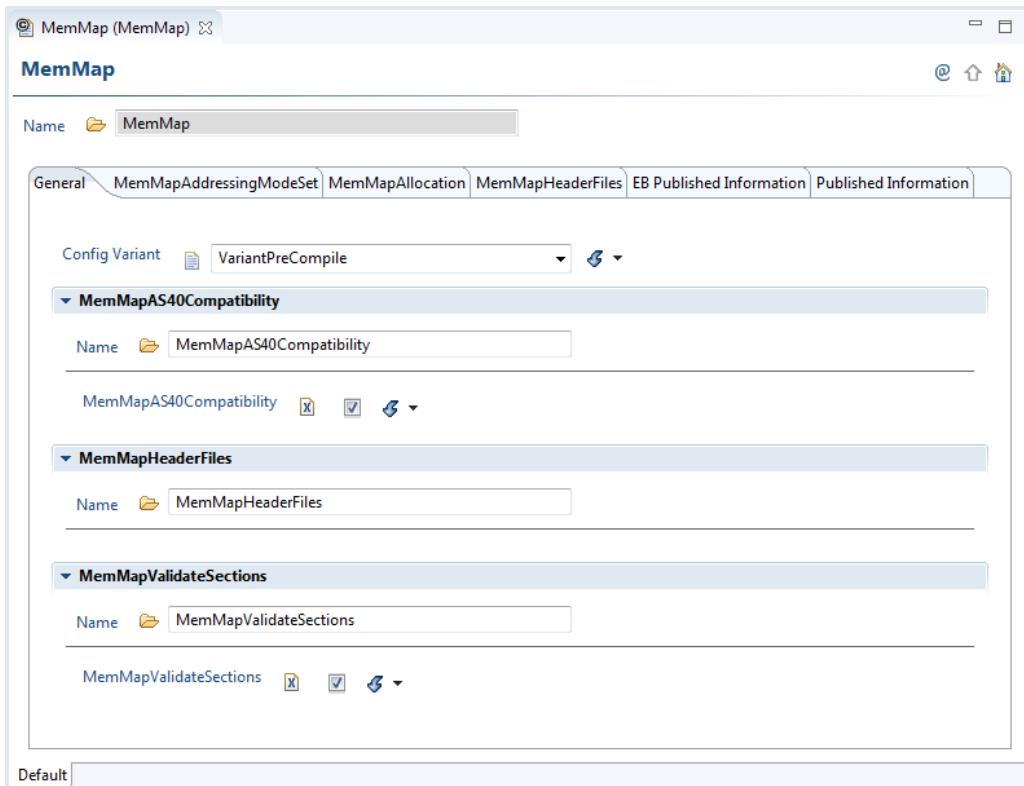
To understand how to configure the `MemMap` module, you must be familiar with the basic concepts of the `MemMap`. For detailed information on the `MemMap` concepts, see [Section 4.3.2, “Background information”](#).

4.3.3.1. Overview

Configure the `MemMap` module for the following use cases:

- ▶ to adapt `MemMap` to the AUTOSAR version that you use for your project: [Section 4.3.3.2, “Configuring MemMap to generate macros compatible to AUTOSAR 4.0.2”](#)
- ▶ to validate the memory sections: [Section 4.3.3.3, “Configuring MemMap to validate memory sections”](#)
- ▶ to create generic mappings for compiler-specific instructions: [Section 4.3.3.4, “Configuring MemMap-GenericMapping”](#)
- ▶ to create specific mappings for compiler-specific instructions: [Section 4.3.3.5, “Configuring MemMapSectionSpecificMapping”](#)

To start the configuration tasks, open the `MemMap` module's editor on the **General** tab.

Figure 4.5. The **General** tab

4.3.3.2. Configuring MemMap to generate macros compatible to AUTOSAR 4.0.2

In this section you learn how to generate MemMap header files that are compatible to AUTOSAR 4.0.2. For details, see [Section 4.3.2.3, “Support for AUTOSAR 4.0.2”](#).

Required features:

- ▶ Support for AUTOSAR 4.0.2
- ▶ Support for MemMap header files generation

Required modules:

- ▶ Required AUTOSAR modules: MemMap module and any other modules



Configuring the **MemMap** module

Step 1

Add the **MemMap** module to your project.

**Step 2**

Import the ARXML file from the configuration into the project using the system description importer of EB tresos Studio.

Step 3

Generate the swcd ARXML files with `generate_swcd`.

Step 4

Run the wizard Update Service Component and BSWM Descriptions.

Step 5

Enable the `MemMapAS40Compatibility` option.

Step 6

Generate the code for your project.

4.3.3.3. Configuring MemMap to validate memory sections

In this section you learn how to generate `MemMap` header files that report errors if the memory sections are not opened and closed in the correct order. For details, see [Section 4.3.2.4, “Support for memory section validation”](#).

Required features:

- ▶ Support for memory section validation
- ▶ Support for `MemMap` header files generation

Required modules:

- ▶ Required AUTOSAR modules: `MemMap` module and any other modules



Configuring the `MemMap` module

Step 1

Add the `MemMap` module to your project.

Step 2

Import the ARXML file from the configuration into the project using the system description importer of EB tresos Studio.

Step 3

Generate the swcd ARXML files with `generate_swcd`.

Step 4

Run the wizard Update Service Component and BSWM Descriptions.

Step 5

Enable the `MemMapValidateSections` option.

Step 6

Generate the code for your project.



4.3.3.4. Configuring MemMapGenericMapping

In this section you learn how to configure the `MemMap` module in order to generate header files with compiler-specific instructions using `MemMapGenericMapping`. For details, see [Section 4.3.2.1, "Support for MemMap header files generation"](#).

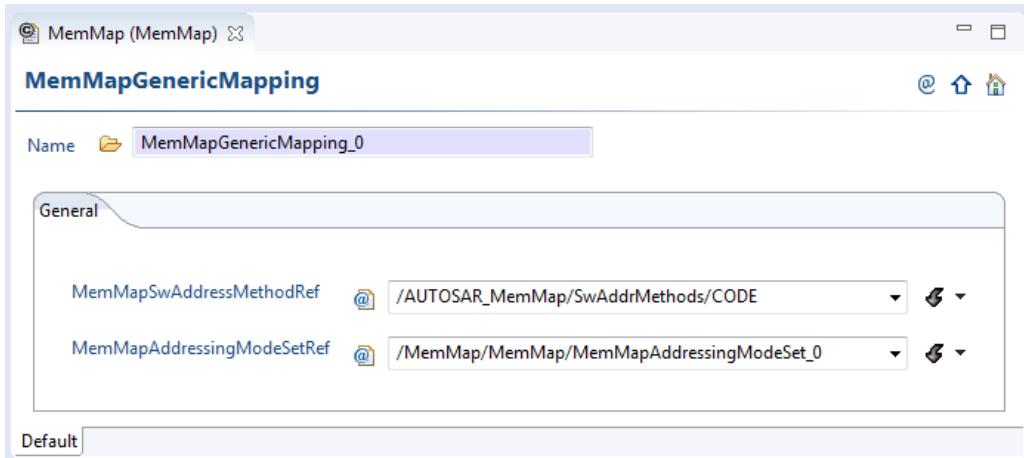


Figure 4.6. The **MemMapGenericMapping**

In order to configure the generic mapping, first configure the `MemMapAddressingModeSet`. Create a `MemMapAddressingModeSet` that applies to the `SwAddrMethod`, referenced in the `MemMapGenericMapping`.

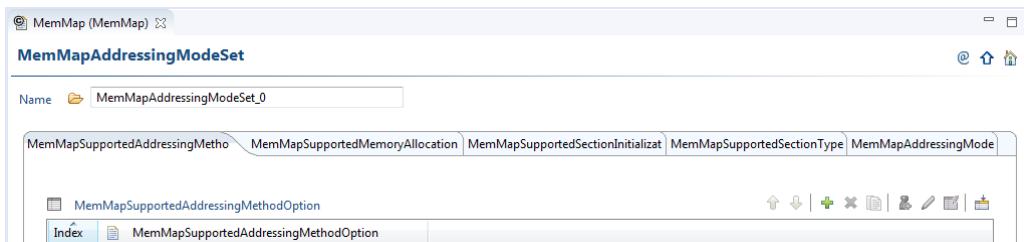


Figure 4.7. The **MemMapAddressingModeSet**

Required features:

- ▶ Support for `MemMap` header files generation

Required modules:

- ▶ Required AUTOSAR modules: `MemMap` module and any other modules



Configuring the `MemMap` module

Step 1

Define a new `MemMapAddressingModeSet` and add a `MemMapAddressingMode` containing the pragmas you want to map.



Step 2

Optionally, add one or more `MemMapSupportedAddressingMethodOption` instances that match at least one of the `SwAddrMethod` options.

Step 3

Optionally, add one or more `MemMapSupportedMemoryAllocationKeywordPolicy` instances that match the `SwAddrMethod` `memoryAllocationKeywordPolicy`.

Step 4

Optionally, add one or more `MemMapSupportedSectionInitializationPolicy` instances that match the `SwAddrMethod` `sectionInitializationPolicy`.

Step 5

Optionally, add one or more `MemMapSupportedSectionType` instances that match the `SwAddrMethod` `sectionType`.

Step 6

Define one or more `MemMapAlignmentSelector` that are the same as the `alignment` attributes of the `MemorySection` elements you want to map.

Step 7

Define a new `MemMapAllocation` and add a `MemMapGenericMapping`.

Step 8

Within the `MemMapGenericMapping`, add a reference to the newly created `MemMapAddressingModeSet` and a reference to the `SwAddrMethod`. The pragmas shall be mapped to the `MemorySection` elements that reference this `SwAddrMethod`. If the drop-down list for valid `SwAddrMethod` references is empty, import the system description file where the `SwAddrMethod` elements are defined.

**NOTE****Mapping warnings and errors**

Warnings are reported if:



- ▶ The SwAddrMethod referenced in MemMapGenericMapping has different attributes as the ones configured in MemMapAddressingModeSet. The MemMapGenericMapping is ignored at generation.
- ▶ The SwAddrMethod referenced in MemMapGenericMapping is not referenced by any of the MemorySection defined in the system description. The MemMapGenericMapping is ignored at generation.
- ▶ The SwAddrMethod referenced in MemMapGenericMapping is from a different package than the one referenced by the MemorySections. The MemMapGenericMapping is taken into account at generation.

An error is reported if:

- ▶ More than one MemMapGenericMapping references the same MemMapSwAddressMethodRef. The generation is stopped.

You can configure the MemMap module to report warnings and errors for invalid MemMapGenericMappings by enabling the Enable Log Entries for Generic and Specific Mappings parameter, located in the container MemMapValidateMappings.

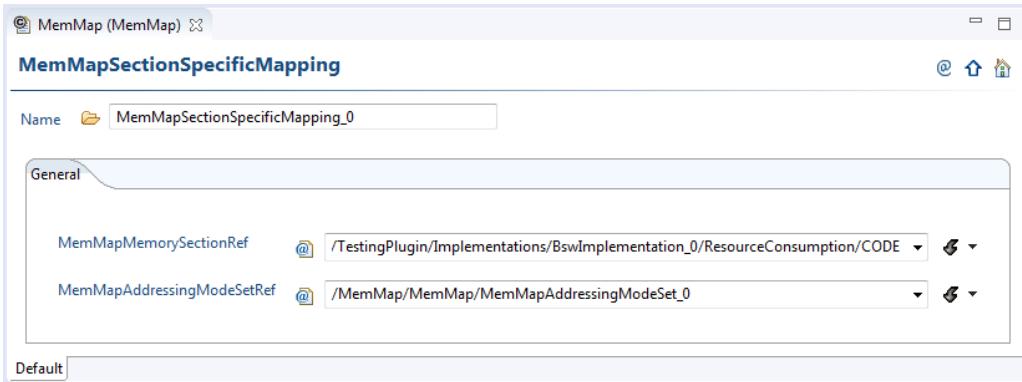
The parameter is enabled by default.

The Enable Log Entries for Generic and Specific Mappings parameter can be:

- ▶ enabled: Errors and warnings are reported for the invalid MemMapGenericMappings.
- ▶ disabled: Errors and warnings are not reported for the invalid MemMapGenericMappings. The invalid MemMapGenericMappings that are not taken into account when Enable Log Entries for Generic and Specific Mappings is enabled, will also be ignored at generation when the parameter is disabled. In case of multiple MemMapGenericMappings that reference the same MemMapSwAddressMethodRef, one of the mappings is taken into account at generation.

4.3.3.5. Configuring MemMapSectionSpecificMapping

In this section you learn how to configure the MemMap module in order to generate header files with compiler-specific instructions using MemMapSectionSpecificMapping. For details, see [Section 4.3.2.1, “Support for MemMap header files generation”](#).

Figure 4.8. The **MemMapSectionSpecificMapping**

Required features:

- ▶ Support for MemMap header files generation

Required modules:

- ▶ Required AUTOSAR modules: MemMap module and any other modules



Configuring the MemMap module

Step 1

Define a new `MemMapAddressingModeSet` and add a `MemMapAddressingMode` containing the pragmas you want to map.

Step 2

Define a new `MemMapAlignmentSelector` that is the same as the alignment attributes of the `MemorySection` you want to map.

Step 3

Define a new `MemMapAllocation` and add a `MemMapSectionSpecificMapping`.

Step 4

Within the `MemMapSectionSpecificMapping`, add a reference to the newly created `MemMapAddressingModeSet` and a reference to the memory section to which the pragmas shall be mapped. If the drop-down list for valid references to memory sections is empty, run the wizard `Update Service Component` and `BSWM Descriptions` first.

**NOTE****Mapping warnings and errors**

A warning is reported if:

- ▶ The `MemMapAlignmentSelector` does not contain the same alignment as the one defined for the `MemorySection`, for which the memory mapping was created. The `MemMapSectionSpecificMapping` is ignored at generation.

An error is reported if:

- ▶ More than one `MemMapSectionSpecificMapping` references the same `MemMapMemorySectionRef`. The generation is stopped.

You can configure the `MemMap` module to validate the `MemMapSectionSpecificMapping` by enabling the `Enable Log Entries for Generic and Specific Mappings` parameter, located in the container `MemMapValidateMappings`.

The parameter is enabled by default.

The `Enable Log Entries for Generic and Specific Mappings` parameter can be:

- ▶ `enabled`: Errors and warnings are reported for the invalid `MemMapSectionSpecificMapping`.
- ▶ `disabled`: Errors and warnings are not reported for the invalid `MemMapSectionSpecificMapping`. The invalid `MemMapSectionSpecificMapping` that is not taken into account when `Enable Log Entries for Generic and Specific Mappings` is enabled, will also be ignored at generation when the parameter is disabled. In case of multiple `MemMapSectionSpecificMappings` that reference the same `MemMapMemorySectionRef`, one of the mappings is taken into account at generation.

4.4. Atomics module user's guide

This chapter provides background information on the `Atomsics` module. It explains the benefits of atomic object access and the terminology. Furthermore, this chapter provides information on how to integrate the `Atomsics` module.

4.4.1. Background information

This chapter explains the atomic functions that are offered by EB tresos AutoCore OS. These functions constitute an efficient foundation for the atomic functions that the `Atomsics` module offers. The first section introduces the problem that can be solved with these atomic functions. With this background, the following sections



explain two important aspects of memory accesses in concurrent environments: *atomicity* and *consistency*. These aspects are the basis for synchronization algorithms that are needed to coordinate concurrent work.

4.4.1.1. Motivation

The development towards more concurrent architectures makes *atomic functions* necessary. Multiple threads of execution may exist at any point in time.

The threads must be coordinated to achieve proper synchronization with the serial parts of a program. Furthermore, the concurrent parts of a program need to work together without much friction. Other synchronization mechanisms offered by EB tresos AutoCore OS can result in a high performance penalty. The performance penalty is caused by context switches and dozens of instructions that are executed if these synchronization mechanisms are used. In some cases, this performance penalty is not acceptable. In these cases, specialized instructions that are offered by the hardware can help. Although these instructions provide less complex synchronization mechanisms, they can prevent performance penalties if an algorithm uses them efficiently.

Key aspects in this field are *atomicity* and *consistency*, which are explained in the following sections.

4.4.1.2. Atomicity of memory accesses

The term *atomicity* relates to a certain trait of memory accesses. Atomic accesses are never interrupted by other concurrent threads, even if they access the same memory location at the same time. They can only be executed either completely or not at all. Therefore, the result of an atomic access is as though there is no contention at all. This also applies to read-modify-write instructions, which are susceptible to interferences of other concurrent threads. The reason for this is that they first have to read from memory, process the data and then write it back. This gives other threads the opportunity to intercept these activities so that the final value in the memory is not as expected.

With an atomicity guarantee, these issues cannot occur and concurrent threads can work on shared data without synchronization mechanisms provided by EB tresos AutoCore OS. However, due to further optimizations that are done in hardware and during compilation, the memory accesses can be reordered.

4.4.1.3. Consistency of memory accesses

The term *consistency* in this field refers to the order in which concurrent threads perceive the write accesses of other threads to shared memory locations. The hardware layer and the compiler layer can affect this ordering.

The microprocessor hardware may use different kinds of buffering to avoid updating the system memory each time a write instruction is executed. Furthermore, read instructions may be executed speculatively ahead of time, if this seems beneficial. The combined effects of these mechanisms must be considered if precise control of memory accesses is required, for example, when peripherals are operated or for synchronization algorithms.



The compiler may also *shuffle* read and write instructions for optimization purposes.

The atomic functions that are provided by EB tresos AutoCore OS enforce *sequential consistency* in the following way:

1. The hardware empties all of its buffers to make the effects of past write instructions visible to other threads. Furthermore, no later read instructions are executed speculatively and no later write instructions are executed. This gives an atomic function the properties of a *memory fence*. Such memory fences prevent the reordering of memory accesses at the hardware level. Thus, a sequentially consistent memory fence prevents read and write instructions from being moved either way across it.
2. Sequential consistency imposes a *total order* on all concurrent accesses to a shared memory location. The consequence is, that all concurrent threads agree upon only one order in which they observe the concurrent accesses of all accessing threads. This is an important property of the atomic functions, because synchronization algorithms depend on it.

4.4.2. Atomics integration notes

The `Atomsics` module alone cannot guarantee atomic access. It relies on an external entity for such guarantees. This external entity may be EB tresos AutoCore OS, an AUTOSAR operating system, or other compatible software. The interface between this external entity and the `Atomsics` module is defined by a set of preprocessor macros and type definitions, which are contained in the following files:

- ▶ `Atomsics_user_types.h`
- ▶ `Atomsics_user_functions.h`

The following macros and types are available:

- ▶ `Atomic_t`
- ▶ `Atomic_value_t`
- ▶ `ATOMICS_USER_AUTOSAR_SPINLOCK`
- ▶ `ATOMICS_USER_MULTICORE_CASE`
- ▶ `ATOMICS_USER_THREAD_FENCE`
- ▶ `ATOMICS_USER_LOCK_OBJECT`
- ▶ `ATOMICS_USER_UNLOCK_OBJECT`
- ▶ `ATOMICS_USER_ATOMIC_T`
- ▶ `ATOMICS_USER_ATOMIC_VALUE_T`
- ▶ `ATOMICS_USER_GET_VALUE`
- ▶ `ATOMICS_USER_SET_VALUE`



► ATOMICS_USER_OBJECT_INITIALIZER

For detailed information on the macros and types, see [Section 5.2.2, “Application programming interface \(API\)”](#).

There are generic implementations for each of these macros, which rely on AUTOSAR synchronization mechanisms. In the single-core case, all interrupts are disabled when atomic objects are accessed. In case there are accessors on more than just one core, an AUTOSAR spinlock is used. This spinlock must also disable all interrupts and you have to provide this spinlock. Because of this, the call environment must permit the use of `SuspendAllInterrupts()` and `ResumeAllInterrupts()` in the first case. In the second case, it must permit the use of `GetSpinlock()` and `ReleaseSpinlock()`. Furthermore, the generic implementation of `ATOMICS_USER_THREAD_FENCE` is empty because the underlying AUTOSAR synchronization mechanisms already take care of memory ordering issues. This means that if you only use functions that the `Atoms` module provides, you do not have to take care of memory ordering issues. If your code depends on a precise ordering in any other way, provide a suitable implementation for this macro.

You can change the generic implementations and hence, adapt the preprocessor macros mentioned above appropriately. For the macros in the `Atoms_user_functions.h` file, you can achieve this by copying the `Atoms_user_functions_customizations.h` file into your project to put the respective `#define` directives there. The header file inclusion order must ensure, that your version of this file is included first.

There is an equivalent file for `Atoms_user_types.h`, which is named `Atoms_user_types_customizations.h`.

To use the services that are offered by the `Atoms` module, include the `Atoms.h` header file.

4.4.3. Migration

You can also use the `Atoms` module without EB tresos AutoCore OS. That is the reason why it only relies on AUTOSAR synchronization mechanisms. This allows you to use a compatible OS of a different vendor.

However, if you use an OS of a different vendor, you cannot tightly integrate the `Atoms` module with the OS to use certain hardware features. This can cause a high synchronization overhead, which may be prohibitive.

If you use EB tresos AutoCore OS instead, you do not need to modify the code that previously used the services of the `Atoms` module. You only require the definition of the `ATOMICS_USE_GENERIC_IMPL` macro on the Make command line with its value set to 0. In this case, optimized synchronization primitives are used in the background, which efficiently use the underlying hardware. This reduces the overhead of all atomic functions.

If you use EB tresos AutoCore OS, you do not need to adapt the macros mentioned in [Section 4.4.2, “Atoms integration notes”](#). All required definitions are already provided and optimized for the hardware platform that is used.

If the `ATOMICS_USE_GENERIC_IMPL` macro is set to the value 1, the generic implementation that is provided by the `Atoms` module is used and [Section 4.4.2, “Atoms integration notes”](#) applies.



4.4.3.1. ATOMICS_USE_GENERIC_IMPL

Purpose	Selects the generic or specialized implementation of atomic functions.
Value	0
Description	<p>This macro is used to select which kind of implementation of atomic functions to use.</p> <p>If set to 1, the generic implementation provided by the <code>Atomics</code> module is used. This implementation makes no assumptions about the underlying hardware and only depends on the interface of an AUTOSAR compatible OS. The call environment must allow the use of EB tresos AutoCore OS spinlocks, i.e. <code>GetSpinlock</code> and <code>ReleaseSpinlock</code>, and the control of interrupts, i.e., <code>SuspendAllInterrupts()</code> and <code>ResumeAllInterrupts()</code>, depending on the setting of the <code>ATOMICS_USER_MULTICORE_CASE</code> macro.</p> <p>If set to 0, the specialized implementation offered by EB tresos AutoCore OS is selected. Compilation errors occur if EB tresos AutoCore OS is not part of your configuration. Furthermore, see the atomic functions documentation that ships with EB tresos AutoCore OS.</p> <p>You must make <code>ATOMICS_USE_GENERIC_IMPL</code> known to Make and the compiler. To do this, call Make as follows: <code>make ATOMICS_USE_GENERIC_IMPL=x</code>. The letter x is either 0 or 1.</p> <p>If you use your own build environment, you must take care to pass <code>ATOMICS_USE_GENERIC_IMPL</code> to the C language preprocessor. For GCC, e.g., you write <code>-DATOMICS_USE_GENERIC_IMPL=x</code> on the command line. Furthermore, you must only compile the files <code>Atomics.c</code> and <code>Atomics_TSPlatforms.c</code> when the generic implementation is selected, i.e. when <code>ATOMICS_USE_GENERIC_IMPL</code> is set to 1.</p> <p>The specialized implementation is an enhanced version of the generic implementation because it uses specific hardware features.</p>

4.4.3.2. Backward compatibility with Platforms module

In the past, the `Platforms` module provided some types and atomic functions for bit-manipulation and to control interrupts. These functions and types start with the `TS_` prefix and are now deprecated. However, you do not need to rewrite legacy code completely, which was designed for the `Platforms` module. The `Atomics` module provides the `Atomics_TSPlatforms.h` header file, which you need to include to use these deprecated functions and types. However, it is recommended for new code that you switch to the new functions and types of the `Atomics` module, which are declared in the `Atomics.h` header file.

The compile-time switch that is represented by the `ATOMICS_USE_GENERIC_IMPL` macro affects these compatibility functions as well. This means that if this macro is set to 0, there is a mandatory dependency on EB



tresos AutoCore OS, and compiler errors occur if EB tresos AutoCore OS is not part of your configuration. The benefit of this dependency is that specific hardware features are used in the background to minimize the run-time overhead of these functions.

If the `ATOMICS_USE_GENERIC_IMPL` macro is set to 1, a generic implementation is selected, which only depends on the API defined by AUTOSAR. This removes the dependency on EB tresos AutoCore OS and allows you to use an AUTOSAR OS of a different vendor.

Sequential consistency is added to the following functions:

- ▶ `TS_AtomicSetBit_x()`
- ▶ `TS_AtomicClearBit_x()`

The letter `x` is either 8, 16, 32, or 64.

The availability of these functions depends on the availability of the respective type.

WARNING



ts functions

If you use the generic implementation, the `Atomics` module uses functions of the underlying AUTOSAR OS and, therefore, its error reporting infrastructure. If these functions fail to lock/unlock an atomic object, you need to use the error reporting of the OS to detect these errors, because the `TS` functions do not report any errors and simply ignore them.

If you use the `Atomics` module in combination with EB tresos AutoCore OS, see the atomic function documentation if there are any usage restrictions pertaining to atomic objects for your hardware.

4.4.4. Example of use

The following example shows how you can use the atomic functions that the `Atomics` module provides. This example shows how you can use atomic objects globally and their values locally and how you can pass them to your own functions. It also shows how atomic objects are used in C language expressions.

Before you use atomic objects, you must initialize them. For this, the `ATOMICS_OBJECT_INITIALIZER` macro as well as the `Atomics_Init()` function exist. The example shows the initialization of the `globalAtomicStaticInit` atomic object via `ATOMICS_OBJECT_INITIALIZER`. It also shows the initialization of the `globalAtomicDynInit` atomic object via `Atomics_Init()`. Note that you do not need to initialize atomic objects with static storage duration if the default initialization to zero is suitable for your use case.

You should always pass atomic objects to functions by reference. Their values, though, may be passed by value. As you can see in the example below, the `func()` function follows these rules.

The `func()` function also shows how you can use atomic objects in expressions. As you can see in the code, objects of type `Atomic_t` are non-transparent and you must always access them via `Atomics_*` () functions.



Objects of the type `Atomic_value_t` are plain numbers. You can use them directly in expressions like any other basic numeric type of the C language.

```
#include <Atomics.h>

/* This atomic object is initialized, before the application is started. */
Atomic_t globalAtomicStaticInit = ATOMICS_OBJECT_INITIALIZER(42);
/* This atomic object must be initialized by the application itself. */
static Atomic_t globalAtomicDynInit;

static Atomic_value_t func(Atomic_t *obj0, Atomic_t *obj1, Atomic_value_t value) {
    Atomic_value_t ret = 0;
    ret = Atomics_FetchAdd(obj0, value) * Atomics_Load(&obj1);
    if (ret > 42) {
        ret = Atomics_Exchange(obj1, ret);
    }
    return ret;
}

int main(void) {
    Atomic_value_t value = 72;
    Atomics_Init(&globalAtomicDynInit, 102);
    value = func(&globalAtomicStaticInit, &globalAtomicDynInit, value);
    return (value > 5 ? 0 : -1);
}
```

4.5. Application template and application demos

4.5.1. Overview

The idea behind the application template and application demos is to give an example of how to start a new project. Everything is kept as simple as possible. Keep in mind that application demos are only a rudimentary starting point and must not be used as the basis for a real ECU. Projects for real ECUs are much more complex and you need knowledge about several parts of the AUTOSAR standard.

The application template and application demos chapter provides you with background information and the set-up of the specific application demos.

The application template and the various application demos are EB tresos Studio projects. These projects can be built and loaded into the ECU target, where they can then be executed. Instructions on how to build and run the application demos are provided in later sections of this chapter.



The application demos are created based on the EB tresos AutoCore Generic basic template project, various supplementary files, and specific steps that need to be taken and are described in guided workflows. The difference between the basic template and an application demo is that the basic template only provides a minimal skeleton for ECU start-up, but does not provide any real application functionality. The application functionality is added later on using the system description, the software component (SWC) description, and the software component (SWC) source files. For details on the basic template, see [Section 4.5.2, “Basic template”](#). For details on the supplementary files, see [Section 4.5.3, “Supplementary files”](#).

Certain application demos are provided directly with EB tresos AutoCore Generic and you can import them directly into EB tresos Studio. In this chapter, you will learn the general steps from importing an application demo into EB tresos Studio to building the application demo.

EB tresos AutoCore Generic also contains a set of guided workflows that enable you to create application demos easily. Instead of providing one big workflow to create one big application demo, a set of individual workflows is provided to create an application demo and to enhance this demo step-by-step. This stepwise approach supports a better understanding of the AUTOSAR configuration process by allowing you to compile and test each step individually.

For an introduction into the EB tresos Studio workflows, see the EB tresos Studio user's guide, chapter Workflows view. The user's guide is located in \$TRESOS_BASE/doc/2.0_EB_tresos_Studio/2.1_Studio_documentation_users_guide.pdf.

Currently the following application demos and workflows exist:

- ▶ The simple RTE application demo (see [Section 4.5.4, “Simple RTE application demo”](#)), based on the Rte workflow
- ▶ The simple CAN application demo (see [Section 4.5.5, “Simple CAN application demo”](#)), based on the CAN Stack workflow
- ▶ The simple FlexRay application demo (see [Section 4.5.6, “Simple FlexRay application demo”](#)), based on the FlexRay Stack workflow
- ▶ The simple LIN application demo (see [Section 4.5.7, “Simple LIN application demo”](#)), based on the LIN Stack workflow
- ▶ The simple Ethernet application demo (see [Section 4.5.8, “Simple Ethernet application demo”](#)), based on the IP Stack workflow
- ▶ The simple Memory application demo (see [Section 4.5.9, “Simple Memory application demo”](#)), based on the Memory Stack workflow
- ▶ The simple Post-build application demo (see [Section 4.5.10, “Simple Post-build application demo”](#)), based on Post-Build Setup, Post-Build Only, and Post-Build Update workflows

[Figure 4.9, “Workflows and application demos overview”](#) shows the dependencies between different workflows and application demos. When you perform a specific workflow step, which is mentioned in the box, you will create the corresponding application demo, which is named in brackets.

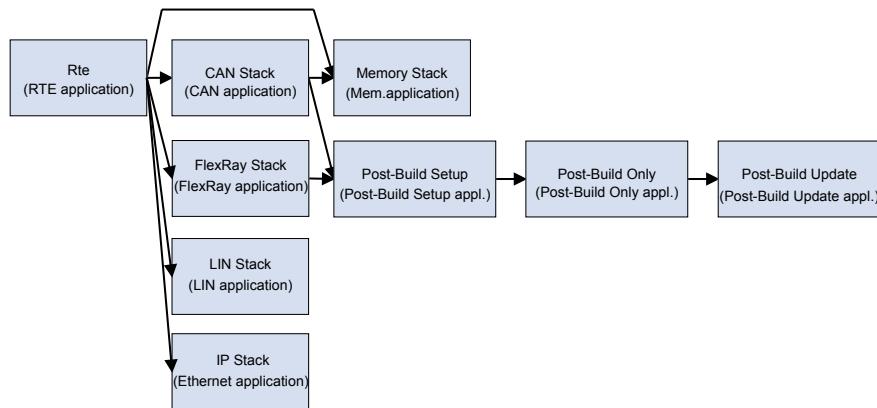


Figure 4.9. Workflows and application demos overview

[Figure 4.10, “Overview of the modules used in the application demos”](#) shows the interaction between the AUTOSAR layers for application demos.

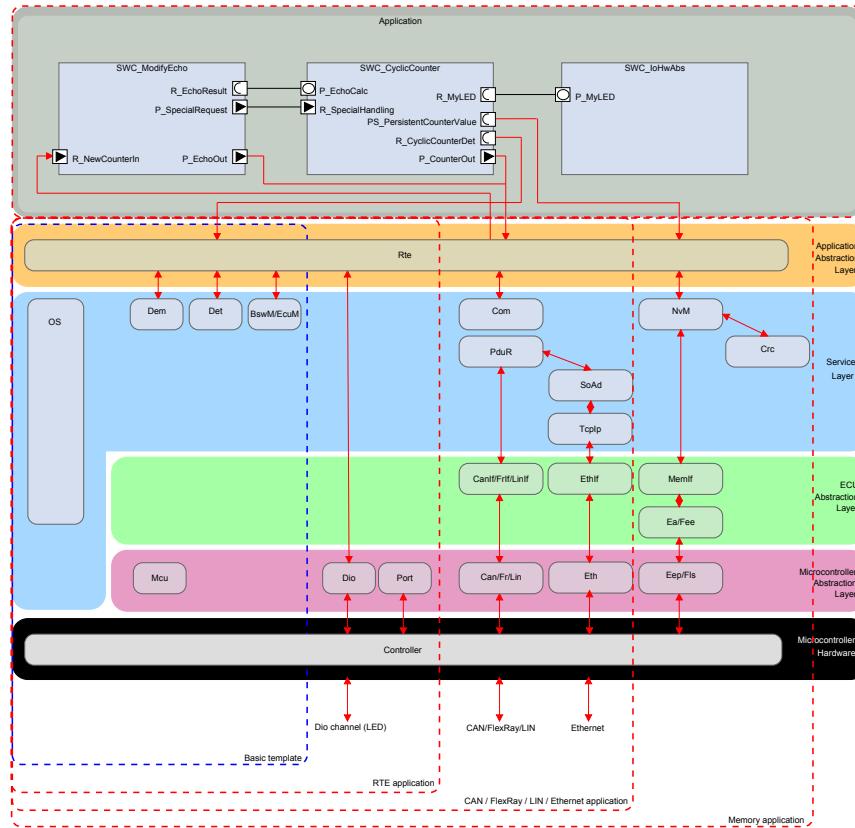


Figure 4.10. Overview of the modules used in the application demos

4.5.2. Basic template



4.5.2.1. Background information

The `basic template` is an EB tresos Studio project that contains a minimal set of required modules and files to generate, compile, and link an AUTOSAR application. It is therefore the perfect candidate when you want to start a new AUTOSAR project without having to care about the absolute basics and the set-up of the build environment.

The following target-independent modules are used:

- ▶ Atomics
- ▶ Base
- ▶ BswM
- ▶ Compiler
- ▶ Dem
- ▶ Det
- ▶ EcuM
- ▶ Make
- ▶ MemMap
- ▶ PbCfgM
- ▶ Rte

The following target-dependent modules are used:

- ▶ Mcu
- ▶ Os
- ▶ Platforms
- ▶ Resource (if applicable for the platform)

4.5.2.2. Applicable workflow

None, the `basic template` is provided as part of EB tresos AutoCore Generic.

4.5.2.3. Content description

The `basic template` is an EB tresos Studio project that provides a pre-configured AUTOSAR application. You can immediately import the `basic template` into EB tresos Studio and build it using an appropriate



compiler. Note that the `basic template` does not provide any real application functionality. Rather, it ensures ECU start-up and all necessary micro-controller initialization, e.g. correct clock. The content of the `basic template` is dependent on the target derivative used.

Sub-directories overview:

- ▶ `config`: modules configuration, and EB tresos Studio project database
- ▶ `source`: BswM, EcuM user-callouts, OS init_task, main, modules stubs, target ECU board, compiler and memory abstraction implementation
- ▶ `util`: user build environment, that can be directly used for project build purposes

4.5.2.4. Availability

The `basic template` is part of the product ACG8 Base and is located in `$TRESOS_BASE/templates/AutoCore/<derivative>/basicTemplate`.

4.5.3. Supplementary files

4.5.3.1. Background information

When an application demo is created based on workflow steps, various supplementary files are required:

- ▶ Module configuration snippets that ease the manual configuration
- ▶ Software components (SWCs) source files that provide actual C-language implementation
- ▶ SWC description files that describe SWC properties in .arxml files
- ▶ System description files that describe system properties in an .arxml file

In order to simplify user application demo debug capabilities, appropriate EB tresos Inspector configuration files are provided for certain application demos.

The SWC description files, the system description files, and the SWC source files are used in all of the application demos. Therefore, different application demos behave similarly. The difference between the application demos is based on the type of the underlying communication that is used.

The description in the following sections is simplified in order to provide only significant details of the overall functionality. For more specific details it is recommended to investigate file content either manually in a standard text-editor or using the dedicated tooling for SWC or system description.



4.5.3.2. SWC and system description files

The following SWC and system description files are provided:

- ▶ AUTOSAR_Datatypes.arxml
- ▶ BswMMMode.arxml
- ▶ CanSystem.arxml
- ▶ FlexraySystem.arxml
- ▶ LinSystem.arxml
- ▶ SoftwareComponents.arxml

Normally, only a certain sub-set of files is used in each application demo.

The SWC composition consists of the following components:

- ▶ SWC_CyclicCounter
- ▶ SWC_ModifyEcho
- ▶ SWC_IoHwAbs

[Figure 4.11, “SWC composition and port connections”](#) shows the SWC composition of SWC used and their port connections.

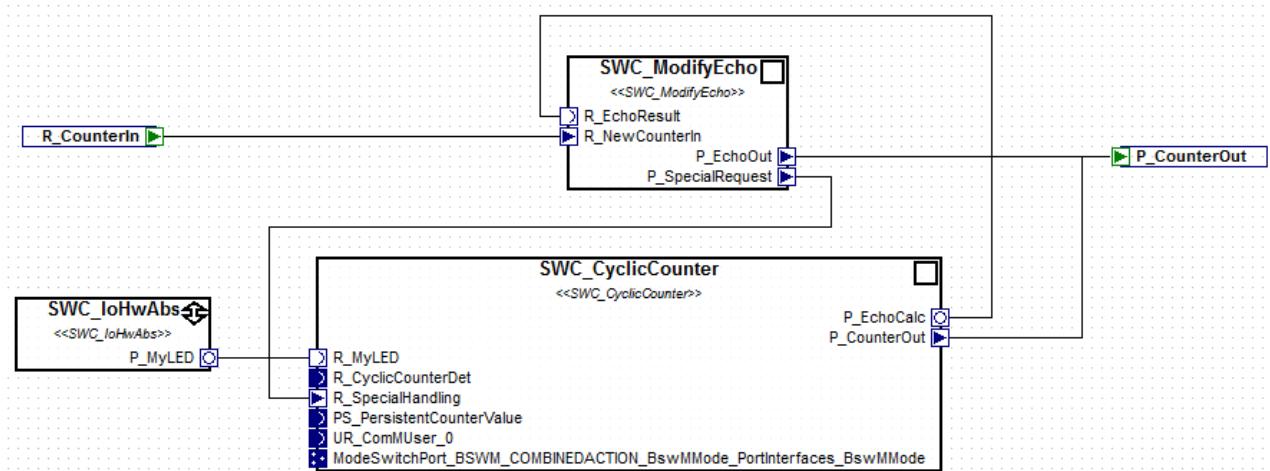


Figure 4.11. SWC composition and port connections

4.5.3.3. SWC source files

For each SWC, one dedicated C-language implementation file is provided:

- ▶ `SWC_CyclicCounter.c` implements functions:



- ▶ SWC_CyclicCounter_Cyclic which is executed periodically and which implements an increasing counter that is transmitted using port P_CounterOut. Also, R_SpecialHandling is checked for reception of special values (pseudo Det error, or shutdown command) from the SWC SWC_ModifyEcho, and LED-light toggling is called on port R_MyLED.
- ▶ SWC_CyclicCounter_SetCounter which is executed when requested on port P_EchoCalc from the SWC SWC_ModifyEcho. It adds the new counter value given to the current counter value and returns the result. In addition, the current counter value will be reset to the newly passed value.
- ▶ SWC_ModifyEcho.c implements function:
 - ▶ SWC_ModifyEcho_ModifyEcho which is executed when a new message on port R_NewCounterIn is received. If a special value is recognized the value is sent to P_SpecialRequest that has a receiver side queue, otherwise it uses port R_EchoResult to call function SWC_CyclicCounter_SetCounter (in the SWC SWC_CyclicCounter), the returned value is transmitted on port P_EchoOut.
- ▶ SWC_IoHwAbs.c implements function:
 - ▶ SWC_IoHwAbs_SetDiscreteValue which implements LED-light toggling using the Dio_WriteChannel from the Dio module when requested on port P_MyLED from the SWC SWC_CyclicCounter.

4.5.3.4. OS tasks' execution context

In the application demos, the provided SWC runnables are mapped into the OS tasks in the following way:

- ▶ SWC_CyclicCounter/SWC_CyclicCounter_Cyclic: Rte_Time_Task which is consequently activated periodically e.g. every 100ms.
- ▶ SWC_ModifyEcho/SWC_ModifyEcho_ModifyEcho: Rte_Event_Task

4.5.3.5. Availability

Supplementary files are part of the product ACG8 Base and are located in \$TRESOS_BASE/demos/AutoCore/<version>/supplement.

4.5.4. Simple RTE application demo

4.5.4.1. Functional behavior

The EB tresos AutoCore Generic simple RTE application demo demonstrates an AUTOSAR application that includes the Rte module and software components (SWCs). It does not include communication stack nor memory stack support. The application functionality is defined by the system description, SWCs descriptions, and SWCs sources, which are shared among various application demos.



The application demo exhibits after an ECU start-up:

- ▶ An application counter that is increased periodically every second and a print-out on standard output (EB tresos WinCore only). The counter starts at default value 99.
- ▶ A Dio channel (usually connected to one of the available board LED lights) that is toggled periodically every second or a print-out on standard output (EB tresos WinCore only).

The following target-independent modules are used:

- ▶ Atomics
- ▶ Base
- ▶ Compiler
- ▶ BswM
- ▶ Dem
- ▶ Det
- ▶ EcuM
- ▶ Make
- ▶ MemMap
- ▶ PbCfgM
- ▶ Rte

The following target-dependent modules are used:

- ▶ Mcu
- ▶ Os
- ▶ Platforms
- ▶ Resource (if applicable for the platform)
- ▶ Port (if applicable for the platform)
- ▶ Dio (if applicable for the platform)

To get the simple RTE application demo to run, first follow the instructions in [Section 4.5.11, “Getting an application demo to run”](#).

4.5.4.2. Applicable workflow

To create the simple RTE application demo step-by-step, you can use the Rte workflow together with the basic template project.



4.5.4.3. Availability

If your delivery contains the simple RTE application demo, you can find it in \$TRESOS_BASE/demos/AutoCore/<derivative>/simple_demo_rte.

4.5.5. Simple CAN application demo

4.5.5.1. Functional behavior

The EB tresos AutoCore Generic simple CAN application demo demonstrates an AUTOSAR application that includes the Rte module, software components (SWCs), and communication stack support. It does not include memory stack support. The application functionality is defined by the system description, SWCs descriptions, and SWCs sources, which are shared among various application demos.

The system demonstrates CAN communication using the Com, PduR, CanIf, and Can module.

The application demo exhibits after an ECU start-up :

- ▶ An application counter that is increased periodically every second and a print-out on standard output (EB tresos WinCore only). The counter starts at default value 99.
- ▶ A Dio channel (usually connected to one of the available board LED lights) that is toggled periodically every second or a print-out on standard output (EB tresos WinCore only).
- ▶ One CAN message Pdu_CounterOut that is sent periodically from the ECU every second and which contains 1 payload byte containing current counter value.
- ▶ One CAN message Pdu_CounterIn that can be accepted by the ECU.
- ▶ Support for application counter set/reset (Pdu_CounterIn with payload value less than 0xF0), pseudo development error (Pdu_CounterIn with payload value equal to 0xF1), and ECU shut-down (Pdu_CounterIn with payload value equal to 0xFF) commands.

In addition, an EB tresos Inspector configuration file simple_demo_can_rte.ipf is provided in the supplementary files.

The following target-independent modules are used:

- ▶ Atomics
- ▶ Base
- ▶ Compiler
- ▶ BswM
- ▶ Dem



- ▶ Det
- ▶ EcuM
- ▶ Make
- ▶ MemMap
- ▶ PbCfgM
- ▶ Rte
- ▶ EcuC
- ▶ CanIf
- ▶ CanSM
- ▶ Com
- ▶ ComM
- ▶ PduR

The following target-dependent modules are used:

- ▶ Mcu
- ▶ Os
- ▶ Platforms
- ▶ Resource (if applicable for the platform)
- ▶ Port (if applicable for the platform)
- ▶ Dio (if applicable for the platform)
- ▶ Can

To get the simple CAN application demo to run, first follow the instructions in [Section 4.5.11, “Getting an application demo to run”](#).

4.5.5.2. Applicable workflow

To create the simple CAN application demo step-by-step, you can use the CAN Stack workflow together with the simple RTE application demo project.

4.5.5.3. PDU allocation

I-PDU	CAN-ID	Data bytes	Direction (EcuTestNode/DebugNode)	Baud rate
Pdu_CounterIn	256	1 byte	Receive/Transmit	100 kbit/s



I-PDU	CAN-ID	Data bytes	Direction (EcuTestNode/DebugNode)	Baud rate
Pdu_CounterOut	272	1 byte	Transmit/Receive	100 kbit/s

Table 4.7. Overview of I-PDUs in this application demo

4.5.5.4. Availability

If your delivery contains the simple CAN application demo you can find it in \$TRESOS_BASE/demos/ AutoCore/<derivative>/simple_demo_can_rte.

4.5.6. Simple FlexRay application demo

4.5.6.1. Functional behavior

The EB tresos AutoCore Generic simple FlexRay application demo demonstrates an AUTOSAR application that includes the Rte module, software components (SWCs), and communication stack support. It does not include memory stack support. The application functionality is defined by the system description, SWCs descriptions, and SWCs sources, which are shared among various application demos.

The system demonstrates FlexRay communication using the Com, PduR, FrSM, FrIf, and Fr module.

The application demo exhibits after an ECU start-up:

- ▶ An application counter that is increased periodically every second and a print-out on standard output (EB tresos WinCore only). The counter starts at default value 99.
- ▶ A Dio channel (usually connected to one of the available board LED lights) that is toggled periodically every second or a print-out on standard output (EB tresos WinCore only).
- ▶ One FlexRay frame CounterOutFrame_001, that is sent periodically from the ECU every second and which contains 8 payload bytes containing current counter value.
- ▶ One FlexRay frame CounterInFrame_001 that can be accepted by the ECU.
- ▶ Support for application counter set/reset (CounterInFrame_001 with payload value less than 0xF0), pseudo development error (CounterInFrame_001 with payload value equal to 0xF1), and ECU shut-down (CounterInFrame_001 with payload value equal to 0xFF) commands.

In addition, an EB tresos Inspector configuration file simple_demo_fr_rte.ipf is provided in the supplementary files.

The following target-independent modules are used:



- ▶ Atomics
- ▶ Base
- ▶ Compiler
- ▶ BswM
- ▶ Dem
- ▶ Det
- ▶ EcuM
- ▶ Make
- ▶ MemMap
- ▶ PbCfgM
- ▶ Rte
- ▶ EcuC
- ▶ FrIf
- ▶ FrSM
- ▶ Com
- ▶ ComM
- ▶ PduR

The following target-dependent modules are used:

- ▶ Mcu
- ▶ Os
- ▶ Platforms
- ▶ Resource (if applicable for the platform)
- ▶ Port (if applicable for the platform)
- ▶ Dio (if applicable for the platform)
- ▶ Fr

To get the simple FlexRay application demo to run, first follow the instructions in [Section 4.5.11, "Getting an application demo to run".](#)

4.5.6.2. Applicable workflow

To create the simple FlexRay application demo step-by-step, you can use the FlexRay Stack workflow together with the simple RTE application demo project.



4.5.6.3. PDU allocation

I-PDU	Slot-ID	Direction (EcuTestNode/DebugNode)	Base cycle	Cycle repetition
CounterInFrame_001	1	Receive/Transmit	0	1
CounterOut-Frame_001	10	Transmit/Receive	0	1

Table 4.8. Overview of I-PDUs in this application demo

4.5.6.4. Availability

If your delivery contains the simple FlexRay application demo you can find it in \$TRESOS_BASE/demos/AutoCore/<derivative>/simple_demo_fr_rte.

4.5.7. Simple LIN application demo

4.5.7.1. Functional behavior

The EB tresos AutoCore Generic simple LIN application demo demonstrates an AUTOSAR application that includes the Rte module, software components (SWCs), and communication stack support. It does not include memory stack support. The application functionality is defined by the system description, SWCs descriptions, and SWCs sources, which are shared among various application demos.

The system demonstrates LIN communication using the Com, PduR, LinIf, and Lin module.

The application demo exhibits after an ECU start-up :

- ▶ An application counter that is increased periodically every second and a print-out on standard output (EB tresos WinCore only). The counter starts at default value 99.
- ▶ A Dio channel (usually connected to one of the available board LED lights) that is toggled periodically every second or a print-out on standard output (EB tresos WinCore only).
- ▶ One LIN message Pdu_CounterOut that is sent periodically from the ECU every second and which contains 1 payload byte containing current counter value.
- ▶ One LIN message Pdu_CounterIn that can be accepted by the ECU.
- ▶ Support for application counter set/reset (Pdu_CounterIn with payload value less than 0xF0), pseudo development error (Pdu_CounterIn with payload value equal to 0xF1), and ECU shut-down (Pdu_CounterIn with payload value equal to 0xFF) commands.

The following target-independent modules are used:



- ▶ Atomics
- ▶ Base
- ▶ Compiler
- ▶ BswM
- ▶ Dem
- ▶ Det
- ▶ EcuM
- ▶ Make
- ▶ MemMap
- ▶ PbCfgM
- ▶ Rte
- ▶ EcuC
- ▶ LinIf
- ▶ LinSM
- ▶ Com
- ▶ ComM
- ▶ PduR

The following target-dependent modules are used:

- ▶ Mcu
- ▶ Os
- ▶ Platforms
- ▶ Resource (if applicable for the platform)
- ▶ Port (if applicable for the platform)
- ▶ Dio (if applicable for the platform)
- ▶ Lin

To get the simple LIN application demo to run, first follow the instructions in [Section 4.5.11, “Getting an application demo to run”](#).

4.5.7.2. Applicable workflow

To create the simple LIN application demo step-by-step, you can use the LIN Stack workflow together with the simple RTE application demo project.



4.5.7.3. PDU allocation

I-PDU	LIN-ID	Data bytes	Direction (EcuTestNode/DebugNode)	Baud rate
Pdu_CounterIn	0	2 byte	Receive/Transmit	10 kbit/s
Pdu_CounterOut	1	2 byte	Transmit/Receive	10 kbit/s

Table 4.9. Overview of I-PDUs in this application demo

4.5.7.4. Availability

If your delivery contains the simple LIN application demo you can find it in \$TRESOS_BASE/demos/AutoCore/<derivative>/simple_demo_lin_rte.

4.5.8. Simple Ethernet application demo

4.5.8.1. Functional behavior

The EB tresos AutoCore Generic simple Ethernet application demo demonstrates an AUTOSAR application that includes the Rte module, software components (SWCs), and communication stack support. It does not include memory stack support. The application functionality is defined by the system description, SWCs descriptions, and SWCs sources, which are shared among various application demos.

The system demonstrates the Ethernet communication between SWCs via the Rte using the Com, PduR, SoAd, TcpIP, EthIf, and Eth module.

The application demo exhibits after an ECU start-up:

- ▶ An application counter that is increased periodically every second and a print-out on standard output (EB tresos WinCore only). The counter starts at default value 99.
- ▶ A Dio channel (usually connected to one of the available board LED lights) that is toggled periodically every second or a print-out on standard output (EB tresos WinCore only).
- ▶ One Ethernet message Pdu_CounterOut that is sent periodically from the ECU every second and which contains 1 payload byte containing current counter value.
- ▶ One Ethernet message Pdu_CounterIn that can be accepted by the ECU.
- ▶ Support for application counter set/reset (Pdu_CounterIn with payload value less than 0xF0), pseudo development error (Pdu_CounterIn with payload value equal to 0xF1), and ECU shut-down (Pdu_CounterIn with payload value equal to 0xFF) commands.



The following target-independent modules are used:

- ▶ Atomics
- ▶ Base
- ▶ Compiler
- ▶ BswM
- ▶ Dem
- ▶ Det
- ▶ EcuM
- ▶ Make
- ▶ MemMap
- ▶ PbCfgM
- ▶ Rte
- ▶ EcuC
- ▶ EthIf
- ▶ EthSM
- ▶ Com
- ▶ ComM
- ▶ PduR
- ▶ SoAd
- ▶ TcpIp

The following target-dependent modules are used:

- ▶ Mcu
- ▶ Os
- ▶ Platforms
- ▶ Resource (if applicable for the platform)
- ▶ Port (if applicable for the platform)
- ▶ Dio (if applicable for the platform)
- ▶ Eth

To get the simple Ethernet application demo to run, first follow the instructions in [Section 4.5.11, “Getting an application demo to run”](#).



4.5.8.2. Applicable workflow

To create the simple Ethernet application demo step-by-step, you can use the IP Stack workflow together with the simple RTE application demo project.

4.5.8.3. PDU allocation

I-PDU	Direction (EcuTest-Node/DebugNode)	Source IP	Source port	Destination IP	Destination port
Pdu_CounterIn	Receive/Transmit	192.168.88.77	60000 / UDP	192.168.88.73	50000 / UDP
Pdu_CounterOut	Transmit/Receive	192.168.88.73	50000 / UDP	192.168.88.77	60000 / UDP

Table 4.10. Overview of I-PDUs in this application demo

4.5.8.4. Availability

If your delivery contains the simple Ethernet application demo, you can find it in \$TRESOS_BASE/demos/AutoCore/<derivative>/simple_demo_eth_rte.

