

AC784xx_MCAL_API_Reference_Manual

V5.0.0-440

Generated by Doxygen 1.8.13

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	6
3.1	File List	6
4	Class Documentation	8
4.1	_Crypto_QueueType Struct Reference	8
4.1.1	Detailed Description	8
4.1.2	Member Data Documentation	8
4.1.2.1	JobPtr	8
4.1.2.2	Next	9
4.2	Acmp_ConfigType Struct Reference	9
4.2.1	Detailed Description	9
4.2.2	Member Data Documentation	9
4.2.2.1	HwConfigPtr	9
4.2.2.2	MaxChannelNum	10
4.3	Can_BitrateConfigType Struct Reference	10
4.3.1	Detailed Description	10
4.3.2	Member Data Documentation	10
4.3.2.1	PRESC	10
4.3.2.2	SEG_1	10
4.3.2.3	SEG_2	11

4.3.2.4	SJW	11
4.4	Can_ConfigType Struct Reference	11
4.4.1	Detailed Description	11
4.4.2	Member Data Documentation	11
4.4.2.1	CanHOHPtr	11
4.4.2.2	ControlerNumber	12
4.4.2.3	ControllerDescriptorsPtr	12
4.4.2.4	HohNumber	12
4.5	Can_ControllerBaudrateConfigType Struct Reference	12
4.5.1	Detailed Description	12
4.5.2	Member Data Documentation	12
4.5.2.1	ControllerBaudRateConfigID	13
4.5.2.2	ControllerBaudRateKbps	13
4.5.2.3	ControllerBitrate	13
4.5.2.4	ControllerFDConfig	13
4.6	Can_ControllerDescriptorType Struct Reference	13
4.6.1	Detailed Description	14
4.6.2	Member Data Documentation	14
4.6.2.1	AutoBusOffRecover	14
4.6.2.2	BusoffInterrupt	14
4.6.2.3	CanControllerActivation	14
4.6.2.4	Channel	14
4.6.2.5	ControlerID	14
4.6.2.6	ControllerBaudrateConfigsPtr	15
4.6.2.7	DefaultBaudRateIndex	15
4.6.2.8	ECUMWakeupSourceId	15
4.6.2.9	MaxBaudRateCount	15
4.6.2.10	MemEccEn	15
4.6.2.11	OverflowMode	15
4.6.2.12	RxInterrupt	16
4.6.2.13	TimeStampEn	16

4.6.2.14	TxInterrupt	16
4.6.2.15	WakeUpEn	16
4.6.2.16	WakeUpInterrupt	16
4.7	Can_ControllerFdConfigType Struct Reference	16
4.7.1	Detailed Description	17
4.7.2	Member Data Documentation	17
4.7.2.1	CanBrsEn	17
4.7.2.2	CanContrTrcvDelayCompensation	17
4.7.2.3	CanFdBitrate	17
4.7.2.4	CanFdEnable	17
4.7.2.5	CanFdIsoMode	18
4.8	Can_FilterControlType Struct Reference	18
4.8.1	Detailed Description	18
4.8.2	Member Data Documentation	18
4.8.2.1	Code	18
4.8.2.2	IdType	18
4.8.2.3	Mask	19
4.9	Can_HardwareObjectType Struct Reference	19
4.9.1	Detailed Description	19
4.9.2	Member Data Documentation	19
4.9.2.1	CanControllerRef	19
4.9.2.2	CanFdPaddingValue	20
4.9.2.3	CanFilterListPtr	20
4.9.2.4	CanHandle	20
4.9.2.5	CanHWObjectCount	20
4.9.2.6	CanHwObjectUsesPolling	20
4.9.2.7	CanIdType	20
4.9.2.8	CanMainFunctionRWPeriodRef	21
4.9.2.9	CanObjectIndex	21
4.9.2.10	CanRwMode	21
4.9.2.11	FilterNumber	21