4.5.9. Simple Memory application demo

4.5.9.1. Functional behavior

The EB tresos AutoCore Generic simple Memory application demo demonstrates an AUTOSAR application that includes the Rte module, software components (SWCs), communication stack support, and memory stack support. The application functionality is defined by the system description, SWCs descriptions, and SWCs sources, which are shared among various application demos.

The system demonstrates CAN communication using the Com, PduR, CanIf, and Can modules.

It demonstrates the memory stack using the NvM, MemIf, Ea or Fee, Crc, and Eep or Fls module.

The application demo exhibits after an ECU start-up:

- ▶ An application counter that is increased periodically every second and a print-out on standard output (EB tresos WinCore only). The counter starts at default value 99. Once ECU shut-down is executed, the counter next starts at the value that has been stored beforehand.
- ▶ A Dio channel (usually connected to one of the available board LED lights) that is toggled periodically every second or a print-out on standard output (EB tresos WinCore only).



- ▶ One CAN message Pdu_CounterOut that is sent periodically from the ECU every second and that contains 1 payload byte containing current counter value.
- ▶ One CAN message Pdu_CounterIn that can be accepted by the ECU.
- ▶ Support for application counter set/reset (Pdu_CounterIn with payload value less than 0xF0), pseudo development error (Pdu_CounterIn with payload value equal to 0xF1), and ECU shut-down (Pdu_CounterIn with payload value equal to 0xFF) commands. The ECU shut-down command ensures that that current counter value is stored into non-volatile (NV) memory. This can then be retrieved as the initial counter value after the next ECU start-up.

In addition, an EB tresos Inspector configuration file `simple_demo_can_rte.ipf` is provided in the supplementary files. The `simple` Memory application demo is an extension of the `simple` CAN demo so the same file can be used.

The following target-independent modules are used:

- ▶ Atomics
- ▶ Base
- ▶ BswM
- ▶ Compiler
- ▶ Dem
- ▶ Det
- ▶ EcuM
- ▶ Make
- ▶ MemMap
- ▶ PbCfgM
- ▶ Rte
- ▶ EcuC
- ▶ CanIf
- ▶ CanSM
- ▶ Com
- ▶ ComM
- ▶ PduR
- ▶ NvM
- ▶ MemIf
- ▶ Ea or Fee
- ▶ Crc



The following target-dependent modules are used:

- ▶ Mcu
- ▶ Os
- ▶ Platforms
- ▶ Resource (if applicable for the platform)
- ▶ Port (if applicable for the platform)
- ▶ Dio (if applicable for the platform)
- ▶ Can
- ▶ Eep or Fls

To get the simple Memory application demo to run, first follow the instructions in [Section 4.5.11, “Getting an application demo to run”](#).

4.5.9.2. Applicable workflow

To create the simple Memory application demo step-by-step, you can use the Memory Stack workflow together with the simple CAN application demo project.

4.5.9.3. PDU allocation

I-PDU	CAN-ID	Data bytes	Direction (EcuTestNode/DebugNode)	Baud rate
Pdu_CounterIn	256	1 byte	Receive/Transmit	100 kbit/s
Pdu_CounterOut	272	1 byte	Transmit/Receive	100 kbit/s

Table 4.11. Overview of I-PDUs in this application demo

4.5.9.4. Availability

If your delivery contains the simple Memory application demo you can find it in \$TRESOS_BASE/demos/AutoCore/<derivative>/simple_demo_mem_can_rte.



4.5.10. Simple Post-build application demo

4.5.10.1. Functional behavior

The EB tresos AutoCore Generic simple Post-build application demo demonstrates the usage of the EB tresos Studio post-build support feature. The simple Post-build application demo is not a typical application demo such as the simple CAN application demo or the simple FlexRay application demo. Rather, it is a generic procedure how to add post-build support into an existing project. The procedure is applicable to any EB tresos Studio project. As the starting point for the simple Post-build application demo, you need an already existing project, i.e. either the simple CAN application demo or the simple FlexRay application demo.

Post-build support allows you to change certain configuration parameters later on, without having to rebuild the whole ECU software. The changed sub-set of configuration parameters can be loaded separately onto the ECU later on, after the project has been built. AUTOSAR defines two post-build scenarios: the Post-build loadable scenario and the Post-build selectable scenario.

For a general introduction to the EB tresos Studio configuration-times feature, see the EB tresos Studio user's guide, chapter Configuration time of projects. The user's guide is located in \$TRESOS_BASE/doc/2.0_EB_tresos_Studio/2.1_Studio_documentation_users_guide.pdf.

Three sub-workflows are provided with which you can create three sub-modifications of the project:

- ▶ Post-Build Setup: In order to add post-build support to an existing project, you need to add the EB tresos AutoCore Generic additional module PbcfgM (Post-Build Configuration Manager) to the project and provide the configuration for this module.
- ▶ Post-Build Only: Describes how you can modify an existing project to generate and build solely a post-build configuration.
- ▶ Post-Build Update: Describes how you can update the post-build configuration.

4.5.10.2. Applicable workflow

To create any of the three sub-modifications of the simple Post-build application demo step-by-step, you can use the Post-Build Setup, the Post-Build Only or the Post-Build Update workflow together with simple CAN application demo or the simple FlexRay application demo project.

4.5.10.3. Availability

If your delivery contains the simple Post-build application demo, you can find it in \$TRESOS_BASE/demos/AutoCore/<derivative>/simple_demo_postbuild.



4.5.11. Getting an application demo to run

4.5.11.1. Prerequisites for starting a new project

Before you start using an application demo, install the compiler supported by EB tresos AutoCore.

See the EB tresos AutoCore release notes for further information about the compiler and compiler version for your architecture.

NOTE**The installation path must not contain any blanks**

Ensure that you install the compiler in a path without any blanks in the pathname. Due to limitations in the build environment, pathnames containing blanks will not work.

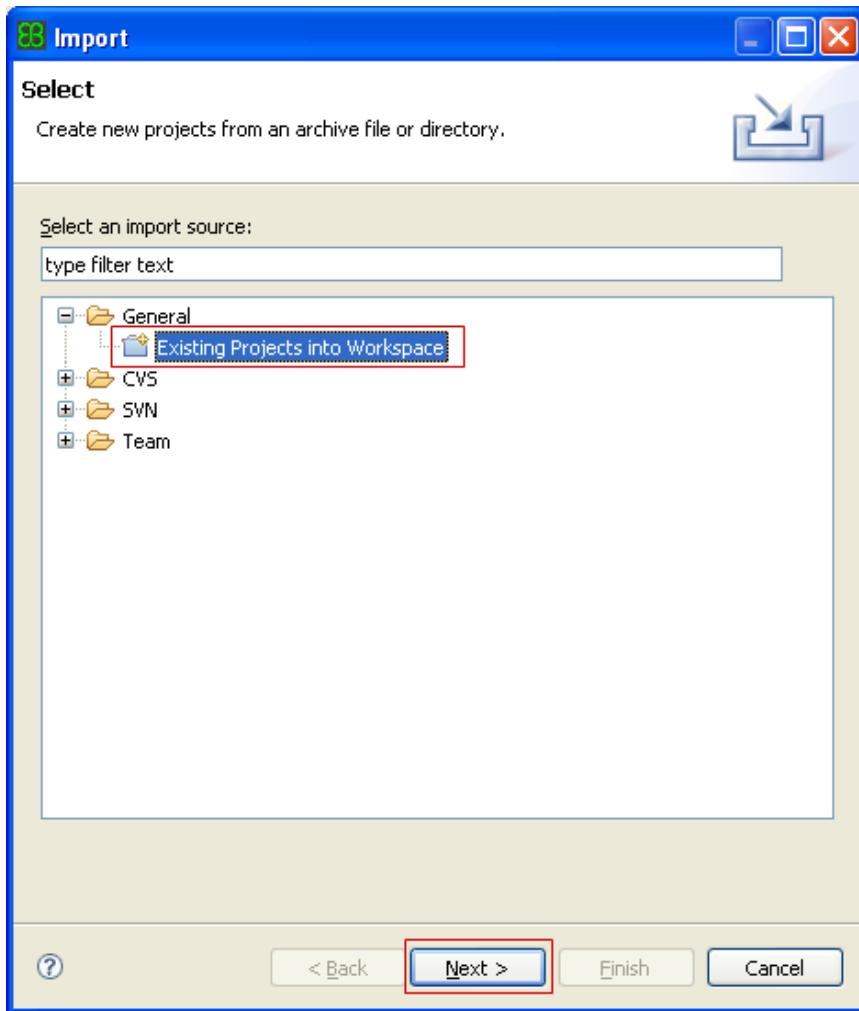
4.5.11.2. Importing the application demo

The application demos are delivered as EB tresos Studio project. You need to import the project into your EB tresos Studio workspace, e.g. at \$TRESOS_BASE/workspace. The \$TRESOS_BASE is the directory into which you installed EB tresos Studio, e.g. C:/EB/tresos.

To import the application demo into your workspace:

- ▶ Locate \$TRESOS_BASE/bin/tresos_gui.exe.
 - ▶ Double-click tresos_gui.exe.
- EB tresos Studio opens up.
- ▶ In the **File** menu, select **Import**.

The **Import** dialog opens up.

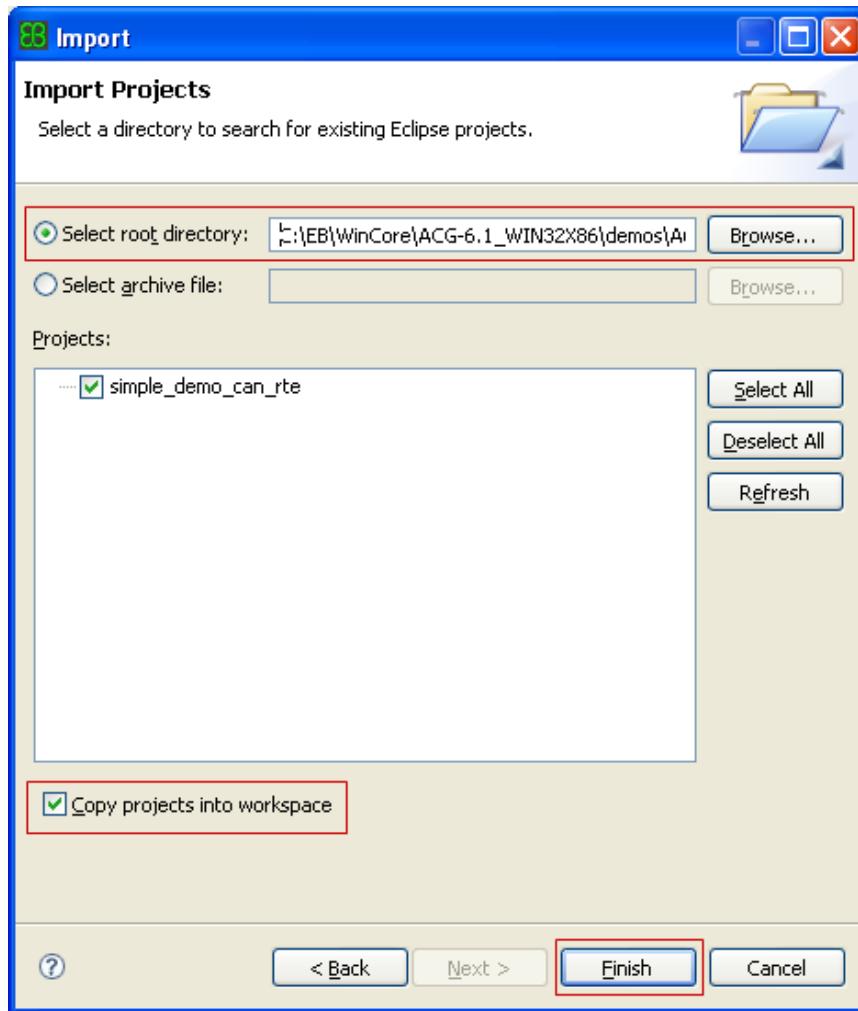


- ▶ Select **Existing Projects into Workspace**.
- ▶ Click **Next**.
- ▶ Select **Select root directory** and browse to \$TRESOS_BASE/demos/AutoCore/\$DERIVATE/<name of application demo>.

And choose one of the available application demo, e.g. simple_demo_rte in order to work with simple RTE demo application demo.

- ▶ Click **OK**.

The project appears in the **Projects** window of the **Import** dialog.



► Select **Copy projects into workspace**.

The project is now copied to the default workspace at `$TRESOS_BASE/workspace`.

► Click **Finish**.

You are done.

In the **Project Explorer** view you now see the project:

- Open it by double-clicking on the project name.
- Open the configuration by clicking the gray chip symbol .

4.5.11.3. Adapting the build environment

This section describes how to adapt the build environment in order to build the application demo. For a more detailed description of the build environment, see [Section 4.6, “Build environment”](#).



To change the settings or the path of the compiler, follow the steps below:

1. Locate the project folder in the workspace directory, e.g. \$TRESOS_BASE/workspace/<name of application demo>.
2. Open the file util/launch_cfg.bat in a text editor.
3. Set the environment variable TRESOS_BASE to the directory into which you have installed EB tresos Studio.
4. If the plugins of EB tresos Studio are not installed in the default location (folder \$TRESOS_BASE/plugins of the installation directory), set the environment variable PLUGINS_BASE to the directory into which you have installed the plugins of EB tresos Studio.
5. Save the file.

4.5.11.3.1. Changing the compiler

NOTE
For EB tresos WinCore, no compiler has to be configured


EB tresos WinCore is delivered with the MinGW development environment, which is installed automatically when you build the application demo. The MinGW development environment also contains a compiler. Therefore, you only need to follow these instructions when you want to configure a different compiler.

If multiple compilers are supported on your architecture you can change the compiler. To change the compiler:

1. Locate the project folder in the workspace directory, e.g. \$TRESOS_BASE/workspace/<name of application demo>.
2. Open the file util/launch_cfg.bat in a text editor.
3. Set the variable TOOLCHAIN to the name of the toolchain, e.g. set TOOLCHAIN=dcc.
4. Set the environment variable TOOLPATH_COMPILER to the actual installation path of your compiler. For example: set TOOLPATH_COMPILER=C:/WindRiver/diab/5.5.1.0.
5. Save the file.

4.5.11.3.2. Changing the board settings

If you want to use a different board, you need to change the board settings.

The directory \$PROJECT_ROOT/source/boards/<board name> contains board-specific make and source files.

In order to use a different board, follow the steps below:

1. Locate the project folder in the workspace directory, e.g. \$TRESOS_BASE/workspace/<name of application demo>.



2. Open the file `util/<target>_<derivative>_Makefile.mak` in a text editor.
3. Change the variable `BOARD`.

The value of the variable has to be the same as the name of the board directory, e.g. `BOARD = EvaBoardMPC5567`.

4.5.11.4. Building the application demo

In order to build an EB tresos AutoCore project, you need to perform the following three steps in the order they are described:

1. Generate the project.
2. Create the project dependencies.
3. Build the project.

To build the application demo, make sure that EB tresos Studio is not running anymore. Then follow these steps:

1. In the `$TRESOS_BASE/workspace/<application_demo_name>/util` directory, double-click the file `launch.bat`. The first start of `launch.bat` takes some time.
2. Type **make generate** and hit the **Enter** key.

The project is being generated.

3. Type **make depend** and hit the **Enter** key.

The project dependencies are being created.

4. Type **make** (or **make -j -O** for compiling in parallel) and hit the **Enter** key.

Your application demo is being built.

If you work with EB tresos WinCore, the MinGW development environment is being installed with the build of the application demo.

You find the resulting binary file in the `$TRESOS_BASE/workspace/<application_demo_name>/output/bin` directory.

4.5.11.5. Taking further steps

1. Find more information about EB tresos Studio in the EB tresos Studio user's guide.

This is located at `$TRESOS_BASE/doc/2.0_EB_tresos_Studio/2.1_Studio_documentation_users_guide.pdf`.



2. Find further information about the build environment used for the application demo at [Section 4.6, “Build environment”](#).
3. For more information about EB tresos AutoCore modules have a look into [Chapter 4, “ACG8 Base user's guide”](#)

4.6. Build environment

The build environment helps you to compile complete projects and create an executable program that can be loaded to the target. It consists mainly of the three EB tresos Studio plug-ins `Make`, `Compiler` and `Platforms`.

The `Make` plug-in provides a set of GNU makefiles, and is located in the directory `$(TRESOS_BASE)/plugins/Make_TS_<version string>`.

The `Compiler` plug-in is located in the directory `$(TRESOS_BASE)/plugins/Compiler_TS_<version string>`. It provides the toolchain-specific GNU makefiles and header files, whereas the `Make` plug-in is toolchain-independent.

The `Platforms` plug-in is located in the directory `$(TRESOS_BASE)/plugins/Platforms_TS_<version string>`. It contains the platform-specific GNU makefiles, header and source files.

NOTE
Make plug-in is incompatible with older Platform plug-ins


The `Make` plug-in version 4.0.8, which was released with product release 7.3, has been refactored and is not compatible anymore with `Platforms` plug-ins that were released before product release 7.3. So if you update the `Make` plug-in, be sure to also update the `Platforms` plug-in.

NOTE
Platforms plug-in is incompatible with older Platform plug-ins


Starting with ACG8.5, the toolchain-specific makefiles and header files are moved out of the `Platforms` plug-in into the new `Compiler` plug-in. So if you update the `Platforms` plug-in, be sure to also install the `Compiler` plug-in.

In the following chapter, build environment and `Make/Compiler/ Platforms` plug-ins are used as synonyms.

4.6.1. Overview

The following sections in the chapter build environment show you how the *user build environment* is configured and used. The information helps integrators in setting up projects, which use EB tresos AutoCore modules together with application code.



Although it is possible to integrate EB tresos AutoCore modules with other build environments, EB strongly recommends using the build environment integrated in EB tresos AutoCore.

4.6.2. Background information

The background information section introduces you to the concepts of the build environment. It provides the knowledge that is necessary to understand the instructions for configuring the build process. Step-by-step instructions are available starting in [Section 4.6.3, “Configuring the build process”](#).

[Figure 4.12, “From ECU configuration to code generation”](#) shows an overview of the workflow from the configuration of an ECU to its code generation:

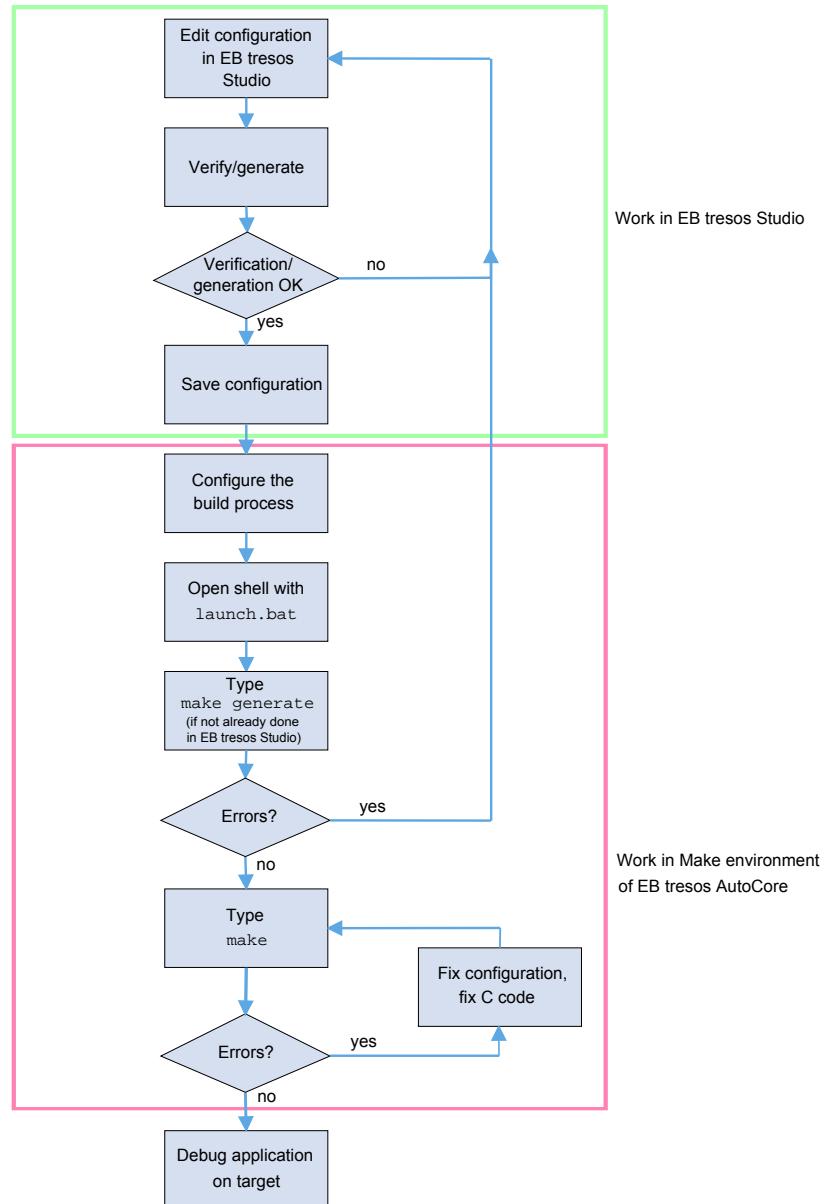


Figure 4.12. From ECU configuration to code generation



4.6.2.1. Improve build speed with GNU Make command line parameters

The build environment is designed for distributing the build steps over as many CPU cores as possible to maximize build speed. To use this feature, run make with the following parameters:

```
make -j -O
```

The `-j` parameter triggers make to spawn new processes as long as the CPU load on all cores is less than 100%. This parameter is recommended for maximum speed.

If you want to compile files that are stored on drives with slow file system access, e.g network drives or ClearCase dynamic views, and you run `make -j`, your system may become unresponsive. This behavior occurs because the CPU load will never reach 100% due to the file system bottle neck. In this case you should limit the amount of concurrent processes with the `-jN` parameter. `N` is the amount of concurrent processes. For example, `make -j16` spawns 16 concurrent processes.

The `-O` parameter is a feature which came with GNU Make v4.0. Use this feature to synchronize the output from different processes to avoid that the output is scrambled as seen in GNU Make v3.82.

4.6.2.2. Folder structure of a project

Every project consists at least of the following folders:

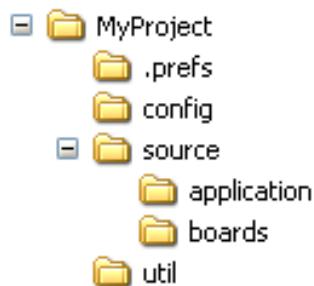


Figure 4.13. Project directory structure

The folders contain the following:

- ▶ **MyProject:** This is the main folder of the project. This path is stored in the `make` variable `PROJECT_PATH`.
- ▶ **.prefs:** Contains the file `preferences.xdm` which holds all project settings.
- ▶ **config:** Contains all configuration files. The files are stored in the XDM format by EB tresos Studio.
- ▶ **source/application:** Contains the application source files.
- ▶ **source/boards:** Contains the board support package files. The board support package contains e.g. the linker scripts and parts of the startup code. For further information on the board support package, see [Section 4.6.2.5, “Board support packages”](#).



You may need to manually adapt some files in the boards directory. It is not possible to use EB tresos Studio to configure the necessary parameters.

- ▶ `util`: This directory is the starting point for using the build environment. If you want to compile an application, start the batch file `launch.bat` from the `util` directory to open a command shell and setup some environment variables. All compile runs have to be started within this command shell.

For instructions how to compile your application, see [Section 4.6.3.4, “Compiling the application”](#).

Within the `util` directory, you can find the main makefiles `Makefile.mak` and `common.mak`. Moreover, it contains the architecture and compiler-specific makefiles.

4.6.2.3. Basic concepts of the makefiles

Every EB tresos AutoCore module comes with a set of makefiles:

- ▶ `xxx_defs.mak`: Defines several variables, for example the path to the module implementation or the output path for the generator.
- ▶ `xxx_rules.mak`: The main purpose of these files is to define the files which have to be compiled for the module libraries.

All configuration-dependent files of EB modules are mapped to the library `xxx_src_FILES` by default. The following example shows the `Det_src_FILES`:



Example 4.1. `Det_src_FILES`

```
Det_src_FILES      := \
$(Det_CORE_PATH)/src/Det.c
```

After defining the file sets, the next step is to configure the build environment to actually create those libraries.

The variable `LIBRARIES_TO_BUILD` holds a list of libraries that are to be build.

In the following example, `Det_lib` and `Det_src` are added to `LIBRARIES_TO_BUILD`.



Example 4.2. `LIBRARIES_TO_BUILD`

```
LIBRARIES_TO_BUILD += Det_lib Det_src
```



The build environment can also directly compile and link source files without adding them to a library.

The variables `CC_FILES_TO_BUILD`, `CPP_FILES_TO_BUILD`, and `ASM_FILES_TO_BUILD` contain a list of C, C++ and assembler files which are to be compiled and linked.



Example 4.3. Source files, which are to be compiled and linked

```
CC_FILES_TO_BUILD      += appl.$(CC_FILE_SUFFIX)
CPP_FILES_TO_BUILD    += appl.$(CPP_FILE_SUFFIX)
ASM_FILES_TO_BUILD    += appl.$(ASM_FILE_SUFFIX)
```

4.6.2.4. Make plug-ins

The makefile framework consists of several plug-ins. [Figure 4.14, “Make plug-ins”](#) gives you an overview of the plug-ins:

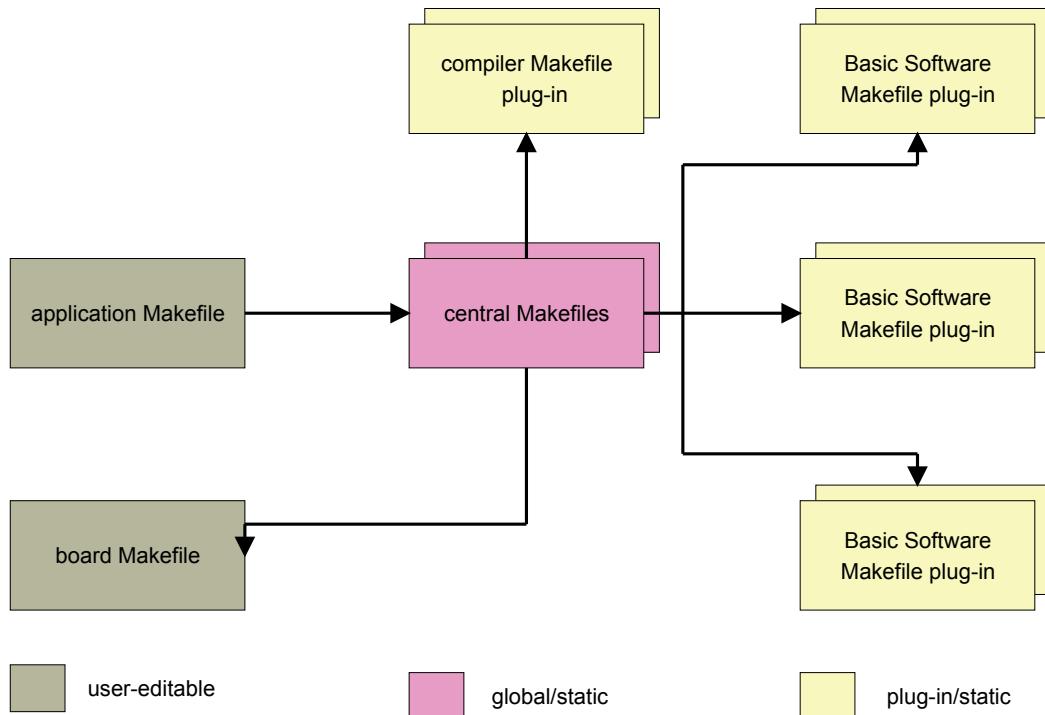


Figure 4.14. Make plug-ins



Starting from the application makefile, the build environment automatically includes several other makefiles. All inclusions are optional.

Every EB tresos AutoCore plug-in *can* implement all makefiles as described in [Section 4.6.2.3, “Basic concepts of the makefiles”](#). However, if some of the files do not exist, this will *not* result in an error message.

On the one hand, this behavior may be confusing, as it does not indicate whether important files are missing. On the other hand, this behavior keeps you from a permanent need to implement the complete set of files, even if the set is not necessary for a specific use case.

Almost all central functionality of the build environment is implemented in the following files:

The file names are sorted alphabetically according to the filename:

Filename	Function
global.mak	Provides all global rules and configuration of the build environment, and includes all other plug-in makefiles. To make use of the makefile framework, include <code>global.mak</code> in your application makefile.
tresos2_rules.mak	Provides rules and configuration for executing the code generation of EB tresos Studio from the command line.

Table 4.12. Central makefiles

Only a few files are located within the project directory. To edit configuration options in the makefiles, you may find the files in `$(PROJECT_ROOT)/util`. Depending on the target and toolchain selected, you may also find the files in `$(PROJECT_ROOT)/source/boards`:

Filename	Function
Makefile.mak	This file is the starting point for the build process, and it includes the <code>common.mak</code> . Usually, you don't need to edit it.
common.mak	This file contains common settings for the make environment such as include paths and C files to built, and it includes the <code>global.mak</code> . You need to edit it if you want to add additional include or source files.
<TARGET>_<DERIVATIVE>_Makefile.mak	This file contains the target-specific settings of the build process. It is included by the <code>common.mak</code> . You need to edit the target-specific options in this file.
<TARGET>_<DERIVATIVE>_<TOOLCHAIN>_cfg.-mak	This file contains toolchain-specific options. For example, you may select the C compiler options here. The file is included by <code>common.mak</code> .



Filename	Function
<MODULE>_cfg.mak	Optionally you may add module configuration files in the util directory. For example, if you add a file Det_cfg.mak here, it will override the settings in \$(TRESOS_BASE)/plugins/Det_TS_<version string>/make/Det_cfg.mak

Table 4.13. Application makefiles in \$(PROJECT_ROOT)/util

Filename	Function
<BOARD_MAKEFILE>.mak	This file contains settings for the selected board support package. Depending on the board selected, you may need to edit this file. Check Section 4.6.2.5, "Board support packages" for details.

Table 4.14. Application makefiles in \$(PROJECT_ROOT)/source/boards

4.6.2.5. Board support packages

A board support package contains code fragments which are necessary to adapt an application to specific hardware, for example a specific startup code. The board support package also contains the linker command file.

You may find all files in the directory \$(PROJECT_ROOT)/source/boards.

It is not possible to configure the board support package with EB tresos Studio. If you do need to configure the board support package, edit the files in this directory manually.

Depending on the actual target, several boards may be available. To select one of the boards, set the variable BOARD in `Makefile.mak`.

When you use the AUTOSAR memory abstraction, you must at least edit the linker script to map your sections in `MemMap.h` to the appropriate location. For details on the syntax of the linker script, take a look at the documentation of your compiler toolchain.

4.6.3. Configuring the build process

You need to configure the build environment in the makefiles listed in [Section 4.6.2.4, "Make plug-ins"](#). It is not possible to edit any configuration of the build environment (EB tresos AutoCore Make module) in the EB tresos Studio configuration editor.



The pink field in the workflow overview indicates where in the workflow you currently are:

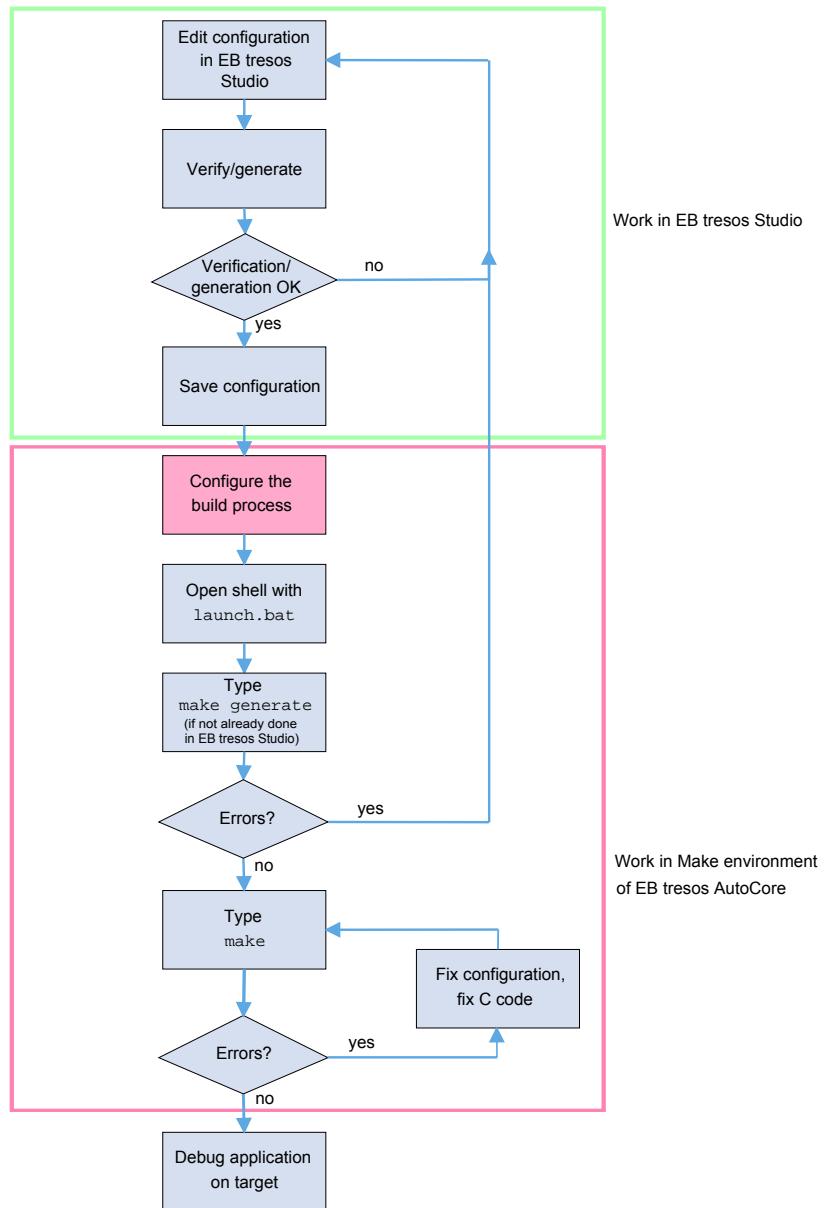


Figure 4.15. Workflow step: configuring the build process

4.6.3.1. Setting the application path



You need to set up some global parameters, before you can successfully compile your application.

The first and most important global parameter to set is `TRESOS_BASE`, which is located in the `util/launch_cfg.bat` batch file. This parameter points to the location of the EB tresos Studio application.

This parameter is set during installation of EB tresos Studio. If you change the default value, you need to manually set the location of EB tresos Studio now.

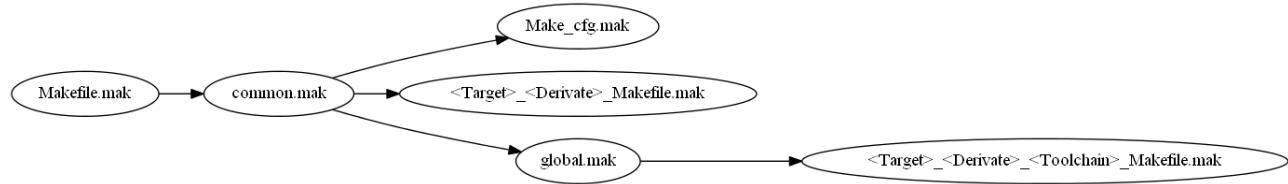
The second global parameter you may have to set is `PLUGINS_BASE`, which is also located in the `util/launch_cfg.bat` batch file. This parameter points to the base directory containing the EB tresos AutoCore plug-ins. If you do not use the default location `TRESOS_BASE/plugins`, you need to manually set the location of the plug-ins now.

To change these parameters manually:

- ▶ Open the `launch_cfg.bat` batch file in `util`.
- ▶ Find the line which contains `TRESOS_BASE`.
- ▶ Change `SET TRESOS_BASE=` to the actual path name, e.g. to `D:/EB/tresos`.
- ▶ Find the line containing `PLUGINS_BASE`.
- ▶ Change `SET PLUGINS_BASE=` to the actual pathname, e.g. to `D:/EB/tresos/AutoCore`.

Set all other global parameters in the makefile `common.mak`. For a list of configuration parameters, see [Section 4.6.3.3, “Configuring the makefiles”](#).

4.6.3.2. Setting the compiler path



To configure the compiler, the variables for `TARGET`, `DERIVATIVE` and `TOOLCHAIN` have to be set according to the naming used in the makefiles delivered along with the `util` directory.

The makefile `<TARGET>_<DERIVATIVE>_<TOOLCHAIN>.cfg.mak` contains a variable `TOOLPATH_COMPILER` which can be defined over an environment variable as well. This variable has to point to a valid path of the compiler executable. If the executable is not found at this location, the make process will exit with an error message.

**NOTE****White-spaces in file names**

The `Make` toolchain is not able to handle white-spaces in file names. This is because all list operations in `Make` are operating on white-space separated lists. Therefore `Make` will handle any white-space as a separator to another element of the list. It is further not possible to handle white-spaces via escape sequences.

If you have a white-space in the compiler directory, there are the following options to resolve this problem:

- rename the compiler directory
- map the compiler directory to a drive letter via the Windows **subst** command
- map the compiler directory to a non-white-space directory via the Windows **junction** command
- define the compiler directory via its 8dot3 representation as obtained via the Windows **dir /X** command

4.6.3.3. Configuring the makefiles

The application makefiles, located in the `util` directory in the project folder, contain all global configuration parameters for the build process. You may change configuration parameters by editing variables in these files.

To ensure that all changes take effect, you need to rebuild your project completely.

The following table contains a summary of all configuration options. The table is sorted alphabetically by parameter name. The detailed setup of each parameter is described in the following sections and is linked to from within the list below. The parameter location is based on the location in the (optional) application example delivered with the EB tresos AutoCore.

Configuration parameter	Configuration option	Parameter location
ASM_FILES_TO_BUILD	Define the list of ASM files to build. For instructions, see Section 4.6.3.3.4, “Defining a list of assembler files” .	<code>\$(PROJECT_ROOT)/util/common.mak</code> , module makefiles
BOARD	Set the name of the board you use for the project. For instructions, see Section 4.6.3.3.9, “Selecting a board support package” .	<code>\$(PROJECT_ROOT)/util/\$(TARGET)_\$(DERIVATE)_Makefile.mak</code>
CC_FILES_TO_BUILD	Define the list of C files to build. For instructions, refer to Section 4.6.3.3.4, “Defining a list of assembler files” .	<code>\$(PROJECT_ROOT)/util/common.mak</code> , module makefiles



Configuration parameter	Configuration option	Parameter location
	Section 4.6.3.3.3, “Defining C files to be compiled”.	
CC/CPP/ASM_INCLUDE_PATH	Extend the include path of the C,C++ or assembler tools. For instructions, see Section 4.6.3.8. “Extending the include path with project-specific path names”.	<code>\$(PROJECT_ROOT)/util/common.mak, module makefiles</code>
CC/CPP/ASM_OPT	Set extra options for the C, C++ or assembler tools.	<code>\$(PROJECT_ROOT)/util/common.mak, module makefiles</code>
TOOLCHAIN	Define the name of the compiler tool chain you use. For instructions, see Section 4.6.3.10, “Selecting a toolchain/a different compiler”.	<code>\$(PROJECT_ROOT)/util/\$(TARGET)_\$(DERIVATIVE)_Makefile.mak</code>
CPP_FILES_TO_BUILD	This variable is only supported on Windows platforms. Other platforms need adaptation of the rules in <code>compiler_rules.mak</code> in the Compiler plug-in.	<code>\$(PROJECT_ROOT)/util/common.mak, module makefiles</code>
LIBRARIES_LINK_ONLY	Define a list of libraries to link.	<code>\$(PROJECT_ROOT)/util/common.mak, module makefiles</code>
LIBRARIES_TO_BUILD	Define a list of libraries to build. For instructions, refer to Section 4.6.3.5, “Building libraries of compiled object files”.	<code>\$(PROJECT_ROOT)/util/common.mak, module makefiles</code>
OBJECTS_LINK_ONLY	Define a list of precompiled objects to link. For instructions, see Section 4.6.3.7, “Linking binary objects to a project”.	<code>\$(PROJECT_ROOT)/util/common.mak, module makefiles</code>
PROJECT	Define the name of the project. For instructions, see Section 4.6.3.2, “Defining the name of the project”.	<code>\$(PROJECT_ROOT)/util/common.mak</code>

Table 4.15. Configuration parameters

4.6.3.3.1. Starting the command shell

The command shell launcher is located at `MyProject/util/launch.bat`.



To open the command shell.

- ▶ Double-click on the launch.bat.

To execute any of the makefile commands:

- ▶ Type the command in the command shell opened with launch.bat.

4.6.3.3.2. Defining the name of the project

The variable `PROJECT` holds the name of the project. Several other variables in the build environment are derived from the project name. For example, the name of the binary executable starts with the project name.

Usually the project and the name of the base directory of the project are the same.

To manually set up a different name:

- ▶ Set the parameter `PROJECT` to the name desired, e.g. `MyProject` as shown in the example below.



Example 4.4. Renaming the project to MyProject with the aid of the `PROJECT` parameter

```
PROJECT = MyProject
```

4.6.3.3.3. Defining C files to be compiled

The `CC_FILES_TO_BUILD` variable allows you to define a list of C files to compile.

To define the list of C files:

- ▶ Set the parameter `CC_FILES_TO_BUILD` to the file names as shown in the example below.



Example 4.5. Defining C files to be compiled with `CC_FILES_TO_BUILD`

```
C_FILES_TO_BUILD += \
$(PROJECT_ROOT)/source/application/test/file1.$(CC_FILE_SUFFIX) \
$(PROJECT_ROOT)/source/application/test/file2.$(CC_FILE_SUFFIX) \
$(PROJECT_ROOT)/source/application/test/file3.$(CC_FILE_SUFFIX)
```



4.6.3.3.4. Defining a list of assembler files

The `ASM_FILES_TO_BUILD` variable allows you to define a list of assembler files, which are processed by the build environment.

To define the list of assembler files:

- ▶ Add all assembler files you want to process to the variable `ASM_FILES_TO_BUILD` as shown in the following example.

The list of assembler files is defined.



Example 4.6. Defining a list of assembler files

```
ASM_FILES_TO_BUILD += \
$(PROJECT_ROOT)/source/application/test/file7.$(ASM_FILE_SUFFIX) \
$(PROJECT_ROOT)/source/application/test/file8.$(ASM_FILE_SUFFIX) \
$(PROJECT_ROOT)/source/application/test/file9.$(ASM_FILE_SUFFIX)
```

The file suffix depends on the compiler makefile plug-in you choose.

The `<filename>_ASM_OPT` variable can be used for ASM files in the same way as for C files.

4.6.3.3.5. Building libraries of compiled object files

It is possible to organize the compiled object files in libraries. Such a library can then for example be delivered to a customer and linked against an application.

To build libraries of compiled object files, follow the instructions in step 1 and step 2.

Step 1: To build a library, the first step is to define a list of libraries that are to be built. The variable `LIBRARIES_TO_BUILD` holds a list of libraries to build.

To define, which libraries are added to the list:

- ▶ Set the parameter `LIBRARIES_TO_BUILD` to the library name(s) as shown in the example below.

The `LIBRARIES_TO_BUILD` now stores the list of libraries to be built.



Example 4.7. Defining the list of libraries

```
LIBRARIES_TO_BUILD += BrakePedalSensor SWC_IndicatorAtomic
```

Step 2: The second step when building a library is to define a list of source files. For each library on your list, you need to define a list of source files in the <library_name>_FILES variable. The build environment compiles all files from this list and adds them to the library.

To define the list of source files:

- ▶ Set the parameter <filename> you defined in step 1 and add _FILES = to the name.
- ▶ Add the source files as shown in the example below.

The list of source files for the first library is now defined.

- ▶ Repeat this step as shown in the example below until you have defined all source files for all libraries.



Example 4.8. Defining the list of source files

```
BrakePedalSensor_FILES = \
$(PROJECT_ROOT)/source/application/blinker_main.$(CC_FILE_SUFFIX) \
$(PROJECT_ROOT)/source/application/blinker_init.$(CC_FILE_SUFFIX) \
$(PROJECT_ROOT)/source/application/blinker_cyclic.$(CC_FILE_SUFFIX)

SWC_IndicatorAtomic_FILES = \
$(PROJECT_ROOT)/source/application/light_main.$(CC_FILE_SUFFIX) \
$(PROJECT_ROOT)/source/application/light_on.$(CPP_FILE_SUFFIX) \
$(PROJECT_ROOT)/source/application/light_off.$(ASM_FILE_SUFFIX)
```

The file list may contain C, C++ and assembler files.

The result of step 1 and step 2: two library files will be created in the directory \$(PROJECT_OUTPUT_PATH) / lib:

1. BrakePedalSensor.<toolchain specific lib extension>
2. SWC_IndicatorAtomic.<toolchain specific lib extension>

In the following, two typical examples of makefile targets are explained in detail. See [Section 4.6.3.4.2, “Other make targets”](#) for a list of other make targets.

To delete the library files in \$(PROJECT_OUTPUT_PATH) / lib:



- ▶ Type **make clean** in the command shell and hit **Enter**.

4.6.3.3.6. Linking libraries to a project

If you want to link binary libraries to your project without building them, add them to the variable `LIBRARIES_LINK_ONLY`.

To link libraries to your project:

- ▶ Set the `LIBRARIES_LINK_ONLY` variable to the name of the library to be included as shown in the example below.

The library is now *linked* to your project rather than included.



Example 4.9. Linking a library

```
LIBRARIES_LINK_ONLY += c:/Programme/matlab/libs/matlab.$(LIB_FILE_SUFFIX)
```

The extension of the library file depends on the compiler makefile plug-in you choose.

4.6.3.3.7. Linking binary objects to a project

You may link binary object files to your project by adding them to the variable `OBJECTS_LINK_ONLY`.

To link a binary object to your project:

- ▶ Set the `OBJECTS_LINK_ONLY` variable to the name of the binary object as shown in the example below.

Your binary object is now linked to your project.



Example 4.10. Linking a binary object

```
OBJECTS_LINK_ONLY = c:/Programme/matlab/libs/matlabobj.obj
```

4.6.3.3.8. Extending the include path with project-specific path names

You may extend the include path with project-specific path names by extending the variables `CC_INCLUDE_PATH` for C files, `CPP_INCLUDE_PATH` for C++ files and `ASM_INCLUDE_PATH` for assembler files.



To extend the include path with project-specific path names:

- ▶ Depending on the type of file, set the variable
 - ▶ `CC_INCLUDE_PATH` for C files,
 - ▶ `CPP_INCLUDE_PATH` for C++ files,
 - ▶ or `ASM_INCLUDE_PATH` for assembler files
- to the include path as shown in the example below.

The include path is now extended.

The build environment adds the additional path names when invoking the C-compiler, the C++-compiler, or the assembler compiler.



Example 4.11. Defining the include path with `CC_INCLUDE_PATH`

```
CC_INCLUDE_PATH = \
$(PROJECT_ROOT)/source/common/include \
$(PROJECT_ROOT)/source/diag/include \
$(PROJECT_ROOT)/source/network/include
```

4.6.3.3.9. Selecting a board support package

Via the variable `BOARD` you may select a board support package for your project. For details on board support packages, see [Section 4.6.2.5, “Board support packages”](#). Depending on your target device, one or more board support packages are available.

To select a board support package:

- ▶ Set the `BOARD` variable to the board support package of your choice as shown in the example below.

In the example, the evaluation board `eva168_2` is selected.

The board support package is now selected.

The build environment includes the makefile `$(PROJECT_ROOT)/source/boards/$(BOARD) / $(BOARD).mak` to find instructions specific to this board.



Example 4.12. Selecting a board support package

```
BOARD ?= eva168_2
```



4.6.3.3.10. Selecting a toolchain/a different compiler

Every target defines a default toolchain in the build environment.

You can have different compiler plug-ins for a specific target. Each compiler plug-in supports one toolchain. If you want to select a toolchain, set the variable `TOOLCHAIN`.

To select a different compiler/another toolchain:

- ▶ Set the `TOOLCHAIN` variable to the name of the compiler as shown in the example below.



Example 4.13. Selecting a different compiler

```
TOOLCHAIN := tasking
```

4.6.3.3.11. Extending compiler and assembler options

You can extend the options for the assembler, the C compiler and the C++ compiler by adding values to the variables `ASM_OPT`, `CC_OPT`, and `CPP_OPT`, respectively.

To extend the compiler options:

- ▶ Set either of the following, depending on which compiler options you are extending:
 - ▶ `ASM_OPT` for the assembler
 - ▶ `CC_OPT` for C compilers
 - ▶ or `CPP_OPT` for C++ compilers
- to the option by which you want to extend the compiler. The following example shows you how to do this.
- ▶ Your compiler option is now extended.

The build environment will send these values to the appropriate tool.



Example 4.14. Extending the C compiler option

```
CC_OPT += -ggdb
```



4.6.3.3.12. Providing file-specific or library-specific compiler options

For a single C file or all C files of a single library, you can provide a specific file with compiler options that is then passed by the build environment to the compiler instead of the common one.

To specify a file-specific compiler options file, set the `{file name}_OPTMAP` variable to the path of the compiler options file. Replace `{file name}` with the file name of the related C file.

To specify a library-specific compiler options file, set the `{library name}_OPTMAP` variable to the path of the compiler options file. Replace `{library name}` with the name of the related library.

If you want to create such a specific compiler options file, you should create a copy of the common one and change options in or add options to this copy. The common compiler options file is created during the build in `$(PROJECT_ROOT)/output/make`, file `compiler.inc`.

See the examples below how to set the variables to use the compiler options file `C:/EB/tresos/workspace/MyProject/util/appl_options.inc` for a C file `appl.c`, and the compiler options file `C:/EB/tresos/workspace/MyProject/util/Det_src_options.inc` for all C files of a library `Det_src`:



Example 4.15. Use specific compiler options file for a single C-file

```
appl_OPTMAP := C:/EB/tresos/workspace/MyProject/util/appl_options.inc
```



Example 4.16. Use specific compiler options file for all C files of a library

```
Det_src_OPTMAP := C:/EB/tresos/workspace/MyProject/util/Det_src_options.inc
```

4.6.3.3.13. Adding preprocessor defines

If you need to define additional preprocessor definitions, you should not add them to `CC_OPT` but use the key/ value approach supplied by the build environment. The reason for this is that in most cases the additional definitions have to be available for the processing of dependencies and the processing of the source files.

All preprocessor definitions that you define as key/ value pairs are promoted by the build environment to every build step.



To set up a key/ value pair:

- ▶ Set the `PREPROCESSOR_DEFINES` to your values, e.g. as shown in the example below.



Example 4.17. Setting up a key/value pair

```
PREPROCESSOR_DEFINES    += myDefine  
myDefine_KEY            := MY_DEFINE  
myDefine_VALUE          := TRUE
```

This results in the following command line option:

```
-DMY_DEFINE=TRUE
```

This command line option is passed to the preprocessor *and* to the compiler.

4.6.3.4. Compiling the application

The pink field in the workflow overview indicates where in the workflow you currently are:

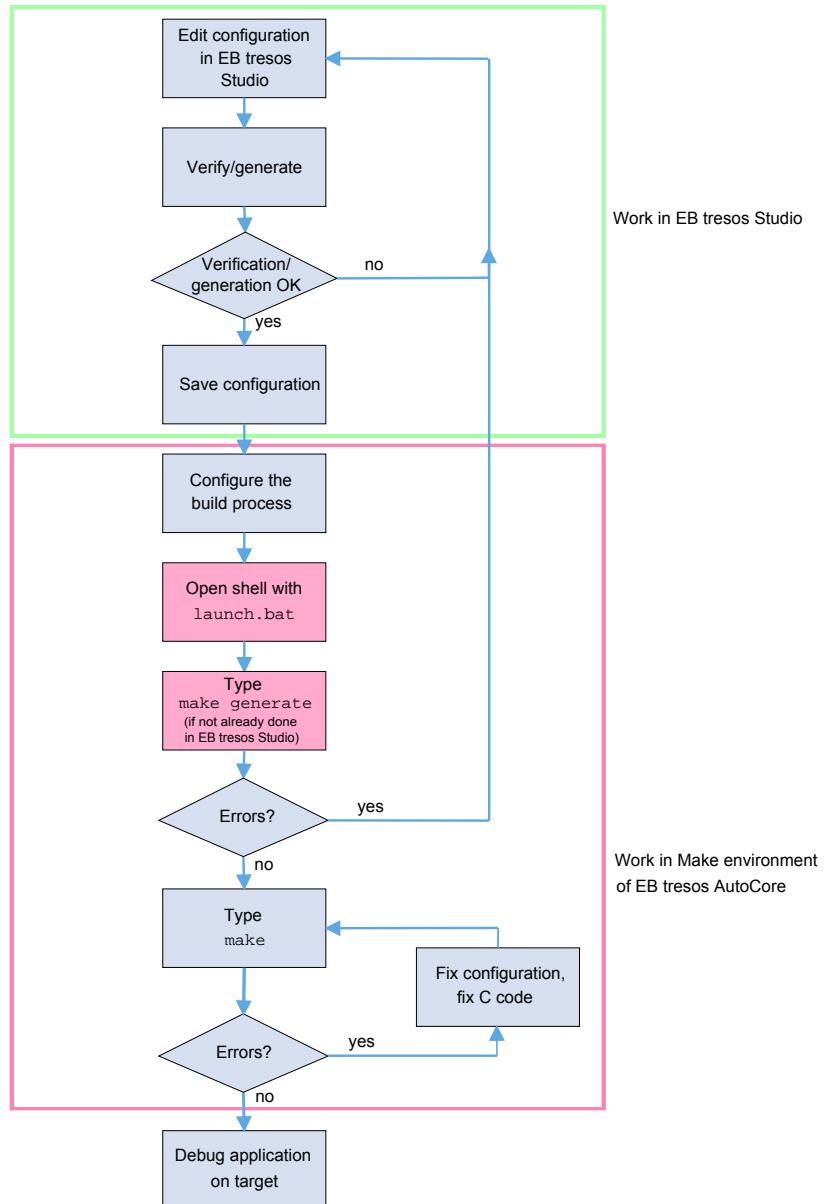


Figure 4.16. Workflow step: compiling the application

To generate, compile, and link your application from scratch:

1. Start the `launch.bat` batch file from the `$(PROJECT_ROOT)/util` directory.



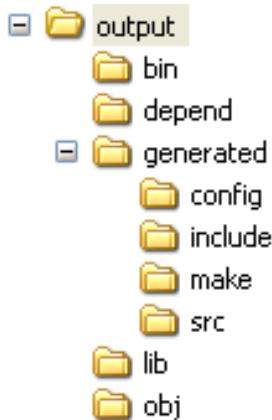
The command shell opens up.

2. To generate all configuration-dependent files, type **make generate** into the command shell.
3. To process the dependency information and build the binary executable, type **make** into the command shell. To speed up compilation, you can use GNU Make's built-in job parallelization by calling **make -j**.

The target **make** processes the dependency information by analyzing all source files by the GCC pre-processor, compiles all objects files, builds all libraries and links the application binary.

4.6.3.4.1. Locating output directories

You may find all folders and files generated by **make** and also by **make generate** if the default generator output path is not changed in EB tresos Studio in the `$(PROJECT_ROOT)/output` directory of your project:



The following list gives you a short overview of the content of the different output directories:

- ▶ **bin:** Contains the binary executable file.

The format and the filename extension differ depending on the compiler toolchain you choose. Files in the ELF format often have the extension `.elf`. Some toolchains also create other files here, e.g. HEX files that can be used with specific FLASH programming tools.

- ▶ **depend:** For every source file that is processed in the dependency processing, a makefile fragment is created in this directory.

This fragment defines on which other files a source file depends. For a file `Can.c` e.g. **make depend** creates a file `Can.mak` in this directory.

- ▶ **generated:** All output files of the generator tools are created in this directory if the generator output path is not changed in EB tresos Studio. The directory contains some subdirectories such as `source` and `include` to organize the generated files.
- ▶ **lib:** This is the default output path for libraries. All libraries are archived in this directory, unless you define an alternate output path for a library.



4.6.3.4.2. Other `make` targets

The build environment provides some more targets that are important for your everyday work. This chapter will give you an short overview of these `make` targets.

`make show_rules`

`make show_rules` gives an overview of all available targets. You can register additional messages to display information about specific targets you have implemented.

`make single_file SF=filename`

`make single_file` compiles only one certain source file into an object file. This target is useful to verify only a single source file.

`make single_lib SL=libname`

`make single_lib` compiles and links only those source files that are required to build a certain single library. This target is useful when you need to recompile only one library because of changed settings.

`make single_lib_clean SL=libname`

`make single_lib_clean` removes the library, all object and dependency files that are needed to build a certain single library. This target is useful when you need to recompile libraries because of changed compiler switches or a toolchain update.

`make clean`

`make clean` removes files created by the compiler (including dependency information) or by the build environment (deletes folders `bin`, `depend`, `lib`, `obj`, `make` in directory given by `PROJECT_OUTPUT_PATH`). You can use this target to rebuild all your application code with `make` without the need to run `make generate`.

`make clean_all`

`make clean_all` deletes all files generated by the compiler, the build environment, or by the code generator.

In addition to `make clean`, `make clean_all` also deletes the directory where files are placed created by a code generator. The path is set in EB tresos Studio and is provided as the makefile variable `GEN_OUTPUT_PATH`.

`make check_dups`

`make check_dups` iterates over all directories in the list defined by `CC_INCLUDE_PATH` and checks if any header file with the same names exists in multiple directories. If duplicates are found, a warning is printed out.

When including a file, the compiler searches the include paths in the given order and stops searching as soon as it finds a file matching the file name. This is dangerous if multiple files of the same name exist. Additionally, you cannot rely on the order because the make environment uses the sort function to remove duplicates from the include file list.

`make flat_src`

`make flat_src` creates the directory `output/flat_src` and copies all headers and source files into this location. Additionally, it creates files containing lists defining which files are linked as objects directly. It further copies all static libs and objects.



This target creates an extract of the sources which is as flat as possible. Since some headers or sources are using folder names within the **include** directives, the respective sub folders with the header files are created.

4.7. Service Needs Calculator

4.7.1. Overview

This chapter provides information about the Service Needs Calculator. This chapter also describes how to use this unattended wizard to automatically resolve configuration dependencies between different EB tresos AutoCore modules.

- ▶ To learn about the concept of the Service Needs Calculator, see [Section 4.7.2, “Background information”](#).
- ▶ To learn about how to calculate the service Needs, see [Section 4.7.3, “Using the Service Needs Calculator”](#).

4.7.2. Background information

In AUTOSAR, a service is a logical entity that offers general functionality to software components and BSW modules. For example, the **NvM** offers services to read and write data blocks to the memory and is regarded as a service provider. To use such a service, a software component or BSW module may have a configuration dependency on the service provider.

For example, to use the **NvM** services, memory blocks must be configured in the **NvM**. References to these blocks must be configured in the software component itself or in the BSW module that wants to use the **NvM** services. This kind of module dependency is known in AUTOSAR as service Needs.

In EB tresos AutoCore, the concept of service Needs is extended to include further dependencies that may not have a direct reference in the service requester.

For example, the **init** function of BSW modules must be configured in the **EcuM** module. But the BSW modules themselves do not need direct **EcuM** references for this. This kind of dependency is also treated as service Needs between the BSW modules and the **EcuM**.

Service Needs can be seen as a contract between modules that provide services and modules that request services. This contract between service providers and service requesters is communicated by exchanging XML



fragments that describe the requested service Needs. The contract contains information on the configuration dependencies between BSW modules. The Service Needs Calculator coordinates and provides the lifecycle and graphical user interface.

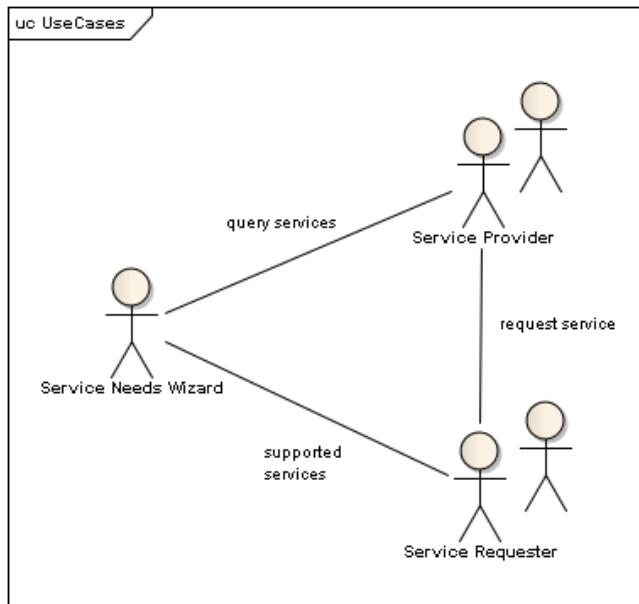


Figure 4.17. Relation between the involved parties of the Service Needs Calculator

The Service Needs Calculator is an unattended wizard. The Service Needs Calculator configures parameters that one BSW module needs in the configuration of another BSW module. These parameters configure a service that a module can offer to other modules. This works the same way as calling a main function.

The Service Needs Calculator automatically collects the requests from the service requester and performs the necessary configuration changes in the service provider.

The Service Needs Calculator is supplied with the EB tresos AutoCore and provides support for the automatic configuration of the following Service Needs:

- ▶ EcuM init functions
- ▶ Dem events
- ▶ NvM blocks
- ▶ Os Tasks
- ▶ Os ISRs
- ▶ Os resources
- ▶ Os schedule tables
- ▶ Os events
- ▶ Os IOC channels



- ▶ Os spinlocks
- ▶ Os alarms
- ▶ Com signals
- ▶ Com signal groups
- ▶ LdCom
- ▶ Xfrm Transformers (ComXf, E2EXf, SomelpXf)

In addition, the Service Needs Calculator provides the following features:

- ▶ If you modify a parameter or reference that had been configured by the Service Needs Calculator, the Service Needs Calculator will not overwrite your manual changes if you run the wizard again.
- ▶ The Service Needs Calculator removes containers, parameters, and references from a previous run if these configurations are not needed anymore in the following run. This happens only if you didn't modify these configurations manually.
- ▶ Several modules that require a Dem event or an NvM block have a reference to the Dem event or the NvM block in their own configuration. The Service Needs Calculator sets this reference automatically.

4.7.3. Using the Service Needs Calculator

Before using the Service Needs Calculator for the first time, you have to configure the wizard to suit your project needs. After you configured the Service Needs Calculator, you can enable or disable the wizard as required.

4.7.3.1. Configuring the Service Needs Calculator



Opening the Service Needs Calculator Configuration

Prerequisite:

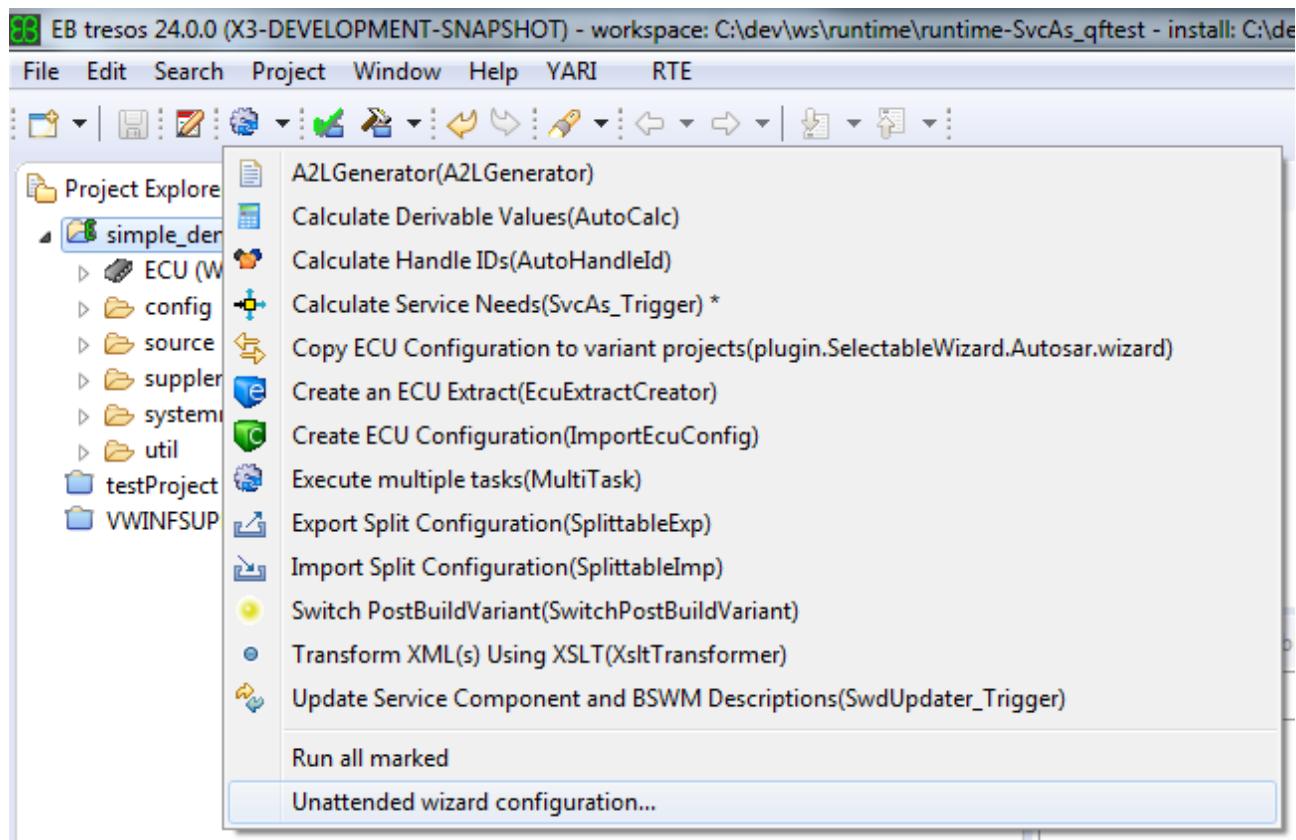
- EB tresos Studio is installed and licensed.

Step 1

To open the Service Needs Calculator configuration dialog, click on the black arrow in the button located in the menu bar of EB tresos Studio.

Step 2

In the **Unattended Wizards** list, click **Unattended wizard configuration....**

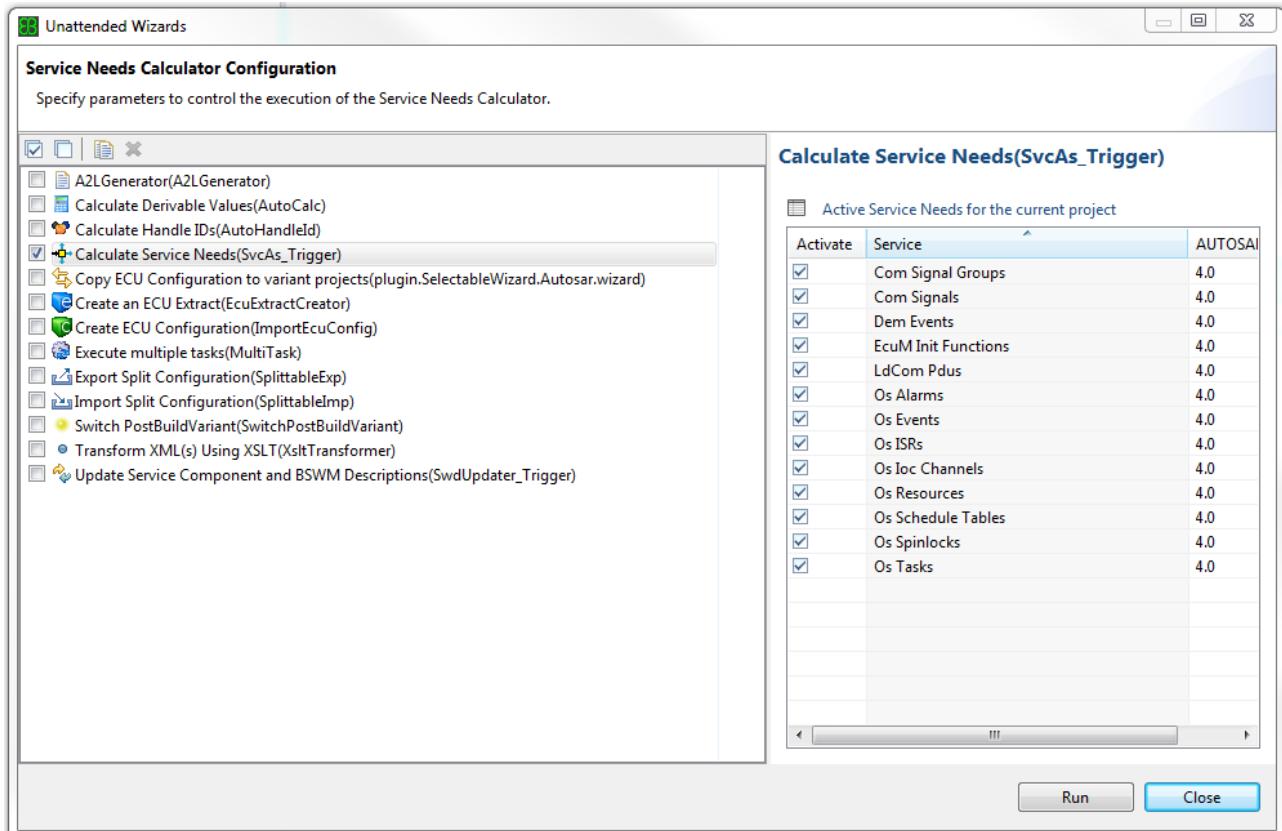


Step 3

To enable the automatic configuration of Service Needs, enable the check box of the needed Service Needs item.

**TIP****Active Service Needs for the current project**

The content in the **Active Service Needs** list for the current project is automatically derived depending on your project configuration. This list only shows Service Needs that are applicable in the current project.

**NOTE****The Service Needs Calculator issues a warning if OsSecondsPerTick is incorrect**

If the `OsSecondsPerTick` parameter is not set correctly in the `Os` configuration, the Service Needs Calculator cannot configure the timing values of the schedule table correctly. In this situation, a warning is shown during the execution of the Service Needs Calculator.

4.7.3.2. Running the Service Needs Calculator

Run the Service Needs Calculator under the following circumstances:

- ▶ Always run the Service Needs Calculator after you add new modules to your project.
- ▶ Always run the Service Needs Calculator after you remove modules from your project.



- ▶ Always run the Service Needs Calculator after you modify parameters in the configuration of a module, which has a service that is required by another module. For example, when you add an additional init configuration to the CanIf module, the CanIf module requires additional events to be configured in the Dem.



Prerequisite:

- Service Needs Calculator is configured and enabled.

Step 1

To run the Service Needs Calculator, click

Information on added, modified, or removed elements is shown in the **Results** view. This view shows warnings if problems occurred during the configuration of the Service Needs.

The screenshot shows the EB tresos IDE interface with the 'Results' tab selected. The main area displays a hierarchical tree of configuration elements under the project 'simple_demo_can_rte'. The tree includes nodes for 'ECU (WINDOWS, WIN32X86)', 'Com Services' (with 'Com (V6.3.32, AS4.0.3)' and its sub-nodes like 'ComConfig', 'ComSignal', and 'SGCounterOut_272T'), and 'OS/RTE' (with 'Os (V6.0.132, AS4.0.3)' and its sub-nodes like 'OsScheduleTable' and 'SchM_DefaultScheduleTable*'). The bottom pane, titled 'Message', lists several log entries with icons indicating their type (information, warning, error). Some key messages include:

- (SVCECUM_1) EcuM Init Functions
- (SVCCOM_1) Com Signals
- (SVCLIBA_4) Removing obsolete entry /AUTOSAR/TOP-LEVEL-PACKAGES/Com/ELEMENTS/Com/ComConfig/ComConfig/ComSignal/SGCounterOut_272T/ComDataInvalidAction.
- (SVCLIBA_4) Removing obsolete entry /AUTOSAR/TOP-LEVEL-PACKAGES/Com/ELEMENTS/Com/ComConfig/ComConfig/ComSignal/SGCounterOut_272T/ComRxDataTimeoutAction.
- (SVCLIB_3) Ignoring imported parameter /AUTOSAR/TOP-LEVEL-PACKAGES/Com/ELEMENTS/Com/ComConfig/ComConfig/ComSignal/SGCounterOut_272T.
- (SVCCOM_2) Com Signal Groups
- (SVCOS_1) Os Tasks
- (SVCLIB_3) Ignoring manually edited parameter /AUTOSAR/TOP-LEVEL-PACKAGES/Os/ELEMENTS/Os/OsTask/Rte_Event_Task.
- (SVCLIB_3) Ignoring manually edited parameter /AUTOSAR/TOP-LEVEL-PACKAGES/Os/ELEMENTS/Os/OsTask/SchMDiagStateTask_20ms.
- (SVCLIB_3) Ignoring manually edited parameter /AUTOSAR/TOP-LEVEL-PACKAGES/Os/ELEMENTS/Os/OsTask/SchMDiagStateTask_20ms/OsStacksize.
- (SVCLIB_3) Ignoring manually edited parameter /AUTOSAR/TOP-LEVEL-PACKAGES/Os/ELEMENTS/Os/OsTask/Rte_Time_Task.
- (SVCOS_5) Os Events
- (SVCLIB_2) Adding entry Rte_OSShutdownEvent.
- (SVCOS_6) Os Alarms
- (SVCOS_7) Os Resources
- (SVCOS_8) Os Schedule Tables
- (SVCOS_11) Os Ioc Channels
- (SVCOS_12) Os ISRs



5. ACG8 Base module references

5.1. Overview

This chapter provides module references for the ACG8 Base product modules. These include a detailed description of all configuration parameters. Furthermore this chapter lists the application programming interface with all data types, constants and functions.

The content of the sections is sorted alphabetically according the EB tresos AutoCore Generic module names.

For further information on the functional behavior of these modules, refer to the chapter ACG8 Base user's guide.

5.1.1. Notation in EB module references

EB notation may differ from the AUTOSAR standard notation in the software specification documents (SWS). This section describes the notation of *default value* and *range* fields in the EB module references.

5.1.1.1. Default value of configuration parameters

If there is no default value specified for a parameter, the default value field is omitted to prevent ambiguity with parameters that have -- as default values.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has no default value field, therefore it is omitted.

5.1.1.2. Range information of configuration parameters

The range of a configuration parameter contains an upper and a lower boundary. However, in special cases the range of allowed values can be computed by means of an XPath function that is evaluated at configuration time. An XPath function can either be a standard `xpath:<function>()` or a custom `cxpath:<function>()` function. The range of a configuration parameter may be computed based on other configuration parameters that are referenced from the XPath function. For more information on custom XPath functions, see section *Custom XPath Functions API* of the EB tresos Studio developer's guide.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has the custom XPath function `cxpath:getCompuMethodsVT()` in the range field which provides the allowed values.



5.2. Atomics

5.2.1. Configuration parameters

There are no configuration parameters.

5.2.2. Application programming interface (API)

5.2.2.1. Type definitions

5.2.2.1.1. Atomic_t

Purpose	The type of atomic objects.
Type	uint8
Description	All atomic objects are of this type. The underlying type is defined by the macro ATOMICS_USER_ATOMIC_T. Multiple threads of execution may work on objects of this type concurrently. Objects of this type are opaque and you must access them only via the <code>Atoms_*()</code> functions. This guarantees atomicity and sequential consistency.

5.2.2.1.2. Atomic_value_t

Purpose	The value type of atomic objects.
Type	uint8
Description	All atomic objects store a value of this type. Its underlying type is defined by the macro ATOMICS_USER_ATOMIC_VALUE_T. Objects of this type must only be used locally, because they are not designed for concurrent access: there are no atomicity or sequential consistency guarantees. You can use them directly, as unsigned integers, in expressions.

5.2.2.1.3. TS_IntStatusType

Purpose	Type to store the interrupt status.
----------------	-------------------------------------



Type	uint32
Description	Objects of this type can be used to store the (enabled/disabled) status of hardware interrupts. These are usually exchanged between TS_IntDisable() and TS_IntRestore() , to disable and restore this status.

5.2.2.2. Macro constants

5.2.2.2.1. ATOMICS_OBJECT_INITIALIZER

Purpose	Expands into an initializer for an atomic object.
Value	ATOMICS_USER_OBJECT_INITIALIZER(initValue)
Description	<p>This macro must be used to initialize an atomic object of type Atomic_t at compile-time. You can assign its expansion directly to an object of type Atomic_t.</p> <p>Atomic objects with static storage duration are initialized with the initial value zero at program load time. In this case, you don't have to use ATOMICS_OBJECT_INITIALIZER(0) explicitly.</p> <p>See also Atomics_Init(), which does the same at runtime.</p> <p>It is an error to use an atomic object without prior initialization.</p> <p>The behavior is the same as what C11 specifies for the following: ATOMIC_VAR_INIT(initValue)</p>

5.2.2.2.2. ATOMICS_USER_ATOMIC_T

Purpose	The type of atomic objects.
Value	uint8
Description	<p>This macro defines the type of atomic objects. It is used in the typedef declaration of the type Atomic_t.</p> <p>The generic implementation assumes, that it's an unsigned integer, which has the native size of the architecture (e.g., 32 bits on 32-bit, or 64 bits on 64-bit CPUs) and is naturally aligned.</p> <p>If you change this macro, please ensure that it's consistent with all ATOMICS_USER_*(*) macros.</p>



5.2.2.2.3. ATOMICS_USER_ATOMIC_VALUE_T

Purpose	The value type of atomic objects.
Value	uint8
Description	<p>This macro defines the value type of atomic objects. All atomic objects internally store objects of this type. It is used in the typedef declaration of the type Atomic_value_t.</p> <p>The generic implementation assumes, that it's an unsigned integer, which has the native size of the architecture (e.g., 32 bits on 32-bit, or 64 bits on 64-bit CPUs).</p> <p>If you change this macro, please ensure that it's consistent with all ATOMICS_USER_*() macros.</p>

5.2.2.2.4. ATOMICS_USER_AUTOSAR_SPINLOCK

Purpose	Id of user-provided AUTOSAR spinlock.
Value	0
Description	<p>When the macro ATOMICS_USER_MULTICORE_CASE expands to 1, this macro must expand to the id of the AUTOSAR spinlock to use to facilitate atomic access to atomic objects. This AUTOSAR spinlock must be provided by you and must also disable all interrupts.</p> <p>If ATOMICS_USER_MULTICORE_CASE expands to 0, this macro has no effect and may be empty.</p>

5.2.2.2.5. ATOMICS_USER_GET_VALUE

Purpose	User callout to get the value of an atomic object.
Value	(* (objaddr))

5.2.2.2.6. ATOMICS_USER_LOCK_OBJECT

Purpose	User callout to lock an atomic object.
Value	do { \ SuspendAllInterrupts(); \ *(successaddr) = TRUE; \ } while (0)
Description	This callout is used by all Atomic_*() and TS_AtomicSet/ClearBit_*() functions, which are passed an atomic object. This object may be of type Atomic_t, uint8, uint16, uint32, or uint64. It must lock the passed atomic object to prevent concurrent access-



es to it. The necessary measures depend on the group of accessors, but usually, interrupts must be disabled and, in case of accessors on multiple cores, an AUTOSAR spinlock must be acquired beforehand. See also the macro ATOMICS_USER_MULTICORE_CASE for further details about these cases.

To identify the atomic object to be locked, its address is passed to this macro. This allows to be more specific about how to lock a specific atomic object.

The success indicator is an output parameter of this macro and tells the caller, whether the atomic object can be safely accessed.

If you adapt one of ATOMICS_USER_THREAD_FENCE, ATOMICS_USER_LOCK_OBJECT or ATOMICS_USER_UNLOCK_OBJECT, make sure that these three macros are consistent and provide the intended functionality.

5.2.2.2.7. ATOMICS_USER_MULTICORE_CASE

Purpose	User switch to select between single- and multi-core cases.
Value	1
Description	<p>The generic implementations for the lock and unlock macros ATOMICS_USER_LOCK_OBJECT and ATOMICS_USER_UNLOCK_OBJECT, rely on AUTOSAR mechanisms. This macro selects between the single- and multi-core case. In the first case, all interrupts are disabled, when atomic objects are accessed. In the second case, an AUTOSAR spinlock is used, which must be provided by you. The macro ATOMICS_USER_AUTOSAR_SPINLOCK must expand to the id of this AUTOSAR spinlock.</p> <p>This also means, that in the first case the functions DisableAllInterrupts() and EnableAllInterrupts() are used, while in the second case the functions GetSpinlock() and ReleaseSpinlock() are used. The call environment must permit the use of these functions respectively.</p> <p>If the generic implementation is not used, this macro can be removed.</p>

5.2.2.2.8. ATOMICS_USER_OBJECT_INIT

Purpose	Initializes the atomic object with the given value.
Value	do {*(objaddr) = (initValue);} while (0)
Description	The initialization must be consistent with ATOMICS_USER_ATOMIC_T.



5.2.2.2.9. ATOMICS_USER_OBJECT_INITIALIZER

Purpose	Expands into an initializer for an atomic object.
Value	(initValue)
Description	See ATOMICS_OBJECT_INITIALIZER for further details. The initializer must be consistent with ATOMICS_USER_ATOMIC_T.

5.2.2.2.10. ATOMICS_USER_SET_VALUE

Purpose	User callout to set the value of an atomic object.
Value	do {} *(objaddr) = (value); } while (0)

5.2.2.2.11. ATOMICS_USER_THREAD_FENCE

Purpose	User callout for thread fence.
Value	do {} while (0)
Description	<p>This macro is called by Atomics_ThreadFence() as well as other Atomics_*() functions to establish a sequentially consistent memory barrier.</p> <p>The generic implementation of this macro is empty, as it is not necessary for the generic implementations of the other Atomics_*() functions. Note that this means, that Atomics_ThreadFence(), which is based on this macro, does actually not implement a memory barrier.</p> <p>If you adapt one of ATOMICS_USER_THREAD_FENCE, ATOMICS_USER_LOCK_OBJECT or ATOMICS_USER_UNLOCK_OBJECT, make sure that these three macros are consistent and provide the intended functionality.</p>

5.2.2.2.12. ATOMICS_USER_UNLOCK_OBJECT

Purpose	User callout to unlock an atomic object.
Value	do {} ResumeAllInterrupts(); *(successaddr) = TRUE; } while (0)
Description	This callout is used by all Atomic_*() and TS_AtomicSet/ClearBit_*() functions, which are passed an atomic object. This object may be of type Atomic_t, uint8, uint16, uint32, or uint64. It is the inverse function of ATOMICS_USER_LOCK_OBJECT. It



shall unlock the given atomic object, so that other accessors get a chance to work on it. It has the same parameters as ATOMICS_USER_LOCK_OBJECT.

If you adapt one of ATOMICS_USER_THREAD_FENCE, ATOMICS_USER_LOCK_OBJECT or ATOMICS_USER_UNLOCK_OBJECT, make sure that these three macros are consistent and provide the intended functionality.

5.2.2.2.13. ATOMICS_VALUE_MAX

Purpose	The maximum value that an object of type Atomic_value_t can store.
Value	255U

5.2.2.2.14. TS_AtomicClearBit

Purpose	Atomically clears one bit in an object.
Value	<pre>do { \ if (sizeof(*Address)) == 1U) \{ \ TS_AtomicClearBit_8((volatile P2VAR(uint8, AUTOMATIC, BASE_APPL_DATA)) (Address), (Bit)); \ } \ else if (sizeof(*Address)) == 2U) \{ \ TS_AtomicClearBit_16((volatile P2VAR(uint16, AUTOMATIC, BASE_APPL_DATA)) (Address), (Bit)); \ } \ else if (sizeof(*Address)) == 4U) \{ \ TS_AtomicClearBit_32((volatile P2VAR(uint32, AUTOMATIC, BASE_APPL_DATA)) (Address), (Bit)); \ } \ else \{ \ TS_AtomicClearBit_64((volatile P2VAR(uint64, AUTOMATIC, BASE_APPL_DATA)) (Address), (Bit)); \ } \ } while (0)</pre>
Description	<p>This macro atomically clears the bit at index Bit in the object pointed to by Address. The maximum value for Bit depends on the number of bits in the object pointed to by Address.</p> <p>Sequential consistency is ensured.</p> <p>This function is deprecated. New code should use the API declared in Atomics.h.</p>

5.2.2.2.15. TS_AtomicSetBit

Purpose	Atomically sets one bit in an object.
Value	<pre>do { \ if (sizeof(*Address)) == 1U) \{ \ TS_AtomicSetBit_8((volatile P2VAR(uint8, AUTOMATIC, BASE_APPL_DATA)) (Address), (Bit)); \ } \ else if (sizeof(*Address)) == 2U) \{ \ TS_AtomicSetBit_16((volatile P2VAR(uint16, AUTOMATIC, BASE_APPL_DATA)) (Address), (Bit)); \ } \ else if (sizeof(*Address)) == 4U) \{ \ TS_AtomicSetBit_32((volatile P2VAR(uint32, AUTOMATIC, BASE_APPL_DATA)) (Address), (Bit)); \ }</pre>



	\ } \ else \{ \ TS_AtomicSetBit_64((volatile P2VAR(uint64, AUTOMATIC, BASE_AP-PL_DATA)) (Address), (Bit)); \ } \ } while (0)
Description	<p>This macro atomically sets the bit at index Bit in the object pointed to by Address. The maximum value for Bit depends on the number of bits in the object pointed to by Address.</p> <p>Sequential consistency is ensured.</p> <p>This function is deprecated. New code should use the API declared in Atomics.h.</p>

5.2.2.3. Functions

5.2.2.3.1. Atomics_ClearFlag

Purpose	Atomically clears a selected bit.	
Synopsis	<pre>void Atomics_ClearFlag (volatile Atomic_t * object , Atomic_value_t flagSelectionMask);</pre>	
Parameters (in)	flagSelectionMask	The bit mask to select one bit in the atomic object.
Parameters (in,out)	object	The pointer to an atomic object to work on.
Description	<p>This function atomically clears the bit selected by 'flagSelectionMask' in the memory location pointed to by 'object'.</p> <p>Sequential consistency is ensured.</p> <p>The behavior is similar as specified in C11 when calling the function: atomic_flag_clear(object); but more than just one flag are fitted into an os_atomic_t object.</p>	

5.2.2.3.2. Atomics_CompareExchange

Purpose	Atomically exchanges the given values if the comparison succeeds.	
Synopsis	<pre>boolean Atomics_CompareExchange (volatile Atomic_t * ob- ject , Atomic_t * expected , Atomic_value_t newValue);</pre>	
Parameters (in)	newValue	The new value to be stored into the atomic object.



Parameters (in,out)	object	The pointer to an atomic object to work on.
	expected	Points to the expected value of the atomic object for the exchange to happen. The location pointed to is updated with the current value of the atomic object, if the comparison fails and hence, the exchange does not happen.
Return Value	If the comparison succeeds and the atomic object is updated with 'newValue', TRUE is returned. Otherwise, FALSE is returned and the atomic object is only read and put into the location pointed to by 'expected'.	
Description	<p>If the values at 'object' and 'expected' are equal, 'newValue' is written atomically into the memory location pointed to by 'object' and TRUE is returned.</p> <p>If the values at 'object' and 'expected' are not equal, the current value at 'object' is written into the memory location pointed to by 'expected' and FALSE is returned.</p> <p>Sequential consistency is ensured.</p> <p>The behavior is as specified in C11 when calling the following function: <code>atomic_compare_exchange_strong(object, expected, newValue);</code></p>	

5.2.2.3. Atomics_Exchange

Purpose	Atomically exchanges the given values.	
Synopsis	<pre>Atomic_value_t Atomics_Exchange (volatile Atomic_value_t * object , Atomic_value_t newValue);</pre>	
Parameters (in)	newValue	The new value to be stored into the atomic object.
Parameters (in,out)	object	The pointer to an atomic object to work on.
Return Value	The value stored in the atomic object before the exchange.	
Description	<p>This function atomically exchanges the value pointed to by 'object' with the value 'newValue'. The value in memory at 'object' before this exchange is returned.</p> <p>Sequential consistency is ensured.</p> <p>The behavior is as specified in C11 when calling the following function: <code>atomic_exchange(object, newValue);</code></p>	



5.2.2.3.4. Atomics_FetchAdd

Purpose	Atomically adds the given number.	
Synopsis	<pre>Atomic_value_t Atomics_FetchAdd (volatile Atomic_t * object , Atomic_value_t operand);</pre>	
Parameters (in)	operand	The number to add to the atomic object.
Parameters (in,out)	object	The pointer to an atomic object to work on.
Return Value	The value of the atomic object before the operation.	
Description	<p>This function atomically adds 'operand' to the value in memory at location 'object'. The value in this location before the operation is returned.</p> <p>Sequential consistency is ensured.</p> <p>The behavior is as specified in C11 when calling the function: atomic_fetch_-add(object, operand);</p>	

5.2.2.3.5. Atomics_FetchAnd

Purpose	Atomically performs a boolean AND operation.	
Synopsis	<pre>Atomic_value_t Atomics_FetchAnd (volatile Atomic_t * object , Atomic_value_t operand);</pre>	
Parameters (in)	operand	The value to AND to the atomic object.
Parameters (in,out)	object	The pointer to an atomic object to work on.
Return Value	The value of the atomic object before the operation.	
Description	<p>This function atomically performs a boolean AND operation of 'operand' and the value in memory pointed to by 'object' and stores the result into that location. It returns the value in that location before the operation.</p> <p>Sequential consistency is ensured.</p> <p>The behavior is as specified in C11 when calling the function: atomic_fetch_-and(object, operand);</p>	

5.2.2.3.6. Atomics_FetchOr

Purpose	Atomically performs a boolean OR operation.
----------------	---



Synopsis	<code>Atomic_value_t Atomics_FetchOr (volatile Atomic_t * object , Atomic_value_t operand);</code>	
Parameters (in)	operand	The value to OR to the atomic object.
Parameters (in,out)	object	The pointer to an atomic object to work on.
Return Value	The value of the atomic object before the operation.	
Description	<p>This function atomically performs a boolean OR operation of 'operand' and the value in memory pointed to by 'object' and stores the result into that location. It returns the value in that location before the operation.</p> <p>Sequential consistency is ensured.</p> <p>The behavior is as specified in C11 when calling the function: <code>atomic_fetch_or(object, operand);</code></p>	

5.2.2.3.7. **Atomics_FetchSub**

Purpose	Atomically subtracts the given number.	
Synopsis	<code>Atomic_value_t Atomics_FetchSub (volatile Atomic_t * object , Atomic_value_t operand);</code>	
Parameters (in)	operand	The number to subtract from the atomic object.
Parameters (in,out)	object	The pointer to an atomic object to work on.
Return Value	The value of the atomic object before the operation.	
Description	<p>This function atomically subtracts 'operand' from the value in memory at location 'object'. The value in this location before the operation is returned.</p> <p>Sequential consistency is ensured.</p> <p>The behavior is as specified in C11 when calling the function: <code>atomic_fetch_sub(object, operand);</code></p>	

5.2.2.3.8. **Atomics_FetchXor**

Purpose	Atomically performs a boolean XOR operation.	
Synopsis	<code>Atomic_value_t Atomics_FetchXor (volatile Atomic_t * object , Atomic_value_t operand);</code>	



Parameters (in)	operand	The value to XOR to the atomic object.
Parameters (in,out)	object	The pointer to an atomic object to work on.
Return Value	The value of the atomic object before the operation.	
Description	<p>This function atomically performs a boolean XOR operation of 'operand' and the value in memory pointed to by 'object' and stores the result into that location. It returns the value in that location before the operation.</p> <p>Sequential consistency is ensured.</p> <p>The behavior is as specified in C11 when calling the function: <code>atomic_fetch_xor(object, operand);</code></p>	

5.2.2.3.9. Atomics_Init

Purpose	Initializes an atomic object.	
Synopsis	<pre>void Atomics_Init (volatile Atomic_t * object , Atomic_value_t initialValue);</pre>	
Parameters (in)	initValue	The initial value.
Parameters (in,out)	object	The pointer to an atomic object to initialize.
Description	<p>Initializes an atomic object at runtime. You must initialize an atomic object before use. Undefined behavior is the consequence, if you fail to do so.</p> <p>Atomic objects with static storage duration are initialized with the initial value zero at program load time. You don't have to use <code>Atomics_Init(object, 0)</code> explicitly in this case.</p> <p>See also ATOMICS_OBJECT_INITIALIZER().</p> <p>The behavior is the same as what C11 specifies for the following call: <code>atomic_init(object, initialValue);</code></p>	

5.2.2.3.10. Atomics_Load

Purpose	Loads an atomic object.
Synopsis	<pre>Atomic_value_t Atomics_Load (volatile const Atomic_t * object);</pre>



Parameters (in)	object	The pointer to an atomic object of which to return its value.
Return Value	The value stored in the atomic object.	
Description	<p>Atomically loads the value in the memory location pointed to by 'object' and returns it. Sequential consistency is ensured.</p> <p>The behavior is as defined by C11 for the following call: atomic_load(object);</p>	

5.2.2.3.11. Atomics_Store

Purpose	Stores into an atomic object.	
Synopsis	<pre>void Atomics_Store (volatile Atomic_t * object , Atomic_value_t newValue);</pre>	
Parameters (in)	newValue	The new value to be stored into the atomic object.
Parameters (in,out)	object	The pointer to an atomic object to work with.
Description	<p>The given 'newValue' is stored atomically into the memory location pointed to by 'object'. Sequential consistency is ensured.</p> <p>The behavior is as defined by C11 for the following call: atomic_store(object, newValue);</p>	

5.2.2.3.12. Atomics_TestAndSetFlag

Purpose	Atomically sets a selected bit.	
Synopsis	<pre>boolean Atomics_TestAndSetFlag (volatile Atom- ic_t * object , Atomic_value_t flagSelectionMask);</pre>	
Parameters (in)	flagSelectionMask	The bit mask to select one bit in the atomic object.
Parameters (in,out)	object	The pointer to an atomic object to work on.
Return Value	The state of the selected flag before the operation.	
Description	Atomically sets the bit selected by 'flagSelectionMask' in the memory location pointed to by 'object'. It returns the state of this bit before the operation, i.e., TRUE, if it was already set and FALSE otherwise. Only one bit may be selected by 'flagSelectionMask'.	



Sequential consistency is ensured.

The behavior is similar as specified in C11 when calling the function: `atomic_flag-test_and_set(object);` but more than just one flag are fitted into an `os_atomic_t` object.

5.2.2.3.13. Atomics_ThreadFence

Purpose	Creates a sequentially consistent acquire and release fence.
Synopsis	<pre>void Atomics_ThreadFence (void);</pre>
Description	<p>The behavior is the same as what C11 specifies for the following call: <code>atomic_thread_fence(memory_order_seq_cst);</code></p> <p>It may also serve as a compiler-barrier, which stops the compiler from moving instructions across it either way for optimization purposes.</p>

5.2.2.3.14. TS_AtomicClearBit_16

Purpose	Atomically clears a bit in a 16-bit object.	
Synopsis	<pre>void TS_AtomicClearBit_16 (volatile uint16 * addr , uint32 bitIdx);</pre>	
Parameters (in)	bitIdx	The index of the bit to clear. It must be out of 0 to 15.
Parameters (in,out)	addr	The pointer to the object to manipulate. Natural alignment is assumed.
Description	<p>Atomically clears the specified bit in the object pointed to by <code>addr</code>. The bit is selected via its index <code>bitIdx</code>. Hence, <code>bitIdx</code> can take values from 0 to 15 (inclusive).</p> <p>Sequential consistency is ensured.</p> <p>This function is deprecated. New code should use the API declared in <code>Atomsics.h</code>.</p>	

5.2.2.3.15. TS_AtomicClearBit_32

Purpose	Atomically clears a bit in a 32-bit object.	
Synopsis	<pre>void TS_AtomicClearBit_32 (volatile uint32 * addr , uint32 bitIdx);</pre>	
Parameters (in)	bitIdx	The index of the bit to clear. It must be out of 0 to 31.



Parameters (in,out)	addr	The pointer to the object to manipulate. Natural alignment is assumed.
Description	<p>Atomically clears the specified bit in the object pointed to by addr. The bit is selected via its index bitIdx. Hence, bitIdx can take values from 0 to 31 (inclusive).</p> <p>Sequential consistency is ensured.</p> <p>This function is deprecated. New code should use the API declared in Atomics.h.</p>	

5.2.2.3.16. TS_AtomicClearBit_64

Purpose	Atomically clears a bit in a 64-bit object.	
Synopsis	<pre>void TS_AtomicClearBit_64 (volatile uint64 * addr , uint32 bitIdx);</pre>	
Parameters (in)	bitIdx	The index of the bit to clear. It must be out of 0 to 63.
Parameters (in,out)	addr	The pointer to the object to manipulate. Natural alignment is assumed.
Description	<p>Atomically clears the specified bit in the object pointed to by addr. The bit is selected via its index bitIdx. Hence, bitIdx can take values from 0 to 63 (inclusive).</p> <p>Sequential consistency is ensured.</p> <p>This function is deprecated. New code should use the API declared in Atomics.h.</p>	

5.2.2.3.17. TS_AtomicClearBit_8

Purpose	Atomically clears a bit in a byte.	
Synopsis	<pre>void TS_AtomicClearBit_8 (volatile uint8 * addr , uint32 bitIdx);</pre>	
Parameters (in)	bitIdx	The index of the bit to clear. It must be out of 0 to 7.
Parameters (in,out)	addr	The pointer to the object to manipulate.
Description	<p>Atomically clears the specified bit in the object pointed to by addr. The bit is selected via its index bitIdx. Hence, bitIdx can take values from 0 to 7 (inclusive).</p> <p>Sequential consistency is ensured.</p>	



This function is deprecated. New code should use the API declared in Atomics.h.

5.2.2.3.18. TS_AtomicSetBit_16

Purpose	Atomically sets a bit in a 16-bit object.	
Synopsis	<pre>void TS_AtomicSetBit_16 (volatile uint16 * addr , uint32 bitIdx);</pre>	
Parameters (in)	bitIdx	The index of the bit to set. It must be out of 0 to 15.
Parameters (in,out)	addr	The pointer to the object to manipulate. Natural alignment is assumed.
Description	<p>Atomically sets the specified bit in the object pointed to by addr. The bit is selected via its index bitIdx. Hence, bitIdx can take values from 0 to 15 (inclusive).</p> <p>Sequential consistency is ensured.</p> <p>This function is deprecated. New code should use the API declared in Atomics.h.</p>	

5.2.2.3.19. TS_AtomicSetBit_32

Purpose	Atomically sets a bit in a 32-bit object.	
Synopsis	<pre>void TS_AtomicSetBit_32 (volatile uint32 * addr , uint32 bitIdx);</pre>	
Parameters (in)	bitIdx	The index of the bit to set. It must be out of 0 to 31.
Parameters (in,out)	addr	The pointer to the object to manipulate. Natural alignment is assumed.
Description	<p>Atomically sets the specified bit in the object pointed to by addr. The bit is selected via its index bitIdx. Hence, bitIdx can take values from 0 to 31 (inclusive).</p> <p>Sequential consistency is ensured.</p> <p>This function is deprecated. New code should use the API declared in Atomics.h.</p>	

5.2.2.3.20. TS_AtomicSetBit_64

Purpose	Atomically sets a bit in a 64-bit object.
----------------	---



Synopsis	<pre>void TS_AtomicSetBit_64 (volatile uint64 * addr , uint32 bitIdx);</pre>	
Parameters (in)	bitIdx	The index of the bit to set. It must be out of 0 to 63.
Parameters (in,out)	addr	The pointer to the object to manipulate. Natural alignment is assumed.
Description	<p>Atomically sets the specified bit in the object pointed to by addr. The bit is selected via its index bitIdx. Hence, bitIdx can take values from 0 to 63 (inclusive).</p> <p>Sequential consistency is ensured.</p> <p>This function is deprecated. New code should use the API declared in Atomics.h.</p>	

5.2.2.3.21. TS_AtomicSetBit_8

Purpose	Atomically sets a bit in a byte.	
Synopsis	<pre>void TS_AtomicSetBit_8 (volatile uint8 * addr , uint32 bitIdx);</pre>	
Parameters (in)	bitIdx	The index of the bit to set. It must be out of 0 to 7.
Parameters (in,out)	addr	The pointer to the object to manipulate.
Description	<p>Atomically sets the specified bit in the object pointed to by addr. The bit is selected via its index bitIdx. Hence, bitIdx can take values from 0 to 7 (inclusive).</p> <p>Sequential consistency is ensured.</p> <p>This function is deprecated. New code should use the API declared in Atomics.h.</p>	

5.2.2.3.22. TS_IntDisable

Purpose	Suspends all interrupts.
Synopsis	<pre>TS_IntStatusType TS_IntDisable (void);</pre>
Return Value	The current interrupt status before all hardware interrupts were suspended. This value may be passed to TS_IntRestore() , which is on the same nesting level.
Description	This function-like macro suspends all interrupts and returns the current interrupt status. This status is passed to TS_IntRestore() to resume interrupts later on.



Please note, that each call of [TS_IntDisable\(\)](#) should be accompanied by a call of [TS_IntRestore\(\)](#) to keep the system responsive. This means, these macros usually occur as pairs in your code. Furthermore, the value passed to [TS_IntRestore\(\)](#) must be the return value of the most recent call of [TS_IntDisable\(\)](#). Violations result in unexpected behavior.

This function is deprecated. New code should use the API declared in Atomics.h.

5.2.2.3.23. TS_IntRestore

Purpose	Restores the given interrupt status.	
Synopsis	<code>void TS_IntRestore (TS_IntStatusType intStatus);</code>	
Parameters (in)	intStatus	The interrupt status to be restored. This is the return value of the most recent TS_IntDisable() call.
Description	<p>This function restores the interrupt status passed to it. This status is the return value of the most recent TS_IntDisable() call.</p> <p>This function is deprecated. New code should use the API declared in Atomics.h.</p>	

5.2.3. Integration notes

5.2.3.1. Exclusive areas

Exclusive areas information is not available for this module.

5.2.3.2. Production errors

Production errors are not reported by the `Atomics` module.

5.2.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section [Memory mapping and compiler abstraction](#) in the [Integration notes](#) section for details.

The following table provides the list of sections that may be mapped for this module:



Memory section

CODE

5.2.3.4. Integration requirements

WARNING

Integration requirements list is not exhaustive



The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the Atomics module.

5.3. Base

5.3.1. Configuration parameters

Containers included		
Container name	Multiplicity	Description
BaseDbg	1..1	This container holds configuration information for the debug base types.
PostBuildSelectable	1..1	Label: Post Build Selectable Variants This container holds configuration information for either a <i>post-build-selectable master project</i> or a <i>Post-Build-selectable slave project</i>
CustomOverrides	1..1	Label: Custom overrides This container holds configuration information for custom overrides of default implementations.
BaseTypes	1..1	Label: Base Types This container holds configuration information for the AU-TOSAR base types.
BaseCpuConfig	1..1	Label: CPU Configuration This container holds configuration information for the CPU.