4.10 Can_HwType Struct Reference	21
4.10.1 Detailed Description	21
4.10.2 Member Data Documentation	22
4.10.2.1 CanId	22
4.10.2.2 ControllerId	22
4.10.2.3 Hoh	22
4.11 Can_PduType Struct Reference	22
4.11.1 Detailed Description	22
4.11.2 Member Data Documentation	23
4.11.2.1 id	23
4.11.2.2 length	23
4.11.2.3 sdu	23
4.11.2.4 swPduHandle	23
4.12 Crc_DmaConfig Struct Reference	23
4.12.1 Detailed Description	24
4.12.2 Member Data Documentation	24
4.12.2.1 DmaChannel	24
4.12.2.2 DmaUsed	24
4.12.2.3 Params	24
4.13 Crypto_AlgorithmInfoType Struct Reference	24
4.13.1 Detailed Description	25
4.13.2 Member Data Documentation	25
4.13.2.1 family	25
4.13.2.2 keyLength	25
4.13.2.3 mode	25
4.13.2.4 secondaryFamily	26
4.14 Crypto_JobInfoType Struct Reference	26
4.14.1 Detailed Description	26
4.14.2 Member Data Documentation	26
4.14.2.1 JobId	26
4.14.2.2 JobPriority	27

4.15 Crypto_JobPrimitiveInfoType Struct Reference	27
4.15.1 Detailed Description	27
4.15.2 Member Data Documentation	27
4.15.2.1 callbackId	27
4.15.2.2 callbackUpdateNotification	28
4.15.2.3 crylfKeyId	28
4.15.2.4 primitiveInfo	28
4.15.2.5 processingType	28
4.16 Crypto_JobPrimitiveInputOutputType Struct Reference	28
4.16.1 Detailed Description	29
4.16.2 Member Data Documentation	29
4.16.2.1 crylfKeyId	29
4.16.2.2 input64	29
4.16.2.3 inputLength	30
4.16.2.4 inputPtr	30
4.16.2.5 mode	30
4.16.2.6 output64Ptr	30
4.16.2.7 outputLengthPtr	30
4.16.2.8 outputPtr	31
4.16.2.9 secondaryInputLength	31
4.16.2.10 secondaryInputPtr	31
4.16.2.11 secondaryOutputLengthPtr	31
4.16.2.12 secondaryOutputPtr	31
4.16.2.13 targetCrylfKeyId	32
4.16.2.14 tertiaryInputLength	32
4.16.2.15 tertiaryInputPtr	32
4.16.2.16 verifyPtr	32
4.17 Crypto_JobRedirectionInfoType Struct Reference	32
4.17.1 Detailed Description	33
4.17.2 Member Data Documentation	33
4.17.2.1 inputKeyElementId	33

4.17.2.2	inputKeyId	33
4.17.2.3	outputKeyElementId	33
4.17.2.4	outputKeyId	34
4.17.2.5	redirectionConfig	34
4.17.2.6	secondaryInputKeyElementId	34
4.17.2.7	secondaryInputKeyId	34
4.17.2.8	secondaryOutputKeyElementId	34
4.17.2.9	secondaryOutputKeyId	35
4.17.2.10	tertiaryInputKeyElementId	35
4.17.2.11	tertiaryInputKeyId	35
4.18	Crypto_JobType Struct Reference	35
4.18.1	Detailed Description	36
4.18.2	Member Data Documentation	36
4.18.2.1	cryptoKeyId	36
4.18.2.2	jobId	36
4.18.2.3	jobInfo	36
4.18.2.4	jobPrimitiveInfo	36
4.18.2.5	jobPrimitiveInputOutput	37
4.18.2.6	jobRedirectionInfoRef	37
4.18.2.7	jobState	37
4.19	Crypto_Key Struct Reference	37
4.19.1	Detailed Description	37
4.19.2	Member Data Documentation	38
4.19.2.1	Count	38
4.19.2.2	KeyId	38
4.19.2.3	KeyTypePtr	38
4.19.2.4	KeyValid	38
4.20	Crypto_ObjectType Struct Reference	38
4.20.1	Detailed Description	39
4.20.2	Member Data Documentation	39
4.20.2.1	DriverObjectId	39