Containers included

GeneralTypes	1..1	<p>Label: General Types</p> <p>This container holds configuration information for the AUTOSAR types part of the <code>Bus</code> _GeneralTypes.</p>
BaseEcuConfig	1..1	<p>Label: Ecu Configuration Export</p> <p>This container holds parameters for the generation of ARXML with ECU parameter configuration. The ARXML files are placed in the folder <code><project output path>/ecuconfig_arxml</code>. For each module in the project a separate file is created.</p>
CommonPublishedInformation	1..1	<p>Label: Common Published Information</p> <p>Common container, aggregated by all modules. It contains published information about vendor and versions.</p>
PublishedInformation	1..1	<p>Label: EB Published Information</p> <p>Additional published parameters not covered by CommonPublishedInformation container.</p>

Parameters included

Parameter name	Multiplicity
CustomStdIncludeFiles	0..100

Parameter Name	CustomStdIncludeFiles
Label	Custom standard header files
Description	<p>List of additional standard header files.</p> <p>The identifiers provided by these header files are accessible by including <code>Platform_Types.h</code> or <code>Std_Types.h</code></p> <p>Make sure the file which is specified here can be found by your compiler in the <i>include file search directories</i>.</p>
Multiplicity	0..100
Type	STRING

5.3.1.1. BaseDbg

Parameters included	
Parameter name	Multiplicity

**Parameters included**

BaseDbgHeaderFile	0..1
-----------------------------------	------

Parameter Name	BaseDbgHeaderFile
Description	<p>Header file for implementation of debug hooks.</p> <p>The identifier provided by this header file is included from <Mod>_Trace.h.</p> <p>Note that the implementation of debug hooks may change the behaviour of the basic software.</p>
Multiplicity	0..1
Type	STRING

5.3.1.2. PostBuildSelectable**Parameters included**

Parameter name	Multiplicity
ProjectType	1..1
VariantNameList	0..n
VariantName	1..1

Parameter Name	ProjectType		
Label	Project Type		
Description	<p>Type of post-build-selectable project.</p> <ul style="list-style-type: none"> ▶ None : This project is not intended to be used as a post-build-selectable project. ▶ Master : This project is a <i>post-build-selectable master project</i>. For a master project the Variant List has to be configured. ▶ Slave : This project is a <i>post-build-selectable slave project</i>. For a slave project the Variant Name has to be configured. 		
Multiplicity	1..1		
Type	ENUMERATION		
Default value	None		
Range	<table border="1"> <tr> <td>None</td> </tr> <tr> <td>Master</td> </tr> </table>	None	Master
None			
Master			



	Slave
--	-------

Parameter Name	VariantNameList
Label	Variant List
Description	List of names of variants of all <i>post-build-selectable slave projects</i> .
Multiplicity	0..n
Type	STRING

Parameter Name	VariantName
Label	Variant Name
Description	Name of the variant of this <i>post-build-selectable slave project</i> .
Multiplicity	1..1
Type	STRING

5.3.1.3. CustomOverrides

Parameters included	
Parameter name	Multiplicity
<u>CustomOverride_Mem-Cpy</u>	1..1
<u>CustomOverride_MemSet</u>	1..1
<u>CustomOverride_Mem-Cmp</u>	1..1
<u>CustomOverride_Mem-BZero</u>	1..1

Parameter Name	CustomOverride_MemCpy
Label	Override memory copy function (macro TS_MemCpy)
Description	Enable or disable the usage of a custom memory copy function. To override the default memory copy function ► define and implement a copy function with signature <code>void custom_memcpy(void* const destination, void const* const source, const usize length)</code> where



	<ul style="list-style-type: none"> ▶ <code>destination</code> is the pointer to where the memory is copied to ▶ <code>source</code> is the pointer to where the memory is copied from ▶ <code>length</code> is number of bytes to be copied from <code>source</code> to <code>destination</code> The memory copy function is expected to copy <code>length</code> bytes from <code>source</code> to <code>destination</code>. ▶ define the macro <code>TS_MemCpy(dst,src,len)</code> that just maps to the custom memory copy function in a custom standard header file ▶ add the custom standard header file to Base configuration parameter <code>CustomStdIncludeFiles</code>
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	CustomOverride_MemSet
Label	Override memory set function (macro <code>TS_MemSet</code>)
Description	<p>Enable or disable the usage of a custom memory set function. To override the default memory set function</p> <ul style="list-style-type: none"> ▶ define and implement a set function with signature <code>void custom_memset(void* const destination, const value, const usize length) where</code> <ul style="list-style-type: none"> ▶ <code>destination</code> is the pointer to where the memory is set ▶ <code>value</code> is the value to be set ▶ <code>length</code> is the number of bytes to be set to specified value The memory set function is expected to set <code>length</code> bytes to <code>value</code> starting from <code>destination</code>. ▶ define the macro <code>TS_MemSet(dst,val,len)</code> that just maps to the custom memory set function in a custom standard header file ▶ add the custom standard header file to Base configuration parameter <code>CustomStdIncludeFiles</code>
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	CustomOverride_MemCmp
Label	Override memory compare function (macro <code>TS_MemCmp</code>)



Description	<p>Enable or disable the usage of a custom memory compare function. To override the default memory compare function</p> <ul style="list-style-type: none"> ▶ define and implement a compare function with signature <code>void custom_memcmp(void const * const str1, void const * const str2, const usize length) where</code> <ul style="list-style-type: none"> ▶ <code>str1</code> is the pointer to one block of memory ▶ <code>str2</code> is the pointer to another block of memory ▶ <code>length</code> is the number of bytes to be compared <p>The memory compare function is expected to compare <code>length</code> bytes from <code>str1</code> to <code>str2</code> and return 0 if they're equal, smaller than 0 if <code>str1</code> is smaller than <code>str2</code>, greater than 0 if <code>str1</code> is greater than <code>str2</code>.</p> ▶ define the macro <code>TS_MemCmp(str1, str2, len)</code> that just maps to the custom memory compare function in a custom standard header file ▶ add the custom standard header file to Base configuration parameter <code>CustomStdIncludeFiles</code>
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	CustomOverride_MemBZero
Label	Override memory zeroing function (macro <code>TS_MemBZero</code>)
Description	<p>Enable or disable the usage of a custom memory zeroing function. To override the default memory zeroing function</p> <ul style="list-style-type: none"> ▶ define and implement a zeroing function with signature <code>void custom_membzero(void* const destination, const usize length) where</code> <ul style="list-style-type: none"> ▶ <code>destination</code> is the pointer to where the memory is zeroed ▶ <code>length</code> is the number of bytes to be zeroed <p>The memory zeroing function is expected to zero <code>length</code> bytes from <code>destination</code>.</p> ▶ define the macro <code>TS_MemBZero(dst, len)</code> that just maps to the custom memory zeroing function in a custom standard header file ▶ add the custom standard header file to Base configuration parameter <code>CustomStdIncludeFiles</code>
Multiplicity	1..1
Type	BOOLEAN



Default value	false
----------------------	-------

5.3.1.4. BaseType

Parameters included	
Parameter name	Multiplicity
BaseType_boolean	1..1
BaseType_uint8	1..1
BaseType_sint8	1..1
BaseType_uint16	1..1
BaseType_sint16	1..1
BaseType_uint32	1..1
BaseType_sint32	1..1
BaseType_uint8_least	1..1
BaseType_sint8_least	1..1
BaseType_uint16_least	1..1
BaseType_sint16_least	1..1
BaseType_uint32_least	1..1
BaseType_sint32_least	1..1
BaseType_uint64	1..1
BaseType_sint64	1..1
BaseType_float32	1..1
BaseType_float64	1..1
DerivedType_TS_Max-AlignedType	1..1
BaseType_Size_char	1..1
BaseType_Alignment_char	1..1
BaseType_AtomicAccess_char	1..1
BaseType_Size_short	1..1
BaseType_Alignment_short	1..1

**Parameters included**

<u>BaseType_AtomicAccess_short</u>	1..1
<u>BaseType_Size_int</u>	1..1
<u>BaseType_Alignment_int</u>	1..1
<u>BaseType_AtomicAccess_int</u>	1..1
<u>BaseType_Size_long</u>	1..1
<u>BaseType_Alignment_long</u>	1..1
<u>BaseType_AtomicAccess_long</u>	1..1
<u>BaseType_Size_longlong</u>	1..1
<u>BaseType_Alignment_longlong</u>	1..1
<u>BaseType_AtomicAccess_longlong</u>	1..1
<u>BaseType_Size_float</u>	1..1
<u>BaseType_Alignment_float</u>	1..1
<u>BaseType_AtomicAccess_float</u>	1..1
<u>BaseType_Size_double</u>	1..1
<u>BaseType_Alignment_double</u>	1..1
<u>BaseType_AtomicAccess_double</u>	1..1
<u>BaseType_Alignment_nearpointer</u>	1..1
<u>BaseType_Size_nearpointer</u>	1..1
<u>BaseType_Size_farpointer</u>	1..1

**Parameters included**

BaseType_Alignment_farpointer	1..1
ComplexType_Alignment_struct	1..1
ComplexType_Alignment_array	1..1
BaseTypes64bit	1..1

Parameter Name**BaseType_boolean****Label**

Mapping for AUTOSAR type 'boolean'

Description

Define the mapping from the AUTOSAR base type 'boolean' to the corresponding C data type.

Multiplicity

1..1

Type

STRING

Parameter Name**BaseType_uint8****Label**

Mapping for AUTOSAR type 'uint8'

Description

Define the mapping from the AUTOSAR base type 'uint8' to the corresponding C data type.

Multiplicity

1..1

Type

STRING

Parameter Name**BaseType_sint8****Label**

Mapping for AUTOSAR type 'sint8'

Description

Define the mapping from the AUTOSAR base type 'sint8' to the corresponding C data type.

Multiplicity

1..1

Type

STRING

Parameter Name**BaseType_uint16****Label**

Mapping for AUTOSAR type 'uint16'

Description

Define the mapping from the AUTOSAR base type 'uint16' to the corresponding C data type.

Multiplicity

1..1

Type

STRING



Parameter Name	BaseType_sint16
Label	Mapping for AUTOSAR type 'sint16'
Description	Define the mapping from the AUTOSAR base type 'sint16' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_uint32
Label	Mapping for AUTOSAR type 'uint32'
Description	Define the mapping from the AUTOSAR base type 'uint32' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_sint32
Label	Mapping for AUTOSAR type 'sint32'
Description	Define the mapping from the AUTOSAR base type 'sint32' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_uint8_least
Label	Mapping for AUTOSAR type 'uint8_least'
Description	Define the mapping from the AUTOSAR base type 'uint8_least' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_sint8_least
Label	Mapping for AUTOSAR type 'sint8_least'
Description	Define the mapping from the AUTOSAR base type 'sint8_least' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_uint16_least
-----------------------	------------------------------



Label	Mapping for AUTOSAR type 'uint16_least'
Description	Define the mapping from the AUTOSAR base type 'uint16_least' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_sint16_least
Label	Mapping for AUTOSAR type 'sint16_least'
Description	Define the mapping from the AUTOSAR base type 'sint16_least' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_uint32_least
Label	Mapping for AUTOSAR type 'uint32_least'
Description	Define the mapping from the AUTOSAR base type 'uint32_least' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_sint32_least
Label	Mapping for AUTOSAR type 'sint32_least'
Description	Define the mapping from the AUTOSAR base type 'sint32_least' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_uint64
Label	Mapping for AUTOSAR type 'uint64'
Description	Define the mapping from the AUTOSAR base type 'uint64' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_sint64
Label	Mapping for AUTOSAR type 'sint64'



Description	Define the mapping from the AUTOSAR base type 'sint64' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_float32
Label	Mapping for AUTOSAR type 'float32'
Description	Define the mapping from the AUTOSAR base type 'float32' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_float64
Label	Mapping for AUTOSAR type 'float64'
Description	Define the mapping from the AUTOSAR base type 'float64' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	DerivedType_TS_MaxAlignedType
Label	Mapping for derived type 'TS_MaxAlignedType'
Description	Define the mapping from the derived type 'TS_MaxAlignedType' to the corresponding C data type.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseType_Size_char
Label	Size of C type 'char'
Description	Define the size of the C data type 'signed/unsigned char' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_Alignment_char
Label	Alignment of C type 'char'
Description	Define the alignment of the C data type 'signed/unsigned char' as number of bytes.



Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_AtomicAccess_char
Label	Atomic access to C type 'char'
Description	<p>Switch to define if atomic access is possible to the C data type 'char'.</p> <ul style="list-style-type: none"> ▶ true: atomic access is possible. ▶ false: atomic access is not possible.
Multiplicity	1..1
Type	BOOLEAN

Parameter Name	BaseType_Size_short
Label	Size of C type 'short'
Description	Define the size of the C data type 'signed/unsigned short' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_Alignment_short
Label	Alignment of C type 'short'
Description	Define the alignment of the C data type 'signed/unsigned short' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_AtomicAccess_short
Label	Atomic access to C type 'short'
Description	<p>Switch to define if atomic access is possible to the C data type 'short'.</p> <ul style="list-style-type: none"> ▶ true: atomic access is possible. ▶ false: atomic access is not possible.
Multiplicity	1..1
Type	BOOLEAN

Parameter Name	BaseType_Size_int
Label	Size of C type 'int'



Description	Define the size of the C data type 'signed/unsigned int' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_Alignment_int
Label	Alignment of C type 'int'
Description	Define the alignment of the C data type 'signed/unsigned int' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_AtomicAccess_int
Label	Atomic access to C type 'int'
Description	Switch to define if atomic access is possible to the C data type 'int'. <ul style="list-style-type: none"> ▶ true: atomic access is possible. ▶ false: atomic access is not possible.
Multiplicity	1..1
Type	BOOLEAN

Parameter Name	BaseType_Size_long
Label	Size of C type 'long'
Description	Define the size of the C data type 'signed/unsigned long' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_Alignment_long
Label	Alignment of C type 'long'
Description	Define the alignment of the C data type 'signed/unsigned long' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_AtomicAccess_long
Label	Atomic access to C type 'long'
Description	Switch to define if atomic access is possible to the C data type 'long'. <ul style="list-style-type: none"> ▶ true: atomic access is possible.



	► <code>false</code> : atomic access is not possible.
Multiplicity	1..1
Type	BOOLEAN

Parameter Name	BaseType_Size_llonglong
Label	Size of C type 'long long'
Description	Define the size of the C data type 'signed/unsigned long long' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_Alignment_llonglong
Label	Alignment of C type 'long long'
Description	Define the alignment of the C data type 'signed/unsigned long long' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_AtomicAccess_llonglong
Label	Atomic access to C type 'long long'
Description	Switch to define if atomic access is possible to the C data type 'long long'. <ul style="list-style-type: none"> ► <code>true</code>: atomic access is possible. ► <code>false</code>: atomic access is not possible.
Multiplicity	1..1
Type	BOOLEAN

Parameter Name	BaseType_Size_float
Label	Size of C type 'float'
Description	Define the size of the C data type 'float' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_Alignment_float
Label	Alignment of C type 'float'
Description	Define the alignment of the C data type 'float' as number of bytes.
Multiplicity	1..1



Type	INTEGER
------	---------

Parameter Name	BaseType_AtomicAccess_float
Label	Atomic access to C type 'float'
Description	Switch to define if atomic access is possible to the C data type 'float'. <ul style="list-style-type: none"> ▶ true: atomic access is possible. ▶ false: atomic access is not possible.
Multiplicity	1..1
Type	BOOLEAN

Parameter Name	BaseType_Size_double
Label	Size of C type 'double'
Description	Define the size of the C data type 'double' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_Alignment_double
Label	Alignment of C type 'double'
Description	Define the alignment of the C data type 'double' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_AtomicAccess_double
Label	Atomic access to C type 'double'
Description	Switch to define if atomic access is possible to the C data type 'double'. <ul style="list-style-type: none"> ▶ true: atomic access is possible. ▶ false: atomic access is not possible.
Multiplicity	1..1
Type	BOOLEAN

Parameter Name	BaseType_Alignment_nearpointer
Label	Alignment of C type 'nearpointer'
Description	Define the alignment of the C data type 'nearpointer' as number of bytes.
Multiplicity	1..1
Type	INTEGER



Parameter Name	BaseType_Size_nearpointer
Label	Size of C type 'nearpointer'
Description	Define the size of the C data type 'nearpointer' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_Size_farpointer
Label	Size of C type 'farpointer'
Description	Define the size of the C data type 'farpointer' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseType_Alignment_farpointer
Label	Alignment of C type 'farpointer'
Description	Define the alignment of the C data type 'farpointer' as number of bytes.
Multiplicity	1..1
Type	INTEGER

Parameter Name	ComplexType_Alignment_struct
Label	Alignment of 'struct' complex types
Description	Define the alignment of complex 'struct' types as number of bytes.
Multiplicity	1..1
Type	STRING

Parameter Name	ComplexType_Alignment_array
Label	Alignment of 'array' complex types
Description	Define the alignment of complex 'array' types as number of bytes.
Multiplicity	1..1
Type	STRING

Parameter Name	BaseTypes64bit
Label	Enable generation of base types for 64bit
Description	Switch to enable/disable the generation of the AUTOSAR 64bit base types (sint64, uint64). Disabled base types are not provided by typedefs. ▶ true: generation of 64bit base types enabled.



	▶ false: generation of 64bit base types disabled.
Multiplicity	1..1
Type	BOOLEAN

5.3.1.5. BaseCpuConfig

Parameters included	
Parameter name	Multiplicity
BaseCpuConfig_type	1..1
BaseCpuConfig_byteorder	1..1
BaseCpuConfig_bitorder	1..1

Parameter Name	BaseCpuConfig_type
Label	Type of CPU
Description	Define the type resp. data unit size of the CPU; allowed values are 64, 32, 16, 8.
Multiplicity	1..1
Type	INTEGER

Parameter Name	BaseCpuConfig_byteorder
Label	Byte order
Description	Define the byte order of the CPU, big endian (BE) or little endian (LE).
Multiplicity	1..1
Type	ENUMERATION
Range	BE LE

Parameter Name	BaseCpuConfig_bitorder
Label	Bit order
Description	Define the bit order of the CPU, big endian (BE) or little endian (LE).
Multiplicity	1..1
Type	ENUMERATION
Range	BE



LE

5.3.1.6. GeneralTypes

Parameters included	
Parameter name	Multiplicity
GeneralTypes_Can_ControllerStateType	1..1

Parameter Name	GeneralTypes_Can_ControllerStateType
Label	Enable generation of Can_ControllerStateType
Description	<p>Switch to enable/disable the generation of the AUTOSAR type <code>Can_ControllerStateType</code>. Disabled types are not provided by typedefs.</p> <ul style="list-style-type: none"> ▶ <code>true</code>: generation of type <code>Can_ControllerStateType</code> enabled. ▶ <code>false</code>: generation of type <code>Can_ControllerStateType</code> disabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	true

5.3.1.7. BaseEcuConfig

Parameters included	
Parameter name	Multiplicity
BaseEcuConfigAlwaysExport	1..1
BaseEcuConfigAsrVersion	1..1

Parameter Name	BaseEcuConfigAlwaysExport
Label	Always Export
Description	<p>Always create ARXML files during code generation.</p> <ul style="list-style-type: none"> ▶ Enabled: ARXML files are created in all generator modes ▶ Disabled: ARXML files are only created in the generator mode <code>export_asr-config</code> (Default)



Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	BaseEcuConfigAsrVersion
Label	Autosar version
Description	Choose the AUTOSAR version for the file format for the ECU configuration.
Multiplicity	1..1
Type	ENUMERATION
Default value	r4_0_3
Range	r2_0 r2_1 r3_0 r3_1 r3_2 r4_0_2 r4_0_3 r4_1_3 r4_2_1 r4_2_2 r4_3_0

5.3.1.8. CommonPublishedInformation

Parameters included	
Parameter name	Multiplicity
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1

**Parameters included**

VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwMajorVersion
-----------------------	-----------------------



Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwMinorVersion
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwPatchVersion
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	26
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ModuleId
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0



Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

Parameter Name	VendorId
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AU-TOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	Release
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.3.1.9. PublishedInformation

Parameters included	
Parameter name	Multiplicity
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport
Label	PbcfgM support
Description	Specifies whether or not the Base can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false



Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

5.3.2. Application programming interface (API)

5.3.2.1. Type definitions

5.3.2.1.1. BufReq_ReturnType

Purpose	type for storage of the result of a buffer request	
Type	enum	
Constants	BUFREQ_OK	Buffer request accomplished successful.
	BUFREQ_E_NOT_OK	Buffer request not successful, buffer cannot be accessed.
	BUFREQ_E_BUSY	Temporarily no buffer available.
	BUFREQ_E_OVFL	no buffer of the required length can be provided

5.3.2.1.2. BusTrcvErrorType

Purpose	Bus status type definition.
Type	uint8
Description	Variables of this type are used to return the bus status evaluated by a transceiver.

5.3.2.1.3. CanTrcv_PNActivationType

Purpose	Datatype used for describing whether PN wakeup functionality in CanTrcv is enabled or disabled.	
Type	enum	
Constants	PN_ENABLED	PN wakeup functionality in CanTrcv is enabled.



	PN_DISABLED	PN wakeup functionality in CanTrcv is disabled.
--	-------------	---

5.3.2.1.4. CanTrcv_TrcvFlagStateType

Purpose	Provides the state of a flag in the transceiver hardware.	
Type	enum	
Constants	CANTRCV_FLAG_SET	The flag is set in the transceiver hardware.
	CANTRCV_FLAG_CLEARED	The flag is cleared in the transceiver hardware.

5.3.2.1.5. CanTrcv_TrcvModeType

Purpose	Operating modes of the CAN Transceiver Driver.	
Type	enum	
Constants	CANTRCV_TRCVMODE_NORMAL	Transceiver mode NORMAL
	CANTRCV_TRCVMODE_SLEEP	Transceiver mode SLEEP
	CANTRCV_TRCVMODE_STANDBY	Transceiver mode STANDBY

5.3.2.1.6. CanTrcv_TrcvWakeupModeType

Purpose	This type shall be used to control the CAN transceiver concerning wake up events and wake up notifications.	
Type	enum	
Constants	CANTRCV_WUMODE_ENABLE	The notification for wakeup events is enabled on the addressed network.
	CANTRCV_WUMODE_DISABLE	The notification for wakeup events is disabled on the addressed network.
	CANTRCV_WUMODE_CLEAR	A stored wakeup event is cleared on the addressed network.

5.3.2.1.7. CanTrcv_TrcvWakeupReasonType

Purpose	This type denotes the wake up reason detected by the CAN transceiver in detail.
---------	---



Type	enum	
Constants	CANTRCV_WU_ERROR	Due to an error wake up reason was not detected. This value may only be reported when error was reported to DEM before.
	CANTRCV_WU_BY_BUS	The transceiver has detected, that the network has caused the wake up of the ECU.
	CANTRCV_WU_BY_PIN	The transceiver has detected a wake-up event at one of the transceiver's pins (not at the CAN bus).
	CANTRCV_WU INTERNALLY	The transceiver has detected, that the network has woken up by the ECU via a request to NORMAL mode.
	CANTRCV_WU_NOT_SUPPORTED	The transceiver does not support any information for the wake up reason.
	CANTRCV_WU_POWER_ON	The transceiver has detected, that the "wake up" is due to an ECU reset after power on.
	CANTRCV_WU_RESET	The transceiver has detected, that the "wake up" is due to an ECU reset.
	CANTRCV_WU_BY_SYSERR	The transceiver has detected, that the "wake up" is due to an HW related device failure.

5.3.2.1.8. Can_ControllerStateType

Purpose	States that are used by the several ControllerMode functions.	
Type	enum	
Constants	CAN_CS_UNINIT	CAN controller state UNINIT
	CAN_CS_STARTED	CAN controller state STARTED
	CAN_CS_STOPPED	CAN controller state STOPPED
	CAN_CS_SLEEP	CAN controller state SLEEP

5.3.2.1.9. Can_ErrorStateType

Purpose	Error states of a CAN controller.
Type	enum



Constants	CAN_ERRORSTATE_ACTIVE	The CAN controller takes fully part in communication.
	CAN_ERRORSTATE_PASSIVE	The CAN controller takes part in communication, but does not send active error frames.
	CAN_ERRORSTATE_BUSOFF	The CAN controller does not take part in communication.

5.3.2.1.10. Can_HwHandleType

Purpose	Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects use extended range (generated).
Type	uint16
Description	<p>Range:</p> <ul style="list-style-type: none"> ➤ uint16 (if configuration parameter 'CanIf/CanIfPublicCfg/CanIfPublicHandleTypeEnum = UINT16') ➤ uint8 (otherwise)

5.3.2.1.11. Can_HwType

Purpose	This type defines a data structure which clearly provides an Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CanId.	
Type	struct	
Members	Can_IdType CanId	Standard/Extended CAN ID of CAN L-PDU
	Can_HwHandleType Hoh	ID of the corresponding Hardware Object Range
	uint8 ControllerId	ControllerId provided by CanIf clearly identify the corresponding controller

5.3.2.1.12. Can_IdType

Purpose	Represents the Identifier of an L-PDU. For extended IDs the most significant bit is set (generated).
----------------	--



Type	uint16
Description	<p>Range:</p> <ul style="list-style-type: none"> ➤ uint16 (if configuration parameter 'CanIf/CanIfPublicCfg/CanIfPublicCanIdType-Enum = UINT16') ➤ uint32 (otherwise)

5.3.2.1.13. Can_PduType

Purpose	This type is used to provide ID, DLC and SDU from CAN interface to CAN driver.	
Type	struct	
Members	uint8 * sdu	Pointer to the data
	Can_IdType id	CAN ID
	PduIdType swPduHandle	PDU ID for Tx confirmation
	uint8 length	Data length code (DLC)

5.3.2.1.14. Can_ReturnType

Purpose	Return values of CAN driver API.	
Type	enum	
Constants	CAN_OK	Success
	CAN_NOT_OK	Error occurred or wakeup event occurred during sleep transition
	CAN_BUSY	Transmit request could not be processed because no transmit object was available.

5.3.2.1.15. Can_StateTransitionType

Purpose	State transitions that are used by the function Can_SetControllerMode.	
Type	enum	
Constants	CAN_T_START	CAN controller transition value to request state STARTED.
	CAN_T_STOP	CAN controller transition value to request state STOPPED.



	CAN_T_SLEEP	CAN controller transition value to request state SLEEP.
	CAN_T_WAKEUP	CAN controller transition value to request state STOPPED from state SLEEP.

5.3.2.1.16. ConstVoidPtr

Purpose	type definition for pointer to const void
Type	const void *

5.3.2.1.17. EthIf_MeasurementIdxType

Purpose	Definition of EthIf_MeasurementIdxType - index to select specific measurement data.
Type	uint8

5.3.2.1.18. EthIf_SignalQualityResultType

Purpose	Signal quality structure type.
Type	struct
Members	uint32 HighestSignalQuality
	uint32 LowestSignalQuality
	uint32 ActualSignalQuality

5.3.2.1.19. EthIf_SwitchPortGroupIdxType

Purpose	Data Type that represents the Ethernet interface switch port group index. The index is zero based and unique for every configured switch port group.
Type	uint8

5.3.2.1.20. EthSwt_MacLearningType

Purpose	Definition of EthSwt_MacLearningType.
Type	uint8



5.3.2.1.21. EthSwt_MgmtInfoType

Purpose	Definition of EthSwt_MgmtInfoType Type for holding the management information received/transmitted on Switches (ports).	
Type	struct	
Members	uint8 SwitchIdx	
	uint8 SwitchPortIdx	

5.3.2.1.22. EthSwt_PortMirrorCfgType

Purpose	Definition of EthSwt_PortMirrorCfgType Type for holding the information of port mirror configuration.	
Type	struct	
Members	uint8 SetSelection	specifies the type selection 0x00 == free configuration, 0x01 - 0x0FF == set number
	uint8 TrafficDirection	specifies whether direction is Ingress or Egress 0 : Ingress 1 : Egress
	uint8 srcMacAddrFilter	Specifies the source MAC address that should be mirrored.
	uint8 dstMacAddrFilter	Specifies the destination MAC address that should be mirrored.
	uint16 VlanIdFilter	VLAN address [0..65535] that should be mirrored.
	uint8 MirroringPacketDivider	Divider if only a subset of received frames should be mirrored.
	uint8 MirroringMode	specifies the mode how the mirrored traffic should be tagged . 0x00 == No VLAN retagging; 0x01 == VLAN retagging; 0x03 == VLAN Double tagging
	uint16 MirroringTimeout	specifies a time constant in seconds after the mirroring configuration should be resumed.

5.3.2.1.23. EthSwt_PortMirrorStateType

Purpose	Definition of EthSwt_PortMirrorStateType.
----------------	---



Type	uint8
------	-------

5.3.2.1.24. EthSwt_StateType

Purpose	Definition of EthSwt_StateType.
Type	uint8

5.3.2.1.25. EthTrcv_BaudRateType

Purpose	Definition of EthTrcv_BaudRateType.
Type	uint8

5.3.2.1.26. EthTrcv_CableDiagResultType

Purpose	Definition of EthTrcv_CableDiagResultType.
Type	uint8

5.3.2.1.27. EthTrcv_DuplexModeType

Purpose	Definition of EthTrcv_DuplexModeType.
Type	uint8

5.3.2.1.28. EthTrcv_LinkStateType

Purpose	Definition of EthTrcv_LinkStateType.
Type	uint8

5.3.2.1.29. EthTrcv_ModeType

Purpose	Definition of EthTrcv_ModeType.
Type	uint8



5.3.2.1.30. EthTrcv_PhysLoopbackModeType

Purpose	Definition of EthTrcv_PhysLoopbackModeType.
Type	uint8

5.3.2.1.31. EthTrcv_PhysTestModeType

Purpose	Definition of EthTrcv_PhysTestModeType.
Type	uint8

5.3.2.1.32. EthTrcv_PhysTxModeType

Purpose	Definition of EthTrcv_PhysTxModeType.
Type	uint8

5.3.2.1.33. EthTrcv_StateType

Purpose	Definition of EthTrcv_StateType.
Type	uint8

5.3.2.1.34. EthTrcv_WakeupModeType

Purpose	Definition of EthTrcv_DuplexModeType.
Type	uint8

5.3.2.1.35. EthTrcv_WakeupReasonType

Purpose	Definition of EthTrcv_WakeupReasonType.
Type	uint8

5.3.2.1.36. Eth_BufListType

Purpose	Buffer list type (used for API Eth_TransmitBufList()).
----------------	--



Type	struct
Members	Eth_DataType * BufPtr
	uint16 LenByte

5.3.2.1.37. Eth_CounterType

Purpose	Definition of Eth_CounterType Statistic counter for diagnostics.
Type	struct
Members	uint32 DropPktBufOverrun uint32 DropPktCrc uint32 UndersizePkt uint32 OversizePkt uint32 AlgnmtErr uint32 SqeTestErr uint32 DiscInbdPkt uint32 ErrInbdPkt uint32 DiscOtbdPkt uint32 ErrOtbdPkt uint32 SnglCollPkt uint32 MultCollPkt uint32 DfrdPkt uint32 LatCollPkt uint32 HwDepCtr0 uint32 HwDepCtr1 uint32 HwDepCtr2 uint32 HwDepCtr3
Description	1.) dropped packets due to buffer overrun 2.) dropped packets due to CRC errors 3.) number of undersize packets which were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757) 4.) number of oversize packets which are longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757) 5.) number of alignment errors, i.e. packets which are received and are not an integral number of octets in length and do not pass the CRC. 6.) SQE test error according to IETF RFC1643 dot3StatsSQETestErrors 7.) The number of inbound packets which



were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. (see IETF RFC 2233 ifInDiscards) 8.) total number of erroneous inbound packets 9.) The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. (see IETF RFC 2233 ifOutDiscards) 10.) total number of erroneous outbound packets 11.) Single collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision. (see IETF RFC1643 dot3StatsSingleCollisionFrames) 12.) Multiple collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision. (see IETF RFC1643 dot3StatsMultipleCollisionFrames) 13.) Number of deferred transmission: A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. (see IETF RFC1643 dot3StatsDeferredTransmissions) 14.) Number of late collisions: The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet. (see IETF RFC1643 dot3StatsLateCollisions) 15-18 .) hardware dependent counter values

5.3.2.1.38. Eth(DataType)

Purpose	Definition of Eth(DataType).
Type	uint8

5.3.2.1.39. Eth_EtherStatsType

Purpose	Definition of Eth_EtherStatsType Statistic counter for diagnostics.	
Type	struct	
Members	uint32 etherStatsDropEvents	
	uint32 etherStatsOctetsLow	
	uint32 etherStatsOctetsHigh	
	uint32 etherStatsPkts	
	uint32 etherStatsBroadcastPkts	
	uint32 etherStatsMulticastPkts	
	uint32 etherStatsCrcAlignErrors	
	uint32 etherStatsUndersizePkts	
	uint32 etherStatsOversizePkts	



	uint32 etherStatsFragments
	uint32 etherStatsJabbers
	uint32 etherStatsCollisions
	uint32 etherStatsRxPkts64Octets
	uint32 etherStatsRxPkts65to127Octets
	uint32 etherStatsRxPkts128to255Octets
	uint32 etherStatsRxPkts256to511Octets
	uint32 etherStatsRxPkts512to1023Octets
	uint32 etherStatsRxPkts1024to1518Octets
	uint32 etherStatsTxPkts64Octets
	uint32 etherStatsTxPkts65to127Octets
	uint32 etherStatsTxPkts128to255Octets
	uint32 etherStatsTxPkts256to511Octets
	uint32 etherStatsTxPkts512to1023Octets
	uint32 etherStatsTxPkts1024to1518Octets

5.3.2.1.40. Eth_FilterActionType

Purpose	Definition of Eth_FilterActionType.
Type	uint8

5.3.2.1.41. Eth_FrameType

Purpose	Definition of Eth_FrameType.
Type	uint16



5.3.2.1.42. Eth_MacVlanType

Purpose	Definition of Eth_MacVlanType .	
Type	struct	
Members	uint8 MacAddr	
	uint16 VlanId	
	uint32 SwitchPort	

5.3.2.1.43. Eth_ModeType

Purpose	Definition of Eth_ModeType.	
Type	uint8	

5.3.2.1.44. Eth_RateRatioType

Purpose	Definition of Eth_RateRatioType .	
Type	struct	
Members	Eth_TimeIntDiffType In- gressTimeStampDelta	
	Eth_TimeIntDiffType Origin- TimeStampDelta	

5.3.2.1.45. Eth_RetransmitInfoType

Purpose	Retransmit info type (used for API Eth_Retransmit()).	
Type	struct	
Members	uint8 Priority	
	boolean TxConfirmation	
	uint8 * PhysAddrPtr	

5.3.2.1.46. Eth_ReturnType

Purpose	Definition Eth_ReturnType.	
Type	uint8	



5.3.2.1.47. Eth_RxStatsType

Purpose	Definition of Eth_RxStatsType Reception statistic counter for diagnostics.
Type	struct
Members	uint32 RxStatsDropEvents
	uint32 RxStatsOctets
	uint32 RxStatsPkts
	uint32 RxStatsBroadcastPkts
	uint32 RxStatsMulticastPkts
	uint32 RxdStatsCrcAlignErrors
	uint32 RxStatsUndersizePkts
	uint32 RxStatsOversizePkts
	uint32 RxStatsFragments
	uint32 RxStatsJabbers
	uint32 RxStatsCollisions
	uint32 RxStatsPkts64Octets
	uint32 RxStatsPkts65to127Octets
	uint32 RxStatsPkts128to255Octets
	uint32 RxStatsPkts256to511Octets
	uint32 RxStatsPkts512to1023Octets
	uint32 RxStatsPkts1024to1518Octets

5.3.2.1.48. Eth_RxStatusType

Purpose	Definition of Eth_FilterActionType.
Type	uint8

5.3.2.1.49. Eth_StateType

Purpose	Definition of Eth_StateType.
----------------	------------------------------



Type	uint8
------	-------

5.3.2.1.50. Eth_TimeIntDiffType

Purpose	Definition of Eth_TimeIntDiffType .	
Type	struct	
Members	Eth_TimeStampType diff	
	boolean sign	

5.3.2.1.51. Eth_TimeStampQualType

Purpose	Definition of Eth_TimeStampQualType.	
Type	uint8	

5.3.2.1.52. Eth_TimeStampType

Purpose	Definition of Eth_TimeStampType .	
Type	struct	
Members	uint32 nanoseconds	
	uint32 seconds	
	uint16 secondsHi	

5.3.2.1.53. Eth_TxErrorCounterValuesType

Purpose	Definition of Eth_TxStatsType Transmission error statistic counter for diagnostics.	
Type	struct	
Members	uint32 TxDroppedNoErrorPkts	
	uint32 TxDroppedErrorPkts	
	uint32 TxDeferredTrans	
	uint32 TxSingleCollision	
	uint32 TxMultipleCollision	
	uint32 TxLateCollision	



	uint32 TxExcessiveCollision	
--	-----------------------------	--

5.3.2.1.54. Eth_TxStatsType

Purpose	Definition of Eth_TxStatsType Transmission statistic counter for diagnostics.	
Type	struct	
Members	uint32 TxNumberOfOctets	
	uint32 TxNUcastPkts	
	uint32 TxUniCastPkts	

5.3.2.1.55. FrTrcv_TrcvModeType

Purpose	This enumerator type represents the FlexRay transceiver driver modes.	
Type	enum	
Constants	FRTRCV_TRCVMODE_NORMAL	FrTrcv mode "Normal"
	FRTRCV_TRCVMODE_STANDBY	FrTrcv mode "Standby"
	FRTRCV_TRCVMODE_SLEEP	FrTrcv mode "Sleep"
	FRTRCV_TRCVMODE_RECEIVEONLY	FrTrcv mode "Receive Only"

5.3.2.1.56. FrTrcv_TrcvWUReasonType

Purpose	This enumerator type represents the FlexRay transceiver wakeup-reasons.	
Type	enum	
Constants	FRTRCV_WU_NOT_SUPPORTED	FrTrcv wakeup not supported
	FRTRCV_WU_BY_BUS	FrTrcv wakeup by bus
	FRTRCV_WU_BY_PIN	FrTrcv wakeup by pin
	FRTRCV_WU_INTERNALY	FrTrcv internally wakeup
	FRTRCV_WU_RESET	FrTrcv reset wakeup
	FRTRCV_WU_POWER_ON	FrTrcv power on wakeup

5.3.2.1.57. Fr_ChannelType

Purpose	This enumerator type represents the FlexRay channels.
----------------	---



Type	enum	
Constants	FR_CHANNEL_A	FlexRay channel A
	FR_CHANNEL_B	FlexRay channel B
	FR_CHANNEL_AB	FlexRay channel A and B

5.3.2.1.58. Fr_ErrorModeType

Purpose	This enumerator type represents the FlexRay controller errormode.	
Type	enum	
Constants	FR_ERRORMODE_ACTIVE	active errormode
	FR_ERRORMODE_PASSIVE	passive errormode
	FR_ERRORMODE_COMM_HALT	communication halted errormode

5.3.2.1.59. Fr_MTSStatusType

Purpose	This enumerator type represents the FlexRay MTS status.	
Type	enum	
Constants	FR_MTS_RCV	MTS received
	FR_MTS_RCV_SYNERR	MTS received and syntax error detected
	FR_MTS_RCV_BVIO	MTS received and boundary violation detected
	FR_MTS_RCV_SYNERR_BVIO	MTS received and syntax error and boundary violation detected
	FR_MTS_NOT_RCV	no MTS received
	FR_MTS_NOT_RCV_SYNERR	no MTS received and syntax error detected
	FR_MTS_NOT_RCV_BVIO	no MTS received and boundary violation detected
	FR_MTS_NOT_RCV_SYNERR_BVIO	no MTS received and syntax error and boundary violation detected

5.3.2.1.60. Fr_OffsetCorrectionType

Purpose	This enumerator type contains the possible offset correction operations.
---------	--



Type	enum	
Constants	FR_OFFSET_INC	apply an incrementing offset correction
	FR_OFFSET_DEC	apply an decrementing offset correction
	FR_OFFSET_NOCHANGE	don't apply any offset correction

5.3.2.1.61. Fr_POCStateType

Purpose	This enumerator type represents the FlexRay controller POC states.	
Type	enum	
Constants	FR_POCTYPE_CONFIG	config state
	FR_POCTYPE_DEFAULT_CONFIG	default config state
	FR_POCTYPE_HALT	halt state
	FR_POCTYPE_NORMAL_ACTIVE	normal active state
	FR_POCTYPE_NORMAL_PASSIVE	normal passive state
	FR_POCTYPE_READY	ready state
	FR_POCTYPE_STARTUP	startup state
	FR_POCTYPE_WAKEUP	wakeup state

5.3.2.1.62. Fr_POCStatusType

Purpose	This structure contains the POC-Status information.	
Type	struct	
Members	Fr_ErrorModeType ErrorCode	CC - error mode
	Fr_SlotModeType SlotMode	CC - slot mode
	Fr_StartupStateType StartupState	CC - startup state
	Fr_POCStateType State	CC - POC state
	Fr_WakeupStatusType WakeupStatus	CC - wakeup state
	boolean CHIHaltRequest	CC - CHI Halt request bit
	boolean ColdstartNoise	CC - Coldstart noise bit
	boolean Freeze	CC - Freeze bit
	boolean CHIReadyRequest	CC - CHI Ready request bit



5.3.2.1.63. Fr_RateCorrectionType

Purpose	This enumerator type contains the possible rate correction operations.	
Type	enum	
Constants	FR_RATE_INC	apply an incrementing rate correction
	FR_RATE_DEC	apply an decrementing rate correction
	FR_RATE_NOCHANGE	don't apply any rate correction

5.3.2.1.64. Fr_RxLPduStatusType

Purpose	This enumerator type represents the LSdu rx status.	
Type	enum	
Constants	FR RECEIVED	LSdu was received
	FR NOT RECEIVED	LSdu was not received
	FR RECEIVED MORE DATA AVAILABLE	LPdu has been received. More instances of this LPdu are available (FIFO usage).

5.3.2.1.65. Fr_SlotModeType

Purpose	This enumerator type represents the FlexRay controller slotmodes.	
Type	enum	
Constants	FR_SLOTMODE_KEY SLOT	key slot mode
	FR_SLOTMODE_ALL_PENDING	all pending slot mode
	FR_SLOTMODE_ALL	all slot mode

5.3.2.1.66. Fr_StartupStateType

Purpose	This enumerator type represents the startup status type.	
Type	enum	
Constants	FR_STARTUP_UNDEFINED	startup state "undefined"
	FR_STARTUP_COLDSTART_LISTEN	startup state "coldstart listen"
	FR_STARTUP_INTEGRATION_COLDSTART_CHECK	startup state "integration coldstart check"



	FR_STARTUP_COLDSTART_JOIN	startup state "coldstart join"
	FR_STARTUP_COLDSTART_COLLISION_RESOLUTION	startup state "collision resolution"
	FR_STARTUP_COLDSTART_CONSISTENCY_CHECK	startup state "consistency check"
	FR_STARTUP_INTEGRATION_LISTEN	startup state "integration listen"
	FR_STARTUP_INITIALIZE_SCHEDULE	startup state "initialize schedule"
	FR_STARTUP_INTEGRATION_CONSISTENCY_CHECK	startup state "integration consistency check"
	FR_STARTUP_COLDSTART_GAP	startup state "coldstart gap"
	FR_STARTUP_EXTERNAL_STARTUP	startup state "external"

5.3.2.1.67. Fr_SyncStateType

Purpose	This enumerator type is used for the FlexRay controller synchronization state.	
Type	enum	
Constants	FR_ASYNC	FlexRay controller is not synchronized to any cluster
	FR_SYNC	FlexRay controller is synchronized to some cluster

5.3.2.1.68. Fr_TxLPduStatusType

Purpose	This enumerator type represents the LSdu tx status.	
Type	enum	
Constants	FR_TRANSMITTED	LSdu was transmitted
	FR_NOT_TRANSMITTED	LSdu was not transmitted
	FR_TRANSMITTED_CONFLICT	A transmission conflict has occurred.

5.3.2.1.69. Fr_WakeupStatusType

Purpose	This enumerator type represents the wakeup status type.	
Type	enum	



Constants	FR_WAKEUP_UNDEFINED	wakeup state "undefined"
	FR_WAKEUP_RECEIVED_HEADER	wakeup state "received header"
	FR_WAKEUP_RECEIVED_WUP	wakeup state "received wakeup pattern"
	FR_WAKEUP_COLLISION_HEADER	wakeup state "collsision header"
	FR_WAKEUP_COLLISION_WUP	wakeup state "collision wakeup pattern"
	FR_WAKEUP_COLLISION_UNKNOWN	wakeup state "collision unknown"
	FR_WAKEUP_TRANSMITTED	wakeup state "transmitted"

5.3.2.1.70. IcomConfigIdType

Purpose	IcomConfigIdType defines the configuration ID.
Type	uint8
Description	An ID of 0 is the default configuration. An ID greater than 0 shall identify a configuration for Pretended Networking. There is more than 1 configuration possible.

5.3.2.1.71. IcomSwitch_ErrorType

Purpose	IcomSwitch_ErrorType defines the errors which can occur when activating or deactivating Pretended Networking.	
Type	enum	
Constants	ICOM_SWITCH_E_OK	
	ICOM_SWITCH_E_FAILED	The activation of Pretended Networking was successful. The activation of Pretended Networking was not successful.

5.3.2.1.72. LinTrcv_TrcvModeType

Purpose	Operating modes of the LIN Transceiver Driver.	
Type	enum	
Constants	LINTRCV_TRCV_MODE_NORMAL	Transceiver mode NORMAL
	LINTRCV_TRCV_MODE_SLEEP	Transceiver mode SLEEP



	LINTRCV_TRCV_MODE_STANDBY	Transceiver mode STANDBY
--	---------------------------	--------------------------

5.3.2.1.73. LinTrcv_TrcvWakeupModeType

Purpose	This type shall be used to control the LIN transceiver concerning wake up events and wake up notifications.	
Type	enum	
Constants	LINTRCV_WUMODE_ENABLE	The notification for wakeup events is enabled on the addressed network.
	LINTRCV_WUMODE_DISABLE	The notification for wakeup events is disabled on the addressed network.
	LINTRCV_WUMODE_CLEAR	A stored wakeup event is cleared on the addressed network.

5.3.2.1.74. LinTrcv_TrcvWakeupReasonType

Purpose	This type denotes the wake up reason detected by the LIN transceiver in detail.	
Type	enum	
Constants	LINTRCV_WU_ERROR	Due to an error wake up reason was not detected. This value may only be reported when error was reported to DEM before.
	LINTRCV_WU_NOT_SUPPORTED	The transceiver does not support any information for the wake up reason.
	LINTRCV_WU_BY_BUS	The transceiver has detected, that the network has caused the wake up of the ECU.
	LINTRCV_WU_BY_PIN	The transceiver has detected a wake-up event at one of the transceiver's pins (not at the LIN bus).
	LINTRCV_WU INTERNALLY	The transceiver has detected, that the network has woken up by the ECU via a request to NORMAL mode.
	LINTRCV_WU_RESET	The transceiver has detected, that the "wake up" is due to an ECU reset.
	LINTRCV_WU_POWER_ON	The transceiver has detected, that the "wake up" is due to an ECU reset after power on.



5.3.2.1.75. Lin_FrameCsModelType

Purpose	This type is used to specify the Checksum model to be used for the LIN Frame.	
Type	enum	
Constants	LIN_ENHANCED_CS	Enhanced checksum model
	LIN_CLASSIC_CS	Classic checksum model

5.3.2.1.76. Lin_FrameDIType

Purpose	This type is used to specify the number of SDU data bytes to copy.
Type	uint8

5.3.2.1.77. Lin_FramePidType

Purpose	Represents all valid protected identifier used by Lin_SendFrame().
Type	uint8
Description	The LIN identifier (0...0x3F) together with its two parity bits.

5.3.2.1.78. Lin_FrameResponseType

Purpose	This type is used to specify whether the frame processor is required to transmit the response part of the LIN frame.	
Type	enum	
Constants	LIN_MASTER_RESPONSE	Response is generated from this (master) node
	LIN_SLAVE_RESPONSE	Response is generated from a remote slave node
	LIN_SLAVE_TO_SLAVE	Response is generated from one slave to another slave, for the master the response will be anonymous, it does not have to receive the response.

5.3.2.1.79. Lin_PduType

Purpose	This Type is used to provide PID, checksum model, data length and SDU pointer from the LIN Interface to the LIN driver.
----------------	---



Type	struct										
Members	<table border="1"> <tr> <td>Lin_FramePidType Pid</td><td></td></tr> <tr> <td>Lin_FrameCsModelType Cs</td><td></td></tr> <tr> <td>Lin_FrameResponseType Drc</td><td></td></tr> <tr> <td>Lin_FrameDlType Dl</td><td></td></tr> <tr> <td>uint8 * SduPtr</td><td>Pointer to the data</td></tr> </table>	Lin_FramePidType Pid		Lin_FrameCsModelType Cs		Lin_FrameResponseType Drc		Lin_FrameDlType Dl		uint8 * SduPtr	Pointer to the data
Lin_FramePidType Pid											
Lin_FrameCsModelType Cs											
Lin_FrameResponseType Drc											
Lin_FrameDlType Dl											
uint8 * SduPtr	Pointer to the data										

5.3.2.1.80. Lin_StatusType

Purpose	LIN operation states for a LIN channel or frame, as returned by the API service Lin_-GetStatus().															
Type	enum															
Constants	<table border="1"> <tr> <td>LIN_NOT_OK</td><td>LIN frame operation return value. Development or production error occurred</td></tr> <tr> <td>LIN_TX_OK</td><td>LIN frame operation return value. Successful transmission.</td></tr> <tr> <td>LIN_TX_BUSY</td><td>LIN frame operation return value. Ongoing transmission (Header or Response).</td></tr> <tr> <td>LIN_TX_HEADER_ERROR</td><td> LIN frame operation return value. Erroneous header transmission such as: <ul style="list-style-type: none"> ▶ Mismatch between sent and read back data ▶ Identifier parity error or ▶ Physical bus error </td></tr> <tr> <td>LIN_RX_ERROR</td><td> LIN frame operation return value. Erroneous response transmission such as: <ul style="list-style-type: none"> ▶ Mismatch between sent and read back data ▶ Physical bus error </td></tr> <tr> <td>LIN_RX_OK</td><td>LIN frame operation return value. Reception of correct response.</td></tr> <tr> <td>LIN_RX_BUSY</td><td>LIN frame operation return value. Ongoing reception: at least one response byte has been received, but the checksum byte has not been received.</td></tr> </table>	LIN_NOT_OK	LIN frame operation return value. Development or production error occurred	LIN_TX_OK	LIN frame operation return value. Successful transmission.	LIN_TX_BUSY	LIN frame operation return value. Ongoing transmission (Header or Response).	LIN_TX_HEADER_ERROR	LIN frame operation return value. Erroneous header transmission such as: <ul style="list-style-type: none"> ▶ Mismatch between sent and read back data ▶ Identifier parity error or ▶ Physical bus error 	LIN_RX_ERROR	LIN frame operation return value. Erroneous response transmission such as: <ul style="list-style-type: none"> ▶ Mismatch between sent and read back data ▶ Physical bus error 	LIN_RX_OK	LIN frame operation return value. Reception of correct response.	LIN_RX_BUSY	LIN frame operation return value. Ongoing reception: at least one response byte has been received, but the checksum byte has not been received.	
LIN_NOT_OK	LIN frame operation return value. Development or production error occurred															
LIN_TX_OK	LIN frame operation return value. Successful transmission.															
LIN_TX_BUSY	LIN frame operation return value. Ongoing transmission (Header or Response).															
LIN_TX_HEADER_ERROR	LIN frame operation return value. Erroneous header transmission such as: <ul style="list-style-type: none"> ▶ Mismatch between sent and read back data ▶ Identifier parity error or ▶ Physical bus error 															
LIN_RX_ERROR	LIN frame operation return value. Erroneous response transmission such as: <ul style="list-style-type: none"> ▶ Mismatch between sent and read back data ▶ Physical bus error 															
LIN_RX_OK	LIN frame operation return value. Reception of correct response.															
LIN_RX_BUSY	LIN frame operation return value. Ongoing reception: at least one response byte has been received, but the checksum byte has not been received.															



	LIN_RX_ERROR	LIN frame operation return value. Erroneous response reception such as: <ul style="list-style-type: none"> ▶ Framing error ▶ Overrun error ▶ Checksum error or ▶ Short response
	LIN_RX_NO_RESPONSE	LIN frame operation return value. No response byte has been received so far.
	LIN_OPERATIONAL	LIN channel state return value. Normal operation; the related LIN channel is ready to transmit next header. No data from previous frame available (e.g. after initialization).
	LIN_CH_SLEEP	LIN channel state return value. Sleep state operation; in this state wake-up detection from slave nodes is enabled.

5.3.2.1.81. NetworkHandleType

Purpose	Identifier of a communication channel.
Type	uint8
Description	Variables of the type NetworkHandleType shall be used to store the identifier of a communication channel

5.3.2.1.82. NotifResultType

Purpose	type for storage of notification result status
Type	uint8

5.3.2.1.83. PNCHandleType

Purpose	Identifier of a partial network cluster.
Type	uint8



Description	Variables of the type PNCHandleType shall be used to store the identifier of a partial network cluster. This type is not specified in the Communication Stack Types SWS of AUTOSAR R4.-0.3 but it is required by the ComM SWS.
--------------------	---

5.3.2.1.84. PduldType

Purpose	Type for a unique identifier for a PDU.
Type	uint16
Description	this type is fixed to uint16 (see deviations of EcuC) ► range: 0 .. Pduldmax

5.3.2.1.85. PduInfoType

Purpose	type for storage of basic information about a PDU	
Type	struct	
Members	uint8 * SduDataPtr	pointer to the SDU of the PDU
	PduLengthType SduLength	length of SDU in bytes
Description	This type shall be used to store the basic information about a PDU of any type, namely a pointer variable pointing to its SDU (payload) and the corresponding length of the SDU in bytes.	

5.3.2.1.86. PduLengthType

Purpose	Type for lengths information of a PDU.
Type	uint16
Description	this type is fixed to uint16 (see deviations of EcuC) ► range: 0 .. PduLengthmax

5.3.2.1.87. RetryInfoType

Purpose	Type to store TP buffer handling information.
----------------	---



Type	struct
Members	TpDataStateType TpDataState
	PduLengthType TxTpDataCnt

5.3.2.1.88. SoAd_MeasurementIdxType

Purpose	Definition of SoAd_MeasurementIdxType - index to select specific measurement data.
Type	uint8

5.3.2.1.89. StatusType

Purpose	Guard macro for type definition of StatusType.
Type	unsigned char
Description	Definition StatusType (OSEK compliance)

5.3.2.1.90. Std_ReturnType

Purpose	Vendor API infix.
Type	uint8
Description	Left empty as this header does not belong to any module Autosar standard API return type

5.3.2.1.91. Std_VersionInfoType

Purpose	return type for xxx_GetVersionInfo() functions of each BSW module
Type	struct
Members	uint16 vendorID
	uint16 moduleID
	uint8 sw_major_version
	uint8 sw_minor_version
	uint8 sw_patch_version
Description	This type is used to request the version of BSW module using the xxx_GetVersionInfo() function.



5.3.2.1.92. TPPParameterType

Purpose	Specifies which TP parameter has to be changed (BS or STMin).	
Type	enum	
Constants	TP_STMIN	This literal identifies TP parameter STMin (Minimum value of TP time-span ST).
	TP_BS	This literal identifies TP parameter BS (Blocksize).
	TP_BC	The Band width control parameter used in FlexRay transport protocol module.
Description	This type shall be used to specify which TP parameter shall be changed when invoking <code>xx_ChangeParameterRequest()</code> .	

5.3.2.1.93. TS_CfgOffsetType

Purpose	Typedef for relative references within the post-build configuration.
Type	uint32
Description	This typedef defines a type that is used for relative references (i.e., offsets) within the post-build configuration

5.3.2.1.94. TS_MaxAlignedType

Purpose	Type definition for each derived type (generated).
Type	double
Description	Platforms Setting: It is set in Platforms plugin by setting the parameter 'Derived-types.[derived type].Mapping:[base type]'.

5.3.2.1.95. TS_VarOffsetType

Purpose	Typedef for relative references within the post-build RAM.
Type	uint16
Description	This typedef defines a type that is used for relative references (i.e., offsets) within the post-build RAM



5.3.2.1.96. Tcplp_MeasurementIdxType

Purpose	Definition of Tcplp_MeasurementIdxType - index to select specific measurement data.
Type	uint8

5.3.2.1.97. TpDataStateType

Purpose	Type to store the state of the TP buffer.	
Type	enum	
Constants	TP_DATACONF	TP_DATACONF indicates that all data, that have been copied so far, are confirmed and can be removed from the TP buffer. Data copied by this API call are excluded and will be confirmed later.
	TP_DATARETRY	TP_DATARETRY indicates that this API call shall copy already copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset of the first byte to be copied by the API call.
	TP_CONF_PENDING	TP_CONF_PENDING indicates that the previously copies data must remain in the TP.
	EB_TP_RETRYINFONULL	EB_TP_RETRYINFONULL indicates that for the structure containing TpDataState a NULL_PTR was originally provided to this API. This value is an EB extension to handle cases where a NULL_PTR is provided for a parameter of type RetryInfoType , which in general is valid and specified by Autosar. However, in the special case the SchM is involved a NULL_PTR cannot be handled because it is dereferenced in the process of saving parameters of a function. Further Note: EB_TP_RETRYINFONULL describes a state similar to TP_-



	DATACONF, with the difference that all data can be removed from TP buffer.
--	--

5.3.2.1.98. VoidPtr

Purpose	type definition for pointer to void
Type	void *

5.3.2.1.99. uint32

Purpose	Macro that is only set if 64bit base types are available (generated, depending on parameter 'Cpu.Type').
Type	unsigned int
Description	<p>Type definition for each AUTOSAR standard type with existing platform specific mapping to a base type (generated) AUTOSAR standard types are: boolean, sint8, sint16, sint32, uint8, uint16, uint32, sint8_least, sint16_least, sint32_least, uint8_least, uint16_least, uint32_least, float32, float64; for CPUs support 64bit types additionally: sint64, uint64</p> <p>Platforms Setting: Definition of platform specific AUTOSAR standard types that are set in the Platforms plugin by parameter 'Basetypes.[AUTOSAR type].Mapping:[base type]'</p>

5.3.2.1.100. usize

Purpose	Type definition of platform specific size type (generated, depending on parameter 'Cpu.Type').
Type	uint32

5.3.2.2. Macro constants

5.3.2.2.1. ADD_TO_FILTER

Purpose	Constant for value ADD_TO_FILTER of Eth_FilterActionType (Compatibility with AUTOSAR 4.1.0).
---------	--



Value	0U
--------------	----

5.3.2.2.2. BASE_DBG_ENABLE

Purpose	Macro for enabling debug header file (generated).
Value	STD_OFF
Description	<p>Range:</p> <ul style="list-style-type: none"> ▶ STD_ON (if configuration parameter 'Base/BaseDbg/BaseDbgHeaderFile' enabled) ▶ STD_OFF (if configuration parameter 'Base/BaseDbg/BaseDbgHeaderFile' disabled)

5.3.2.2.3. BASE_PDUR_CONFIG_PTR

Purpose	Address of the first multiple configuration container for each module, or special handling for specific modules (generated).
Value	NULL_PTR

5.3.2.2.4. BASE_PDUR_ENABLED

Purpose	Enable status for each module (VariantPostBuild) (generated).
Value	STD_ON

5.3.2.2.5. BASE_PDUR_HEADER

Purpose	Name of module's main header file for each module (generated).
Value	<PduR.h>

5.3.2.2.6. BOOLEAN_C

Purpose	Macro to define a constant of type boolean.
----------------	---



Value	((boolean) x ## U)
--------------	--------------------

5.3.2.2.7. BUSTRCV_E_ERROR

Purpose	Bus transceiver detected an unclassified error.
Value	0x01U

5.3.2.2.8. BUSTRCV_OK

Purpose	Code for: no bus error.
Value	0x00U
Description	There is no bus transceiver error seen by the driver of transceiver does not support the detection of bus errors.

5.3.2.2.9. Base_CustomStdIncludeFiles

Purpose	Inclusions of custom standard files (generated).
Value	
Description	<p>For each configuration parameter 'Base/CustomStdIncludeFiles/[file]', an include statement is generated</p> <p>Platforms Setting: Custom standard file can be used for definition of additional standard types that are customer specific.</p>

5.3.2.2.10. COMPILER_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.3.2.2.11. COMPILER_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
----------------	--------------------------------



Value	0U
--------------	----

5.3.2.2.12. COMPILER_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.3.2.2.13. COMPILER_MODULE_ID

Purpose	AUTOSAR module identification.
Value	198U

5.3.2.2.14. COMPILER_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	3U

5.3.2.2.15. COMPILER_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	2U

5.3.2.2.16. COMPILER_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	0U

5.3.2.2.17. COMPILER_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U



5.3.2.2.18. COMSTACK_BF_START

Purpose	
Value	((uint32)(~((uint32)0u)))

5.3.2.2.19. COMSTACK_BYTE_MIRROR16

Purpose	Byte mirror macro to convert uint16 values between little and big endian format. Should usually map to a _ROR(x,8u) or similar compiler-known-function.
Value	COMSTACK_CT_BYTE_MIRROR16(x)

5.3.2.2.20. COMSTACK_BYTE_MIRROR32

Purpose	Byte mirror macro to convert uint32 values between little and big endian format. Should usually map to a _builtin_byteswap or similar compiler-known-function. Can also be implemented via inline assembly, using 4-5 rotates of full & half registers.
Value	COMSTACK_CT_BYTE_MIRROR32(value)

5.3.2.2.21. COMSTACK_BYTE_REPLICATOR32

Purpose	A constant multiplier to replicate a multiplied byte within a 32 bit register. Big constants are costly. Hence we multiply byte constants with byte multipliers to provide clever compilers alternate routes for evaluation: e.g. const / value range analysis -> PC-rel load of all-large consts vs. loading one big multiplier constant & then multiplying it with a number of small consts that each fit in the 8bit immediate operands (if the MUL is cheaper than the LD).
Value	UINT32_C(0x01010101)
Description	If your compiler is not smart enough to evaluate alternate routes or falls for premature constant-expression-evaluation, you may try to predefine the multiplier constants in terms of a volatile variable that effectively hides the constant from the compiler.

5.3.2.2.22. COMSTACK_CT_BYTE_MIRROR16

Purpose	Byte mirror macro to convert uint16 values between little and big endian format. Non-overloadable. For compile-time evaluation.
----------------	---



Value	((uint16)(((uint16)((uint16)(x)) << 8u)) ((uint16)((uint16)(x))>> 8u)))
--------------	---

5.3.2.2.23. COMSTACK_CT_BYTE_MIRROR32

Purpose	Byte mirror macro to convert uint32 values between little and big endian format. Non-overloadable. For compile-time evaluation.
Value	(uint32)\ ((uint32)((uint32)((uint32)(value) & (uint32)0x000000FFUL) << 24U)) \\ (uint32)((uint32)((uint32)(value) & (uint32)0x0000FF00UL) << 8U)) \\ (uint32)((uint32)(value) >> 8U) & ((uint32)0x0000FF00UL)) \\ (uint32)((uint32)(value) >> 24U) & ((uint32)0x000000FFUL)))

5.3.2.2.24. COMSTACK_CT_HTON_UINT16

Purpose	uint16 host to network order conversion macro. Non-overloadable. For compile-time evaluation.
Value	TS_IF_BE_LE((value), COMSTACK_CT_BYTE_MIRROR16(value))

5.3.2.2.25. COMSTACK_CT_HTON_UINT32

Purpose	uint32 host to network order conversion macro Non-overloadable. For compile-time evaluation.
Value	TS_IF_BE_LE((value), COMSTACK_CT_BYTE_MIRROR32(value))

5.3.2.2.26. COMSTACK_CT_NTOH_UINT16

Purpose	uint16 network to host order conversion macro. Non-overloadable. For compile-time evaluation.
Value	TS_IF_BE_LE((value), COMSTACK_CT_BYTE_MIRROR16(value))

5.3.2.2.27. COMSTACK_CT_NTOH_UINT32

Purpose	uint32 network to host order conversion macro Non-overloadable. For compile-time evaluation.
Value	TS_IF_BE_LE((value), COMSTACK_CT_BYTE_MIRROR32(value))



5.3.2.2.28. COMSTACK_GET16

Purpose	macro gets a 16 bit value from a network header.
Value	((uint16)((uint16)((uint16)((ComStack_ConstUint8PtrType)(headerPtr)[(constByteOffset)]))<<8u)) \ ((uint16)((uint8) (((ComStack_ConstUint8PtrType)(headerPtr))[(constByteOffset)+1u]))))

5.3.2.2.29. COMSTACK_GET32

Purpose	macro gets an uint32 value from a network header.
Value	((uint32)((uint32)((ComStack_ConstUint8PtrType)(headerPtr)[(constByteOffset)]))<<24u)) \ ((uint32)((uint32)((ComStack_ConstUint8PtrType)(headerPtr))[(constByteOffset)+1]))<<16u)) \ ((uint32)((uint32)((ComStack_ConstUint8PtrType)(headerPtr))[(constByteOffset)+2]))<<8u)) \ ((uint32)((uint32)((ComStack_ConstUint8PtrType)(headerPtr))[(constByteOffset)+3]))))

5.3.2.2.30. COMSTACK_GETCPY32

Purpose	macro copies 32 bit value (in network byte order) from headerPtr @ constByteOffset to resultPtr
Value	do { \ enum {ofs= (constByteOffset)}; /* only compile-time constant allowed */ \ ComStack_ConstUint8PtrType const hdrPtr= (headerPtr); /* single argument eval. */ \ uint8* const resPtr= (uint8*)(resultPtr); /* cast to correct type */ \ resPtr[0]= hdrPtr[ofs]; \ resPtr[1]= hdrPtr[ofs+1]; \ resPtr[2]= hdrPtr[ofs+2]; \ resPtr[3]= hdrPtr[ofs+3]; \ } while(0)

5.3.2.2.31. COMSTACK_HAMMING_WEIGHT32

Purpose	Hamming weight computes the number of bits set in the input argument (register width for 0). We employ the ISA equivalent of a binary adder tree, using register-width SHIFT & ADD operations, as SIMD operations by ignoring carry overflows, as it's proven they cannot occur. Thus we'd finish in $\lceil \log(\text{registerwidth}) \rceil$ steps. However, we switch back to folding adds, once we reach operand sizes that are likely to be computed cheaper using register-parts or 'extract' operations due to less parallelism. At the special request of the BITE team: https://en.wikipedia.org/wiki/Hamming_weight#Efficient_implementation .
Value	do { \ const uint32 rep32 = COMSTACK_BYTE_REPLICATOR32; \ const uint32 m1 = 0x55u * rep32; \ const uint32 m2 = 0x33u * rep32; \ const uint32 m4 = ((uint32)(rep32



```
<< 4u)) - rep32; \ uint32 x= (uint32) (in32); \ x= ((uint32)(x & m1)) + ((uint32)((uint32)
(x >> 1u)) & m1)); \ x= ((uint32)(x & m2)) + ((uint32)((uint32)(x >> 2u)) & m2)); \ x=
((uint32)(x & m4)) + ((uint32)((uint32)(x >> 4u)) & m4)); \ x+= x << 16u; \ x+= x << 8u;
\ result8_least= x >> ((sizeof(uint32)-1u) << 3u); \ } while(0)
```

5.3.2.2.32. COMSTACK_HTON_UINT16

Purpose	uint16 host to network order conversion macro
Value	TS_IF_BE_LE((value), COMSTACK_BYTE_MIRROR16(value))

5.3.2.2.33. COMSTACK_HTON_UINT32

Purpose	uint32 host to network order conversion macro
Value	TS_IF_BE_LE((value), COMSTACK_BYTE_MIRROR32(value))

5.3.2.2.34. COMSTACK_ISA_WIDTH

Purpose	
Value	32

5.3.2.2.35. COMSTACK_LOWEST_BIT32

Purpose	Compute the lowest bit position for the input in32 z=z^(z-1) : the z-1 lets the under-flow-carry ripple up to the lowest bit clearing it. XOR-ing that with the original value reinstates the lowest bit, but clears all higher bits. Now we only have to count the number of set bits, subtracting one to be zero-based.
Value	do { \ uint8_least y; \ <u>uint32</u> z= (<u>uint32</u>)(in32); \ z= z ^ (z-1); \ COMSTACK_HAMMING_WEIGHT32(y,z); \ result8_least= y-1; \ } while(0)

5.3.2.2.36. COMSTACK_NTOH_UINT16

Purpose	uint16 network to host order conversion macro
Value	TS_IF_BE_LE((value), COMSTACK_BYTE_MIRROR16(value))



5.3.2.2.37. COMSTACK_NTOH_UINT32

Purpose	uint32 network to host order conversion macro
Value	TS_IF_BE_LE((value), COMSTACK_BYTE_MIRROR32(value))

5.3.2.2.38. COMSTACK_SET16

Purpose	macro sets any network header 16 bit field to value16
Value	do { \ enum {ofs= (constByteOffset) }; /* only compile-time constant allowed */ \ ComStack_Uint8PtrType const hdrPtr= (headerPtr); /* single argument eval. */ \ uint16 const val16= (value16); /* ensure correct type */ \ hdrPtr[ofs]= TS_CAST(uint, 8, val16 >> 8u); \ hdrPtr[ofs+1]= TS_CAST(uint, 8, val16); \ } while(0)

5.3.2.2.39. COMSTACK_SET32

Purpose	macro sets any network header 32 bit field to value32
Value	do { \ enum {ofs= (constByteOffset) }; /* only compile-time constant allowed */ \ ComStack_Uint8PtrType const hdrPtr= (headerPtr); /* single argument eval. */ \ uint32 const val32= (value32); /* ensure correct type */ \ hdrPtr[ofs]= TS_CAST(uint, 8, val32 >> 24u); \ hdrPtr[ofs+1]= TS_CAST(uint, 8, val32 >> 16u); \ hdrPtr[ofs+2]= TS_CAST(uint, 8, val32 >> 8u); \ hdrPtr[ofs+3]= TS_CAST(uint, 8, val32); \ } while(0)

5.3.2.2.40. COMSTACK_SETCOPY32

Purpose	macro copies 32 bit value (in network byte order) from sourcePtr to resultPtr @ const-ByteOffset
Value	do { \ enum {ofs= (constByteOffset) }; /* only compile-time constant allowed */ \ ComStack_Uint8PtrType const hdrPtr= (headerPtr); /* single argument eval. */ \ uint8 const* const srcPtr= (uint8 const* const)(sourcePtr); /* cast to correct type */ \ hdrPtr[ofs]= srcPtr[0]; \ hdrPtr[ofs+1]= srcPtr[1]; \ hdrPtr[ofs+2]= srcPtr[2]; \ hdrPtr[ofs+3]= srcPtr[3]; \ } while(0)

5.3.2.2.41. COMTYPE_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
----------------	--------------------------------



Value	4U
--------------	----

5.3.2.2.42. COMTYPE_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.3.2.2.43. COMTYPE_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.3.2.2.44. COMTYPE_MODULE_ID

Purpose	AUTOSAR module identification.
Value	196U
Description	Left empty as this header does not belong to any module

5.3.2.2.45. COMTYPE_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	3U

5.3.2.2.46. COMTYPE_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	2U

5.3.2.2.47. COMTYPE_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	0U



5.3.2.2.48. COMTYPE_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.3.2.2.49. CONSTDEF

Purpose	macro to be used for top-level const -ness of function parameters.
Value	const
Description	Background: In C, the top level const -ness of a parameter identifier is not part of the parameter profile. Hence it is always useful to write-protect incoming parameters, even if they are passed by-value. While this is also useful for scalars (int, etc), the biggest value shows for pointers and pointers to pointers that should not be changed within the function. Hint: as the top level const belongs to the identifier itself, making it a non-L-value (not assignable), it makes much sense to write this CONSTDEF immediately in front of the identifier. If a specific implementation needs to change that value (e.g.: redirect to temporary buffer, ...), the implementation is free to leave out that top-level const and still satisfies the interface (prototype, function-pointer assignability, linker symbol-lookup, ...) Only DIAB has problems with top level const definitions vs declarations: they must be identical. Hence, for Diab, we define this symbol as empty. General const-correctness and safety is still maintained throughout all other architectures.

5.3.2.2.50. CPU_BIT_ORDER

Purpose	Bit order of this CPU (generated).
Value	BE
Description	<p>Platforms Setting: The bit order of the CPU is set in the Platforms plugin by setting the parameter 'Cpu.Bitorder'.</p> <p>Range:</p> <ul style="list-style-type: none"> ▶ LE (LSB_FIRST if parameter 'Cpu.Bitorder:LE') ▶ BE (MSB_FIRST if parameter 'Cpu.Bitorder:BE')

5.3.2.2.51. CPU_BYTE_ORDER

Purpose	Byte order of this CPU (generated).
----------------	-------------------------------------



Value	LE
Description	<p>Platforms Setting: The byte order of the CPU is set in the Platforms plugin by setting the parameter 'Cpu.Byteorder'.</p> <p>Range:</p> <ul style="list-style-type: none"> ▶ BE (HIGH_BYTE_FIRST if parameter 'Cpu.Byteorder:BE') ▶ LE (LOW_BYTE_FIRST if parameter 'Cpu.Byteorder:LE')

5.3.2.2.52. CPU_TYPE

Purpose	Word size of this CPU (generated).
Value	CPU_TYPE_32
Description	<p>Platforms Setting: The word size specifies the natural data unit size of a CPU. It is set in the Platforms plugin by setting the parameter 'Cpu.Type'.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ▶ CPU_TYPE_64 (if parameter 'Cpu.Type:64') ▶ CPU_TYPE_32 (if parameter 'Cpu.Type:32') ▶ CPU_TYPE_16 (if parameter 'Cpu.Type:16') ▶ CPU_TYPE_8 (if parameter 'Cpu.Type:8')

5.3.2.2.53. CPU_TYPE_16

Purpose	cpu identifier for 8-bit CPUs
Value	16U

5.3.2.2.54. CPU_TYPE_32

Purpose	cpu identifier for 32-bit CPUs
Value	32U

5.3.2.2.55. CPU_TYPE_64

Purpose	cpu identifier for 64-bit CPUs
----------------	--------------------------------



Value	64U
--------------	-----

5.3.2.2.56. CPU_TYPE_8

Purpose	cpu identifier for 8-bit CPUs
Value	8U

5.3.2.2.57. ETHIF_MEAS_ALL

Purpose	Constant for the value of Measurement index that represents all measurement index-es.
Value	0xFFU

5.3.2.2.58. ETHIF_MEAS_DROP_CRTLIDX

Purpose	Constant for the value of Measurement index for dropped datagrams caused by invalid CtrlIdx/VLAN.
Value	1U

5.3.2.2.59. ETHSWT_MACLEARING_HWDISABLED

Purpose	Constant for value HWDISABLED of EthSwt_MacLearningType.
Value	0U

5.3.2.2.60. ETHSWT_MACLEARING_HWENABLED

Purpose	Constant for value HWENABLED of EthSwt_MacLearningType.
Value	1U

5.3.2.2.61. ETHSWT_MACLEARING_SWENABLED

Purpose	Constant for value SWENABLED of EthSwt_MacLearningType.
Value	2U



5.3.2.2.62. ETHSWT_STATE_ACTIVE

Purpose	Constant for value ACTIVE of EthSwt_StateType.
Value	2U

5.3.2.2.63. ETHSWT_STATE_INIT

Purpose	Constant for value INIT of EthSwt_StateType.
Value	1U

5.3.2.2.64. ETHSWT_STATE_UNINIT

Purpose	Constant for value UNINIT of EthSwt_StateType.
Value	0U

5.3.2.2.65. ETHTRCV_BAUD_RATE_1000MBIT

Purpose	Constant for value 1000MBIT of EthTrcv_BaudRateType.
Value	2U

5.3.2.2.66. ETHTRCV_BAUD_RATE_100MBIT

Purpose	Constant for value 100MBIT of EthTrcv_BaudRateType.
Value	1U

5.3.2.2.67. ETHTRCV_BAUD_RATE_10MBIT

Purpose	Constant for value 10MBIT of EthTrcv_BaudRateType.
Value	0U

5.3.2.2.68. ETHTRCV_CABLEDIAG_ERROR

Purpose	Constant for value CABLE DIAGNOSTIC FAILED of EthTrcv_CableDiagResultType.
----------------	--



Value	1U
--------------	----

5.3.2.2.69. ETHTRCV_CABLEDIAG_OK

Purpose	Constant for value CABLE DIAGNOSTIC OK of EthTrcv_CableDiagResultType.
Value	0U

5.3.2.2.70. ETHTRCV_CABLEDIAG_OPEN

Purpose	Constant for value OPEN CIRCUIT DETECTED of EthTrcv_CableDiagResultType.
Value	3U

5.3.2.2.71. ETHTRCV_CABLEDIAG_PENDING

Purpose	Constant for value CABLE DIAGNOSTIC PENDING of EthTrcv_CableDiagResultType.
Value	0x04U

5.3.2.2.72. ETHTRCV_CABLEDIAG_SHORT

Purpose	Constant for value SHORT CIRCUIT DETECTED of EthTrcv_CableDiagResultType.
Value	2U

5.3.2.2.73. ETHTRCV_CABLEDIAG_WRONG_POLARITY

Purpose	Constant for value CABLE DIAGNOSTIC WRONG POLARITY of EthTrcv_CableDiagResultType.
Value	0x05U

5.3.2.2.74. ETHTRCV_DUPLEX_MODE_FULL

Purpose	Constant for value FULL of EthTrcv_DuplexModeType.
Value	1U



5.3.2.2.75. ETHTRCV_DUPLEX_MODE_HALF

Purpose	Constant for value HALF of EthTrcv_DuplexModeType.
Value	0U

5.3.2.2.76. ETHTRCV_LINK_STATE_ACTIVE

Purpose	Constant for value ACTIVE of EthTrcv_LinkStateType.
Value	1U

5.3.2.2.77. ETHTRCV_LINK_STATE_DOWN

Purpose	Constant for value DOWN of EthTrcv_LinkStateType.
Value	0U

5.3.2.2.78. ETHTRCV_MODE_ACTIVE

Purpose	Constant for value ACTIVE of EthTrcv_ModeType.
Value	1U

5.3.2.2.79. ETHTRCV_MODE_DOWN

Purpose	Constant for value DOWN of EthTrcv_ModeType.
Value	0U

5.3.2.2.80. ETHTRCV_PHYLOOPBACK_EXTERNAL

Purpose	Constant for value EXTERNAL LOOPBACK of EthTrcv_PhysicalLoopbackModeType.
Value	2U

5.3.2.2.81. ETHTRCV_PHYLOOPBACK_INTERNAL

Purpose	Constant for value INTERNAL LOOPBACK of EthTrcv_PhysicalLoopbackModeType.
----------------	---



Value	1U
--------------	----

5.3.2.2.82. ETHTRCV_PHYLOOPBACK_NONE

Purpose	Constant for value NORMAL OPERATION of EthTrcv_PhysLoopbackModeType.
Value	0U

5.3.2.2.83. ETHTRCV_PHYLOOPBACK_REMOTE

Purpose	Constant for value REMOTE LOOPBACK of EthTrcv_PhysLoopbackModeType.
Value	3U

5.3.2.2.84. ETHTRCV_PHYTESTMODE_1

Purpose	Constant for value TEST TRANSMITTER DROOP of EthTrcv_PhysTestModeType.
Value	1U

5.3.2.2.85. ETHTRCV_PHYTESTMODE_2

Purpose	Constant for value TEST MASTER TIMING JITTER of EthTrcv_PhysTestModeType.
Value	2U

5.3.2.2.86. ETHTRCV_PHYTESTMODE_3

Purpose	Constant for value TEST SLAVE TIMING JITTER of EthTrcv_PhysTestModeType.
Value	3U

5.3.2.2.87. ETHTRCV_PHYTESTMODE_4

Purpose	Constant for value TEST TRANSMITTER DISTORTION of EthTrcv_PhysTestModeType.
----------------	---



Value	4U
--------------	----

5.3.2.2.88. ETHTRCV_PHYTESTMODE_5

Purpose	Constant for value TEST POWER SPECTRAL DENSITY (PSD) MASK of EthTrcv_PhysTestModeType.
Value	5U

5.3.2.2.89. ETHTRCV_PHYTESTMODE_NONE

Purpose	Constant for value NORMAL OPERATION of EthTrcv_PhysTestModeType.
Value	0U

5.3.2.2.90. ETHTRCV_PHYTXMODE_NORMAL

Purpose	Constant for value NORMAL OPERATION of EthTrcv_PhysTxModeType.
Value	0U

5.3.2.2.91. ETHTRCV_PHYTXMODE_SCRAMBLER_OFF

Purpose	Constant for value SCRAMBLER DISABLED of EthTrcv_PhysTxModeType.
Value	2U

5.3.2.2.92. ETHTRCV_PHYTXMODE_TX_OFF

Purpose	Constant for value TRANSMITTER DISABLED of EthTrcv_PhysTxModeType.
Value	1U

5.3.2.2.93. ETHTRCV_STATE_ACTIVE

Purpose	Constant for value ACTIVE of EthTrcv_StateType.
----------------	---



Value	2U
--------------	----

5.3.2.2.94. ETHTRCV_STATE_INIT

Purpose	Constant for value INIT of EthTrcv_StateType.
Value	1U

5.3.2.2.95. ETHTRCV_STATE_UNINIT

Purpose	Constant for value UNINIT of EthTrcv_StateType.
Value	0U

5.3.2.2.96. ETHTRCV_WUM_CLEAR

Purpose	Constant for value transceiver wake up reason cleared of EthTrcv_WakeupModeType.
Value	2U

5.3.2.2.97. ETHTRCV_WUM_DISABLE

Purpose	Constant for value transceiver wake up mode disabled of EthTrcv_WakeupModeType.
Value	0U

5.3.2.2.98. ETHTRCV_WUM_ENABLE

Purpose	Constant for value Transceiver wake up mode enabled of EthTrcv_WakeupModeType.
Value	1U

5.3.2.2.99. ETHTRCV_WUR_BUS

Purpose	Constant for value transceiver wake up reason - Bus wake up detected.
----------------	---



Value	2U
--------------	----

5.3.2.2.100. ETHTRCV_WUR_GENERAL

Purpose	Constant for value Transceiver wake up reason - no distinct reason supported by hardware.
Value	1U

5.3.2.2.101. ETHTRCV_WUR_INTERNAL

Purpose	Constant for value transceiver wake up reason - Internal wake up detected.
Value	3U

5.3.2.2.102. ETHTRCV_WUR_NONE

Purpose	Constant for value transceiver wake up reason - no wake up reason detected.
Value	0U

5.3.2.2.103. ETHTRCV_WUR_PIN

Purpose	Constant for value transceiver wake up reason - Pin wake up detected.
Value	6U

5.3.2.2.104. ETHTRCV_WUR_POWER_ON

Purpose	Constant for value transceiver wake up reason - Power on wake up detected.
Value	5U

5.3.2.2.105. ETHTRCV_WUR_RESET

Purpose	Constant for value Transceiver wake up reason - Reset wake up detected.
----------------	---



Value	4U
--------------	----

5.3.2.2.106. ETHTRCV_WUR_SYSERR

Purpose	Constant for value Transceiver wake up reason - System error wake up detected.
Value	7U

5.3.2.2.107. ETH_ADD_TO_FILTER

Purpose	Constant for value ADD_TO_FILTER of Eth_FilterActionType.
Value	0U

5.3.2.2.108. ETH_BUFLISTTYPE_DEF

Purpose	Buffer list type macro to prevent double definition in Eth (backward compatibility, to prevent double definition in Eth_Types.h).
Value	

5.3.2.2.109. ETH_E_NOT_OK

Purpose	Constant for value NOT_OK of Eth_ReturnType.
Value	1U

5.3.2.2.110. ETH_E_NO_ACCESS

Purpose	Constant for value NO_ACCESS of Eth_ReturnType.
Value	2U

5.3.2.2.111. ETH_INVALID

Purpose	Constant for value INVALID of Eth_TimeStampQualType.
Value	1U



5.3.2.2.112. ETH_MODE_ACTIVE

Purpose	Constant for value ACTIVE of Eth_ModeType.
Value	1U

5.3.2.2.113. ETH_MODE_DOWN

Purpose	Constant for value DOWN of Eth_ModeType.
Value	0U

5.3.2.2.114. ETH_NOT RECEIVED

Purpose	Constant for value NOT_RECEIVED of Eth_FilterActionType.
Value	1U

5.3.2.2.115. ETH_OK

Purpose	Constant for value OK of Eth_ReturnType.
Value	0U

5.3.2.2.116. ETH RECEIVED

Purpose	Constant for value RECEIVED of Eth_FilterActionType.
Value	0U

5.3.2.2.117. ETH RECEIVED_FRAMES LOST

Purpose	Constant for value RECEIVED_FRAMES_LOST of Eth_FilterActionType.
Value	3U

5.3.2.2.118. ETH RECEIVED_MORE DATA AVAILABLE

Purpose	Constant for value RECEIVED_MORE_DATA_AVAILABLE of Eth_FilterActionType.
----------------	--



Value	2U
--------------	----

5.3.2.2.119. ETH_REMOVE_FROM_FILTER

Purpose	Constant for value REMOVE_FROM_FILTER of Eth_FilterActionType.
Value	1U

5.3.2.2.120. ETH_RETRANSMITINFOTYPE_DEF

Purpose	Retransmit info type macro to prevent double definition in Eth (backward compatibility, to prevent double definition in Eth_Types.h).
Value	

5.3.2.2.121. ETH_STATE_ACTIVE

Purpose	Constant for value ACTIVE of Eth_StateType.
Value	2U

5.3.2.2.122. ETH_STATE_INIT

Purpose	Constant for value INIT of Eth_StateType.
Value	1U

5.3.2.2.123. ETH_STATE_UNINIT

Purpose	Constant for value UNINIT of Eth_StateType.
Value	0U

5.3.2.2.124. ETH_UNCERTAIN

Purpose	Constant for value UNCERTAIN of Eth_TimeStampQualType.
----------------	--



Value	2U
--------------	----

5.3.2.2.125. ETH_VALID

Purpose	Constant for value VALID of Eth_TimeStampQualType.
Value	0U

5.3.2.2.126. E_NOT_OK

Purpose	Constant for value NOT_OK of StatusType.
Value	1U

5.3.2.2.127. E_NO_DATA

Purpose	No data available which can be deserialized.
Value	0x01U

5.3.2.2.128. E_OK

Purpose	Constant for value OK of StatusType.
Value	0U

5.3.2.2.129. E_SAFETY_HARD_RUNTIMEERROR

Purpose	A runtime error occurred, safety properties could not be checked and NO output data could be produced.
Value	0xFFU

5.3.2.2.130. E_SAFETY_INIT_ERR

Purpose	Not enough data were received to use them, additionally a check failed.
----------------	---



Value	0x33U
--------------	-------

5.3.2.2.131. E_SAFETY_INIT_NND

Purpose	Not enough data were received to use them, additionally no new data received.
Value	0x35U

5.3.2.2.132. E_SAFETY_INIT_OK

Purpose	Not enough data were received to use them.
Value	0x30U

5.3.2.2.133. E_SAFETY_INIT REP

Purpose	Not enough data were received to use them but some with a repeated counter were received.
Value	0x31U

5.3.2.2.134. E_SAFETY_INIT_SEQ

Purpose	Not enough data were received to use them, additionally a counter jump occurred.
Value	0x32U

5.3.2.2.135. E_SAFETY_INVALID_ERR

Purpose	The data are invalid and cannot be used because a check failed.
Value	0x43U

5.3.2.2.136. E_SAFETY_INVALID_NND

Purpose	Communication is invalid according to safety and no new data received.
Value	0x45U



5.3.2.2.137. E_SAFETY_INVALID_OK

Purpose	The data are invalid and cannot be used.
Value	0x40U

5.3.2.2.138. E_SAFETY_INVALID REP

Purpose	The data are invalid and cannot be used because a repeated counter was received.
Value	0x41U

5.3.2.2.139. E_SAFETY_INVALID_SEQ

Purpose	The data are invalid and cannot be used due to a counter jump.
Value	0x42U

5.3.2.2.140. E_SAFETY_NODATA_ERR

Purpose	No data are available since initialization of transformer. Therefore the check failed.
Value	0x23U

5.3.2.2.141. E_SAFETY_NODATA_NND

Purpose	No data are available since initialization of transformer.
Value	0x25U

5.3.2.2.142. E_SAFETY_NODATA_OK

Purpose	No data are available since initialization of transformer.
Value	0x20U

5.3.2.2.143. E_SAFETY_NODATA REP

Purpose	No data are available since initialization of transformer because a repeated counter was received.
----------------	--



Value	0x21U
--------------	-------

5.3.2.2.144. E_SAFETY_NODATA_SEQ

Purpose	No data are available since initialization of transformer and a counter jump occurred.
Value	0x22U

5.3.2.2.145. E_SAFETY_SOFT_RUNTIMEERROR

Purpose	A runtime error occurred, safety properties could not be checked (state or status cannot be determined) but non-protected output data could be produced nonetheless.
Value	0x77U

5.3.2.2.146. E_SAFETY_VALID_ERR

Purpose	The data are valid according to safety, although the check itself failed.
Value	0x03U

5.3.2.2.147. E_SAFETY_VALID_NND

Purpose	Communication is valid according to safety, but no new data received.
Value	0x05U

5.3.2.2.148. E_SAFETY_VALID REP

Purpose	The data are valid according to safety, although data with a repeated counter were received.
Value	0x01U

5.3.2.2.149. E_SAFETY_VALID_SEQ

Purpose	The data are valid according to safety, although a counter jump occurred.
----------------	---



Value	0x02U
--------------	-------

5.3.2.2.150. E_SEC_NOT_AUTH

Purpose	The data was not authenticated correctly.
Value	0x01U

5.3.2.2.151. E_SEC_NOT_FRESH

Purpose	The data was not fresh.
Value	0x02U

5.3.2.2.152. E_SER_GENERIC_ERROR

Purpose	A generic not precisely detailed error occurred.
Value	0x81U

5.3.2.2.153. E_SER_MALFORMED_MESSAGE

Purpose	The received message is malformed. The transformer is not able to produce an output.
Value	0x89U

5.3.2.2.154. E_SER_WRONG_INTERFACE_VERSION

Purpose	Interface version of serialized data is not supported.
Value	0x88U

5.3.2.2.155. E_SER_WRONG_MESSAGE_TYPE

Purpose	The received message type was not expected.
----------------	---



Value	0x8aU
--------------	-------

5.3.2.2.156. E_SER_WRONG_PROTOCOL_VERSION

Purpose	The version of the receiving transformer did not match the sending transformer.
Value	0x87U

5.3.2.2.157. FALSE

Purpose	false value for boolean type
Value	0U

5.3.2.2.158. FR_CIDX_GCOLDSTARTATTEMPTS

Purpose	Index for config parameter FrIfGColdStartAttempts.
Value	17U

5.3.2.2.159. FR_CIDX_GCYCLECOUNTMAX

Purpose	Index for config parameter FrIfGCycleCountMax.
Value	18U

5.3.2.2.160. FR_CIDX_GACTIONPOINTOFFSET

Purpose	Index for config parameter FrIfGdActionPointOffset.
Value	25U

5.3.2.2.161. FR_CIDX_GDBIT

Purpose	Index for config parameter FrIfGdBit.
Value	26U



5.3.2.2.162. FR_CIDX_GDCASRXLOWMAX

Purpose	Index for config parameter FrIfGdCasRxLowMax.
Value	27U

5.3.2.2.163. FR_CIDX_GDCYCLE

Purpose	Index for config parameter FrIfGdCycle.
Value	0U

5.3.2.2.164. FR_CIDX_GDDYNAMICSLOTIDLEPHASE

Purpose	Index for config parameter FrIfGdDynamicSlotIdlePhase.
Value	28U

5.3.2.2.165. FR_CIDX_GDIGNOREAFTERTX

Purpose	Index for config parameter FrIfGdIgnoreAfterTx.
Value	54U

5.3.2.2.166. FR_CIDX_GDMACROTICK

Purpose	Index for config parameter FrIfGdMacrotick.
Value	4U

5.3.2.2.167. FR_CIDX_GDMINISLOT

Purpose	Index for config parameter FrIfGdMinislot.
Value	30U

5.3.2.2.168. FR_CIDX_GDMINISLOTACTIONPOINTOFFSET

Purpose	Index for config parameter FrIfGdMiniSlotActionPointOffset.
----------------	---



Value	29U
--------------	-----

5.3.2.2.169. FR_CIDX_GDNIT

Purpose	Index for config parameter FrIfGdNit.
Value	7U

5.3.2.2.170. FR_CIDX_GDSAMPLECLOCKPERIOD

Purpose	Index for config parameter FrIfGdSampleClockPeriod.
Value	31U

5.3.2.2.171. FR_CIDX_GDSTATICSLOT

Purpose	Index for config parameter FrIfGdStaticSlot.
Value	8U

5.3.2.2.172. FR_CIDX_GDSYMBOLWINDOW

Purpose	Index for config parameter FrIfGdSymbolWindow.
Value	32U

5.3.2.2.173. FR_CIDX_GDSYMBOLWINDOWACTIONPOINTOFFSET

Purpose	Index for config parameter FrIfGdSymbolWindowActionPointOffset.
Value	33U

5.3.2.2.174. FR_CIDX_GDTSSTRANSMITTER

Purpose	Index for config parameter FrIfGdTssTransmitter.
Value	34U



5.3.2.2.175. FR_CIDX_GDWAKEUPRXIDLE

Purpose	Index for config parameter FrIfGdWakeUpRxIdle.
Value	35U

5.3.2.2.176. FR_CIDX_GDWAKEUPRXLOW

Purpose	Index for config parameter FrIfGdWakeUpRxLow.
Value	36U

5.3.2.2.177. FR_CIDX_GDWAKEUPRXWINDOW

Purpose	Index for config parameter FrIfGdWakeUpRxWindow.
Value	9U

5.3.2.2.178. FR_CIDX_GDWAKEUPTXACTIVE

Purpose	Index for config parameter FrIfGdWakeUpTxActive.
Value	37U

5.3.2.2.179. FR_CIDX_GDWAKEUPTXIDLE

Purpose	Index for config parameter FrIfGdWakeUpTxIdle.
Value	38U

5.3.2.2.180. FR_CIDX_GLISTENNOISE

Purpose	Index for config parameter FrIfGListenNoise.
Value	19U

5.3.2.2.181. FR_CIDX_GMACROPERCYCLE

Purpose	Index for config parameter FrIfGMacroPerCycle.
----------------	--



Value	3U
--------------	----

5.3.2.2.182. FR_CIDX_GMAXWITHOUTCLOCKCORRECTFATAL

Purpose	Index for config parameter FrIfGMaxWithoutClockCorrectFatal.
Value	20U

5.3.2.2.183. FR_CIDX_GMAXWITHOUTCLOCKCORRECTPASSIVE

Purpose	Index for config parameter FrIfGMaxWithoutClockCorrectPassive.
Value	21U

5.3.2.2.184. FR_CIDX_GNETWORKMANAGEMENTVECTORLENGTH

Purpose	Index for config parameter FrIfGNetworkManagementVectorLength.
Value	22U

5.3.2.2.185. FR_CIDX_GNUMBEROFGMINISLOTS

Purpose	Index for config parameter FrIfGNumberOfMinislots.
Value	5U

5.3.2.2.186. FR_CIDX_GNUMBEROFGSTATICSLOTS

Purpose	Index for config parameter FrIfGNumberOfStaticSlots.
Value	6U

5.3.2.2.187. FR_CIDX_GPAYLOADLENGTHSTATIC

Purpose	Index for config parameter FrIfGPayloadLengthStatic.
Value	23U

**5.3.2.2.188. FR_CIDX_GSYNCFRAMEIDCOUNTMAX**

Purpose	Index for config parameter FrIfGSyncFrameIDCountMax.
Value	24U

5.3.2.2.189. FR_CIDX_PALLOWHALTDUETOCLOCK

Purpose	Index for config parameter FrPAllowHaltDueToClock.
Value	55U

5.3.2.2.190. FR_CIDX_PALLOWPASSIVETOACTIVE

Purpose	Index for config parameter FrPAllowPassiveToActive.
Value	39U

5.3.2.2.191. FR_CIDX_PCHANNELS

Purpose	Index for config parameter FrPChannels.
Value	40U

5.3.2.2.192. FR_CIDX_PCLUSTERDRIFTDAMPING

Purpose	Index for config parameter FrPClusterDriftDamping.
Value	41U

5.3.2.2.193. FR_CIDX_PDACCEPTEDSTARTUPRANGE

Purpose	Index for config parameter FrPdAcceptedStartupRange.
Value	16U

5.3.2.2.194. FR_CIDX_PDECODINGCORRECTION

Purpose	Index for config parameter FrPDecodingCorrection.
----------------	---



Value	42U
--------------	-----

5.3.2.2.195. FR_CIDX_PDELAYCOMPENSATIONA

Purpose	Index for config parameter FrPDelayCompensationA.
Value	43U

5.3.2.2.196. FR_CIDX_PDELAYCOMPENSATIONB

Purpose	Index for config parameter FrPDelayCompensationB.
Value	44U

5.3.2.2.197. FR_CIDX_PDLISTENTIMEOUT

Purpose	Index for config parameter FrPdListenTimeout.
Value	2U

5.3.2.2.198. FR_CIDX_PDMICROTICK

Purpose	Index for config parameter FrPdMicrotick.
Value	53U

5.3.2.2.199. FR_CIDX_PEXTERNALSYNC

Purpose	Index for config parameter FrPExternalSync.
Value	56U

5.3.2.2.200. FR_CIDX_PFALLBACKINTERNAL

Purpose	Index for config parameter FrPFallBackInternal.
Value	57U



5.3.2.2.201. FR_CIDX_PKEYSLOTID

Purpose	Index for config parameter FrPKeySlotId.
Value	10U

5.3.2.2.202. FR_CIDX_PKEYSLOTONLYENABLED

Purpose	Index for config parameter FrPKeySlotOnlyEnabled.
Value	58U

5.3.2.2.203. FR_CIDX_PKEYSLOTUSEDFORSTARTUP

Purpose	Index for config parameter FrPKeySlotUsedForStartup.
Value	59U

5.3.2.2.204. FR_CIDX_PKEYSLOTUSEDFORSYNC

Purpose	Index for config parameter FrPKeySlotUsedForSync.
Value	60U

5.3.2.2.205. FR_CIDX_PLATESTTX

Purpose	Index for config parameter FrPLatestTx.
Value	11U

5.3.2.2.206. FR_CIDX_PMACROINITIALOFFSETA

Purpose	Index for config parameter FrPMacroInitialOffsetA.
Value	45U

5.3.2.2.207. FR_CIDX_PMACROINITIALOFFSETB

Purpose	Index for config parameter FrPMacroInitialOffsetB.
----------------	--



Value	46U
--------------	-----

5.3.2.2.208. FR_CIDX_PMICROINITIALOFFSETA

Purpose	Index for config parameter FrPMicroInitialOffsetA.
Value	47U

5.3.2.2.209. FR_CIDX_PMICROINITIALOFFSETB

Purpose	Index for config parameter FrPMicroInitialOffsetB.
Value	48U

5.3.2.2.210. FR_CIDX_PMICROPERCYCLE

Purpose	Index for config parameter FrPMicroPerCycle.
Value	1U

5.3.2.2.211. FR_CIDX_PNMVECTOREARLYUPDATE

Purpose	Index for config parameter FrPNmVectorEarlyUpdate.
Value	61U

5.3.2.2.212. FR_CIDX_POFFSETCORRECTIONOUT

Purpose	Index for config parameter FrPOffsetCorrectionOut.
Value	12U

5.3.2.2.213. FR_CIDX_POFFSETCORRECTIONSTART

Purpose	Index for config parameter FrPOffsetCorrectionStart.
Value	13U



5.3.2.2.214. FR_CIDX_PPAYLOADLENGTHDYNMAX

Purpose	Index for config parameter FrPPayloadLengthDynMax.
Value	49U

5.3.2.2.215. FR_CIDX_PRATECORRECTIONOUT

Purpose	Index for config parameter FrPRateCorrectionOut.
Value	14U

5.3.2.2.216. FR_CIDX_PSAMPLESPERMICROTICK

Purpose	Index for config parameter FrPSamplesPerMicrotick.
Value	50U

5.3.2.2.217. FR_CIDX_PSECONDKEYSLOTID

Purpose	Index for config parameter FrPSecondKeySlotId.
Value	15U

5.3.2.2.218. FR_CIDX_PTWOKEYSLOTMODE

Purpose	Index for config parameter FrPTwoKeySlotMode.
Value	62U

5.3.2.2.219. FR_CIDX_PWAKEUPCHANNEL

Purpose	Index for config parameter FrPWakeupChannel.
Value	51U

5.3.2.2.220. FR_CIDX_PWAKEUPPATTERN

Purpose	Index for config parameter FrPWakeupPattern.
----------------	--



Value	52U
--------------	-----

5.3.2.2.221. FR_SLOTMODE_SINGLE

Purpose	map FR_SLOTMODE_SINGLE to FR_SLOTMODE_KEYSLOT for ASR3.x compliance
Value	FR_SLOTMODE_KEYSLOT

5.3.2.2.222. HIGH_BYTE_FIRST

Purpose	identifier for 'high byte first'
Value	0U

5.3.2.2.223. LOW_BYTE_FIRST

Purpose	cpu identifier for 8-bit CPUs
Value	1U

5.3.2.2.224. LSB_FIRST

Purpose	identifier for 'little endian'
Value	1U

5.3.2.2.225. MSB_FIRST

Purpose	identifier for 'high byte first'
Value	0U

5.3.2.2.226. NTFRSLT_E_ABORT

Purpose	Notification code: Error notification.
Value	0x09U



Description	► Flow control (FC) N_PDU with FlowStatus = ABORT received. It indicates an abort of a transmission. A possible reason for this is that the receiver is currently busy and can not take the request at that point in time. This value can be issued to the service user on the sender side.
--------------------	--

5.3.2.2.227. NTFRSLT_E_CANCELATION_NOT_OK

Purpose	Request cancellation has not been executed.
Value	0x0CU
Description	Due to an internal error the requested cancelation has not been executed. This will happen e.g., if the to be canceled transmission has been executed already.

5.3.2.2.228. NTFRSLT_E_CANCELATION_OK

Purpose	Requested cancellation has been executed.
Value	0x0BU

5.3.2.2.229. NTFRSLT_E_INVALID_FS

Purpose	Notification code: Error notification.
Value	0x06U
Description	► invalid or unknown FlowStatus value has been received in a flow controll (FC) N_PDU This value can be issued to the service user on the sender side only.

5.3.2.2.230. NTFRSLT_E_NOT_OK

Purpose	Notification code: Error notification.
Value	0x01U

5.3.2.2.231. NTFRSLT_E_NO_BUFFER

Purpose	Notification code: Error notification.
----------------	--



Value	0x0AU
Description	<ul style="list-style-type: none"> ➤ flow control (FC) N_PDU with FlowStatus = OVFLW received. It indicates that the buffer on the receiver side of a segmented message transmission cannot store the number of bytes specified by the FirstFrame DataLength (FF_DL) parameter in the FirstFFrame and therefore the transmission of the segmented message was aborted. ➤ no buffer within the TP available to transmit the segmented I-PDU. <p>This value can be issued to the service user on both the sender and receiver side.</p>

5.3.2.2.232. NTFRSLT_E_PARAMETER_NOT_OK

Purpose	Parameter value is not ok.
Value	0x0EU

5.3.2.2.233. NTFRSLT_E_RX_ON

Purpose	Reception ongoing.
Value	0x0FU

5.3.2.2.234. NTFRSLT_E_TIMEOUT_A

Purpose	Notification code: Error notification.
Value	0x02U
Description	<ul style="list-style-type: none"> ➤ timer N_Ar/N_As (according to ISO specification [ISONM]) has passed its time-out value N_Asmax/N_Armax. <p>This value can be issued to service user on both the sender and receiver side.</p>

5.3.2.2.235. NTFRSLT_E_TIMEOUT_BS

Purpose	Notification code: Error notification.
Value	0x03U
Description	<ul style="list-style-type: none"> ➤ timer N_Bs has passed its time-out value N_Bsmax (according to ISO specification [ISONM]).



	This value can be issued to the service user on the sender side only
--	--

5.3.2.2.236. NTFRSLT_E_TIMEOUT_CR

Purpose	Notification code: Error notification.
Value	0x04U
Description	<ul style="list-style-type: none"> ▶ timer N_CR has passed its time-out value N_CRmax <p>This value can be issued to the service user on the receiver side only</p>

5.3.2.2.237. NTFRSLT_E_UNEXP_PDU

Purpose	Notification code: Error notification.
Value	0x07U
Description	<ul style="list-style-type: none"> ▶ unexpected protocol data unit received <p>This value can be issued to the service user on both the sender and receiver side.</p>

5.3.2.2.238. NTFRSLT_E_VALUE_NOT_OK

Purpose	Parameter value is in invalid range.
Value	0x10U

5.3.2.2.239. NTFRSLT_E_WFT_OVRN

Purpose	Notification code: Error notification.
Value	0x08U
Description	<ul style="list-style-type: none"> ▶ flow control WAIT fram that exceeds the maximum counter N_WFTmax received. <p>This value can be issued to the service user on the sender side.</p>

5.3.2.2.240. NTFRSLT_E_WRONG_SN

Purpose	Notification code: Error notification.
----------------	--



Value	0x05U
Description	<ul style="list-style-type: none"> ➤ unexpected sequence number (PCI.SN) value received <p>This value can be issued to the service user on the receiver side only</p>

5.3.2.2.241. NTFRSLT_OK

Purpose	Vendor API infix.
Value	0x00U
Description	<p>Left empty as this header does not belong to any module Notification code: Action has been successfully finished This is used when:</p> <ul style="list-style-type: none"> ➤ message sent out (in case of confirmation) ➤ message received (in case if indication)

5.3.2.2.242. NTFRSLT_PARAMETER_OK

Purpose	Parameter value is not ok.
Value	0x0DU

5.3.2.2.243. NULL_PTR

Purpose	abstraction of the null pointer constant
Value	((void *)0)

5.3.2.2.244. PLATFORM_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	M4_REL_VERSION_MAJOR'U'

5.3.2.2.245. PLATFORM_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
----------------	--------------------------------



Value	M4_REL_VERSION_MINOR`'U``
--------------	---------------------------

5.3.2.2.246. PLATFORM_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	M4_REL_VERSION_PATCH`'U``

5.3.2.2.247. PLATFORM_MODULE_ID

Purpose	AUTOSAR module identification.
Value	199U

5.3.2.2.248. PLATFORM_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	2U

5.3.2.2.249. PLATFORM_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	5U

5.3.2.2.250. PLATFORM_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	0U

5.3.2.2.251. PLATFORM_VENDOR_API_INFIX

Purpose	Vendor API infix.
Value	



Description	Left empty as this header does not belong to any module
--------------------	---

5.3.2.2.252. PLATFORM_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	M4_VENDOR_ID`U`

5.3.2.2.253. PORT_MIRROR_DISABLED

Purpose	Constant for value port mirroring disabled of EthSwt_PortMirrorStateType.
Value	0U

5.3.2.2.254. PORT_MIRROR_ENABLED

Purpose	Constant for value port mirroring enabled of EthSwt_PortMirrorStateType.
Value	1U

5.3.2.2.255. REMOVE_FROM_FILTER

Purpose	Constant for value REMOVE_FROM_FILTER of Eth_FilterActionType (Compatibility with AUTOSAR 4.1.0).
Value	1U

5.3.2.2.256. SINT16_C

Purpose	Macro to define a constant of type sint16.
Value	((sint16)(x))

5.3.2.2.257. SINT32_C

Purpose	Macro to define a constant of type sint32.
----------------	--



Value	(x ## L)
--------------	----------

5.3.2.2.258. SINT8_C

Purpose	Macro to define a constant of type sint8.
Value	((sint8)(x))

5.3.2.2.259. SOAD_MEAS_ALL

Purpose	Constant for the value of Measurement index that represents all measurement index-es.
Value	0xFFU

5.3.2.2.260. SOAD_MEAS_DROP_TCP

Purpose	Constant for the value of Measurement index for dropped PDUs caused by invalid destination TCP-Port.
Value	1U

5.3.2.2.261. SOAD_MEAS_DROP_UDP

Purpose	Constant for the value of Measurement index for dropped PDUs caused by invalid destination UDP-Port.
Value	2U

5.3.2.2.262. STATIC

Purpose	Vendor API infix.
Value	static
Description	Left empty as this header does not belong to any module definition of an automatic memory class To be used for local non static variables definition of an type-definition memory class To be used within type definitions only abstraction of compiler keyword 'static' values: 'static' or empty



5.3.2.2.263. STD_ACTIVE

Purpose	Autosar logical state 'active'.
Value	1U

5.3.2.2.264. STD_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.3.2.2.265. STD_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.3.2.2.266. STD_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.3.2.2.267. STD_HIGH

Purpose	physical state 5V or 3.3V
Value	1U

5.3.2.2.268. STD_IDLE

Purpose	Autosar logical state 'idle'.
Value	0U

5.3.2.2.269. STD_LOW

Purpose	physical state 0V
----------------	-------------------



Value	0U
--------------	----

5.3.2.2.270. STD_MODULE_ID

Purpose	AUTOSAR module identification.
Value	197U
Description	Left empty as this header does not belong to any module

5.3.2.2.271. STD_OFF

Purpose	Autosar definition for 'off'.
Value	0U

5.3.2.2.272. STD_ON

Purpose	Autosar definition for 'on'.
Value	1U

5.3.2.2.273. STD_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	1U

5.3.2.2.274. STD_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	3U

5.3.2.2.275. STD_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
----------------	-------------------------------



Value	0U
--------------	----

5.3.2.2.276. STD_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.3.2.2.277. STD_VERSION_INFO_TYPE_DEFINED

Purpose	Guard for definition of Std_VersionInfoType .
Value	STD_OFF

5.3.2.2.278. TCPIP_MEAS_ALL

Purpose	Constant for the value of Measurement index that represents all measurement index- es.
Value	0xFFU

5.3.2.2.279. TCPIP_MEAS_DROP_ARP

Purpose	Constant for the value of Measurement index for dropped ARP entry add requests.
Value	0x82U

5.3.2.2.280. TCPIP_MEAS_DROP_ICMPV6

Purpose	Constant for the value of Measurement index for dropped PDUs caused by invalid ICMPv6 packets.
Value	0x80U

5.3.2.2.281. TCPIP_MEAS_DROP_IPV4

Purpose	Constant for the value of Measurement index for dropped PDUs caused by invalid IPv4 address.
----------------	---



Value	3U
--------------	----

5.3.2.2.282. TCPIP_MEAS_DROP_IPV6

Purpose	Constant for the value of Measurement index for dropped PDUs caused by invalid IPv6 address.
Value	4U

5.3.2.2.283. TCPIP_MEAS_DROP_TCP

Purpose	Constant for the value of Measurement index for dropped PDUs caused by invalid destination TCP-Port.
Value	1U

5.3.2.2.284. TCPIP_MEAS_DROP_UDP

Purpose	Constant for the value of Measurement index for dropped PDUs caused by invalid destination UDP-Port.
Value	2U

5.3.2.2.285. TCPIP_MEAS_REPLACED_ARP

Purpose	Constant for the value of Measurement index for replaced ARP entries.
Value	0x81U

5.3.2.2.286. TPPARAMETER_BC

Purpose	TP parameter literal of ASR3.2 for the BandwidthControl parameter used in FlexRay transport protocol module.
Value	TP_BC

5.3.2.2.287. TPPARAMETER_BS

Purpose	TP parameter literal of ASR3.2 for the Block size parameter including RfC 58315.
----------------	--



Value	TP_BS
--------------	-------

5.3.2.2.288. TPPARAMETER_STMIN

Purpose	TP parameter literal of ASR3.2 for the ST min parameter.
Value	TP_STMIN

5.3.2.2.289. TPPARAMTERE_BS

Purpose	TP parameter literal of ASR3.2 for the Block size parameter.
Value	TP_BS

5.3.2.2.290. TRUE

Purpose	true value for boolean type
Value	1U

5.3.2.2.291. TS_ADDITIONAL_PADDING_BYTES_INSERTED

Purpose	Check whether additional padding is inserted.
Value	((\ (sizeof(struct {uint8 nMember;}) != sizeof(uint8)) \ (sizeof(struct {uint16 nMember;}) != sizeof(uint16)) \ (sizeof(struct {uint32 nMember;}) != sizeof(uint32)) \) != 0U)
Description	This macro checks whether additional padding bytes are inserted for structures solely consisting of uint8/uint16/uint32 members

5.3.2.2.292. TS_ALIGNMENT_TS_MaxAlignedType

Purpose	Alignment constraints for each derived type (generated).
Value	4U
Description	Platforms Setting: It is derived from the parameters in the Platforms plugin by 'Derivedtypes.[derived type].Mapping:[base type]' and 'Basetypes.[C type].Alignment:1 2 4 8' (C type is the base type without signed or unsigned specifier).