4.20.2.2	HeadOfFree	39
4.20.2.3	HeadOfJobs	39
4.20.2.4	PrimitivesCount	40
4.20.2.5	PrimitivesPtr	40
4.20.2.6	QueuedJobs	40
4.20.2.7	QueueSize	40
4.21	Crypto_PrimitiveInfoType Struct Reference	40
4.21.1	Detailed Description	41
4.21.2	Member Data Documentation	41
4.21.2.1	algorithm	41
4.21.2.2	resultLength	41
4.21.2.3	service	41
4.22	Crypto_PrimitiveType Struct Reference	42
4.22.1	Detailed Description	42
4.22.2	Member Data Documentation	42
4.22.2.1	Family	42
4.22.2.2	Mode	42
4.22.2.3	SecondaryFamily	43
4.22.2.4	Service	43
4.23	CryptoKeyElement Struct Reference	43
4.23.1	Detailed Description	43
4.23.2	Member Data Documentation	43
4.23.2.1	ActualSize	44
4.23.2.2	AllowPartialAccess	44
4.23.2.3	Id	44
4.23.2.4	KeyFormatType	44
4.23.2.5	MaxSize	44
4.23.2.6	Persist	44
4.23.2.7	ReadAccess	45
4.23.2.8	Uniqueld	45
4.23.2.9	ValuePtr	45

4.23.2.10 WriteAccess	45
4.24 Dio_ChannelGroupType Struct Reference	45
4.24.1 Detailed Description	46
4.24.2 Member Data Documentation	46
4.24.2.1 Mask	46
4.24.2.2 Offset	46
4.24.2.3 Port	46
4.25 Dio_ConfigType Struct Reference	46
4.25.1 Detailed Description	47
4.25.2 Member Data Documentation	47
4.25.2.1 ChannelGroupList	47
4.25.2.2 NumChannelGroups	47
4.26 Fee_BlockHeaderType Struct Reference	47
4.26.1 Detailed Description	48
4.26.2 Member Data Documentation	48
4.26.2.1 Assigned	48
4.26.2.2 BlockId	48
4.26.2.3 CheckSum	48
4.26.2.4 ImmediateBlock	49
4.26.2.5 Invalid	49
4.26.2.6 Length	49
4.26.2.7 TargetAddr	49
4.26.2.8 Valid	49
4.27 Fee_BlockInfoType Struct Reference	50
4.27.1 Detailed Description	50
4.27.2 Member Data Documentation	50
4.27.2.1 HeadAddr	50
4.27.2.2 Header	50
4.27.2.3 Status	50
4.28 Fee_BlockType Struct Reference	51
4.28.1 Detailed Description	51

4.28.2	Member Data Documentation	51
4.28.2.1	BlockAssign	51
4.28.2.2	BlockNumber	52
4.28.2.3	BlockSize	52
4.28.2.4	ClusterGrp	52
4.28.2.5	ImmediateData	52
4.29	Fee_ClusterGroupType Struct Reference	52
4.29.1	Detailed Description	53
4.29.2	Member Data Documentation	53
4.29.2.1	BlockPtr	53
4.29.2.2	ClusterCount	53
4.29.2.3	ClusterPtr	53
4.29.2.4	ReservedSize	54
4.30	Fee_ClusterHeaderType Struct Reference	54
4.30.1	Detailed Description	54
4.30.2	Member Data Documentation	54
4.30.2.1	Checksum	54
4.30.2.2	ClusterId	55
4.30.2.3	Invalid	55
4.30.2.4	Length	55
4.30.2.5	StartAddr	55
4.30.2.6	Valid	55
4.31	Fee_ClusterInfoType Struct Reference	56
4.31.1	Detailed Description	56
4.31.2	Member Data Documentation	56
4.31.2.1	Header	56
4.31.2.2	Status	56
4.32	Fee_ClusterType Struct Reference	56
4.32.1	Detailed Description	57
4.32.2	Member Data Documentation	57
4.32.2.1	Length	57

4.32.2.2 StartAddr	57
4.33 Fee_ConfigType Struct Reference	57
4.33.1 Detailed Description	58
4.33.2 Member Data Documentation	58
4.33.2.1 ClusterGrpPtr	58
4.33.2.2 GrpCount	58
4.34 Fee_JobInfoType Struct Reference	58
4.34.1 Detailed Description	59
4.34.2 Member Data Documentation	59
4.34.2.1 BlockIdx	59
4.34.2.2 BufferValid	59
4.34.2.3 ClusterIdx	59
4.34.2.4 DataLength	60
4.34.2.5 DataOffset	60
4.34.2.6 DataPtr	60
4.34.2.7 GroupIdx	60
4.34.2.8 Mode	60
4.34.2.9 ModuleStatus	61
4.34.2.10 OldState	61
4.34.2.11 Result	61
4.34.2.12 SrcDataPtr	61
4.34.2.13 State	61
4.35 Fee_JobPendingInfoType Struct Reference	62
4.35.1 Detailed Description	62
4.35.2 Member Data Documentation	62
4.35.2.1 BlockNumber	62
4.35.2.2 BlockOffset	62
4.35.2.3 Length	63
4.35.2.4 Mode	63
4.35.2.5 PendingJob	63
4.35.2.6 PendingValid	63