5.3.2.2.293. TS_ALIGNMENT_struct

Purpose	Default alignment constraints for each complex type (generated).
Value	2U
Description	Platforms Setting: It is derived from the parameter in the Platforms plugin by 'Complextypes.[complex type].Alignment:1 1->1,2'

5.3.2.2.294. TS_ALIGNMENT_struct_0

Purpose	Value of each alignment constraint for each complex type (generated).
Value	1U
Description	Platforms Setting: It is set in the Platforms plugin by setting the parameter 'Complextypes.[complex type].Alignment:1->1,2'.

5.3.2.2.295. TS_ALIGNMENT_struct_NUM_THRESHOLDS

Purpose	Number of alignment constraints for each complex type (generated).
Value	2U
Description	Platforms Setting: It is set in the Platforms plugin by setting the parameter 'Complextypes.[complex type].Alignment:1 1->1,2'.

5.3.2.2.296. TS_ALIGNMENT_struct_THRESHOLD_0

Purpose	Threshold of each alignment constraint for each complex type (generated).
Value	1U
Description	Platforms Setting: It is set in the Platforms plugin by setting the parameter 'Complextypes.[complex type].Alignment:1->1,2'.

5.3.2.2.297. TS_ALIGNMENT_uint32

Purpose	Alignment constraints for each AUTOSAR type (generated).
Value	4U
Description	Platforms Setting: It is derived from the parameters in the Platforms plugin by 'Basetypes.[AUTOSAR type].Mapping:[base type]' and 'Basetypes.[C type].Alignment:1 2 4 8' (C type is the base type without signed or unsigned specifier).



5.3.2.2.298. TS_ALIGNOF

Purpose	Get alignment of type within structures.
Value	TS_OFFSETOF(struct {uint8 nMember1; type nMember2;}, nMember2)
Description	This macro retrieves the structure alignment requirements of a given type

5.3.2.2.299. TS_ARCH_DERIVATE

Purpose	The target derivative name must be provided externally.
Value	

5.3.2.2.300. TS_ARCH_FAMILY

Purpose	The target architecture family name must be provided externally.
Value	

5.3.2.2.301. TS_ARM

Purpose	constant for ARM target architecture family
Value	31U

5.3.2.2.302. TS_ARM64

Purpose	constant for ARM64 target architecture family
Value	43U

5.3.2.2.303. TS_ARRAYALIGNOF

Purpose	Get alignment of an array of type type within structures.
Value	TS_OFFSETOF(struct {uint8 nMember1; type nMember2[elements];}, nMember2)
Description	This macro retrieves the structure alignment requirements of an array containing elements elements of type type



5.3.2.2.304. TS_AtomicAssign16

Purpose	Default implementation for assigning the 16 bit entity from to to in an atomic fashion if not provided by TSAutosar_Cfg.h.
Value	TS_AtomicAssignGeneric((to), (from))
Description	<p>This macro assigns the 16 bit entity from to to in an atomic fashion</p> <p>The parameters from and to thus have to be of type uint16 or sint16.</p> <p>Platforms Setting: The implementation depends on the parameters in the Platforms plugin by 'Basetypes.[AUTOSAR type].Mapping:[base type]' and 'Basetypes.[C type].-AtomicAccess:true false' (C type is the base type without signed or unsigned specifier).</p>

5.3.2.2.305. TS_AtomicAssign32

Purpose	Default implementation for assigning the 32 bit entity from to to in an atomic fashion if not provided by TSAutosar_Cfg.h.
Value	TS_AtomicAssignGeneric((to), (from))
Description	<p>This macro assigns the 32 bit entity from to to in an atomic fashion</p> <p>The parameters from and to thus have to be of type uint32 or sint32.</p> <p>Platforms Setting: The implementation depends on the parameters in the Platforms plugin by 'Basetypes.[AUTOSAR type].Mapping:[base type]' and 'Basetypes.[C type].-AtomicAccess:true false' (C type is the base type without signed or unsigned specifier).</p>

5.3.2.2.306. TS_AtomicAssign8

Purpose	Default implementation for assigning the 8 bit entity from to to in an atomic fashion if not provided by TSAutosar_Cfg.h.
Value	((to) = (from))
Description	<p>This macro assigns the 8 bit entity from to to in an atomic fashion</p> <p>The parameters from and to thus have to be of type uint8 or sint8.</p>



Platforms Setting: The implementation depends on the parameters in the Platforms plugin by 'Basetypes.[AUTOSAR type].Mapping:[base type]' and 'Basetypes.[C type].-AtomicAccess:true|false' (C type is the base type without signed or unsigned specifier).

5.3.2.2.307. TS_AtomicAssignGeneric

Purpose	Assigns an arbitrary entity from to to in an atomic fashion.
Value	<code>do { \ TS_IntStatusType savedIntStatus = TS_IntDisable(); \ (to) = (from); \ TS_- IntRestore(savedIntStatus); \ } while(0)</code>
Description	<p>This macro assigns the entity from (which can be of arbitrary size) to to in an atomic fashion</p> <p>The parameters from and to have to be of a type that can be assigned via the C assignment operator.</p>

5.3.2.2.308. TS_C16X

Purpose	constant for C16X target architecture family
Value	1U

5.3.2.2.309. TS_CAST

Purpose	Cast expr to the type given by type and bits.
Value	<code>((type##bits)(expr))</code>
Description	<p>usage: e.g: <code>TS_CAST(uint, 8, ((length) >> 8u))</code></p> <p>This macro is used to suppress excessive warnings from some compilers, while keeping the code readable and efficient for well-behaving compilers.</p> <p>Note: Using the default version with the C cast should be sufficient for all compilers. Unless your compiler misbehaves, don't override the default definition.</p> <p>Background:</p> <p>There are many ways for a compiler to do sign/zero extensions: shift back & forth, AND with a constant, move between registers of different sizes, explicit sign/unsigned-extend instructions, bit-insertion instructions, ...</p>



Which route is the cheapest, will depend on the architecture, size of the masking constant, the location of the variable within the register and maybe even on other constants in the program, or the constants currently residing in registers.

Your compiler may select any - and may not even find the cheapest, depending on strategy, phase-ordering within the compiler, abortion due to compile time limits, ...

Add the commutative property of AND, throw in adjacent shifts that may be joined with shifts in the same direction, and the peep-hole optimizer will have a hard time identifying all possible expression tree patterns to eliminate the redundant sign/zero extension with a given compile time budget.

Now, if you still want to proceed, you may need to insert one of the following variants in the Compiler.h for your compiler/architecture, in order to appease your specific compiler - adding Deviation MISRAC2012-2 to the set of deviations:

```
define TS\_CAST\(type, bits, expr\) ((type#bits)((expr) & ((type#bits)(~((type#bits)0u))))))

define TS\_CAST\(type, bits, expr\) ((type#bits)((expr) & ((type#bits)-1)))
```

5.3.2.2.310. TS_CFG_OFFSET_TYPE_MAX

Purpose	Maximum value for a TS_CfgOffsetType.
Value	UINT32_C(0xFFFFFFFF)
Description	This macro defines the maximum value for a TS_CfgOffsetType

5.3.2.2.311. TS_CFG_OFFSET_TYPE_MIN

Purpose	Minimum value for a TS_CfgOffsetType.
Value	UINT32_C(0x00000000)
Description	This macro defines the minimum value for a TS_CfgOffsetType

5.3.2.2.312. TS_CHECKEDGETCFG

Purpose	Get a non-const reference to a post-build config element.
Value	((offset) == 0U) ? \ (NULL_PTR) : \ TS_UNCHECKEDGETCFG(cfgbase, reftype, module, offset))
Description	This macro retrieves a non-const pointer to a post-build config element



5.3.2.2.313. TS_CHECKEDGETCONSTCFG

Purpose	Get a const reference to a post-build config element.
Value	((((offset) == 0U) ? \ ((CONSTP2CONST(reftype, AUTOMATIC, module ## _AP-PL_CONST)) 0) : \ TS_UNCHECKEDGETCONSTCFG(cfgbase, reftype, module, offset))
Description	This macro retrieves a const pointer to a post-build config element

5.3.2.2.314. TS_CHECKEDGETCONSTVAR

Purpose	Get a const reference to a post-build RAM element.
Value	((((offset) == TS_VAR_OFFSET_TYPE_MAX) ? \ ((CONSTP2VAR(reftype, AUTOMATIC, module ## _VAR_NOINIT)) 0) : \ TS_UNCHECKEDGETCONSTVAR(rambase, reftype, module, offset))
Description	This macro retrieves a const pointer to a post-build RAM element

5.3.2.2.315. TS_CHECKEDGETVAR

Purpose	Get a non-const reference to a post-build RAM element.
Value	((((offset) == TS_VAR_OFFSET_TYPE_MAX) ? \ ((P2VAR(reftype, AUTOMATIC, module ## _VAR_NOINIT)) 0) : \ TS_UNCHECKEDGETVAR(rambase, reftype, module, offset))
Description	This macro retrieves a non-const pointer to a post-build RAM element

5.3.2.2.316. TS_CONSTREF2CFG

Purpose	abstraction for references in ROM referencing config elements
Value	CONST(TS_CfgOffsetType , TYPEDEF)
Description	<p>This macro abstracts the declaration and definition of references in ROM referencing config elements</p> <p>The reference itself is not modifiable (read only). The references's target is not modifiable (read only).</p>



5.3.2.2.317. TS_CONSTREF2VAR

Purpose	abstraction for references in ROM referencing PB RAM elements
Value	CONST(TS_VarOffsetType , TYPEDEF)
Description	<p>This macro abstracts the declaration and definition of references in ROM referencing post build RAM elements</p> <p>The reference itself is not modifiable (read only). The references's target is modifiable.</p>

5.3.2.2.318. TS_CORTEXM

Purpose	constant for CORTEXM target architecture family
Value	40U

5.3.2.2.319. TS_DEPRECATED_FUNCTION

Purpose	Tag a function as deprecated.
Value	<code>__attribute__ ((deprecated))</code>
Description	<p>This macro shall be used to tag a function as deprecated.</p> <p>Example use:</p> <pre>STATIC FUNC(void, EBTEST_CODE) TestDeprecatedFunction(void) TS_DEPRECATED_FUNCTION;</pre> <p>This yields the following warning for every <code>_USE_</code> of the function during compilation:</p> <pre>appl.c: In function 'TestMain': appl.c:62: warning: 'TestDeprecatedFunction' is deprecated (declared at appl.c:29)</pre>

5.3.2.2.320. TS_DEPRECATED_MACRO

Purpose	Tag a macro as deprecated.
Value	<code>TS_DOPRAGMA(Macro_ ## x ## _is_deprecated)</code>



Description	<p>This macro shall be used to tag a macro as deprecated.</p> <p>Example use:</p> <pre>define DEPRECATED_MACRO(a,b) \ do \{ \ TS_DEPRECATED_MACRO(DEPRECATED_MACRO) \ (a) = (b); \ } while(0)</pre> <p>This yields the following warning for every _USE_ of the macro during compilation:</p> <p>appl.c: In function 'TestDeprecatedMacro': appl.c:99: warning: ignoring pragma Macro_DEPRECATED_MACRO_is_deprecated</p> <p>Note that this warning that the pragma is ignored is the best we can do, since the only means gcc provides to explicitly trigger warning messages are error and warning, which both cannot be used in macros. - Note further that pragma message is not supported by gcc (although the docu states that is is).</p>
--------------------	---

5.3.2.2.321. TS_DEPRECATED_TYPEDEF

Purpose	Tag a typedef as deprecated.
Value	<code>__attribute__ ((deprecated))</code>
Description	<p>This macro shall be used to tag a typedef as deprecated.</p> <p>Example use:</p> <pre>typedef struct { uint8 member0; uint8 member1; } deprecatedType TS_DEPRECATED_TYPEDEF;</pre> <p>This yields the following warning for every _USE_ of the typedef during compilation:</p> <p>appl.c: In function 'TestDeprecatedTypedef': appl.c:84: warning: 'deprecatedType' is deprecated</p>

5.3.2.2.322. TS_DEPRECATED_VARIABLE

Purpose	Tag a variable as deprecated.
Value	<code>__attribute__ ((deprecated))</code>
Description	<p>This macro shall be used to tag a variable as deprecated.</p> <p>Example use:</p> <pre>uint8 deprecatedVariable TS_DEPRECATED_VARIABLE;</pre>



This yields the following warning for every `_USE_` of the variable during compilation:

`appl.c: In function 'TestDeprecatedVariable': appl.c:91: warning: 'deprecatedVariable' is deprecated (declared at appl.c:48)`

5.3.2.2.323. TS_DOPRAGMA

Purpose	Helper macro for other deprecation macros.
Value	<code>_Pragma (#x)</code>

5.3.2.2.324. TS_DSPIC33

Purpose	constant for DSPIC33 target architecture family
Value	35U

5.3.2.2.325. TS_DefMaxAlignedByteArray

Purpose	Define a byte array with maximum alignment.
Value	<code>TS_DefMaxAlignedByteArray_Hlp(identifier, module, _ ## memclass, numberOfBytes)</code>
Description	This macro defines identifier as an array of at least <code>numberOfBytes</code> bytes in a way that this array is aligned according to the most stringent alignment requirement of any simple data type available on the respective platform.

5.3.2.2.326. TS_DefMaxAlignedByteArray_Hlp

Purpose	Helper macro for <code>TS_DefMaxAlignedByteArray</code> .
Value	<code>VAR(TS_MaxAlignedType, module ## memclass) \ identifier[((numberOfBytes) - 1) + sizeof(TS_MaxAlignedType) / sizeof(TS_MaxAlignedType)]</code>

5.3.2.2.327. TS_EASYCAN

Purpose	constant for EASYCAN target architecture family
----------------	---



Value	23U
--------------	-----

5.3.2.2.328. TS_EBX1XX

Purpose	constant for EBX1XX target architecture family
Value	30U

5.3.2.2.329. TS_EXCALIBUR

Purpose	constant for EXCALIBUR target architecture family
Value	3U

5.3.2.2.330. TS_F2MC16L

Purpose	constant for F2MC16L target architecture family
Value	4U

5.3.2.2.331. TS_FATBOOL

Purpose	Create a fat-boolean of the given type for IF-conversion, using TS_SELECT. Be sure to assign to temporary variables to avoid double-evaluation.
Value	$((\text{type}0) - (\text{type})(\text{cond}))$

5.3.2.2.332. TS_FATBOOL_LT

Purpose	Create a fat-boolean of the given type for IF-conversion, using TS_SELECT. Be sure to assign to temporary variables to avoid double-evaluation.
Value	$((\text{inttype})((\text{TS_SINT}(\text{inttype}))((\text{inttype})(\text{a}) - (\text{inttype})(\text{b}))) >> ((\text{sizeof}(\text{inttype}) << 3u) - 1u))$

5.3.2.2.333. TS_FCR4

Purpose	constant for FCR4 target architecture family
----------------	--



Value	32U
--------------	-----

5.3.2.2.334. TS_GetBit

Purpose	Extracts a bit from a variable of type OpType.
Value	((Result) = (OpType)(((OpType)(((OpType)*(Address)) >> (Bit))) & ((OpType)1u))
Description	This function returns the bit number Bit from the variable of type OpType pointed to by Address. The parameter OpType can either be uint8, uint16 or uint32.

5.3.2.2.335. TS_GetBitGroup

Purpose	Gets the level of a group of bits.
Value	((Result) = (OpType) (((OpType)(*(Address))) & (OpType)(Mask)) >> (Offset))
Description	This macro extracts the level of several consecutive bits indicated by Mask and Offset from the variable pointed to by Address. Preconditions:

5.3.2.2.336. TS_H8

Purpose	constant for H8 target architecture family
Value	6U

5.3.2.2.337. TS_HC08

Purpose	constant for HC08 target architecture family
Value	7U

5.3.2.2.338. TS_HC12

Purpose	constant for HC12 target architecture family
----------------	--



Value	8U
--------------	----

5.3.2.2.339. TS_IsBitSet

Purpose	Check if a bit is set in a variable of type OpType.
Value	((*(Address) & ((OpType)((OpType)1U<<(Bit)))) != 0U)
Description	This function returns non-zero if the bit number Bit from the variable of type OpType pointed to by Address is set. The parameter OpType can either be uint8, uint16 or uint32.

5.3.2.2.340. TS_LINUX

Purpose	constant for LINUX target architecture family
Value	36U

5.3.2.2.341. TS_M32C

Purpose	constant for M32C target architecture family
Value	9U

5.3.2.2.342. TS_M32R

Purpose	constant for M32R target architecture family
Value	25U

5.3.2.2.343. TS_MAKENULLREF2CFG

Purpose	Create an invalid post-build able reference to a config element.
Value	((TS_CfgOffsetType) 0U)
Description	This macro creates an invalid post-build able reference to the given config element.



5.3.2.2.344. TS_MAKENULLREF2VAR

Purpose	Create an invalid post-build able reference to a RAM element.
Value	TS_VAR_OFFSET_TYPE_MAX
Description	This macro creates a invalid post-build able reference to the given RAM element.

5.3.2.2.345. TS_MAKEREF2CFG

Purpose	Create a post-build able reference to a config element.
Value	((TS_CfgOffsetType) TS_OFFSETOF(TS_PB_CFG_LAYOUT_TYPE, member))
Description	This macro creates a post-build able reference to the given config element.

5.3.2.2.346. TS_MAKEREF2VAR

Purpose	Create a post-build able reference to a RAM element.
Value	(offset)
Description	This macro creates a post-build able reference to the given RAM element.

5.3.2.2.347. TS_MB91

Purpose	constant for MB91 target architecture family
Value	5U

5.3.2.2.348. TS_MB96

Purpose	constant for MB96 target architecture family
Value	28U

5.3.2.2.349. TS_MEMZERO_CUSTOM_OVERRIDE

Purpose	Macro for enabling overriding of the memory zeroing function by a custom implementation.
----------------	--



Value	STD_OFF
Description	<p>Range:</p> <ul style="list-style-type: none"> ➤ STD_ON (if configuration parameter 'Base/CustomOverrides/CustomOverride_MemBZero' enabled) ➤ STD_OFF (if configuration parameter 'Base/CustomOverrides/CustomOverride_MemBZero' disabled)

5.3.2.2.350. TS_MEMCMP_CUSTOM_OVERRIDE

Purpose	Macro for enabling overriding of the memory compare function by a custom implementation.
Value	STD_OFF
Description	<p>Range:</p> <ul style="list-style-type: none"> ➤ STD_ON (if configuration parameter 'Base/CustomOverrides/CustomOverride_MemCmp' enabled) ➤ STD_OFF (if configuration parameter 'Base/CustomOverrides/CustomOverride_MemCmp' disabled)

5.3.2.2.351. TS_MEMCPY_CUSTOM_OVERRIDE

Purpose	Macro for enabling overriding of the memory copy function by a custom implementation.
Value	STD_OFF
Description	<p>Range:</p> <ul style="list-style-type: none"> ➤ STD_ON (if configuration parameter 'Base/CustomOverrides/CustomOverride_MemCpy' enabled) ➤ STD_OFF (if configuration parameter 'Base/CustomOverrides/CustomOverride_MemCpy' disabled)

5.3.2.2.352. TS_MEMSET_CUSTOM_OVERRIDE

Purpose	Macro for enabling overriding of the memory set function by a custom implementation.
Value	STD_OFF



Description	<p>Range:</p> <ul style="list-style-type: none"> ▶ STD_ON (if configuration parameter 'Base/CustomOverrides/CustomOverride_MemSet' enabled) ▶ STD_OFF (if configuration parameter 'Base/CustomOverrides/CustomOverride_MemSet' disabled)
--------------------	--

5.3.2.2.353. TS_MERGED_COMPILE

Purpose	All source files are built separately by default.
Value	STD_OFF

5.3.2.2.354. TS_MOD_PRIV_DECL

Purpose	Use internal linkage of function declarations, if merged, fast build is used.
Value	static

5.3.2.2.355. TS_MOD_PRIV_DEFN

Purpose	Use internal linkage of function definitions, if merged, fast build is used.
Value	static

5.3.2.2.356. TS_MPC551X

Purpose	constant for MPC551X target architecture family
Value	22U

5.3.2.2.357. TS_MemBZero

Purpose	This macro maps to the CPU specific macro for filling memory bytes with zero, for 64bit CPU to TS_MemBZero64() .
Value	(TS_MemBZero64((dst), (len)))



5.3.2.2.358. TS_MemCmp

Purpose	This macro maps to the CPU specific macro for comparing memory bytes, for 64bit CPU to TS_MemCmp64() .
Value	(TS_MemCmp64((a),(b),(n)))

5.3.2.2.359. TS_MemCpy

Purpose	This macro maps to the CPU specific macro for copying memory bytes, for 64bit CPU to TS_MemCpy64() .
Value	(TS_MemCpy64((d),(s),(n)))

5.3.2.2.360. TS_MemSet

Purpose	This macro maps to the CPU specific macro for setting memory bytes, for 64bit CPU to TS_MemSet64() .
Value	(TS_MemSet64((dst),(val),(len)))

5.3.2.2.361. TS_NECK8K

Purpose	constant for NEC78K target architecture family
Value	10U

5.3.2.2.362. TS_NIOS2

Purpose	constant for NIOS2 target architecture family
Value	27U

5.3.2.2.363. TS_OFFSETOF

Purpose	Get the byte offset of a structure member.
Value	(offsetof(structure, member))
Description	This macro retrieves the byte offset of a given structure member from the start of the structure



5.3.2.2.364. TS_PA

Purpose	constant for PA target architecture family
Value	2U

5.3.2.2.365. TS_PARAM_UNUSED

Purpose	This macro can be used to avoid compiler warnings.
Value	((void) (x))

5.3.2.2.366. TS_PIC24H

Purpose	constant for PIC24H target architecture family
Value	26U

5.3.2.2.367. TS_PIKEOS

Purpose	constant for PIKEOS target architecture family
Value	33U

5.3.2.2.368. TS_PREVENTEMPTYTRANSLATIONUNIT

Purpose	Prevent empty translation unit.
Value	typedef void UNIQUE(TSPreventEmptyTranslationUnit, __LINE__-);
Description	This macro shall be used to prevent that a translation unit is empty.

5.3.2.2.369. TS_PROD_ERR_DISABLE

Purpose	definition of production error reporting defines - off
Value	2U
Description	The production error reporting could be configured to Dem, Det and off. TS_PROD_ERR_DISABLE is used if the production error reporting is switched off.



5.3.2.2.370. TS_PROD_ERR REP_TO DEM

Purpose	definition of production error reporting defines - report to Dem
Value	0U
Description	The production error reporting could be configured to Dem, Det and off. TS_PROD_ERR REP_TO DEM is used if the production error is reported towards the Dem.

5.3.2.2.371. TS_PROD_ERR REP_TO DET

Purpose	definition of production error reporting defines - report to Det
Value	1U
Description	The production error reporting could be configured to Dem, Det and off. TS_PROD_ERR REP_TO DET is used if the production error is reported towards the Det.

5.3.2.2.372. TS_R32C

Purpose	constant for R32C target architecture family
Value	21U

5.3.2.2.373. TS_REF2CFG

Purpose	abstraction for references in RAM referencing config elements
Value	VAR(TS_CfgOffsetType , TYPEDEF)
Description	This macro abstracts the declaration and definition of references in RAM referencing config elements The reference itself is modifiable. The references's target is not modifiable (read only).

5.3.2.2.374. TS_REF2VAR

Purpose	abstraction for references in RAM referencing PB RAM elements
Value	VAR(TS_VarOffsetType , TYPEDEF)
Description	This macro abstracts the declaration and definition of references in RAM referencing post build RAM elements



The reference itself is modifiable. The references's target is modifiable.

5.3.2.2.375. TS_RH850

Purpose	constant for RH850 target architecture family
Value	37U

5.3.2.2.376. TS_RL78

Purpose	constant for RL78 target architecture family
Value	38U

5.3.2.2.377. TS_RZ

Purpose	constant for RZ target architecture family
Value	39U

5.3.2.2.378. TS_S12X

Purpose	constant for S12X target architecture family
Value	11U

5.3.2.2.379. TS_S12Z

Purpose	constant for S12Z target architecture family
Value	34U

5.3.2.2.380. TS_SAF7780

Purpose	constant for SAF7780 target architecture family
Value	12U



5.3.2.2.381. TS_SELECT

Purpose	Portable conditional move, using a FATBOOL(TM) {0,~0} condition to select the correct input by & masking. Be sure to assign to temporary variables to avoid double-evaluation.
Value	((type)((type)((type)(true) & (type)(fat_bool)) (type)((type)(false) & (type)~(type)(fat_bool))))

5.3.2.2.382. TS_SH2

Purpose	constant for SH2 target architecture family
Value	13U

5.3.2.2.383. TS_SH4

Purpose	constant for SH4 target architecture family
Value	29U

5.3.2.2.384. TS_SIG_ARRAY_ALIGNMENT_MASK

Purpose	macro for mask for extraction of array alignment requirements
Value	UINT32_C(0x00000007)

5.3.2.2.385. TS_SIG_ARRAY_ALIGNMENT_SHIFT

Purpose	macro for shift for extraction of array alignment requirements
Value	UINT32_C(24)

5.3.2.2.386. TS_SIG_CHECK_ALIGNMENT_ARRAY

Purpose	macro to check whether the minimal alignment requirements of arrays that were assumed when generating the post build configuration match the machines minimal alignment requirements for arrays
----------------	---



Value	(TS_SIG_GET_ALIGNMENT(ARRAY, signature) == TS_ARRAYALIGNOF(uint8, 1))
--------------	---

5.3.2.2.387. TS_SIG_CHECK_ALIGNMENT_SINT16

Purpose	macro to check whether the alignment requirements of sint16 that were assumed when generating the post build configuration match the machines alignment requirements for sint16
Value	(TS_SIG_GET_ALIGNMENT(SINT16, signature) == TS_ALIGNOF(sint16))

5.3.2.2.388. TS_SIG_CHECK_ALIGNMENT_SINT32

Purpose	macro to check whether the alignment requirements of sint32 that were assumed when generating the post build configuration match the machines alignment requirements for sint32
Value	(TS_SIG_GET_ALIGNMENT(SINT32, signature) == TS_ALIGNOF(sint32))

5.3.2.2.389. TS_SIG_CHECK_ALIGNMENT_SINT8

Purpose	macro to check whether the alignment requirements of sint8 that were assumed when generating the post build configuration match the machines alignment requirements for sint8
Value	(TS_SIG_GET_ALIGNMENT(SINT8, signature) == TS_ALIGNOF(sint8))

5.3.2.2.390. TS_SIG_CHECK_ALIGNMENT_STRUCT

Purpose	macro to check whether the minimal alignment requirements of structures that were assumed when generating the post build configuration match the machines minimal alignment requirements for structures.
Value	(TS_SIG_GET_ALIGNMENT(STRUCT, signature) == TS_ALIGNOF(struct {uint8 nMember1; uint8 nMember2; \ uint8 nMember3; uint8 nMember4; \ uint8 nMember5; uint8 nMember6; \ uint8 nMember7; uint8 nMember8; }))
Description	Using two uint8 members here to address the fact that the TriCore EABI states that the minimal alignment requirement of structs of 1 byte size is only 1 byte (instead of 2 bytes minimal alignment like all struct of size > 1 byte)



5.3.2.2.391. TS_SIG_CHECK_ALIGNMENT_UINT16

Purpose	macro to check whether the alignment requirements of uint16 that were assumed when generating the post build configuration match the machines alignment requirements for uint16
Value	(TS_SIG_GET_ALIGNMENT(UINT16, signature) == TS_ALIGNOF(uint16))

5.3.2.2.392. TS_SIG_CHECK_ALIGNMENT_UINT32

Purpose	macro to check whether the alignment requirements of uint32 that were assumed when generating the post build configuration match the machines alignment requirements for uint32
Value	(TS_SIG_GET_ALIGNMENT(UINT32, signature) == TS_ALIGNOF(uint32))

5.3.2.2.393. TS_SIG_CHECK_ALIGNMENT_UINT8

Purpose	macro to check whether the alignment requirements of uint8 that were assumed when generating the post build configuration match the machines alignment requirements for uint8
Value	(TS_SIG_GET_ALIGNMENT(UINT8, signature) == TS_ALIGNOF(uint8))

5.3.2.2.394. TS_SIG_CHECK_ENDIANESS

Purpose	macro to check whether the endianness that was assumed when generating the post build configuration match the machines endianness
Value	((signature) & UINT32_C(0x80808080)) == UINT32_C(0x80000000))

5.3.2.2.395. TS_SIG_GET_ALIGNMENT

Purpose	macro to extract the alignment requirements for a specific type from the signature
Value	((uint32)(UINT32_C(1) << TS_SIG_GET_ALIGNMENT_HLP(type, signature)))

5.3.2.2.396. TS_SIG_GET_ALIGNMENT_HLP

Purpose	helper macro for extraction of alignment requirements
----------------	---



Value	TS_SIG_GET_ALIGNMENT_HLP_HLP(TS_SIG_## type, signature)
--------------	---

5.3.2.2.397. TS_SIG_GET_ALIGNMENT_HLP_HLP

Purpose	helper macro for extraction of alignment requirements
Value	((signature) >> prefixedtype ## _ALIGNMENT_SHIFT) & (uint32)prefixedtype ## _-ALIGNMENT_MASK)

5.3.2.2.398. TS_SIG_SINT16_ALIGNMENT_MASK

Purpose	macro for mask for extraction of 16bit signed integer alignment requirements
Value	UINT32_C(0x00000007)

5.3.2.2.399. TS_SIG_SINT16_ALIGNMENT_SHIFT

Purpose	macro for shift for extraction of 16bit signed integer alignment requirements
Value	UINT32_C(8)

5.3.2.2.400. TS_SIG_SINT32_ALIGNMENT_MASK

Purpose	macro for mask for extraction of 32bit signed integer alignment requirements
Value	UINT32_C(0x00000007)

5.3.2.2.401. TS_SIG_SINT32_ALIGNMENT_SHIFT

Purpose	macro for shift for extraction of 32bit signed integer alignment requirements
Value	UINT32_C(0)

5.3.2.2.402. TS_SIG_SINT8_ALIGNMENT_MASK

Purpose	macro for mask for extraction of 8bit signed integer alignment requirements
----------------	---



Value	UINT32_C(0x00000007)
--------------	----------------------

5.3.2.2.403. TS_SIG_SINT8_ALIGNMENT_SHIFT

Purpose	macro for shift for extraction of 8bit signed integer alignment requirements
Value	UINT32_C(16)

5.3.2.2.404. TS_SIG_STRUCT_ALIGNMENT_MASK

Purpose	macro for mask for extraction of struct alignment requirements
Value	UINT32_C(0x00000007)

5.3.2.2.405. TS_SIG_STRUCT_ALIGNMENT_SHIFT

Purpose	macro for shift for extraction of struct alignment requirements
Value	UINT32_C(27)

5.3.2.2.406. TS_SIG_UINT16_ALIGNMENT_MASK

Purpose	macro for mask for extraction of 16bit unsigned integer alignment requirements
Value	UINT32_C(0x00000007)

5.3.2.2.407. TS_SIG_UINT16_ALIGNMENT_SHIFT

Purpose	macro for shift for extraction of 16bit unsigned integer alignment requirements
Value	UINT32_C(11)

5.3.2.2.408. TS_SIG_UINT32_ALIGNMENT_MASK

Purpose	macro for mask for extraction of 32bit unsigned integer alignment requirements
Value	UINT32_C(0x00000007)



5.3.2.2.409. TS_SIG_UINT32_ALIGNMENT_SHIFT

Purpose	macro for shift for extraction of 32bit unsigned integer alignment requirements
Value	UINT32_C(3)

5.3.2.2.410. TS_SIG_UINT8_ALIGNMENT_MASK

Purpose	macro for mask for extraction of 8bit unsigned integer alignment requirements
Value	UINT32_C(0x00000007)

5.3.2.2.411. TS_SIG_UINT8_ALIGNMENT_SHIFT

Purpose	macro for shift for extraction of 8bit unsigned integer alignment requirements
Value	UINT32_C(19)

5.3.2.2.412. TS_SIZE_PduldType

Purpose	Size of AUTOSAR standard type PduldType.
Value	2U
Description	this value is fixed to 2U (see deviations of EcuC)

5.3.2.2.413. TS_SIZE_PduLengthType

Purpose	Size of AUTOSAR standard type PduLengthType.
Value	2U
Description	this value is fixed to 2U (see deviations of EcuC)

5.3.2.2.414. TS_SIZE_TS_MaxAlignedType

Purpose	Size of each derived type (generated).
Value	8U
Description	Platforms Setting: It is derived from the parameters in the Platforms plugin by 'Derivedtypes.[derived type].Mapping:[base type]' and 'Basetypes.[C type].Size:1 2 4 8' (C type is the base type without signed or unsigned specifier).



5.3.2.2.415. TS_SIZE_uint32

Purpose	Size of each AUTOSAR type (generated).
Value	4U
Description	Platforms Setting: It is derived from the parameters in the Platforms plugin by 'Basetypes.[AUTOSAR type].Mapping:[base type]' and 'Basetypes.[C type].Size:1 2 4 8' (C type is the base type without signed or unsigned specifier).

5.3.2.2.416. TS_ST30

Purpose	constant for ST30 target architecture family
Value	14U

5.3.2.2.417. TS_STATIC_ASSERT

Purpose	Report a static assertion (a.k.a. compile-time assert).
Value	<code>typedef uint8 TS_STATIC_ASSERT_EVAL(msg_id,__LINE__-) [(expr) ? 1 : -1]</code>
Description	<p>Use this macro to check a static compile time assertion as part of defensive programming. With this macro you can check</p> <ul style="list-style-type: none"> ▶ size of types and size of global variables ▶ alignment of global variables ▶ padding in structures <p>The macro cannot be used for non-static checks involving quantities which cannot be computed at compile time by the compiler.</p> <p>This macro can be used both at global and at local scope, but it should better be used at function level, right after the local variables, before the first statement of the scope.</p> <p>Note1: At global scope, there is potential to confuse tools like doxygen, which mistakes it as a function declaration, unless DOXYGEN_EXCLUDE_SYMBOLS += TS_STATIC_ASSERT is added to the respective make file. (e.g. asc_MODULE:)</p> <p>Note2: You have to use a trailing semicolon after TS_STATIC_ASSERT().</p> <p>Examples:</p> <p>Ensure the correct size of a data type:</p> <ul style="list-style-type: none"> ▶ <code>typedef uint8 Mod_IdHandleType;</code>



- ▶ TS_STATIC_ASSERT(sizeof(Mod_IdHandleType) == 1U, Incorrect_allocation_size)
- Ensure correct alignment of the data type (32-bit in this case):
- ▶ Mod_IdHandleType Mod_GlobalIdHandle;
 - ▶ TS_STATIC_ASSERT((&Mod_GlobalIdHandle & 3U) == 0U, Incorrect_alignment)

Technical background: If the expr evaluates to false, the macro defines an array type with negative array size. This is an invalid construct in ANSI C and leads to a compiler error. The identifier of the type is typically included in the compiler's error message for the user to see. It includes the string "TS_STATIC_ASSERT", the supplied msg_id and also the line number on which the macro [TS_STATIC_ASSERT\(\)](#) is placed. From this information the user can infer which static assert failed.

5.3.2.2.418. TS_STATIC_ASSERT_EVAL

Purpose	internal parameter evaluation helper macro to serve evaluated tokens to TS_STATIC_ASSERT_JOIN's ## operators
Value	TS_STATIC_ASSERT_JOIN(msg_id,line)

5.3.2.2.419. TS_STATIC_ASSERT_JOIN

Purpose	internal concatenation helper macro
Value	TS_STATIC_ASSERT_##msg_id##_in_line_##line

5.3.2.2.420. TS_STM8A

Purpose	constant for STM8A target architecture family
Value	41U

5.3.2.2.421. TS_TMS320

Purpose	constant for TMS320 target architecture family
Value	42U



5.3.2.2.422. TS_TMS470

Purpose	constant for TMS470 target architecture family
Value	15U

5.3.2.2.423. TS_TRICORE

Purpose	constant for TRICORE target architecture family
Value	16U

5.3.2.2.424. TS_UNCHECKEDGETCFG

Purpose	Get a non-const reference to a post-build config element - no check for invalid refs.
Value	((P2CONST(reftype, AUTOMATIC, module ## _APPL_CONST)) \ (P2CONST(void, AUTOMATIC, module ## _APPL_CONST)) \ &(((P2CONST(uint8, AUTOMATIC, module ## _APPL_CONST)) \ (P2CONST(void, AUTOMATIC, module ## _APPL_CONST)) \ cfgbase) [(offset)]))
Description	This macro retrieves a non-const pointer to a post-build config element without checking for invalid references

5.3.2.2.425. TS_UNCHECKEDGETCONSTCFG

Purpose	Get a const reference to a post-build config element - no check for invalid refs.
Value	((CONSTP2CONST(reftype, AUTOMATIC, module ## _APPL_CONST)) \ (CONSTP2CONST(void, AUTOMATIC, module ## _APPL_CONST)) \ &((CONSTP2CONST(uint8, AUTOMATIC, module ## _APPL_CONST)) \ (CONSTP2CONST(void, AUTOMATIC, module ## _APPL_CONST)) \ cfgbase) [(offset)]))
Description	This macro retrieves a const pointer to a post-build config element without checking for invalid references

5.3.2.2.426. TS_UNCHECKEDGETCONSTVAR

Purpose	Get a const reference to a post-build RAM element - no check for invalid refs.
----------------	--



Value	((CONSTP2VAR(reftype, AUTOMATIC, module ## _VAR_NOINIT)) \ \ (CONSTP2VAR(void, AUTOMATIC, module ## _VAR_NOINIT)) \ &(((CONSTP2VAR(uint8, AUTOMATIC, module ## _VAR_NOINIT)) \ (CONSTP2VAR(void, AUTOMATIC, module ## _VAR_NOINIT)) \ rambase) [(offset)]))
Description	This macro retrieves a const pointer to a post-build RAM element without checking for invalid references

5.3.2.2.427. TS_UNCHECKEDGETVAR

Purpose	Get a non-const reference to a post-build RAM element - no check for invalid refs.
Value	((P2VAR(reftype, AUTOMATIC, module ## _VAR_NOINIT)) \ (P2VAR(void, AUTOMATIC, module ## _VAR_NOINIT)) \ &((P2VAR(uint8, AUTOMATIC, module ## _VAR_NOINIT)) \ (P2VAR(void, AUTOMATIC, module ## _VAR_NOINIT)) \ rambase) [(offset)]))
Description	This macro retrieves a non-const pointer to a post-build RAM element without checking for invalid references

5.3.2.2.428. TS_V850

Purpose	constant for V850 target architecture family
Value	17U

5.3.2.2.429. TS_VAR_OFFSET_TYPE_MAX

Purpose	Maximum value for a TS_VarOffsetType.
Value	UINT16_C(0xFFFF)
Description	This macro defines the maximum value for a TS_VarOffsetType

5.3.2.2.430. TS_VAR_OFFSET_TYPE_MIN

Purpose	Minimum value for a TS_VarOffsetType.
Value	UINT16_C(0x0000)



Description	This macro defines the minimum value for a TS_VarOffsetType
--------------------	---

5.3.2.2.431. TS_VENDOR_ID_3SOFT

Purpose	definition of the unique vendor ID
Value	1U
Description	This vendor ID is based on the vendor ID published by the HIS. It should be used for each module

5.3.2.2.432. TS_VENDOR_ID_EB

Purpose	map vendor ID of EB to 3SOFT
Value	TS_VENDOR_ID_3SOFT

5.3.2.2.433. TS_WINDOWS

Purpose	constant for WINDOWS target architecture family
Value	19U

5.3.2.2.434. TS_XC16X

Purpose	constant for XC16X target architecture family
Value	18U

5.3.2.2.435. TS_XC2000

Purpose	constant for XC2000 target architecture family
Value	20U

5.3.2.2.436. TS_XPC56XX

Purpose	constant for XPC56XX target architecture family
Value	24U



5.3.2.2.437. UINT16_C

Purpose	Macro to define a constant of type uint16.
Value	((uint16) x ## U)

5.3.2.2.438. UINT32_C

Purpose	Macro to define a constant of type uint32.
Value	(x ## UL)

5.3.2.2.439. UINT8_C

Purpose	Macro to define a constant of type uint8.
Value	((uint8) x ## U)

5.3.2.2.440. UNIQUE

Purpose	
Value	UNIQUETYPEDEF(x, y)

5.3.2.2.441. UNIQUETYPEDEF

Purpose	
Value	x ## y

5.3.2.2.442. USIZE_C

Purpose	Macro to define a constant of platform specific type usize (generated, depending on parameter 'Cpu.Type').
Value	(x ## UL)

5.3.2.2.443. _STATIC_

Purpose	map _STATIC_ to value of STATIC for Autosar 2.1 backward compatibility
----------------	--



Value	static
--------------	--------

5.3.2.2.444. include_Atomics_TSPlatforms

Purpose	Common Autosar definitions and common macros are located in Atomics_TSPlatforms.h, which includes header files with common type definitions.
Value	
Description	Platforms Setting: Non-Autosar platform specific definitions related to atomic assignment are included by the header file Atomics_TSPlatforms.h which is located in Atomics. It contains macros and functions like TS_AtomicSetBit_8/16/32/64, TS_AtomicClearBit_8/16/32/64.

5.3.2.2.445. include_Platforms_TSPlatforms

Purpose	Common Autosar definitions and common macros are located in Platforms_TSPlatforms.h, which includes header files with common type definitions.
Value	
Description	Platforms Setting: Non-Autosar platform specific definitions are included by the header file Platforms_TSPlatforms.h which is located in Platforms.

5.3.2.3. Functions

5.3.2.3.1. ComStack_FindNextOne

Purpose	
Synopsis	uint32 ComStack_FindNextOne (const void *const bitfield , const uint32 prev);
Return Value	

5.3.2.3.2. TS_MemBZero16_NoCheck

Purpose	Fills len bytes with zero at destination dst without alignment checks (16-bit version).
----------------	---



Synopsis	<pre>void TS_MemBZero16_NoCheck (void *const dst , const usize len);</pre>	
Parameters (in)	dst	Pointer to the destination to fill
	len	Amount of bytes to fill

5.3.2.3.3. TS_MemBZero32_NoCheck

Purpose	Fills len bytes with zero at destination dst.	
Synopsis	<pre>void TS_MemBZero32_NoCheck (void *const dst , const usize len);</pre>	
Parameters (in)	dst	Pointer to the destination to fill
	len	Amount of bytes to fill

5.3.2.3.4. TS_MemBZero64

Purpose	Fills len bytes with zero at destination dst.	
Synopsis	<pre>void TS_MemBZero64 (void *const dst , const usize len);</pre>	
Parameters (in)	dst	Pointer to the destination to fill
	len	Amount of bytes to fill

5.3.2.3.5. TS_MemCmp16_NoCheck

Purpose	Compares n bytes of memory at locations a and b without alignment checks (16-bit version).	
Synopsis	<pre>Std_ReturnType TS_MemCmp16_NoCheck (const void *con- st a , const void *const b , const usize len);</pre>	
Parameters (in)	a	Pointer to the first memory block to compare
	b	Pointer to the second memory block to compare
	len	Amount of bytes to compare
Return Value	Comparison status	
	E_OK	Both memory blocks compare equal
	E_NOT_OK	The memory blocks are not equal



Description	This function compares a memory block of length len located at address a to a memory block located at address b.
--------------------	--

5.3.2.3.6. TS_MemCmp32_NoCheck

Purpose	Compares n bytes of memory at locations a and b without alignment checks (32-bit version).	
Synopsis	<pre>Std_ReturnType TS_MemCmp32_NoCheck (const void *const a , const void *const b , const usize len);</pre>	
Parameters (in)	a	Pointer to the first memory block to compare
	b	Pointer to the second memory block to compare
	len	Amount of bytes to compare
Return Value	Comparison status	
	E_OK	Both memory blocks compare equal
	E_NOT_OK	The memory blocks are not equal
Description	This function compares a memory block of length len located at address a to a memory block located at address b.	

5.3.2.3.7. TS_MemCmp64

Purpose	Compares n bytes of memory at locations a and b (64-bit version).	
Synopsis	<pre>Std_ReturnType TS_MemCmp64 (const void *const a , const void *const b , const usize len);</pre>	
Parameters (in)	a	Pointer to the first memory block to compare
	b	Pointer to the second memory block to compare
	len	Amount of bytes to compare
Return Value	Comparison status	
	E_OK	Both memory blocks compare equal
	E_NOT_OK	The memory blocks are not equal



Description	This function compares a memory block of length len located at address a to a memory block located at address b.
--------------------	--

5.3.2.3.8. TS_MemCpy16_NoCheck

Purpose	Copies len bytes of memory from dst to src without overlapping or alignment checks (16-bit version).	
Synopsis	<pre>void TS_MemCpy16_NoCheck (void *const dst , const void *const src , const usize len);</pre>	
Parameters (in)	dst	Pointer to the destination address
	src	Pointer to the source address
	len	Amount of bytes to copy
Description	This function copies a memory block of length len from the source src to the destination dst.	

5.3.2.3.9. TS_MemCpy32_NoCheck

Purpose	Copies len bytes of memory from dst to src without overlapping or alignment checks (32-bit version).	
Synopsis	<pre>void TS_MemCpy32_NoCheck (void *const dst , const void *const src , const usize len);</pre>	
Parameters (in)	dst	Pointer to the destination address
	src	Pointer to the source address
	len	Amount of bytes to copy
Description	This function copies a memory block of length len from the source src to the destination dst.	

5.3.2.3.10. TS_MemCpy64

Purpose	Copies len bytes of memory from dst to src (64-bit version).	
Synopsis	<pre>void TS_MemCpy64 (void *const dst , con- st void *const src , const usize len);</pre>	
Parameters (in)	dst	Pointer to the destination address



	src	Pointer to the source address
	len	Amount of bytes to copy
Description	This function copies a memory block of length len from the source src to the destination dst.	

5.3.2.3.11. TS_MemMove

Purpose	Moves len bytes of memory from location pointed to by src to the location pointed to by dst.	
Synopsis	<pre>void TS_MemMove (void *const dst , const void *const src , const usize len);</pre>	
Parameters (in)	dst	Pointer to the destination memory block
	src	Pointer to the source memory block
	len	Amount of bytes to copy

5.3.2.3.12. TS_MemSet16_NoCheck

Purpose	Assigns len copies of value val to destination dst without alignment checks (16-bit version).	
Synopsis	<pre>void TS_MemSet16_NoCheck (void *const dst , const uint8 val , const usize len);</pre>	
Parameters (in)	dst	Pointer to assign the value to
	val	Value to be assigned
	len	Amount of copies to assign

5.3.2.3.13. TS_MemSet32_NoCheck

Purpose	Assigns len copies of value val to destination dst without alignment checks (32-bit version).	
Synopsis	<pre>void TS_MemSet32_NoCheck (void *const dst , const uint8 val , const usize len);</pre>	
Parameters (in)	dst	Pointer to assign the value to
	val	Value to be assigned
	len	Amount of copies to assign



5.3.2.3.14. TS_MemSet64

Purpose	Assigns len copies of value val to destination dst.	
Synopsis	<pre>void TS_MemSet64 (void *const dst , const uint8 val , const usize len);</pre>	
Parameters (in)	dst	Pointer to assign the value to
	val	Value to be assigned
	len	Amount of copies to assign

5.3.2.3.15. TS_PlatformSigIsValid

Purpose	Checks whether the platform signature signature is valid.	
Synopsis	<pre>boolean TS_PlatformSigIsValid (uint32 signature);</pre>	
Parameters (in)	signature	Platform signature
Return Value	Validity status	
	TRUE	the platform signature signature is valid
	FALSE	the platform signature signature is invalid

5.3.3. Integration notes

5.3.3.1. Exclusive areas

Exclusive areas information is not available for this module.

5.3.3.2. Production errors

Production errors are not reported by the `Base` module.

5.3.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section [Memory mapping and compiler abstraction](#) in the [Integration notes](#) section for details.



The following table provides the list of sections that may be mapped for this module:

Memory section
CODE

5.3.3.4. Integration requirements

WARNING Integration requirements list is not exhaustive



The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the Base module.

5.3.3.5. Platform integration

Search for Platforms Setting in the source code of the Base plugin or in this document to find all locations where a platform specific setting is required.

5.3.3.5.1. Optimized memory copy function

The default platform independent implementation of the memory copy function can be overridden by an optimized implementation. To provide a custom memory copy function, do the following:

- ▶ Implement the memory copy function with the signature `void custom_memcpy(void* const destination, void const* const source, const usize length)` where
 - ▶ `destination` is the pointer to where the memory is copied to
 - ▶ `source` is the pointer to where the memory is copied from
 - ▶ `length` is number of bytes to be copied from `source` to `destination`
 The memory copy function is expected to copy `length` bytes from `source` to `destination`.
- ▶ Provide a custom standard header file
- ▶ Define the macro `TS_MemCpy(dst, src, len)` that just maps to the custom memory copy function in the custom standard header file
- ▶ Add the name of the custom standard header file to the Base configuration by the parameter `CustomStdIncludeFiles`



- ▶ Enable overriding of the memory copy function in the Base configuration by the parameter `CustomOverride_MemCopy`, `container CustomOverrides`

5.3.3.5.2. Optimized memory set function

The default platform independent implementation of the memory set function can be overridden by an optimized implementation. To provide a custom memory set function, do the following:

- ▶ Implement the memory set function with the signature `void custom_memset(void* const destination, const value, const usize length)` where
 - ▶ `destination` is the pointer to where the memory is set
 - ▶ `value` is the value to be set
 - ▶ `length` is the number of bytes to be set to specified value
 The memory set function is expected to set `length` bytes to `value` starting from `destination`.
- ▶ Provide a custom standard header file
- ▶ Define the macro `TS_MemSet(dst, val, len)` that just maps to the custom memory set function in the custom standard header file
- ▶ Add the name of the custom standard header file to the Base configuration by the parameter `CustomStdIncludeFiles`
- ▶ Enable overriding of the memory set function in the Base configuration by the parameter `CustomOverride_MemSet`, `container CustomOverrides`

5.3.3.5.3. Optimized memory compare function

The default platform independent implementation of the memory compare function can be overridden by an optimized implementation. To provide a custom memory compare function, do the following:

- ▶ Implement the memory compare function with the signature `void custom_memcmp(void const * const str1, void const * const str2, const usize length)` where
 - ▶ `str1` is the pointer to one block of memory
 - ▶ `str2` is the pointer to another block of memory
 - ▶ `length` is the number of bytes to be compared
 The memory compare function is expected to compare `length` bytes from `str1` to `str2` and return 0 if they're equal, smaller than 0 if `str1` is smaller than `str2`, greater than 0 if `str1` is greater than `str2`.
- ▶ Provide a custom standard header file
- ▶ Define the macro `TS_MemCmp(str1, str2, len)` that just maps to the custom memory compare function in the custom standard header file



- ▶ Add the name of the custom standard header file to the Base configuration by the parameter `CustomStdIncludeFiles`
- ▶ Enable overriding of the memory compare function in the Base configuration by the parameter `CustomOverride_MemCmp`, `container CustomOverrides`

5.3.3.5.4. Optimized memory zeroing function

The default platform independent implementation of the memory zeroing function can be overridden by an optimized implementation. To provide a custom memory zeroing function, do the following:

- ▶ Implement the memory zeroing function with the signature `void custom_membzero(void* const destination, const usize length)` where
 - ▶ `destination` is the pointer to where the memory is zeroed
 - ▶ `length` is the number of bytes to be zeroed
 The memory zeroing function is expected to zero `length` bytes from `destination`.
- ▶ Provide a custom standard header file
- ▶ Define the macro `TS_MemBZero(dst, len)` that just maps to the custom memory zeroing function in the custom standard header file
- ▶ Add the name of the custom standard header file to the Base configuration by the parameter `CustomStdIncludeFiles`
- ▶ Enable overriding of the memory zeroing function in the Base configuration by the parameter `CustomOverride_MemBZero`, `container CustomOverrides`

5.3.3.5.5. Low Level Primitives

Base supplies a layer of common operations that are not standardized in C90. Since they are required by several ComStack modules, Base provides the layer of abstraction with provisions to predefine them from Platforms or Compiler packages, effectively overriding them with a more targeted version using either non-standard compiler-known-functions, C or inline assembly.