4.35.2.7 RdDataPtr	63
4.35.2.8 WrDataPtr	64
4.36 Fls_ConfigType Struct Reference	64
4.36.1 Detailed Description	64
4.36.2 Member Data Documentation	64
4.36.2.1 DefaultMode	65
4.36.2.2 EraseTimeout	65
4.36.2.3 JobEndNotificationPtr	65
4.36.2.4 JobErrorNotificationPtr	65
4.36.2.5 MaxReadFastMode	65
4.36.2.6 MaxReadNormalMode	65
4.36.2.7 MaxWriteFastMode	66
4.36.2.8 MaxWriteNormalMode	66
4.36.2.9 PCallBackPtr	66
4.36.2.10 PDataType	66
4.36.2.11 PErasePtr	66
4.36.2.12 PWritePtr	66
4.36.2.13 SectorCount	67
4.36.2.14 SectorEndAddr	67
4.36.2.15 SectorFlags	67
4.36.2.16 SectorPageSize	67
4.36.2.17 SectorSize	67
4.36.2.18 SectorStartAddr	67
4.36.2.19 WriteTimeout	68
4.37 FlsTst_BlockConfigType Struct Reference	68
4.37.1 Detailed Description	68
4.37.2 Member Data Documentation	68
4.37.2.1 BlockBaseAddress	68
4.37.2.2 BlockIndex	68
4.37.2.3 BlockSize	69
4.37.2.4 NumberOfTestedCells	69

4.37.2.5	SignatureAddress	69
4.37.2.6	TestAlgorithm	69
4.38	FIsTst_CommonVariableType Struct Reference	69
4.38.1	Detailed Description	70
4.38.2	Member Data Documentation	70
4.38.2.1	BgndBlockCellLength	70
4.38.2.2	BgndBlockCellNumber	70
4.38.2.3	BgndBlockCellSignatureValue	70
4.38.2.4	BgndBlockCellStartId	70
4.38.2.5	BgndBlockIndex	70
4.38.2.6	BgndLastSignature	71
4.38.2.7	BgndOverallResult	71
4.38.2.8	BgndTestAbortOrSuspendFlag	71
4.38.2.9	FgndLastResult	71
4.38.2.10	FgndLastSignature	71
4.38.2.11	TestCompletion	71
4.38.2.12	TestIntervalld	72
4.39	FIsTst_ConfigType Struct Reference	72
4.39.1	Detailed Description	72
4.39.2	Member Data Documentation	72
4.39.2.1	BgndBlockConfig	72
4.39.2.2	BgndBlockMaxNumber	72
4.39.2.3	FgndBlockConfig	73
4.39.2.4	FgndBlockMaxNumber	73
4.39.2.5	TestCompletedNotification	73
4.40	FIsTst_ErrorDetailsType Struct Reference	73
4.40.1	Detailed Description	73
4.40.2	Member Data Documentation	73
4.40.2.1	EccFaultAddress	74
4.40.2.2	EccStatus	74
4.41	FIsTst_TestResultBgndType Struct Reference	74

4.41.1 Detailed Description	74
4.41.2 Member Data Documentation	74
4.41.2.1 TestIntervalld	74
4.41.2.2 TestResultBgnd	75
4.42 FlsTst_TestSignatureBgndType Struct Reference	75
4.42.1 Detailed Description	75
4.42.2 Member Data Documentation	75
4.42.2.1 TestIntervalld	75
4.42.2.2 TestSignatureValue	75
4.43 FlsTst_TestSignatureFgndType Struct Reference	76
4.43.1 Detailed Description	76
4.43.2 Member Data Documentation	76
4.43.2.1 TestSignatureValue	76
4.44 I2c_CHConfigType Struct Reference	76
4.44.1 Detailed Description	76
4.44.2 Member Data Documentation	77
4.44.2.1 ChConfigPtr	77
4.44.2.2 HwChannel	77
4.45 I2c_ConfigType Struct Reference	77
4.45.1 Detailed Description	77
4.45.2 Member Data Documentation	77
4.45.2.1 ChannelConfigPtr	77
4.45.2.2 MaxChannelNum	78
4.46 Icu_ChannelConfigType Struct Reference	78
4.46.1 Detailed Description	78
4.46.2 Member Data Documentation	78
4.46.2.1 Channel	78
4.46.2.2 ChannelNotification	79
4.46.2.3 DefaultStartEdge	79
4.46.2.4 HwSelect	79
4.46.2.5 LogicId	79

4.46.2.6	MeasurementMode	79
4.46.2.7	ModuleId	80
4.46.2.8	SignalMeasurementPropertyType	80
4.46.2.9	TimestampBufferType	80
4.46.2.10	WakeupCapable	80
4.46.2.11	WakeupSource	80
4.47	Icu_ConfigType Struct Reference	81
4.47.1	Detailed Description	81
4.47.2	Member Data Documentation	81
4.47.2.1	ChannelConfigPtr	81
4.47.2.2	IpConfigPtr	81
4.48	Icu_DutyCycleType Struct Reference	81
4.48.1	Detailed Description	82
4.48.2	Member Data Documentation	82
4.48.2.1	ActiveTime	82
4.48.2.2	PeriodTime	82
4.49	Icu_IpConfigType Struct Reference	82
4.49.1	Detailed Description	83
4.49.2	Member Data Documentation	83
4.49.2.1	PwmModuleConfigPtr	83
4.49.2.2	PwmModuleCount	83
4.50	Icu_PwmModuleConfigType Struct Reference	83
4.50.1	Detailed Description	83
4.50.2	Member Data Documentation	84
4.50.2.1	ClockSource	84
4.50.2.2	Instance	84
4.50.2.3	Prescaler	84
4.51	Lin_ConfigType Struct Reference	84
4.51.1	Detailed Description	84
4.51.2	Member Data Documentation	85
4.51.2.1	Lin_HwConfigPtr	85

4.52 Mcu_ConfigType Struct Reference	85
4.52.1 Detailed Description	85
4.52.2 Member Data Documentation	85
4.52.2.1 Mcu_ClkConfigNum	85
4.52.2.2 Mcu_ClockConfigPtr	86
4.52.2.3 Mcu_CommonConfigPtr	86
4.52.2.4 Mcu_DemConfigPtr	86
4.52.2.5 Mcu_ModeConfigNum	86
4.52.2.6 Mcu_ModeConfigPtr	86
4.52.2.7 Mcu_RamConfigNum	86
4.52.2.8 Mcu_RamConfigPtr	87
4.53 Ocu_ChannelConfigType Struct Reference	87
4.53.1 Detailed Description	87
4.53.2 Member Data Documentation	87
4.53.2.1 Channel	87
4.53.2.2 ChannelDefaultThreshold	87
4.53.2.3 ChannelInitLevel	88
4.53.2.4 ChannelNotification	88
4.53.2.5 ChannelOutputPinUsed	88
4.53.2.6 ChannelPinAction	88
4.53.2.7 LogicId	88
4.53.2.8 PwmModulesId	88
4.54 Ocu_ConfigType Struct Reference	89
4.54.1 Detailed Description	89
4.54.2 Member Data Documentation	89
4.54.2.1 IpConfig	89
4.54.2.2 OcuChannelConfig	89
4.54.2.3 OcuChannelCount	89
4.55 Ocu_IpConfigType Struct Reference	89
4.55.1 Detailed Description	90
4.55.2 Member Data Documentation	90