The default implementation is a portable default. A `_CT_` infix designates a non-overridable version that can still be evaluated at compile-time and should be used for constants only. Note: the 64bit variants only exist, if the Compiler supports 64 bit types (`uint64` and `sint64`).

- ▶ `COMSTACK[_CT]_BYTE_MIRROR16/32` Hint: rotating a 16 bit value by 8 is effectively swapping its bytes.
For 32 bit split-registers, the sequence `ROL32,8; ROL16,8; ROL16,8; ROL32,24` does the trick.
- ▶ `COMSTACK[_CT]_HTON_UINT16/32` - utilize `COMSTACK[_CT]_BYTE_MIRROR16/32`
- ▶ `COMSTACK[_CT]_NTOH_UINT16/32` - utilize `COMSTACK[_CT]_BYTE_MIRROR16/32`



ComStack packet header setter/getter macros, that copy and shift content bytewise from and to network-order byte streams.

Depending on the respective buffer alignment and unaligned read/write capabilities, wider-access macros utilizing the NTOH and HTON macros are employed in different ComStack modules. The macros in Base serve as a common fall-back.

To adapt these macros to the respective module's Compiler abstraction, the following types are used for pointers to header areas:

- ▶ ComStack_ConstUInt8PtrType
- ▶ ComStack_Uint8PtrType

They must be defined by the respective module to match their pointer types for network byte streams.

- ▶ COMSTACK_GET16/32
- ▶ COMSTACK_SET16/32
- ▶ COMSTACK_GETCPY32 - verbatim copy without HTON / NTOH reordering
- ▶ COMSTACK_SETCOPY32 - verbatim copy without HTON / NTOH reordering
- ▶ COMSTACK_BYTE_REPLICATOR32/64

A macro that supplies a constant of the lowest bit set for every byte. This allows for relatively cheap replication of byte constants to a whole register.

Big constants are costly. Hence we multiply byte constants with byte replication constants to provide clever compilers alternate routes for evaluation: e.g. const / value range analysis -> PC-rel load of all-large consts vs. loading one big multiplier constant and then multiplying it with a number of small consts that each fit in the 8bit immediate operands (if the MUL is cheaper than the LD).

If your compiler is not smart enough to evaluate alternate routes or falls for premature constant-expression-evaluation, you may try to predefine the multiplier constants in terms of a volatile variable that effectively hides the constant from the compiler.

- ▶ COMSTACK_HAMMING_WEIGHT32/64 - utilizes COMSTACK_BYTE_REPLICATOR32/64

Hamming weight computes the number of bits set in the input argument (register width for 0). We employ the ISA equivalent of a binary adder tree, using register-width SHIFT and ADD operations, as SIMD operations by ignoring carry overflows, as it's proven they cannot occur. Thus we'd finish in $\lceil \log(\text{registerwidth}) \rceil$ steps.

However, we switch back to folding adds, once we reach operand sizes that are likely to be computed cheaper using register-parts or 'extract' operations due to less parallelism. An in-depth discussion can be found in good text books and under: [Hamming Weight \(Wikipedia\)](#)

- ▶ COMSTACK_LOWEST_BIT32/64 - utilizes COMSTACK_HAMMING_WEIGHT32/64



5.4. Compiler

5.4.1. Configuration parameters

There are no configuration parameters.

5.4.2. Application programming interface (API)

5.4.2.1. Macro constants

5.4.2.1.1. COMPILERCFG_EXTENSION_FILE

Purpose	Includes an additional Compiler_CfgExt.h file if macro is available.
Value	
Description	If the macro COMPILERCFG_EXTENSION_FILE is defined, the EB Compiler_Cfg.h file will include an additional file Compiler_CfgExt.h. This file can be used to add memory and pointer classes of 3rd-party modules.

5.4.2.1.2. COMPILERCFG_EXTENSION_MCAL_FILE

Purpose	Includes an additional Compiler_CfgExtMCAL.h file if macro is available.
Value	
Description	The compiler abstraction macro header file can be extended with 3rd-party specific macros. If the macro COMPILERCFG_EXTENSION_MCAL_FILE is defined, the EB's Compiler.h file will include an additional file Compiler_CfgExtMCAL.h. This file can be used to add additional compiler abstraction macros of 3rd-party MCAL modules. This feature is used during a platform porting by EB.

5.4.2.1.3. CONST

Purpose	abstraction for declaration and definition of constants
Value	consttype const memclass



Description	This macro abstracts the declaration and definition of constants.
--------------------	---

5.4.2.1.4. CONSTP2CONST

Purpose	abstraction for pointers in ROM pointing to ROM
Value	ptrtype const * const memclass ptrclass
Description	<p>This macro abstracts the declaration and definition of pointers in ROM pointing to constants in ROM.</p> <p>The pointer itself is not modifiable (read only). The pointer's target is not modifiable (read only).</p>

5.4.2.1.5. CONSTP2FUNC

Purpose	abstraction for declaration and definition of constant function pointers
Value	rettype (* const fctname)
Description	This macro abstracts the declaration and definition of constant pointers to functions.

5.4.2.1.6. CONSTP2VAR

Purpose	abstraction for pointers in ROM pointing to RAM
Value	ptrtype * const memclass ptrclass
Description	<p>This macro abstracts the declaration and definition of pointers in ROM pointing to variables in RAM.</p> <p>The pointer is not modifiable. (read only). The pointer's target is modifiable.</p>

5.4.2.1.7. FUNC

Purpose	abstraction for function declaration and definition
Value	rettype memclass
Description	This macro abstracts the declaration and definition of functions and ensures the correct syntax of function declaration as required by the specific compiler.



5.4.2.1.8. FUNC_P2CONST

Purpose	abstraction for function declaration and definition
Value	const rettype * memclass
Description	This macro abstracts the declaration and definition of functions returning a pointer to a constant and ensures the correct syntax of function declarations as required by a specific compiler.

5.4.2.1.9. FUNC_P2VAR

Purpose	abstraction for function declaration and definition
Value	rettype * memclass
Description	This macro abstracts the declaration and definition of functions returning a pointer to a variable and ensures the correct syntax of function declarations as required by a specific compiler.

5.4.2.1.10. LOCAL_INLINE

Purpose	definition of a keyword for 'static inline' functions
Value	static INLINE
Description	To be used for 'static inline' functions.

5.4.2.1.11. P2CONST

Purpose	abstraction for pointers in RAM pointing to ROM
Value	ptrtype const * memclass ptrclass
Description	<p>This macro abstracts the declaration and definition of pointers in RAM pointing to constants in ROM.</p> <p>The pointer itself is modifiable. The pointer's target is not modifiable (read only).</p>

5.4.2.1.12. P2FUNC

Purpose	abstraction for declaration and definition of function pointers
----------------	---



Value	rettype (* fctname) ptrclass
Description	This macro abstracts the declaration and definition of pointers to functions.

5.4.2.1.13. P2VAR

Purpose	abstraction for pointers in RAM pointing to RAM
Value	ptrtype * memclass ptrclass
Description	<p>This macro abstracts the declaration and definition of pointers in RAM pointing to variables in RAM.</p> <p>The pointer itself is modifiable. The pointer's target is modifiable.</p>

5.4.2.1.14. VAR

Purpose	abstraction for the declaration and definition of variables
Value	vartype memclass
Description	This macro abstracts the declaration and definition of variables.

5.4.2.1.15. _INLINE_

Purpose	definition of an inline keyword
Value	INLINE
Description	To be used for inlining functions values: 'inline' or empty to strictly stick to the strict ANSI C90 standard inline is not an option map _INLINE_ to INLINE for Autosar 2.1 backward compatibility

5.4.3. Integration notes

5.4.3.1. Exclusive areas

Exclusive areas information is not available for this module.



5.4.3.2. Production errors

Production errors are not reported by the `Compiler` module.

5.4.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

Memory mapping information is not available for this module.

5.4.3.4. Integration requirements

WARNING
Integration requirements list is not exhaustive


The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the `Compiler` module.

5.5. Det

5.5.1. Configuration parameters

Containers included		
Container name	Multiplicity	Description
CommonPublishedInformation	1..1	Label: Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
DetGeneral	1..1	General parameter definitions for the Det
DetNotification	0..1	Configuration of the notification functions.
DetDefensiveProgramming	1..1	Parameters for defensive programming

**Containers included**

SoftwareComponentList	0..n	<p>Definition of all Software Components that use the Det service port operations. Its multiplicity describes the number of applications accessing the Det.</p> <p>The name of list entries are taken to construct the provide port names. These provide ports must then be connected to the require ports of the corresponding SW-C with the RteAssistant. Therefore it is advisable to take the real SW-C name to make these connection process easy.</p>
PublishedInformation	1..1	<p>Label: EB Published Information</p> <p>Additional published parameters not covered by CommonPublishedInformation container.</p>

Parameters included

Parameter name	Multiplicity
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT
Label	Config Variant
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPreCompile
Range	VariantPreCompile

5.5.1.1. CommonPublishedInformation**Parameters included**

Parameter name	Multiplicity
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1

**Parameters included**

ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH



Parameter Name	SwMajorVersion
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	6
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwMinorVersion
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwPatchVersion
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ModuleId
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL



Default value	15
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	VendorId
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AU-TOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	Release
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.5.1.2. DetGeneral

Containers included		
Container name	Multiplicity	Description
DetServiceAPI	1..1	<p>Label: Service API Parameters</p> <p>Container for configuration of the service API of Det.</p> <p>Check "Enable Rte Usage" in order to enable this configuration item.</p>

Parameters included	
Parameter name	Multiplicity
DetForwardToDlt	0..1

**Parameters included**

DetVersionInfoApi	1..1
LoggingMode	1..1
BufferSize	1..1
KeepFirst	1..1
DetRteUsage	1..1
EnableReportErrorStopExecution	1..1

Parameter Name**DetForwardToDlt****Description**

The functionality related to this parameter is not supported by the current implementation.

Only if the parameter is present and set to true, the Det requires the Dlt interface and forwards its call to the function `Dlt_DetForwardErrorTrace()`. In this case the optional interface to `Dlt_Det` is required.

Multiplicity

0..1

Type

BOOLEAN

Default value

true

Configuration class**PreCompile:**

VariantPreCompile

Origin

AUTOSAR_ECUC

Parameter Name**DetVersionInfoApi****Label**

Enable Version Info API

Description

Switch to enable/disable the API function `Det_GetVersionInfo()` to read out the module's version information.

- ▶ true: Version info API enabled.

- ▶ false: Version info API disabled.

Multiplicity

1..1

Type

BOOLEAN

Default value

false

Configuration class**VariantPreCompile:**

VariantPreCompile

Origin

AUTOSAR_ECUC

Parameter Name**LoggingMode**



Description	Logging mode of the Det:	
	<ul style="list-style-type: none"> ▶ Internal: error reports are stored in an internal buffer ▶ Breakpoint: the debugger halts the system 	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	Internal	
Range	Internal Breakpoint	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	BufferSize	
Description	Number of recordable entries used if error reports are buffered depending of the logging mode	
Multiplicity	1..1	
Type	INTEGER	
Default value	10	
Range	>=1 <=32767	
Configuration class	PreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	KeepFirst	
Description	Reports to keep if buffer overflows: either the first (value is true) or last entries (value is false)	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	PreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	DetRteUsage	
Label	Enable Rte Usage	
Description	This parameter enables the usage of the RTE for this module.	



	For an easy integration it is recommended to disable the usage of the RTE at the beginning of the integration work.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	EnableReportErrorStopExecution
Description	If this configuration parameter is set to TRUE, the Det_ReportError() shall stop execution.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.5.1.3. DetServiceAPI

Parameters included	
Parameter name	Multiplicity
DetEnableASR32ServiceAPI	
DetEnableASR40ServiceAPI	
DetEnableASR42ServiceAPI	
DetEnableASR43ServiceAPI	
DetDefaultASRSer-viceAPI	1..1

Parameter Name	DetEnableASR32ServiceAPI
Label	Enable AUTOSAR 3.2 service API
Description	<p>Configures whether the AUTOSAR 3.2 service API shall be provided.</p> <ul style="list-style-type: none"> ▶ TRUE = Enables AUTOSAR 3.2 service API. ▶ FALSE = Disables AUTOSAR 3.2 service API.



Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	DetEnableASR40ServiceAPI
Label	Enable AUTOSAR 4.0 service API
Description	<p>Configures whether the AUTOSAR 4.0 service API shall be provided.</p> <ul style="list-style-type: none"> ▶ TRUE = Enables AUTOSAR 4.0 service API. ▶ FALSE = Disables AUTOSAR 4.0 service API.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	DetEnableASR42ServiceAPI
Label	Enable AUTOSAR 4.2 service API
Description	<p>Configures whether the AUTOSAR 4.2 service API shall be provided.</p> <ul style="list-style-type: none"> ▶ TRUE = Enables AUTOSAR 4.2 service API. ▶ FALSE = Disables AUTOSAR 4.2 service API.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	DetEnableASR43ServiceAPI
Label	Enable AUTOSAR 4.3 service API
Description	<p>Configures whether the AUTOSAR 4.3 service API shall be provided.</p> <ul style="list-style-type: none"> ▶ TRUE = Enables AUTOSAR 4.3 service API. ▶ FALSE = Disables AUTOSAR 4.3 service API.



Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	DetDefaultASRServiceAPI	
Label	Default AUTOSAR service API	
Description	<p>Defines the default AUTOSAR service API.</p> <ul style="list-style-type: none"> ▶ AUTOSAR_32 = AUTOSAR 3.2 service API is the default one. ▶ AUTOSAR_40 = AUTOSAR 4.0 service API is the default one. ▶ AUTOSAR_42 = AUTOSAR 4.2 service API is the default one. ▶ AUTOSAR_43 = AUTOSAR 4.3 service API is the default one. ▶ NONE = No default AUTOSAR service API is provided. 	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	AUTOSAR_40	
Range	AUTOSAR_32 AUTOSAR_40 AUTOSAR_42 AUTOSAR_43 NONE	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.5.1.4. DetNotification

Parameters included	
Parameter name	Multiplicity
DetErrorHook	0..n
DetReportRuntimeError-Callout	0..n

**Parameters included**

DetReportTransient-FaultCallout	0..n
DetHeaderFile	0..n

Parameter Name **DetErrorHook**

Description	Optional list of functions to be called by the Development Error Tracer in context of each call of Det_ReportError. The type of these functions must be identical to the type of <code>Det_ReportError()</code> itself: <code>Std_ReturnType (*f) (uint16, uint8, uint8, uint8)</code> .
Multiplicity	0..n
Type	FUNCTION-NAME
Default value	XXX_DetCallback
Configuration class	PreCompile: VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name **DetReportRuntimeErrorCallout**

Description	Optional list of functions to be called by the Default Error Tracer in context of each call of Det_ReportRuntimeError. The type of these functions must be identical to the type of <code>Det_ReportRuntimeError()</code> itself: <code>Std_ReturnType (*f) (uint16, uint8, uint8, uint8)</code> .
Multiplicity	0..n
Type	FUNCTION-NAME
Default value	XXX_DetRuntimeCallout
Configuration class	PreCompile: VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name **DetReportTransientFaultCallout**

Description	Optional list of functions to be called by the Default Error Tracer in context of each call of Det_ReportTransientFault. The type of these functions must be identical to the type of <code>Det_ReportTransientFault()</code> itself: <code>Std_ReturnType (*f) (uint16, uint8, uint8, uint8)</code> .
Multiplicity	0..n



Type	FUNCTION-NAME	
Default value	XXX_DetTransientFaultCallout	
Configuration class	PreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DetHeaderFile	
Description	Optional list of header files which are included. These header files must contain the function declarations of the notification functions given in DetErrorHook.	
Multiplicity	0..n	
Type	STRING	
Default value	XXX_cbk.h	
Configuration class	PreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.5.1.5. DetDefensiveProgramming

Parameters included	
Parameter name	Multiplicity
<u>DetDefensiveProgram-</u> <u>mingEnabled</u>	1..1
<u>DetPrecondi-</u> <u>tionAssertEnabled</u>	1..1
<u>DetPostcondi-</u> <u>tionAssertEnabled</u>	1..1
<u>DetStaticAssertEnabled</u>	1..1
<u>DetUnreachableCode-</u> <u>AssertEnabled</u>	1..1
<u>DetInvariantAssertEn-</u> <u>abled</u>	1..1

Parameter Name	DetDefensiveProgrammingEnabled
Description	Enables or disables the feature 'defensive programming' for all software modules
Multiplicity	1..1
Type	BOOLEAN



Default value	false	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	DetPreconditionAssertEnabled	
Description	Enables handling of precondition assertion checks reported from functions in other modules.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	DetPostconditionAssertEnabled	
Description	Enables handling of postcondition assertion checks reported from functions in other modules.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	DetStaticAssertEnabled	
Description	Enables handling of static assertion checks reported from functions in other modules.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	DetUnreachableCodeAssertEnabled	
Description	Enables handling of unreachable code assertion checks reported from functions in other modules.	
Multiplicity	1..1	



Type	BOOLEAN	
Default value	true	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	DetInvariantAssertEnabled	
Description	Enables handling of invariant assertion checks reported from functions in other modules.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.5.1.6. SoftwareComponentList

Containers included		
Container name	Multiplicity	Description
InstanceIdList	0..n	This container describes one entry for the SWC's Instance Id.

Parameters included	
Parameter name	Multiplicity
SoftwareComponentAutosarVersion43	1..1
ModuleId	1..1

Parameter Name	SoftwareComponentAutosarVersion43
Description	<p>This parameter enables the usage of the RTE Service Ports for DET according to AUTOSAR version greater than or equal to 4.3</p> <p>If the configuration parameter is set to TRUE then both ModuleId and InstanceId are used to identify the component and component instance by using "port defined argument values"</p> <p>If the configuration parameter is set to FALSE then only ModuleId is used to identify the component by using "port defined argument values"</p>



	The functionality of this parameter is enabled only if DetEnableASR43ServiceAPI or DetDefaultASRServiceAPI = ASR43 are selected from the ServiceAPI container.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	ModuleId
Description	This parameter defines the unique module ID of the Software Component. When reporting errors to the DET, a symbolic name derived from the moduleID has to be used to identify the reporter Values in the range of 0..255 are reserved for Basic Software Modules. Valid module IDs for external applications range from 256..65535.
Multiplicity	1..1
Type	INTEGER
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.5.1.7. InstanceIdList

Parameters included	
Parameter name	Multiplicity
InstanceId	1..1

Parameter Name	InstanceId
Description	Configuration parameter which defines the Instance ID of the Software Component used for the according Service Port. If the Software Component is configured according to AUTOSAR version greater than or equal to 4.3 this parameter is mandatory. If the Software Component is configured according to AUTOSAR version less than 4.3 this parameter is not required.



	This parameter is editable if the SoftwareComponenetAutosarVersion43 parameter is enabled.
	Valid instance IDs for external applications range from 0..255.
Multiplicity	1..1
Type	INTEGER
Configuration class	PbcfgMSupport: VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.5.1.8. PublishedInformation

Parameters included	
Parameter name	Multiplicity
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport
Label	PbcfgM support
Description	Specifies whether or not the Det can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.5.2. Application programming interface (API)

5.5.2.1. Macro constants

5.5.2.1.1. DET_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
----------------	--------------------------------



Value	4U
--------------	----

5.5.2.1.2. DET_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.5.2.1.3. DET_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.5.2.1.4. DET_E_INVARIANT_ASSERT_FAILED

Purpose	Reserved error value for invariant assertions.
Value	233U

5.5.2.1.5. DET_E_PARAM_POINTER

Purpose	Development error code used if Det_GetVersionInfo() is called with null parameter pointer.
Value	1U

5.5.2.1.6. DET_E_POSTCONDITION_ASSERT_FAILED

Purpose	Reserved error value for postcondition assertions.
Value	231U

5.5.2.1.7. DET_E_PRECONDITION_ASSERT_FAILED

Purpose	Reserved error value for precondition assertions.
----------------	---



Value	230U
--------------	------

5.5.2.1.8. DET_E_UNREACHABLE_CODE_ASSERT_FAILED

Purpose	Reserved error value for unreachable code assertions.
Value	232U

5.5.2.1.9. DET_INTERNAL_API_ID

Purpose	Service id for internal functions.
Value	0xFFU

5.5.2.1.10. DET_INVARIANT_ASSERT

Purpose	Report a invariant assertion.
Value	do { \ if (!Condition) \ { \ DET_INVARIANT_ASSERT_PRINT(\ (Condition), (ModuleId), (InstanceId), (ApId)); \ (void)Det_ReportError(\ (ModuleId), (InstanceId), (ApId), \ DET_E_INVARIANT_ASSERT_FAILED); \ } \ } \ while (0)
Description	Use this macro in BSW modules to check a defensive programming invariant assertion.

5.5.2.1.11. DET_INVARIANT_ASSERT_PRINT

Purpose	Internal helper macro that in fact is a wrapper of Det_InvariantAssertPrint that have as few arguments initialized with compiler macros to identify place of call.
Value	Det_InvariantAssertPrint(\ (ModuleId), (InstanceId), (ApId), __FILE__-, __LINE__-, #Condition)

5.5.2.1.12. DET_MODULE_ID

Purpose	AUTOSAR module identification.
----------------	--------------------------------



Value	15U
--------------	-----

5.5.2.1.13. DET_POSTCONDITION_ASSERT

Purpose	Report a postcondition assertion.
Value	<code>do \{ \ if (!(Condition)) \{ \ DET_POSTCONDITION_ASSERT_PRINT(\ (Condition), (ModuleId), (Instanceld), (Apild)); \ (void)Det_ReportError(\ (ModuleId), (Instanceld), (Apild), \ DET_E_POSTCONDITION_ASSERT_FAILED); \ } \ } \ while (0)</code>
Description	Use this macro in BSW modules to check a defensive programming postcondition assertion.

5.5.2.1.14. DET_POSTCONDITION_ASSERT_PRINT

Purpose	Internal helper macro that in fact is a wrapper of Det_PostconditionAssertPrint that have as few arguments initialized with compiler macros to identify place of call.
Value	<code>Det_PostconditionAssertPrint(\ (ModuleId), (Instanceld), (Apild), __FILE__-, __LINE__-, #Condition)</code>

5.5.2.1.15. DET_PRECONDITION_ASSERT

Purpose	Report a precondition assertion.
Value	<code>do \{ \ if (!(Condition)) \{ \ DET_PRECONDITION_ASSERT_NO_EVAL(\ (Condition), (ModuleId), (Instanceld), (Apild)); \ } \ } \ while (0)</code>
Description	Use this macro in BSW modules to check a defensive programming precondition assertion with an evaluation of the condition.

5.5.2.1.16. DET_PRECONDITION_ASSERT_NO_EVAL

Purpose	Report a precondition assertion without condition evaluation.
Value	<code>do \{ \ DET_PRECONDITION_ASSERT_PRINT(\ (Condition), (ModuleId), (Instanceld), (Apild)); \ (void)Det_ReportError(\ (ModuleId), (Instanceld), (Apild), \ DET_E_PRECONDITION_ASSERT_FAILED); \ } \ while (0)</code>



Description	Use this macro in BSW modules to check a defensive programming precondition assertion without evaluating the condition.
--------------------	---

5.5.2.1.17. DET_PRECONDITION_ASSERT_PRINT

Purpose	Internal helper macro that in fact is a wrapper of Det_PreconditionAssertPrint that have as few arguments initialized with compiler macros to identify place of call.
Value	<code>Det_PreconditionAssertPrint(\ (ModuleId), (InstanceId), (ApId), __FILE__-, __LINE__-, #Condition)</code>

5.5.2.1.18. DET_SID_GET_VERSION_INFO

Purpose	AUTOSAR API service ID.
Value	0x03U
Description	Definition of service ID for Det_GetVersionInfo() .

5.5.2.1.19. DET_SID_INIT

Purpose	AUTOSAR API service ID.
Value	0x00U
Description	Definition of service ID for Det_Init() .

5.5.2.1.20. DET_SID_REPORT_ERROR

Purpose	AUTOSAR API service ID.
Value	0x01U
Description	Definition of service ID for Det_ReportError() .

5.5.2.1.21. DET_SID_REPORT_RUNTIME_ERROR

Purpose	AUTOSAR API service ID.
----------------	-------------------------



Value	0x04U
Description	Definition of service ID for Det_ReportRuntimeError() .

5.5.2.1.22. DET_SID_REPORT_TRANSIENT_FAULT

Purpose	AUTOSAR API service ID.
Value	0x05U
Description	Definition of service ID for Det_ReportTransientFault() .

5.5.2.1.23. DET_SID_START

Purpose	AUTOSAR API service ID.
Value	0x02U
Description	Definition of service ID for Det_Strat() .

5.5.2.1.24. DET_STATIC_ASSERT

Purpose	Report a static assertion.
Value	<code>typedef uint8 DET_STATIC_ASSERT_JOIN(DetStaticAssertFailedInLine,__LINE__-)[\ (expr) ? 1 : -1];</code>
Description	<p>Use this macro in BSW modules to check a static compile time assertion as part of defensive programming. With this macro you can check</p> <ul style="list-style-type: none"> ▶ size of types and size of global variables ▶ alignment of global variables ▶ padding in structures <p>The macro cannot be used for non-static check involving quantities which cannot be computed by the compiler at compile time.</p> <p>This macro must only be used from global file scope, especially at places where global declarations can be done. Do not use it inside of a function scope. Do not use a trailing semicolon after DET_STATIC_ASSERT(). In the case of a disabled DET_STATIC_ASSERT() macro a trailing semicolon would lead to a single semicolon outside of a function body which is not allowed in C syntax.</p> <p>Examples:</p>



Ensure the correct size of a data type:

- ▶ `typedef uint8 Mod_IdHandleType;`
- ▶ `DET_STATIC_ASSERT(sizeof(Mod_IdHandleType) == 1U)`

Ensure correct alignment of the data type (32-bit in this case):

- ▶ `Mod_IdHandleType Mod_GlobalIdHandle;`
- ▶ `DET_STATIC_ASSERT((&Mod_GlobalIdHandle & 3U) == 0U)`

Technical background: If the expr is evaluated to be false the macro defines an array type with negative array size. This is an invalid construct in ANSI C and leads to a compiler error. The type name is typically included in the compiler's error message. It includes the string "StaticAssertFailed" and also the line number on which the macro [DET_STATIC_ASSERT\(\)](#) is placed. From that the user can infer which static assert failed.

5.5.2.1.25. DET_STATIC_ASSERT_JOIN

Purpose	Internal helper macro.
Value	<code>DET_STATIC_ASSERT_JOIN_HLP(X,Y)</code>

5.5.2.1.26. DET_STATIC_ASSERT_JOIN_HLP

Purpose	Internal helper macro.
Value	<code>X##Y</code>

5.5.2.1.27. DET_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	6U

5.5.2.1.28. DET_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
----------------	-------------------------------



Value	5U
--------------	----

5.5.2.1.29. DET_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	1U

5.5.2.1.30. DET_UNREACHABLE_CODE_ASSERT

Purpose	Report a unreachable code assertion.
Value	<code>do \{ \ DET_UNREACHABLE_CODE_ASSERT_PRINT(\ (ModuleId), (Instanceld), (Apild)); \ (void)Det_ReportError(\ (ModuleId), (Instanceld), (Apild), \ DET_E_UN-REACHABLE_CODE_ASSERT_FAILED); \ } \ while (0)</code>
Description	Use this macro in BSW modules to mark unreachable code as part of defensive programming.

5.5.2.1.31. DET_UNREACHABLE_CODE_ASSERT_PRINT

Purpose	Internal helper macro that in fact is a wrapper of Det_UnreachableCodeAssertPrint that have as few arguments initialized with compiler macros to identify place of call.
Value	<code>Det_UnreachableCodeAssertPrint(\ (ModuleId), (Instanceld), (Apild), __FILE__-, __-LINE__-)</code>

5.5.2.1.32. DET_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.5.2.1.33. Det_ReportError

Purpose	Internal helper macro.
----------------	------------------------



Value	Det_ASR40_ReportError
Description	Wrapping macro to provide AUTOSAR 4.0 API as default to other BSW modules

5.5.2.1.34. Det_ReportRuntimeError

Purpose	Internal helper macro.
Value	Det_ASR43_ReportRuntimeError
Description	Wrapping macro to provide AUTOSAR 4.3 API as default to other BSW modules

5.5.2.1.35. Det_ReportTransientFault

Purpose	Internal helper macro.
Value	Det_ASR43_ReportTransientFault
Description	Wrapping macro to provide AUTOSAR 4.3 API as default to other BSW modules

5.5.2.2. Objects

5.5.2.2.1. Det_ErrorIndex

Purpose	Index for writing errors into buffer.
Type	uint8
Description	Det_ErrorBuffer[Det_ErrorIndex][Det_WriteIndex_<error_type>] is the next field to be overwritten. Variable definition is only available for internal or external logging mode. Error type: DevelopmentError / RuntimeError / TransientFault Variable shall only be used for debugging.

5.5.2.2.2. Det_ErrorLost_DevelopmentError

Purpose	Counter for lost development errors because of buffer overflow.
Type	uint16



Description	Variable definition is only available for internal or external logging mode. Variable shall only be used for debugging.
--------------------	--

5.5.2.2.3. Det_ErrorLost_RuntimeError

Purpose	Counter for lost runtime errors because of buffer overflow.
Type	uint16
Description	Variable definition is only available for internal or external logging mode. Variable shall only be used for debugging.

5.5.2.2.4. Det_FaultLost_TransientFault

Purpose	Counter for lost transient faults because of buffer overflow.
Type	uint16
Description	Variable definition is only available for internal or external logging mode. Variable shall only be used for debugging.

5.5.2.2.5. Det_UsedSlots_DevelopmentError

Purpose	Slot count of used Det_ErrorBuffer development entries.
Type	uint16
Description	Variable definition is only available for internal or external logging mode. Variable shall only be used for debugging.

5.5.2.2.6. Det_UsedSlots_RuntimeError

Purpose	Slot count of used Det_ErrorBuffer runtime entries.
Type	uint16
Description	Variable definition is only available for internal or external logging mode.



Variable shall only be used for debugging.

5.5.2.2.7. Det_UsedSlots_TransientFault

Purpose	Slot count of used Det_ErrorBuffer transient entries.
Type	uint16
Description	<p>Variable definition is only available for internal or external logging mode.</p> <p>Variable shall only be used for debugging.</p>

5.5.2.2.8. Det_WriteIndex_DevelopmentError

Purpose	Index for writing development errors into buffer.
Type	uint16
Description	<p>Det_ErrorBuffer[Det_ErrorIndex][Det_WriteIndex_DevelopmentError] is the next field to be overwritten. Variable definition is only available for internal or external logging mode.</p> <p>Variable shall only be used for debugging.</p>

5.5.2.2.9. Det_WriteIndex_RuntimeError

Purpose	Index for writing runtime errors into buffer.
Type	uint16
Description	<p>Det_ErrorBuffer[Det_ErrorIndex][Det_WriteIndex_RuntimeError] is the next field to be overwritten. Variable definition is only available for internal or external logging mode.</p> <p>Variable shall only be used for debugging.</p>

5.5.2.2.10. Det_WriteIndex_TransientFault

Purpose	Index for writing transient faults into buffer.
Type	uint16
Description	<p>Det_ErrorBuffer[Det_ErrorIndex][Det_WriteIndex_TransientFault] is the next field to be overwritten. Variable definition is only available for internal or external logging mode.</p>



Variable shall only be used for debugging.

5.5.2.3. Functions

5.5.2.3.1. Det_ASR32_ReportError

Purpose	Report development errors.	
Synopsis	<pre>void Det_ASR32_ReportError (uint16 ModuleId , uint8 InstanceId , uint8 ApiId , uint8 ErrorId);</pre>	
Service ID	<u>DET_SID_REPORT_ERROR</u>	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ModuleId	Module ID of calling module.
	InstanceId	Instance ID of calling module.
	ApiId	ID of API service in which error is detected.
	ErrorId	ID of detected development error.
Description	This function is the AUTOSAR 3.2 specific implementation to report development errors. This function is called from modules XXX for development errors in case of activated preprocessor switch XXX_DEV_ERROR_DETECT. Production relevant errors shall be reported to the Diagnostics Event Manager (DEM).	

5.5.2.3.2. Det_ASR40_ReportError

Purpose	Report development errors.	
Synopsis	<pre>Std_ReturnType Det_ASR40_ReportError (uint16 ModuleId , uint8 InstanceId , uint8 ApiId , uint8 ErrorId);</pre>	
Service ID	<u>DET_SID_REPORT_ERROR</u>	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ModuleId	Module ID of calling module.
	InstanceId	Instance ID of calling module.



	ApiId	ID of API service in which error is detected.
	ErrorId	ID of detected development error.
Return Value	always E_OK	
Description	This function reports development errors. This function is called from modules XXX for development errors in case of activated preprocessor switch XXX_DEV_ERROR_DETECT. Production relevant errors shall be reported to the Diagnostics Event Manager (DEM).	

5.5.2.3.3. Det_ASR43_ReportRuntimeError

Purpose	Report runtime errors.	
Synopsis	<pre>Std_ReturnType Det_ASR43_ReportRuntimeError (uint16 ModuleId , uint8 InstanceId , uint8 ApiId , uint8 ErrorId);</pre>	
Service ID	<u>DET_SID_REPORT_RUNTIME_ERROR</u>	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ModuleId	Module ID of calling module.
	InstanceId	Instance ID of calling module.
	ApiId	ID of API service in which error is detected.
	ErrorId	ID of detected runtime error.
Return Value	always E_OK	
Description	This function reports runtime errors. This function is called from modules XXX for runtime errors in case of activated preprocessor switch XXX_DEV_ERROR_DETECT. Production relevant errors shall be reported to the Diagnostics Event Manager (DEM).	

5.5.2.3.4. Det_ASR43_ReportTransientFault

Purpose	Report transient faults.	
Synopsis	<pre>Std_ReturnType Det_ASR43_ReportTransientFault (uint16 ModuleId , uint8 InstanceId , uint8 ApiId , uint8 FaultId);</pre>	
Service ID	<u>DET_SID_REPORT_TRANSIENT_FAULT</u>	



Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ModuleId	Module ID of calling module.
	InstanceId	Instance ID of calling module.
	ApiId	ID of API service in which transient fault is detected (defined in SWS of calling module)
	FaultId	ID of detected transient fault (defined in SWS of calling module).
Return Value	: If no callout exists it shall return E_OK, otherwise it shall return the value of the configured callout. In case several callouts are configured the logical or (sum) of the callout return values shall be returned. Rationale: since E_OK=0, E_OK will be only returned if all are E_OK, and for multiple error codes there is a good chance to detect several of them.	
Description	This function reports transient faults. This function is called from modules XXX for transient faults in case of activated preprocessor switch XXX_DEV_ERROR_DETECT. Production relevant transient faults shall be reported to the Diagnostics Event Manager (DEM).	

5.5.2.3.5. Det_GetVersionInfo

Purpose	Get version information of the Development Error Tracer.	
Synopsis	<pre>void Det_GetVersionInfo (Std_VersionInfoType * VersionInfoPtr);</pre>	
Service ID	DET_SID_GET_VERSION_INFO	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	VersionInfoPtr	Pointer to where to store the version information of this module.
Description	<p>This service returns the version information of this module. The version information includes:</p> <ul style="list-style-type: none"> ▶ Vendor Id ▶ Module Id ▶ Instance Id ▶ Vendor specific version numbers 	



	Precondition: Function is only available if parameter DetVersionInfoApi is set to true.
--	---

5.5.2.3.6. Det_Init

Purpose	Initialize the Development Error Tracer.
Synopsis	<code>void Det_Init (void);</code>
Service ID	DET_SID_INIT
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	<p>This function initializes the Development Error Tracer. The initialization function resets all error counters and error logging data.</p> <p>If an initialization, or error information (e.g. after reset) is not desired, this function shall not be called. Det_Start() must be called after Det_Init() if external logging via PduR is enabled.</p>

5.5.2.3.7. Det_InvariantAssertPrint

Purpose	Print failed invariant assertion.	
Synopsis	<code>void Det_InvariantAssertPrint (uint16 ModuleId , uint8 InstanceId , uint8 ApiId , const char * File , uint32 Line , const char * Condition);</code>	
Parameters (in)	ModuleId	The ID of the reporting module
	InstanceId	The instance ID of the reporting module
	ApiId	The API function where the check failed
	File	The file where the check failed
	Line	The line of file where the check failed
	Condition	The condition check that enable print
Description	Internal helper function to print a failed invariant assertion.	

5.5.2.3.8. Det_PostconditionAssertPrint

Purpose	Print failed postcondition assertion.
----------------	---------------------------------------



Synopsis	<pre>void Det_PostconditionAssertPrint (uint16 ModuleId , uint8 InstanceId , uint8 ApiId , const char * File , uint32 Line , const char * Condition);</pre>	
Parameters (in)	ModuleId	The ID of the reporting module
	InstanceId	The instance ID of the reporting module
	ApiId	The API function where the check failed
	File	The file where the check failed
	Line	The line of file where the check failed
	Condition	The condition check that enable print
Description	Internal helper function to print a failed Postcondition.	

5.5.2.3.9. Det_PreconditionAssertPrint

Purpose	Print failed precondition assertion.	
Synopsis	<pre>void Det_PreconditionAssertPrint (uint16 ModuleId , uint8 InstanceId , uint8 ApiId , const char * File , uint32 Line , const char * Condition);</pre>	
Parameters (in)	ModuleId	The ID of the reporting module
	InstanceId	The instance ID of the reporting module
	ApiId	The API function where the check failed
	File	The file where the check failed
	Line	The line of file where the check failed
	Condition	The condition check that enable print
Description	Internal helper function to print a failed precondition.	

5.5.2.3.10. Det_Start

Purpose	Start the Development Error Tracer.
Synopsis	<pre>void Det_Start (void);</pre>
Service ID	DET_SID_START
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This function starts the Development Error Tracer after Det_Init() has been called.



If external logging via PduR is enabled, this function must be called after the PduR_-Init() function. This function must be called to enable Det to send error reports via PduR.

5.5.2.3.11. Det_UnreachableCodeAssertPrint

Purpose	Print unreachable code assertion.	
Synopsis	<pre>void Det_UnreachableCodeAssertPrint (uint16 ModuleId , uint8 InstanceId , uint8 ApiId , const char * File , uint32 Line);</pre>	
Parameters (in)	ModuleId	The ID of the reporting module
	InstanceId	The instance ID of the reporting module
	ApiId	The API function where the check failed
	File	The file where the check failed
	Line	The line of file where the check failed
Description	Internal helper function to print if execution reaches unreachable code.	

5.5.3. Integration notes

5.5.3.1. Exclusive areas

This section describes the exclusive areas used by the `Det` module.

5.5.3.1.1. SCHM_DET_EXCLUSIVE_AREA_0

Protected data structures	All shared data that shall be protected from mutual access.
Recommended locking mechanism	This exclusive area should be secured by using some kind of interrupt locking mechanism; e.g. <code>SuspendAllInterrupts()</code> provided by the AUTOSAR Operating System. The reasons is that the Det is called by many modules including modules of the MCAL layer. The different options for interrupt locking mechanisms are described in the EB tresos AutoCore Generic documentation. See the section Mapping exclusive areas in the basic software modules in the Integration notes section for details.



5.5.3.2. Production errors

Production errors are not reported by the Det module.

5.5.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section [Memory mapping and compiler abstraction](#) in the [Integration notes](#) section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
VAR_NO_INIT_UNSPECIFIED
CODE
VAR_INIT_8
VAR_NO_INIT_8
VAR_NO_INIT_16

5.5.3.4. Integration requirements

WARNING

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.5.3.4.1. Det.EB.IntReq.EB_INTREQ_Det_0001

Description	Callout functions of the Det module must not call Det_ReportError().
Rationale	According to DET026, recursive calls of Det_ReportError() are not allowed.

5.5.3.4.2. Det.EB.IntReq.EB_INTREQ_Det_0002

Description	In case the Multi-Core functionality of the Det module is used in a project with Memory Protection enabled, the memory sections of the Det module shall be configured in a way that write access is granted for all Os partitions from which Det errors can be reported.
-------------	--



Rationale	If Memory Protection is enabled and Det errors are reported from Os Partitions which do not have write access to the Det_ErrorBuffer, memory protection errors will be triggered.
------------------	---

5.5.3.4.3. Det.EB.IntReq.EB_INTREQ_Det_0003

Description	Callout functions of the Det module must not call Det_ReportRuntimeError().
Rationale	According to SWS_Det_00026, recursive calls of Det_ReportRuntimeError() are not allowed.

5.5.3.4.4. Det.EB.IntReq.EB_INTREQ_Det_0004

Description	Callout functions of the Det module must not call Det_ReportTransientFault().
Rationale	According to SWS_Det_00026, recursive calls of Det_ReportTransientFault() are not allowed.

5.6. EcuC

5.6.1. Configuration parameters

Containers included		
Container name	Multiplicity	Description
CommonPublishedInformation	1..1	Label: Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
EcucGeneral	1..1	This container is a subcontainer of Ecuc and specifies the general configuration parameters
EcucHardware	0..1	Hardware definition of this ECU. This parameter enables the container EcucCoreDefinition which is currently not supported. Do not edit this container. Otherwise, you invalidate the quality level as described in the quality statement in your delivery.
EcucPartitionCollection	0..1	Collection of partitions defined for this ECU.

**Containers included**

EcucPduCollection	0..1	Collection of all Pdu objects flowing through the Com-Stack.
EcucPostBuildVariants	0..1	Collection of toplevel PostBuildSelectable variants. The Pre-definedVariants linked inside this container will determine how many PostBuildSelectableVariants exist. If this container exist the name pattern for initialization of BSW modules will be <Mip>_Config_<PredefinedVariant.shortName>. If this container does not exist the name pattern for initialization of BSW modules will be <Mip>_Config. Manage Post-Build-Loadable and Post-Build-Selectable variants of an ECU configuration project.
EcucVariationResolver	0..1	
PublishedInformation	1..1	Label: EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included

Parameter name	Multiplicity
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT
Label	Config Variant
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPostBuild
Range	VariantPostBuild

5.6.1.1. CommonPublishedInformation**Parameters included**

Parameter name	Multiplicity
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1

**Parameters included**

SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL



Default value	3
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwMajorVersion
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwMinorVersion
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwPatchVersion
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	16
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ModuleId
-----------------------	-----------------



Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	10
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	VendorId
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AU-TOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	Release
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.6.1.2. EcucGeneral

Parameters included	
Parameter name	Multiplicity
EcucDevErrorDetect	1..1
EcucMetaDataHandlin-gEnabled	1..1



Parameter Name	EcucDevErrorDetect	
Description	If true then EcuC will enable the error-reporting to the Development Error Tracer (DET)	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	EcucMetaDataHandlingEnabled	
Description	If true then EcuC will enable the Meta Data Handling	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.6.1.3. EcucHardware

Containers included		
Container name	Multiplicity	Description
EcucCoreDefinition	0..n	Definition of one core on this ECU.

5.6.1.4. EcucCoreDefinition

Parameters included	
Parameter name	Multiplicity
EcucCoreId	1..1
EcucCoreHwRef	0..1

Parameter Name	EcucCoreId
Description	ID of the core.
Multiplicity	1..1



Type	INTEGER	
Range	<=65535 >=0	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcucCoreHwRef	
Description	Optional reference to the HwElement of HwCategory ProcessingUnit that represents this core in the ECU Resource Template.	
Multiplicity	0..1	
Type	FOREIGN-REFERENCE	
Configuration class	PreCompile:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.6.1.5. EcucPartitionCollection

Containers included		
Container name	Multiplicity	Description
EcucPartition	0..n	Definition of one partition on this ECU. One partition will be implemented using one OS-application.

5.6.1.6. EcucPartition

Containers included		
Container name	Multiplicity	Description
EcucPartitionSoftware-ComponentInstanceRef	0..n	References the SW Component instances from the ECU Extract that shall be executed in this partition.

Parameters included

Parameter name	Multiplicity
EcucPartitionBswModuleExecution	1..1
EcucPartitionQmB-swModuleExecution	1..1

**Parameters included**

PartitionCanBeRestarted	1..1
EcucPartitionBswModuleDistinguishedPartition	0..n

Parameter Name	EcucPartitionBswModuleExecution	
Description	Denotes that this partition will execute all BSW modules. BSW modules can only be executed in one partition.	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcucPartitionQmBswModuleExecution	
Description	Denotes that this partition will execute QM BSW. This parameter is currently not supported. Do not edit this parameter. Otherwise, you invalidate the quality level as described in the quality statement in your delivery.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	PartitionCanBeRestarted	
Description	Specifies the requirement whether the partition can be restarted. If set to true, all software executing in this partition shall be capable of handling a restart.	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcucPartitionBswModuleDistinguishedPartition	
Description	This maps the abstract partition of the BSW module to a concrete partition existing in the ECU. This container is currently not supported. Do not edit this contain-	



	er. Otherwise, you invalidate the quality level as described in the quality statement in your delivery.
Multiplicity	0..n
Type	FOREIGN-REFERENCE
Configuration class	PreCompile: VariantPostBuild
Origin	AUTOSAR_ECUC

5.6.1.7. EcucPartitionSoftwareComponentInstanceRef

Parameters included	
Parameter name	Multiplicity
TARGET	1..1
CONTEXT	0..n

Parameter Name	TARGET
Multiplicity	1..1
Type	REFERENCE
Origin	AUTOSAR_ECUC

Parameter Name	CONTEXT
Multiplicity	0..n
Type	REFERENCE
Range	ROOT-SW-COMPOSITION-PROTOTYPE
Origin	AUTOSAR_ECUC

5.6.1.8. EcucPduCollection

Containers included		
Container name	Multiplicity	Description
MetaDataType	0..n	Meta data serves to transport information through the AUTOSAR layers. It is transported by the PduInfoType structure via a separate pointer to a byte array alongside the length of and a pointer to the payload of the PDU. This container defines the content of the meta data.

**Containers included**

Pdu	0..n	One Pdu flowing through the COM-Stack. This Pdu is used by all Com-Stack modules to agree on referencing the same Pdu.
---------------------	------	--

Parameters included

Parameter name	Multiplicity
PduldTypeEnum	1..1
PduLengthTypeEnum	1..1

Parameter Name	PduldTypeEnum
Description	The PduldType is used within the entire AUTOSAR Com Stack except for bus drivers. The size of this global type depends on the maximum number of PDUs used within one software module. If no software module deals with more PDUs than 256, this type can be set to uint8. If at least one software module handles more than 256 PDUs, this type must be set to uint16. See AUTOSAR_SWS_CommunicationStackTypes for more details.
Multiplicity	1..1
Type	ENUMERATION
Default value	UINT16
Range	UINT16 UINT8
Configuration class	VariantPostBuild: VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	PduLengthTypeEnum
Description	The PduLengthType is used within the entire AUTOSAR Com Stack except for bus drivers. The size of this global type depends on the maximum length of PDUs to be sent by an ECU. If no segmentation is used the length depends on the maximum payload size of a frame of the underlying communication system (for FlexRay maximum size is 255 bytes, therefore uint8). If segmentation is used it depends on the maximum length of a segmented N-PDU (in general uint16 is used). See AUTOSAR_SWS_CommunicationStackTypes for more details.
Multiplicity	1..1
Type	ENUMERATION
Default value	UINT16
Range	UINT32



	UINT16
	UINT8
Configuration class	VariantPostBuild: VariantPostBuild
Origin	AUTOSAR_ECUC

5.6.1.9. MetaDataType

Containers included		
Container name	Multiplicity	Description
MetaDataItem	1..n	The content of meta data in a Pdu consists of an ordered list of meta data items.

5.6.1.10. MetaDataItem

Parameters included	
Parameter name	Multiplicity
MetaDataMemberLength	1..1
MetaDataMemberType	1..1

Parameter Name	MetaDataMemberLength
Description	This parameter defines the length of a meta data item in bytes.
Multiplicity	1..1
Type	INTEGER
Range	<=64 >=1
Configuration class	VariantPostBuild: VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	MetaDataMemberType
Description	This parameter defines the type of a meta data item.
Multiplicity	1..1
Type	ENUMERATION



Range	ADDRESS_EXTENSION_8 CAN_ID_32 ETHERNET_MAC_64 LIN_NAD_8 PRIORITY_8 SOCKET_CONNECTION_ID_16 SOURCE_ADDRESS_16 TARGET_ADDRESS_16 PAYLOAD_TYPE_16
Configuration class	VariantPostBuild:
Origin	Elektrobit Automotive GmbH

5.6.1.11. Pdu

Containers included		
Container name	Multiplicity	Description
EcucPduDedicatedPartition	0..n	Container to overrule partition references for specific BSW modules.

Parameters included	
Parameter name	Multiplicity
PduLength	1..1
MetaDataTypeRef	0..1
EcucPduDefaultPartitionRef	0..1
SysTpduToFrameMappingRef	0..1
Pduld	0..1

Parameter Name	PduLength
Description	Length of the Pdu in bytes. It should be noted that in former AUTOSAR releases (Rel 2.1, Rel 3.0, Rel 3.1, Rel 4.0 Rev. 1) this parameter was defined in bits.
Multiplicity	1..1
Type	INTEGER



Configuration class	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	MetaDataTypeRef	
Description	Reference to meta data that is transported in the Pdu through the AUTOSAR layers.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	PreCompile:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	EcucPduDefaultPartitionRef	
Description	Reference to the EcucPartition this Pdu is associated with.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	PreCompile:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	SysTPduToFrameMappingRef	
Description	Optional reference to the PduToFrameMapping from the SystemTemplate which this Pdu represents.	
Multiplicity	0..1	
Type	FOREIGN-REFERENCE	
Configuration class	PreCompile:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	Pduld	
Description	Pdu table identifier. Parameter only required for meta data handling	
Multiplicity	0..1	
Type	INTEGER	
Range	<=65535 >=0	
Configuration class	PreCompile:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	



5.6.1.12. EcucPduDedicatedPartition

Parameters included	
Parameter name	Multiplicity
EcucPduDedicatedPartitionBswModuleRef	1..1
EcucPduDedicatedPartitionRef	1..1

Parameter Name	EcucPduDedicatedPartitionBswModuleRef	
Description	Reference to one BSW module's configuration.	
Multiplicity	1..1	
Type	CHOICE-REFERENCE	
Range	node:paths(/AUTOSAR/TOP-LEVEL-PACKAGES/*/ELEMENTS/*[(@-type='MODULE-CONFIGURATION')])	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	EcucPduDedicatedPartitionRef	
Description	Reference to the EcucPartition this Pdu is associated with within the specific BSW module.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.6.1.13. EcucPostBuildVariants

Parameters included	
Parameter name	Multiplicity
ActiveLoadableVariantRef	1..1
ActiveSelectableVariantRef	1..1

**Parameters included**

EcucPostBuildVariantRef	1..n
---	------

Parameter Name	ActiveLoadableVariantRef	
Label	Post-Build-Loadable Variant	
Description	Select the currently active Post-Build-Loadable variant.	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Range	as:paths(node:filter(node:refs('ASTyped:PredefinedVariant'), 'starts-with(as:path(.), "/EB/PostBuildLoadable"))'), true())	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	ActiveSelectableVariantRef	
Label	Post-Build-Selectable Variant	
Description	Choose the currently active Post-Build-Selectable variant. The variant must be part of <i>List of Post-Build-Selectable Variants</i> .	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Range	../EcucPostBuildVariantRef/*	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	EcucPostBuildVariantRef	
Description	Reference to a Post-Build-Selectable variant used in this project. Reference to a PredefinedVariant that defines one toplevel postBuild configuration set (covering all post-build capable BSW modules). PredefinedVariants that are referenced here shall contain only PostBuildVariantCriterionValueSets.	
Multiplicity	1..n	
Type	FOREIGN-REFERENCE	
Range	as:paths(node:filter(node:refs('ASTyped:PredefinedVariant'), 'not(starts-with(as:path(.), "/EB/PostBuildLoadable"))'), true())	
Configuration class	VariantPostBuild:	VariantPostBuild
	VariantPostBuild:	VariantPostBuild



Origin	AUTOSAR_ECUC
---------------	--------------

5.6.1.14. EcucVariationResolver

Parameters included	
Parameter name	Multiplicity
PredefinedVariantRef	1..n

Parameter Name	PredefinedVariantRef
Multiplicity	1..n
Type	FOREIGN-REFERENCE
Configuration class	PreCompile: VariantPostBuild
Origin	AUTOSAR_ECUC

5.6.1.15. PublishedInformation

Parameters included	
Parameter name	Multiplicity
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport
Label	PbcfgM support
Description	Specifies whether or not the EcuC can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.6.2. Application programming interface (API)



5.6.2.1. Macro constants

5.6.2.1.1. ECUC_COMPATIBILITY_VERSION_CHECK

Purpose	Interface compatibility check macro for EcuC library user.
Value	((ECUC_SW_MAJOR_VERSION == (ecuc_sw_major_version)) && \ (ECUC_SW_MINOR_VERSION == (ecuc_sw_minor_version)) && \ (ECUC_SW_PATCH_VERSION >= (ecuc_sw_patch_version)))

5.6.2.1.2. ECUC_METADATA_ITEM_ADDRESS_EXTENSION_8

Purpose	Address extension field CAN ID of ISO 15765-2 meta data item type.
Value	5U

5.6.2.1.3. ECUC_METADATA_ITEM_CAN_ID_32

Purpose	CAN ID according to ISO 11898-2 meta data item type.
Value	1U

5.6.2.1.4. ECUC_METADATA_ITEM_ETHERNET_MAC_64

Purpose	Ethernet MAC address meta data item type.
Value	0U

5.6.2.1.5. ECUC_METADATA_ITEM_LIN_NAD_8

Purpose	LIN node address meta data item type.
Value	6U

5.6.2.1.6. ECUC_METADATA_ITEM_PAYLOAD_TYPE_16

Purpose	DolP payload meta data item type.
----------------	-----------------------------------



Value	8U
--------------	----

5.6.2.1.7. ECUC_METADATA_ITEM_PRIORITY_8

Purpose	Priority field of SAE J1939 IDs, or Ethernet QoS meta data item type.
Value	7U

5.6.2.1.8. ECUC_METADATA_ITEM_SOCKET_CONNECTION_ID_16

Purpose	SoAd socket connection ID meta data item type.
Value	2U

5.6.2.1.9. ECUC_METADATA_ITEM_SOURCE_ADDRESS_16

Purpose	Source address of CanTp, FrTp, or DoIP meta data item type.
Value	3U

5.6.2.1.10. ECUC_METADATA_ITEM_TARGET_ADDRESS_16

Purpose	Target address of CanTp, FrTp, or DoIP data item type.
Value	4U

5.6.2.2. Functions

5.6.2.2.1. EcuC_CopyRxData

Purpose	copy received data	
Synopsis	<pre>BufReq_ReturnType EcuC_CopyRxData (EcuC_RxControlInfoType * RxControlInfoPtr , PduInfoType * RxBufferInfoPtr , const PduInfoType * PduInfoPtr , PduLengthType * RxBufferSizePtr);</pre>	
Parameters (in)	RxControlInfoPtr	control information of the passed TP-receive buffer



	RxBufferInfoPtr	pointer to the beginning of the TP-receive buffer and size of the buffer
	PduInfoPtr	Pointer to a PduInfoType which indicates the number of bytes to be copied and the location of the source data
Parameters (out)	RxBufferSizePtr	Remaining receive buffer after completion of this call
Return Value	Function execution success status	
	BUFREQ_OK:	Data has been copied
	BUFREQ_E_NOT_OK:	Request failed
Description	This function copies received data to the receive TP buffer	

5.6.2.2.2. EcuC_CopyRxDataDet

Purpose	perform parameter checks	
Synopsis	<pre>Std_ReturnType EcuC_CopyRxDataDet (const PduInfo- Type * PduInfoPtr , PduLengthType * RxBufferSizePtr);</pre>	
Parameters (in)	PduInfoPtr	Pointer to a PduInfoType which indicates the number of bytes to be copied and the location of the source data
Parameters (out)	RxBufferSizePtr	Assigned Rx buffer
Return Value	Function execution success status	
	E_OK:	parameters are valid
	E_NOT_OK:	parameter contains an unacceptable value
Description	This function performs parameter checks for <module>_CopyRxData	

5.6.2.2.3. EcuC_CopyTxData

Purpose	Copies the requested transmit data from TxBufferInfoPtr to PduInfoPtr.
Synopsis	<pre>BufReq_ReturnType EcuC_CopyTxData (EcuC_TxControlInfo- Type * TxControlInfoPtr , const PduInfoType * TxBuffer- InfoPtr , PduInfoType * PduInfoPtr , RetryInfoType * RetryInfoPtr , PduLengthType * TxDataCntPtr);</pre>



Parameters (in)	TxControlInfoPtr	Contains the control information of the passed TPtransmit buffer
	TxBufferInfoPtr	Contains pointer to the beginning of the TP-transmit buffer and size of the buffer
	PduInfoPtr	Pointer to a PduInfoType, which indicates the number of bytes to be copied (SduLength) and the location where the data have to be copied to (SduDataPtr). An SduLength of 0 is possible in order to poll the available transmit data count. In this case no data are to be copied and SduDataPtr might be invalid.
	RetryInfoPtr	TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position. NULL_PTR or !TP_DATARETRY copy the remaining data which has not been copied yet
Parameters (out)	TxDataCntPtr	Contains the length of the remaining data that needs to be copied
Return Value	Result of operation	
	BUFREQ_OK	Data has been copied to the transmit buffer completely as requested.
	BUFREQ_E_NOT_OK	Data has not been copied. Request failed, in case the corresponding I-PDU was stopped
	BUFREQ_E_BUSY	if number of bytes to copy is greater than remaining length
Description	At invocation of EcuC_CopyTxData the function copies the requested transmit data from TxBufferInfoPtr to the location specified by the PduInfoPtr. The function EcuC_CopyTxData also calculates and sets the TxDataCntPtr to the amount of remaining bytes for the transmission of this large I-PDU.	

5.6.2.2.4. EcuC_GetMetaDataAdapterExtension

Purpose	Retrieve address extension meta data item.
----------------	--



Synopsis	<pre>Std_ReturnType EcuC_GetMetaDataAddrExtension (EcuC_PduIdType PduId , const PduInfoType * PduInfoPtr , uint8 * AddrExtension);</pre>	
Parameters (in)	PduId	ID of the PDU
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
	AddrExtension	pointer to stored address extension field
Return Value	Result of operation	
	E_OK	value has been copied to addrExtension
	E_NOT_OK	addrExtension not configured for this PDU.
Description	Retrieves the address extension meta data item of the referenced PDU from the meta data memory	

5.6.2.2.5. EcuC_GetMetaDataCanId

Purpose	Retrieve Can Id meta data item.	
Synopsis	<pre>Std_ReturnType EcuC_GetMetaDataCanId (EcuC_PduIdType PduId , const PduInfoType * PduInfoPtr , uint32 * CanId);</pre>	
Parameters (in)	PduId	ID of the PDU
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
	CanId	pointer to stored CAN ID according to ISO 11898-2
Return Value	Result of operation	
	E_OK	value has been copied to canId
	E_NOT_OK	canId not configured for this PDU.
Description	Retrieves the Can Id meta data item of the referenced PDU from the meta data memory	

5.6.2.2.6. EcuC_GetMetaDataEthMac

Purpose	Retrieve Ethernet Mac Address meta data item.
----------------	---



Synopsis	<code>Std_ReturnType EcuC_GetMetaDataTableEthMac (EcuC_PduIdType PduId , const PduInfoType * PduInfoPtr , uint8 * EthMacPtr);</code>	
Parameters (in)	PduId	ID of the PDU
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
	EthMacPtr	pointer to the Ethernet MAC address
Return Value	Result of operation	
	E_OK	values has been copied to ethMacPtr
	E_NOT_OK	Ethernet MAC not configured for this PDU.
Description	Retrieves the Ethernet Mac Address meta data item of the referenced PDU from the meta data memory	

5.6.2.2.7. EcuC_GetMetaDataTableLinNad

Purpose	Retrieve Lin node address meta data item.	
Synopsis	<code>Std_ReturnType EcuC_GetMetaDataTableLinNad (EcuC_PduIdType PduId , const PduInfoType * PduInfoPtr , uint8 * LinNad);</code>	
Parameters (in)	PduId	ID of the PDU
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
	LinNad	LIN node address as used in the LIN transport protocol
Return Value	Result of operation	
	E_OK	value has been copied to linNad
	E_NOT_OK	lin Nad not configured for this PDU.
Description	Retrieves the Lin node address meta data item of the referenced PDU from the meta data memory	

5.6.2.2.8. EcuC_GetMetaDataTablePayloadType

Purpose	Retrieve payload type meta data item.	
Synopsis	<code>Std_ReturnType EcuC_GetMetaDataTablePayloadType (EcuC_PduIdType PduId , const PduInfoType * PduInfoPtr , uint16 * PldType);</code>	



Parameters (in)	PduId	ID of the PDU
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
	PldType	payload type
Return Value	Result of operation	
	E_OK	value has been copied to PldType
	E_NOT_OK	payload type not configured for this PDU.
Description	Retrieves the payload type data item of the referenced PDU from the meta data memory	

5.6.2.2.9. EcuC_GetMetaDataPriority

Purpose	Retrieve priority field meta data item.	
Synopsis	<pre>Std_ReturnType EcuC_GetMetaDataPriority (EcuC_PduIdType PduId , const PduInfoType * PduInfoPtr , uint8 * Priority);</pre>	
Parameters (in)	PduId	ID of the PDU
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
	Priority	Priority field of SAE J1939 IDs
Return Value	Result of operation	
	E_OK	value has been copied to priority
	E_NOT_OK	priority field not configured for this PDU.
Description	Retrieves the priority field meta data item of the referenced PDU from the meta data memory	

5.6.2.2.10. EcuC_GetMetaDataSoConId

Purpose	Retrieve SoAd socket connection id meta data item.	
Synopsis	<pre>Std_ReturnType EcuC_GetMetaDataSoConId (EcuC_PduIdType PduId , const PduInfoType * PduInfoPtr , uint16 * SoConId);</pre>	
Parameters (in)	PduId	ID of the PDU
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)



	SoConId	SoAd socket connection ID
Return Value	Result of operation	
	E_OK	value has been copied to SoConId
	E_NOT_OK	SoAd socket connection ID not configured for this PDU.
Description	Retrieves the SoAd socket connection id meta data item of the referenced PDU from the meta data memory	

5.6.2.2.11. EcuC_GetMetaDataSourceAddr

Purpose	Retrieve source address meta data item.	
Synopsis	<pre>Std_ReturnType EcuC_GetMetaDataSourceAddr (EcuC_PduIdType PduId , const PduInfoType * PduInfoPtr , uint16 * SrcAddr);</pre>	
Parameters (in)	PduId	ID of the PDU
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
	SrcAddr	Source address of CanTp, FrTp, or DolP transport protocol messages, or of SAE J1939 messages
Return Value	Result of operation	
	E_OK	value has been copied to srcAddr
	E_NOT_OK	Source Address not configured for this PDU.
Description	Retrieves the source address meta data item of the referenced PDU from the meta data memory	

5.6.2.2.12. EcuC_GetMetaDataTargetAddr

Purpose	Retrieve target address meta data item.	
Synopsis	<pre>Std_ReturnType EcuC_GetMetaDataTargetAddr (EcuC_PduIdType PduId , const PduInfoType * PduInfoPtr , uint16 * TargetAddr);</pre>	
Parameters (in)	PduId	ID of the PDU
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)



	TargetAddr	Target address of CanTp, FrTp, or DolP transport protocol messages, or destination address of SAE J1939 messages
Return Value	Result of operation	
	E_OK	value has been copied to TargetAddr
	E_NOT_OK	Target address not configured for this PDU.
Description	Retrieves the target address meta data item of the referenced PDU from the meta data memory	

5.6.2.2.13. EcuC_InitRx

Purpose	Initialize RxInit.	
Synopsis	<code>void EcuC_InitRx (EcuC_RxControlInfoType * RxControlInfoPtr);</code>	
Parameters (in)	RxControlInfoPtr	control information of the passed TP-receive buffer
Description	This function initializes the control data structure of the receive buffer, e.g. state of the receive buffer.	

5.6.2.2.14. EcuC_InitTx

Purpose	Initialize transmit buffer.	
Synopsis	<code>void EcuC_InitTx (EcuC_TxControlInfoType * TxControlInfoPtr);</code>	
Parameters (in)	TxControlInfoPtr	Contains the control information of the passed TPtransmit buffer
Description	Initializes the control data structure of the transmit buffer, e.g. state of the transmit buffer.	

5.6.2.2.15. EcuC_RxBufferIsLocked

Purpose	current status of reception
Synopsis	<code>boolean EcuC_RxBufferIsLocked (EcuC_RxControlInfoType * RxControlInfoPtr);</code>



Parameters (in)	RxControlInfoPtr	control information of the passed TP-receive buffer
Return Value	Function execution success status	
	TRUE:	Reception is currently ongoing
	FALSE:	Reception finished
Description	This function Indicates if reception is currently ongoing or received data is stopped	

5.6.2.2.16. EcuC_SetMetaDataAddrExtension

Purpose	Set address extension meta data.	
Synopsis	Std_ReturnType EcuC_SetMetaDataAddrExtension (EcuC_PduId- Type PduId , PduInfoType * PduInfoPtr , uint8 AddrExtension);	
Parameters (in)	PduId	ID of the PDU
	AddrExtension	addrExtension
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
Return Value	Result of operation	
	E_OK	value has been saved
	E_NOT_OK	addrExtension not configured for this PDU
Description	Sets the address extension meta data item of the referenced PDU in the meta data memory	

5.6.2.2.17. EcuC_SetMetaDataCanId

Purpose	Set Can id meta data item.	
Synopsis	Std_ReturnType EcuC_SetMetaDataCanId (EcuC_PduId- Type PduId , PduInfoType * PduInfoPtr , uint32 CanId);	
Parameters (in)	PduId	ID of the PDU
	CanId	CAN ID
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
Return Value	Result of operation	
	E_OK	value has been saved



	E_NOT_OK	canId not configured for this PDU
Description	Sets the Can id meta data item of the referenced PDU in the meta data memory	

5.6.2.2.18. EcuC_SetMetaDataMember

Purpose	Set Ethernet Mac Address meta data.	
Synopsis	<pre>Std_ReturnType EcuC_SetMetaDataMember (EcuC_PduIdType PduId , PduInfoType * PduInfoPtr , const uint8 * EthMacPtr);</pre>	
Parameters (in)	PduId	ID of the PDU
	EthMacPtr	Ethernet Mac Address
Parameters (out)	PduInfoPtr	
	pointer to the PDU info including the pointer to the meta data (currently not used)	
Return Value	Result of operation	
	E_OK	value has been saved
	E_NOT_OK	ethMac not configured for this PDU
Description	Sets the Ethernet Mac Address meta data item of the referenced PDU in the meta data memory	

5.6.2.2.19. EcuC_SetMetaDataMember

Purpose	Set Lin node address meta data item.	
Synopsis	<pre>Std_ReturnType EcuC_SetMetaDataMember (EcuC_PduIdType PduId , PduInfoType * PduInfoPtr , uint8 LinNad);</pre>	
Parameters (in)	PduId	ID of the PDU
	LinNad	LIN node address as used in the LIN transport protocol
Parameters (out)	PduInfoPtr	
	pointer to the PDU info including the pointer to the meta data (currently not used)	
Return Value	Result of operation	
	E_OK	value has been saved
	E_NOT_OK	linNad not configured for this PDU
Description	Sets the Lin node address meta data item of the referenced PDU in the meta data memory	



5.6.2.2.20. EcuC_SetMetaDataPayloadType

Purpose	Set payload type meta data item.	
Synopsis	<pre>Std_ReturnType EcuC_SetMetaDataPayloadType (EcuC_PduId- Type PduId , PduInfoType * PduInfoPtr , uint16 PldType);</pre>	
Parameters (in)	PduId	ID of the PDU
	PldType	payload type
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
Return Value	Result of operation	
	E_OK	value has been saved
	E_NOT_OK	PldType not configured for this PDU
Description	Sets the payload type meta data item of the referenced PDU in the meta data memory	

5.6.2.2.21. EcuC_SetMetaDataPriority

Purpose	Sets priority field meta data item.	
Synopsis	<pre>Std_ReturnType EcuC_SetMetaDataPriority (EcuC_PduId- Type PduId , PduInfoType * PduInfoPtr , uint8 Priority);</pre>	
Parameters (in)	PduId	ID of the PDU
	priority	Priority field of SAEI
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
Return Value	Result of operation	
	E_OK	value has been saved
	E_NOT_OK	priority not configured for this PDU
Description	Sets the priority field meta data item of the referenced PDU in the meta data memory	

5.6.2.2.22. EcuC_SetMetaDataSoConId

Purpose	Set SoAd socket connection ID meta data item.	
Synopsis	<pre>Std_ReturnType EcuC_SetMetaDataSoConId (EcuC_PduId- Type PduId , PduInfoType * PduInfoPtr , uint16 SoConId);</pre>	



Parameters (in)	PduId	ID of the PDU
	SoConId	SoAd socket connection ID
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
Return Value	Result of operation	
	E_OK	value has been saved
	E_NOT_OK	SoConId not configured for this PDU
Description	Sets the SoAd socket connection ID meta data item of the referenced PDU in the meta data memory	

5.6.2.2.23. EcuC_SetMetaDataSourceAddr

Purpose	Set source address meta data item.	
Synopsis	Std_ReturnType EcuC_SetMetaDataSourceAddr (EcuC_PduId- Type PduId , PduInfoType * PduInfoPtr , uint16 SrcAddr);	
Parameters (in)	PduId	ID of the PDU
	SrcAddr	Source address of CanTp, FrTp, or DolP transport protocol messages, or of SAE J1939 messages
Parameters (out)	pointer to the PDU info including the pointer to the meta data (currently not used)	
Return Value	Result of operation	
	E_OK	value has been saved
	E_NOT_OK	srcAddr not configured for this PDU
Description	Sets the source address meta data item of the referenced PDU in the meta data memory	

5.6.2.2.24. EcuC_SetMetaDataTargetAddr

Purpose	Set target address meta data item.	
Synopsis	Std_ReturnType EcuC_SetMetaDataTargetAddr (EcuC_PduId- Type PduId , PduInfoType * PduInfoPtr , uint16 TargetAddr);	
Parameters (in)	PduId	ID of the PDU



	TargetAddr	Target address of CanTp, FrTp, or DoIP transport protocol messages, or destination address of SAE J1939 messages
Parameters (out)	PduInfoPtr	pointer to the PDU info including the pointer to the meta data (currently not used)
Return Value	Result of operation	
	E_OK	value has been saved
	E_NOT_OK	TargetAddr not configured for this PDU
Description	Sets the target address meta data item of the referenced PDU in the meta data memory	

5.6.2.2.25. EcuC_StartOfReception

Purpose	Start of reception.	
Synopsis	<pre>BufReq_ReturnType EcuC_StartOfReception (EcuC_RxControlInfoType * RxControlInfoPtr , PduInfoType * RxBufferInfoPtr , PduLengthType TpSduLength , PduLengthType * RxBufferSizePtr);</pre>	
Parameters (in)	RxControlInfoPtr	control information of the passed TP-receive buffer
	RxBufferInfoPtr	pointer to the beginning of the TP-receive buffer and size of the buffer
	TpSduLength	Complete length of the TP I-PDU to be received
Parameters (out)	RxBufferSizePtr	Pointer to the size of internal TP-receive buffer
Return Value	Function execution success status	
	BUFREQ_OK:	Connection has been accepted.
	BUFREQ_E_NOT_OK:	Connection has been rejected
	BUFREQ_E_OVFL:	In case the receive buffer size is smaller than TpSduLength
Description	This function returns the pointer to the size of the internal receive buffer	

5.6.2.2.26. EcuC_StartOfReceptionDet

Purpose	perform parameter checks
----------------	--------------------------



Synopsis	<code>Std_ReturnType EcuC_StartOfReceptionDet (PduLengthType TpSduLength , PduLengthType * RxBufferSizePtr);</code>	
Parameters (in)	TpSduLength	Complete length of the TP I-PDU to be received
Parameters (out)	RxBufferSizePtr	Assigned Rx buffer
Return Value	Function execution success status	
	E_OK:	parameters are valid
	E_NOT_OK:	parameter contains an unacceptable value
Description	This function performs parameter checks for <module>_StartOfReception	

5.6.2.2.27. EcuC_TpRxIndication

Purpose	indicates completion of reception	
Synopsis	<code>Std_ReturnType EcuC_TpRxIndication (EcuC_RxControlInfoType * RxControlInfoPtr , NotifResultType Result , PduLengthType * CopiedDataSize);</code>	
Parameters (in)	RxControlInfoPtr	control information of the passed TP-receive buffer
	Result	Result of PDU reception
		copied message length
Parameters (in,out)		copied message length
Return Value	Function execution success status	
	E_OK:	message successfully received
	E_NOT_OK:	error occurred during reception
Description	This function indicates that the reception has been completed	

5.6.2.2.28. EcuC_TpTransmit

Purpose	Transmission of a TP message.
Synopsis	<code>Std_ReturnType EcuC_TpTransmit (EcuC_TxControlInfoType * TxControlInfoPtr , PduIdType PduId , const PduInfoType * TxBufferInfoPtr , EcuC_LoTpTransmitFunctPtrType LoTpTransmitFunctPtr);</code>



Parameters (in)	TxControlInfoPtr	Contains the control information of the passed TPtransmit buffer
	PduId	Identification of the I-PDU.
	TxBufferInfoPtr	Length and pointer to the buffer of the I-PDU
	LoTpTransmitFunctPtr	Transmit function of the lower layer e.g. PduR_ComTransmit
Return Value	Result of operation	
	E_OK	Request is accepted
	E_NOT_OK	Request is not accepted e.g. transmission currently ongoing
Description	This function triggers the transmission of a TP message.	

5.6.2.2.29. EcuC_TpTxConfirmation

Purpose	Indication that the transmission has been completed.	
Synopsis	Std_ReturnType EcuC_TpTxConfirmation (EcuC_TxControlInfoType * TxControlInfoPtr , NotifResultType Result);	
Parameters (in)	TxControlInfoPtr	Contains the control information of the passed TPtransmit buffer
	Result	NTFRSLT_OK: the complete I-PDU has been transmitted <ANY other="" value="">: the I-PDU has not been transmitted, the transmit buffer can be unlocked
Return Value	Result of operation	
	E_OK	Message successfully transmitted
	E_NOT_OK	Error occurred during transmission, data was not fully transmitted
Description	EcuC_TpTxConfirmation is called to indicate that the transmission has been completed.	

5.6.2.2.30. EcuC_TxBufferIsLocked

Purpose	Indicates the state of the transmission.
----------------	--



Synopsis	<pre>boolean EcuC_TxBufferIsLocked (EcuC_- TxControlInfoType * TxControlInfoPtr);</pre>	
Parameters (in)	TxControlInfoPtr	Contains the control information of the passed TPtransmit buffer
Return Value	Result of operation	
	TRUE	Transmission is currently ongoing.
	FALSE	Transmission is finished
Description	Indicates if transmission is currently ongoing (buffer is locked) or transmission is stopped (Idle).	

5.6.3. Integration notes

5.6.3.1. Exclusive areas

Exclusive areas information is not available for this module.

5.6.3.2. Production errors

Production errors are not reported by the `EcuC` module.

5.6.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section [Memory mapping and compiler abstraction](#) in the [Integration notes](#) section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE
VAR_NO_INIT_64
VAR_INIT_UNSPECIFIED
CONST_UNSPECIFIED



5.6.3.4. Integration requirements

WARNING

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.6.3.4.1. EcuC.EB_CALLSEQREQ_EcuC_0001

Description	Before calling EcuC_CopyRxData() it must be ensured by EcuC_CopyRxDataDet() that the arguments PduInfoPtr and rxBufferSizePtr are valid.
--------------------	--

5.6.3.4.2. EcuC.EB_CALLSEQREQ_EcuC_0002

Description	It needs to be ensured that rxControlInfoPtr and rxBufferInfoPtr are not NULL_POINTERS, when calling EcuC_CopyRxData().
--------------------	---

5.6.3.4.3. EcuC.EB_CALLSEQREQ_EcuC_0003

Description	Before calling EcuC_CopyTxData() it must be verified by EcuC_CopyTxDataDet() that the arguments RetryInfoPtr, TxDATAcntPtr and PduInfoPtr are valid.
--------------------	--

5.6.3.4.4. EcuC.EB_CALLSEQREQ_EcuC_0004

Description	It needs to be ensured that txControlInfoPtr and txBufferInfoPtr are not NULL_POINTERS, when calling EcuC_CopyTxData()
--------------------	--

5.6.3.4.5. EcuC.EB_CALLSEQREQ_EcuC_0005

Description	It needs to be ensured that rxControlInfoPtr is not a NULL_POINTER, when calling EcuC_TpRxIndication().
--------------------	---

5.6.3.4.6. EcuC.EB_CALLSEQREQ_EcuC_0006

Description	Before calling EcuC_TpRxIndication() it must be ensured by EcuC_TpRxIndication-Det() that copiedDataSize is valid.
--------------------	--



5.6.3.4.7. EcuC.EB_CALLSEQREQ_EcuC_0007

Description	It needs to be ensured that txControlInfoPtr, txBufferInfoPtr and IoTpTransmitFunctPtr are not NULL_POINTERS, when calling EcuC_TpTransmit().
-------------	---

5.6.3.4.8. EcuC.EB_CALLSEQREQ_EcuC_0008

Description	It needs to be ensured that rxControlInfoPtr and rxBufferInfoPtr are not NULL_POINTERS, when calling EcuC_StartOfReception().
-------------	---

5.6.3.4.9. EcuC.EB_CALLSEQREQ_EcuC_0009

Description	Before calling EcuC_StartOfReception() it must be ensured by EcuC_StartOfReceptionDet() that the argument RxBufferSizePtr is valid.
-------------	---

5.6.3.4.10. EcuC.EB_CALLSEQREQ_EcuC_0010

Description	It needs to be ensured that txControlInfoPtr is not a NULL_POINTER, when calling EcuC_InitTx().
-------------	---

5.6.3.4.11. EcuC.EB_CALLSEQREQ_EcuC_0011

Description	It needs to be ensured that rxControlInfoPtr is not a NULL_POINTER, when calling EcuC_InitRx().
-------------	---

5.6.3.4.12. EcuC.EB_CALLSEQREQ_EcuC_0012

Description	It needs to be ensured that txControlInfoPtr is not a NULL_POINTER, when calling EcuC_TxBufferIsLocked().
-------------	---

5.6.3.4.13. EcuC.EB_CALLSEQREQ_EcuC_0013

Description	It needs to be ensured that rxControlInfoPtr is not a NULL_POINTER, when calling EcuC_RxBufferIsLocked().
-------------	---

5.6.3.4.14. EcuC.EB_CALLSEQREQ_EcuC_0014

Description	It needs to be ensured that txControlInfoPtr is not a NULL_POINTER, when calling EcuC_TpTxConfirmation().
-------------	---



5.7. MemMap

5.7.1. Configuration parameters

Containers included		
Container name	Multiplicity	Description
CommonPublishedInformation	1..1	Label: Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
MemMapAddressingModeSet	0..n	Defines a set of addressing modes which might apply to a SwAddrMethod.
MemMapAllocation	0..n	Defines the mapping of MemMapAddressingModeSets to SwAddrMethods.
MemMapAS40Compatibility	1	
MemMapGenerateEmptyHeaderFile	1..1	
MemMapHeaderFiles	1..1	
MemMapValidateMappings	1..1	
MemMapValidateSections	1..1	
PublishedInformation	1..1	Label: EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included	
Parameter name	Multiplicity
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT
Label	Config Variant
Multiplicity	1..1



Type	ENUMERATION
Default value	VariantPreCompile
Range	VariantPreCompile

5.7.1.1. CommonPublishedInformation

Parameters included	
Parameter name	Multiplicity
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL



Default value	4
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH
Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH
Parameter Name	SwMajorVersion
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH
Parameter Name	SwMinorVersion
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH
Parameter Name	SwPatchVersion
Label	Software Patch Version



Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ModuleId
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	195
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	VendorId
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AU-TOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	Release
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH



5.7.1.2. MemMapAddressingModeSet

Containers included		
Container name	Multiplicity	Description
MemMapAddressing-Mode	1..n	Defines a addressing mode with a set of #pragma statements implementing the start and the stop of a section.
Parameters included		
Parameter name	Multiplicity	
MemMapSupportedAddressingMethodOption	0..n	
MemMapSupportedMemoryAllocationKeywordPolicy	0..n	
MemMapSupportedSectionInitializationPolicy	0..n	
MemMapSupportedSectionType	0..n	
Parameter Name	MemMapSupportedAddressingMethodOption	
Description	This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods. The attribute option of a swAddrMethod mapped via MemMapGenericMapping to this MemMapAddressingModeSet shall be equal to one of the configured MemMapSupportedAddressMethodOption's.	
Multiplicity	0..n	
Type	STRING	
Configuration class	PreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	
Parameter Name	MemMapSupportedMemoryAllocationKeywordPolicy	
Description	This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods. The attribute MemoryAllocationKeywordPolicy of a swAddrMethod mapped via MemMapGenericMapping to this MemMapAddressingModeSet shall be equal to one of the configured MemMapSupportedMemoryAllocationKeywordPolicy's. <ul style="list-style-type: none"> ▶ MEMMAP_ALLOCATION_KEYWORD_POLICY_ADDR_METHOD_SHORT_NAME: The Memory Allocation Keyword is build with the short name of the SwAddrMethod. This is the default value if the attribute does not exist in the SwAddrMethod. 	



	▶ MEMMAP_ALLOCATION_KEYWORD_POLICY_ADDR_METHOD_SHORT_NAME_AND_ALIGNMENT: The Memory Allocation Keyword is build with the the short name of the SwAddrMethod and the alignment attribute of the MemorySection. This requests a separation of objects in memory dependent from the alignment and is not applicable for RunnableEntitys and BswSchedulableEntitys.
Multiplicity	0..n
Type	ENUMERATION
Range	MEMMAP_ALLOCATION_KEYWORD_POLICY_ADDR_METHOD_SHORT_NAME MEMMAP_ALLOCATION_KEYWORD_POLICY_ADDR_METHOD_SHORT_NAME_AND_ALIGNMENT
Configuration class	PreCompile: VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	MemMapSupportedSectionInitializationPolicy
Description	<p>This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods. The sectionInitializationPolicy attribute value of a swAddrMethod mapped via MemMapGenericMapping to this MemMapAddressingModeSet shall be equal to one of the configured MemMapSupportedSectionInitializationPolicy's. Please note that SectionInitializationPolicyType describes the intended initialization of MemorySections. The following values are standardized in AUTOSAR Methodology:</p> <ul style="list-style-type: none"> ▶ NO-INIT: No initialization and no clearing is performed. Such data elements must not be read before one has written a value into it. * ""INIT"": To be used for data that are initialized by every reset to the specified value (initValue). ▶ POWER-ON-INIT: To be used for data that are initialized by 'Power On' to the specified value (initValue). Note: there might be several resets between power on resets. ▶ CLEARED: To be used for data that are initialized by every reset to zero. ▶ POWER-ON-CLEARED: To be used for data that are initialized by 'Power On' to zero. Note: there might be several resets between power on resets. Please note that the values are defined similar to the representation of enumeration types in the XML schema to ensure backward compatibility.
Multiplicity	0..n
Type	STRING
Configuration class	PreCompile: VariantPreCompile



Origin	AUTOSAR_ECUC
---------------	--------------

Parameter Name	MemMapSupportedSectionType
Description	<p>This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods. The attribute sectionType of a swAddrMethod mapped via MemMapGenericMapping or MemMapSectionSpecificMapping to this MemMapAddressingModeSet shall be equal to one of the configured MemMapSupportedSectionType's.</p> <ul style="list-style-type: none"> ▶ MEMMAP_SECTION_TYPE_CALIBRATION_OFFLINE: Program data which can only be used for offline calibration. Note: This value is deprecated and shall be substituted by calPrm. ▶ MEMMAP_SECTION_TYPE_CALIBRATION_ONLINE: Program data which can be used for online calibration. Note: This value is deprecated and shall be substituted by calPrm. ▶ MEMMAP_SECTION_TYPE_CAL_PRM: To be used for calibratable constants of ECU-functions. ▶ MEMMAP_SECTION_TYPE_CODE: To be used for mapping code to application block, boot block, external flash etc. ▶ MEMMAP_SECTION_TYPE_CONFIG_DATA: Constants with attributes that show that they reside in one segment for module configuration. ▶ MEMMAP_SECTION_TYPE_CONST: To be used for global or static constants. ▶ MEMMAP_SECTION_TYPE_EXCLUDE_FROM_FLASH: Values existing in the ECU but not dropped down in the binary file. No upload should be needed to obtain access to the ECU data. The ECU will never be touched by the instrumentation tool, with the exception of upload. These are memory areas which are not overwritten by downloading the executable. ▶ MEMMAP_SECTION_TYPE_USER_DEFINED: No specific categorization of sectionType possible. Note: This value is deprecated and shall be substituted by var, code, const, calPrm, configData, excludeFromFlash and the appropriate values of the orthogonal attributes sectionInitializationPolicy, memoryAllocationKeywordPolicy and option. ▶ MEMMAP_SECTION_TYPE_VAR: To be used for global or static variables. The expected initialization is specified with the attribute sectionInitializationPolicy. ▶ MEMMAP_SECTION_TYPE_VAR_FAST: To be used for all global or static variables that have at least one of the following properties: - accessed bit-wise - frequently used - high number of accesses in source code Some plat-



	<p>forms allow the use of bit instructions for variables located in this specific RAM area as well as shorter addressing instructions. This saves code and runtime. Note: This value is deprecated and shall be substituted by var and the appropriate values of the orthogonal attributes sectionInitializationPolicy, memoryAllocationKeywordPolicy and option.</p> <ul style="list-style-type: none"> ▶ <code>MEMMAP_SECTION_TYPE_VAR_NO_INIT</code>: To be used for all global or static variables that are never initialized. Note: This value is deprecated and shall be substituted by var and the appropriate values of the orthogonal attributes sectionInitializationPolicy, memoryAllocationKeywordPolicy and option. ▶ <code>MEMMAP_SECTION_TYPE_VAR_POWER_ON_INIT</code>: To be used for all global or static variables that are initialized only after power on reset. Note: This value is deprecated and shall be substituted by var and the appropriate values of the orthogonal attributes sectionInitializationPolicy, memoryAllocationKeywordPolicy and option.
Multiplicity	0..n
Type	ENUMERATION
Range	<code>MEMMAP_SECTION_TYPE_CALIBRATION_OFFLINE</code> <code>MEMMAP_SECTION_TYPE_CALIBRATION_ONLINE</code> <code>MEMMAP_SECTION_TYPE_CAL_PRM</code> <code>MEMMAP_SECTION_TYPE_CODE</code> <code>MEMMAP_SECTION_TYPE_CONFIG_DATA</code> <code>MEMMAP_SECTION_TYPE_CONST</code> <code>MEMMAP_SECTION_TYPE_EXCLUDE_FROM_FLASH</code> <code>MEMMAP_SECTION_TYPE_USER_DEFINED</code> <code>MEMMAP_SECTION_TYPE_VAR</code> <code>MEMMAP_SECTION_TYPE_VAR_FAST</code> <code>MEMMAP_SECTION_TYPE_VAR_NO_INIT</code> <code>MEMMAP_SECTION_TYPE_VAR_POWER_ON_INIT</code>
Configuration class	PreCompile: <code>VariantPreCompile</code>
Origin	AUTOSAR_ECUC

5.7.1.3. MemMapAddressingMode

Parameters included	
Parameter name	Multiplicity



Parameters included

MemMapAddressing-ModeStart	1..1
MemMapAddressing-ModeStop	1..1
MemMapAlignmentSelector	1..n

Parameter Name	MemMapAddressingModeStart	
Description	Defines a set of #pragma statements implementing the start of a section.	
Multiplicity	1..1	
Type	MULTILINE-STRING	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	MemMapAddressingModeStop	
Description	Defines a set of #pragma statements implementing the end of a section.	
Multiplicity	1..1	
Type	MULTILINE-STRING	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	MemMapAlignmentSelector
Description	<p>Defines the alignments for which the MemMapAddressingMode applies. If at least one alignment of the MemMapAlignmentSelector fits to alignment attribute of the MemorySection, the content of this MemMapAddressingMode is used to implement the start and the stop of a section.</p> <p>Please note that the same MemMapAddressingMode can be applicable for several alignments, e.g. "8" bit and "UNSPECIFIED".</p> <p>Allowed values are:</p> <ul style="list-style-type: none"> ▶ Numerical values (decimal, hexadecimal, octal, dual) ▶ BOOLEAN ▶ UNSPECIFIED ▶ UNKNOWN ▶ or an empty value (matches MemorySections without the alignment attribute)



Multiplicity	1..n	
Type	STRING	
Default value		
Configuration class	PreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.7.1.4. MemMapAllocation

Containers included		
Container name	Multiplicity	Description
MemMapGenericMapping	0..n	Defines which SwAddrMethod is implemented with which MemMapAddressingModeSet.
MemMapSectionSpecificMapping	0..n	Defines which MemorySection of a BSW Module or a Software Component is implemented with which MemMapAddressingModeSet.

5.7.1.5. MemMapGenericMapping

Parameters included	
Parameter name	Multiplicity
MemMapSwAddressMethodRef	1..1
MemMapAddressingModeSetRef	1..1

Parameter Name	MemMapSwAddressMethodRef	
Description	Reference to the SwAddrMethod which applies to the MemMapGenericMapping.	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Configuration class	VariantPreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	MemMapAddressingModeSetRef	
Description	Reference to the MemMapAddressingModeSet which applies to the MemMapGenericMapping.	



Multiplicity	1..1
Type	REFERENCE
Configuration class	VariantPreCompile: VariantPreCompile
Origin	AUTOSAR_ECUC

5.7.1.6. MemMapSectionSpecificMapping

Parameters included	
Parameter name	Multiplicity
<u>MemMapMemorySectionRef</u>	1..1
<u>MemMapAddressingModeSetRef</u>	1..1

Parameter Name	MemMapMemorySectionRef
Description	Reference to the MemorySection which applies to the MemMapSectionSpecificMapping.
Multiplicity	1..1
Type	FOREIGN-REFERENCE
Configuration class	VariantPreCompile: VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	MemMapAddressingModeSetRef
Description	Reference to the MemMapAddressingModeSet which applies to the MemMapModuleSectionSpecificMapping.
Multiplicity	1..1
Type	REFERENCE
Configuration class	VariantPreCompile: VariantPreCompile
Origin	AUTOSAR_ECUC

5.7.1.7. MemMapAS40Compatibility

Parameters included	
Parameter name	Multiplicity

**Parameters included**

MemMapAS40Compatibility	1
---	---

Parameter Name	MemMapAS40Compatibility
Description	<ul style="list-style-type: none"> ▶ If enabled, MemMap macros for the MemorySection CONFIG_DATA are defined as [PREFIX]_[START STOP]_SEC_CONFIG_DATA_[ALIGNMENT] and PREFIX_[START STOP]_CONFIG_DATA_[ALIGNMENT] ▶ If disabled, MemMap macros for the MemorySection CONFIG_DATA are defined as [PREFIX]_[START STOP]_SEC_CONFIG_DATA_[ALIGNMENT]
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive Software

5.7.1.8. MemMapGenerateEmptyHeaderFile**Parameters included**

Parameter name	Multiplicity
MemMapGenerateEmptyHeaderFile	1..1

Parameter Name	MemMapGenerateEmptyHeaderFile
Description	<ul style="list-style-type: none"> ▶ If enabled, the MemMap header empty files for BSW/SWC modules with no memory sections will be generated. ▶ If disabled, the MemMap header empty files for BSW/SWC modules with no memory sections will not be generated. <p>When the BSW and/or SWC implementations do not have any memory sections defined, the header files are generated empty. The files ({Mip}_MemMap.h and {componentTypeName}_MemMap.h) will not contain any memory allocation keywords and report a MEMMAP_ERROR if they are included. In order to prevent this generation the MemMapGenerateEmptyHeaderFile parameter can be used.</p> <p>When the BSW module BswModuleDescription shortName is equal to the SWC module componentTypeName the common generated header file will be empty (will not contain any memory allocation keywords) if both the BSW and SWC implementation do not have any memory sections defined. If either one of the mod-</p>



	ules have memory sections defined the common header file will always be generated even if the MemMapGenerateEmptyHeaderFile parameter is disabled.
	The MemMap.h file is not affected by the MemMapGenerateEmptyHeaderFile parameter and will always be generated.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.7.1.9. MemMapHeaderFiles

Parameters included	
Parameter name	Multiplicity
MemMapHeaderFiles	0..n

Parameter Name	MemMapHeaderFiles
Description	A list of additional header files included by the generated MemMap.h. The files are included in alphabetical order.
Multiplicity	0..n
Type	STRING
Configuration class	PreCompile: VariantPreCompile
Origin	Elektrobit Automotive Software

5.7.1.10. MemMapValidateMappings

Parameters included	
Parameter name	Multiplicity
MemMapValidateMappings	1..1

Parameter Name	MemMapValidateMappings
Label	Enable Log Entries for Generic and Specific Mappings



Description	<p>Enables log entries, warnings and errors, for the invalid MemMapGenericMappings and MemMapSectionSpecificMappings.</p> <ul style="list-style-type: none"> ▶ If enabled, warnings and errors are reported for the invalid MemMapGenericMappings and MemMapSectionSpecificMappings. ▶ If disabled, invalid MemMapGenericMappings and MemMapSectionSpecificMappings will be silently ignored. <p>Warnings are reported if:</p> <ul style="list-style-type: none"> ▶ The SwAddrMethod referenced in MemMapGenericMapping has different attributes as the ones configured in MemMapAddressingModeSet. ▶ The SwAddrMethod referenced in MemMapGenericMapping is not referenced by any of the MemorySection defined in the system description. ▶ The SwAddrMethod referenced in MemMapGenericMapping is from a different package than the one referenced by the MemorySections. ▶ The MemMapAlignmentSelector does not contain the same alignment as the one defined for the MemorySection, for which the memory mapping was created. <p>Errors are reported if:</p> <ul style="list-style-type: none"> ▶ More than one MemMapGenericMapping references the same MemMapSwAddressMethodRef. ▶ More than one MemMapSectionSpecificMapping references the same MemMapMemorySectionRef.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive Software

5.7.1.11. MemMapValidateSections

Parameters included	
Parameter name	Multiplicity
MemMapValidateCoreScope	1..1

**Parameters included**

MemMapValidateSafety	1..1
MemMapValidateSections	1..1

Parameter Name	MemMapValidateCoreScope
Description	<p>Enables the coreScope validation.</p> <ul style="list-style-type: none"> ▶ If enabled, the usage of coreScope is validated ▶ If disabled, the usage of coreScope is not validated <p>Supported coreScope keywords:</p> <ul style="list-style-type: none"> ▶ GLOBAL - default value ▶ LOCAL <p>Supported coreScope options:</p> <ul style="list-style-type: none"> ▶ coreGlobal - default value ▶ coreLocal <p>Errors are reported if:</p> <ul style="list-style-type: none"> ▶ The coreScope is set multiple times. ▶ CoreLocal is not set with the correct SwAddrMethod sectionInitializationPolicy (Cleared or INIT). ▶ CoreLocal is not present in both the name and options of the MemorySection and the SwAddrMethod's options.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive Software

Parameter Name**MemMapValidateSafety**

Description	Enables the safety level validation.
	<ul style="list-style-type: none"> ▶ If enabled, the usage of safety levels is validated ▶ If disabled, the usage of safety levels is not validated



	<p>Supported safety level keywords:</p> <ul style="list-style-type: none"> ▶ QM - default value ▶ ASIL_A ▶ ASIL_B ▶ ASIL_C ▶ ASIL_D <p>Supported safety level options:</p> <ul style="list-style-type: none"> ▶ safetyQM - default value ▶ safetyAsilA ▶ safetyAsilB ▶ safetyAsilC ▶ safetyAsilD <p>Errors are reported if:</p> <ul style="list-style-type: none"> ▶ The safety level is set multiple times. ▶ The safety level is not present in both the name and options of the MemorySection and the SwAddrMethod's options.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive Software

Parameter Name	MemMapValidateSections
Description	<ul style="list-style-type: none"> ▶ If enabled, the memory sections will be checked that they are opened and closed in the right order ▶ If disabled, the memory sections will not be checked that they are opened and closed in the right order
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	VariantPreCompile: VariantPreCompile
Origin	Elektrobit Automotive Software



5.7.1.12. PublishedInformation

Parameters included	
Parameter name	Multiplicity
PbcfgMSupport	1..1
Parameter Name	
Label	PbcfgM support
Description	Specifies whether or not the MemMap can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.7.2. Application programming interface (API)

The `MemMap` module does not have any public API.

5.7.3. Integration notes

5.7.3.1. Exclusive areas

Exclusive areas are not used by the `MemMap` module.

5.7.3.2. Production errors

Production errors are not reported by the `MemMap` module.

5.7.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.



The `MemMap` module does not define any memory sections to be mapped.

5.7.3.4. Integration requirements

WARNING Integration requirements list is not exhaustive



The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.7.3.4.1. `lim.MemMap.EB_INTREQ_MemMap_0001`

Description	When a module has a memory section referencing the <code>SwAddrMethod</code> <code>CODE</code> in both the basic software and software component description, the same <code>MemMapAddressingMode</code> should be used for both memory sections when creating the <code>MemMapSectionSpecificMappings</code> .
Rationale	The memory sections <code>CODE</code> need to be mapped to the same <code>MemMapAddressingMode</code> to ensure the mapping is consistent.

5.8. PcfgM

5.8.1. Configuration parameters

Containers included		
Container name	Multiplicity	Description
<u>CommonPublishedInformation</u>	1..1	Label: Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
<u>PublishedInformation</u>	1..1	Label: EB Published Information Additional published parameters not covered by <code>CommonPublishedInformation</code> container.
<u>PcfgMBswModules</u>	1..n	Each container references a module configuration and module instance which are merged to one <code>PcfgM</code> configuration.

**Containers included**

PbcfgMGeneral	1..1	This container specifies the general configuration parameters of the PbcfgM.
-------------------------------	------	--

Parameters included

Parameter name	Multiplicity
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT
Label	Config Variant
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPostBuild
Range	VariantPostBuild

5.8.1.1. CommonPublishedInformation**Parameters included**

Parameter name	Multiplicity
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.



Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwMajorVersion
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH



Parameter Name	SwMinorVersion
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwPatchVersion
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	20
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ModuleId
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	32769
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	VendorId
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AU-TOSAR vendor list
Multiplicity	1..1



Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	Release
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.8.1.2. PublishedInformation

Parameters included	
Parameter name	Multiplicity
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport
Label	PbcfgM support
Description	Specifies whether or not the PbcfgM can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.8.1.3. PbcfgMBswModules

Parameters included	
Parameter name	Multiplicity

**Parameters included**

PbcfgMModuleInstanceID	0..1
PbcfgMBswModuleRef	1..1

Parameter Name**PbcfgMModuleInstanceID****Description** Instance ID of the referenced module**Multiplicity** 0..1**Type** INTEGER**Default value** 0**Range** >=0

<=255

Configuration class **PreCompile:** VariantPostBuild**Origin** Elektrobit Automotive GmbH**Parameter Name****PbcfgMBswModuleRef****Description** This is a reference to one BSW module's configuration.**Multiplicity** 1..1**Type** CHOICE-REFERENCE**Range** node:paths(/AUTOSAR/TOP-LEVEL-PACKAGES/*/ELEMENTS/*[(@-type='MODULE-CONFIGURATION') and (IMPLEMENTATION_CONFIG_VARIANT = 'VariantPostBuild') and (PublishedInformation/PbcfgMSupport = 'true')])**Configuration class** **VariantPostBuild:** VariantPostBuild**Origin** Elektrobit Automotive GmbH**5.8.1.4. PbcfgMGeneral****Parameters included**

Parameter name	Multiplicity
PbcfgMDevErrorDetect	1..1
PbcfgMRelocatableCfgEnable	1..1



Parameters included

PbcfgMConstCfgAddress	1..1
PbcfgMBinarySupportEnable	1..1
PbcfgMMapisValidFunctionToMemSection	1..1

Parameter Name	PbcfgMDevErrorDetect
Description	Switches the Development Error Detection and Notification ON or OFF. Optimization Effect: <ul style="list-style-type: none">▶ ROM reduction (code): Disabling this parameter reduces the ROM consumption of the module code.▶ Execution time reduction (code): Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	VariantPostBuild: VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name

PbcfgMRelocatableCfgEnable

Description	Enables or disables the post-build-time configuration data to be used either by relative offsets to the configuration start address (relocatable) or by absolute pointers (not relocatable). ▶ TRUE: Relocateable configuration is in use (switched on). ▶ FALSE: Relocateable configuration is not in use (switched off). Optimization Effect: <ul style="list-style-type: none">▶ ROM reduction (config): Enabling this parameter reduces the ROM consumption of the module configuration.▶ Execution time reduction (code): Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN



Default value	true	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	
Parameter Name	PbcfgMConstCfgAddress	
Description	Defines the fix address where the configuration starts.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	>=0 <4294967295	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	
Parameter Name	PbcfgMBinarySupportEnable	
Description	Enables or disables the support for generation of binary post build configuration (i.e. generation of the Motorola srec file)	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	
Parameter Name	PbcfgMMapisValidFunctionToMemSection	
Description	Enables or disables the memory mapping of the <mod>_IsValidConfig functions to a separate memory section.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.8.2. Application programming interface (API)



5.8.2.1. Functions

5.8.2.1.1. PbcfgM_GetConfig

Purpose	Get a module configuration from the PbcfgM.	
Synopsis	<pre>Std_ReturnType PbcfgM_GetConfig (uint16 PbcfgM_- ModuleId , uint16 PbcfgM_InstanceID , PbcfgM_- ModuleConfigPtrType * PbcfgM_ModuleConfigPtr);</pre>	
Service ID	0x02	
Sync/Async	synchronous	
Reentrancy	reentrant	
Parameters (in)	PbcfgM_ModuleId	Module ID
	PbcfgM_InstanceID	Instance ID, if a module has just one instance zero shall be used as default value.
Parameters (out)	PbcfgM_ModuleConfigPtr	Pointer to the requested module configuration. The pointer is not changed if the configuration is not found.
Return Value	Function execution success status	
	E_OK	In the case a configuration was found.
	E_NOT_OK	In the case no configuration was found.
Description	Used to get a module configuration from the post build configuration manager.	

5.8.2.1.2. PbcfgM_Init

Purpose	Initializes all PbcfgM global variables.	
Synopsis	<pre>void PbcfgM_Init (const PbcfgM_- ConfigType * PbcfgM_ConfigPtr);</pre>	
Service ID	0x01	
Sync/Async	synchronous	
Reentrancy	non reentrant	
Parameters (in)	PbcfgM_ConfigPtr	Configuration pointer to post build configuration of the PbcfgM
Description	Initializes all PbcfgM global variables including default values, default selector field and state of timeout monitors.	



5.8.2.1.3. PbcfgM_IsValidConfig

Purpose	Checks if the PbcfgM configuration is valid.	
Synopsis	<pre>Std_ReturnType PbcfgM_IsValidConfig (const PbcfgM_ConfigType * PbcfgM_ConfigPtr);</pre>	
Sync/Async	synchronous	
Reentrancy	reentrant	
Parameters (in)	PbcfgM_ConfigPtr	Pointer to the PbcfgM configuration.
Return Value	Function execution status.	
	E_OK	In the case the PbcfgM and all referenced module configurations are valid.
	E_NOT_OK	In the case the PbcfgM or one of the referenced module configuration is invalid.
Description	Checks if the PbcfgM configuration and all referenced configurations are valid.	

5.8.3. Integration notes

5.8.3.1. Exclusive areas

Exclusive areas are not used by the PbcfgM module.

5.8.3.2. Production errors

Production errors are not reported by the PbcfgM module.

5.8.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section [Memory mapping and compiler abstraction](#) in the [Integration notes](#) section for details.

The following table provides the list of sections that may be mapped for this module:

**Memory section**

CODE

VAR_FAST_INIT_UNSPECIFIED

VAR_FAST_INIT_8

CONST_32

CONST_UNSPECIFIED

CONFIG_DATA_UNSPECIFIED

5.8.3.4. Integration requirements

WARNING**Integration requirements list is not exhaustive**

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the PbcfgM module.



6. Bibliography

Bibliography

- [1] *AUTOSAR BSW Makefile Interface*, Issue Version 0.3 Draft, Publisher: AUTOSAR
- [2] *AUTOSAR System Template*, Issue Version 4.2.0, Release 4.0, Revision 3, Publisher: AUTOSAR
- [3] *LIN Specification Package Revision 2.0*, Publish date: September 23, 2003, Publisher: LIN Consortium
- [4] *AUTOSAR System Template*, Issue Version 4.6.0, Release 4.2, Revision 2, Publisher: AUTOSAR
- [5] *AUTOSAR Specification of Memory Mapping*, Issue Version 1.4.0, Release 4.0, Revision 3, Publisher: AUTOSAR
- [6] *AUTOSAR Specification of Memory Mapping*, Issue Version 1.4.0, Release 4.3, Revision 1, Publisher: AUTOSAR