4.55.2.1	OcuModuleConfig	90
4.55.2.2	OcuModuleCount	90
4.56	Ocu_ModuleConfigType Struct Reference	90
4.56.1	Detailed Description	90
4.56.2	Member Data Documentation	91
4.56.2.1	ClockSource	91
4.56.2.2	ModuleId	91
4.56.2.3	ModuleMaxValue	91
4.56.2.4	Prescaler	91
4.57	Port_ConfigType Struct Reference	91
4.57.1	Detailed Description	92
4.57.2	Member Data Documentation	92
4.57.2.1	DigitalFilterConfig	92
4.57.2.2	NumDigitalFilterPorts	92
4.57.2.3	NumPins	92
4.57.2.4	NumUnusedPins	93
4.57.2.5	UnusedPadConfig	93
4.57.2.6	UnusedPads	93
4.57.2.7	UsedPadConfig	93
4.58	Port_DigitalFilterConfigType Struct Reference	93
4.58.1	Detailed Description	94
4.58.2	Member Data Documentation	94
4.58.2.1	Clock	94
4.58.2.2	PinMask	94
4.58.2.3	PortID	94
4.58.2.4	Width	94
4.59	Port_PinConfigType Struct Reference	95
4.59.1	Detailed Description	95
4.59.2	Member Data Documentation	95
4.59.2.1	DataOut	95
4.59.2.2	DirChangeable	95

4.59.2.3	Direction	96
4.59.2.4	GPIOMode	96
4.59.2.5	ModeChangeable	96
4.59.2.6	PCR	96
4.59.2.7	Pin	96
4.60	Port_UnUsedPinConfigType Struct Reference	97
4.60.1	Detailed Description	97
4.60.2	Member Data Documentation	97
4.60.2.1	DataOut	97
4.60.2.2	Direction	97
4.60.2.3	PCR	97
4.61	Pwm_ChannelConfigType Struct Reference	98
4.61.1	Detailed Description	98
4.61.2	Member Data Documentation	98
4.61.2.1	Channel	98
4.61.2.2	ChannelClass	98
4.61.2.3	ChannelMode	99
4.61.2.4	ChannelNotification	99
4.61.2.5	DeadTimeValue	99
4.61.2.6	EnableCombaineComple	99
4.61.2.7	EnableDeadTime	99
4.61.2.8	EnableInterrupt	99
4.61.2.9	EnableMatchTrigger	100
4.61.2.10	IdleState	100
4.61.2.11	LogicId	100
4.61.2.12	Polarity	100
4.61.2.13	PwmDefaultDutycycle	100
4.61.2.14	PwmModulesId	100
4.61.2.15	TimePsc	101
4.62	Pwm_ConfigType Struct Reference	101
4.62.1	Detailed Description	101

4.62.2	Member Data Documentation	101
4.62.2.1	ChannelCfg	101
4.62.2.2	IpConfigPtr	101
4.62.2.3	PwmChannelCount	102
4.63	Pwm_IpConfigType Struct Reference	102
4.63.1	Detailed Description	102
4.63.2	Member Data Documentation	102
4.63.2.1	ModuleCfg	102
4.63.2.2	PwmModuleCount	102
4.64	Pwm_ModuleConfigType Struct Reference	103
4.64.1	Detailed Description	103
4.64.2	Member Data Documentation	103
4.64.2.1	ClockSource	103
4.64.2.2	CountMode	103
4.64.2.3	EnableInitTrigger	103
4.64.2.4	EnableInterrupt	104
4.64.2.5	EnableMaxTrigger	104
4.64.2.6	EnableOverflowEvent	104
4.64.2.7	Instance	104
4.64.2.8	OverflowNotification	104
4.64.2.9	Period	104
4.64.2.10	Prescaler	105
4.64.2.11	TriggerRatio	105
4.65	Sent_ConfigType Struct Reference	105
4.65.1	Detailed Description	105
4.65.2	Member Data Documentation	105
4.65.2.1	HwConfigPtr	105
4.65.2.2	MaxChannelNum	106
4.65.2.3	ModuleConfig	106
4.66	Uart_ConfigType Struct Reference	106
4.66.1	Detailed Description	106
4.66.2	Member Data Documentation	106
4.66.2.1	HwConfigPtr	106
4.66.2.2	MaxChannelNum	107
4.67	Wdg_ConfigType Struct Reference	107
4.67.1	Detailed Description	107
4.67.2	Member Data Documentation	107
4.67.2.1	DefaultMode	107
4.67.2.2	ModeSetting	107

5	File Documentation	108
5.1	Adc.h File Reference	108
5.1.1	Detailed Description	109
5.1.2	Macro Definition Documentation	109
5.1.2.1	ADC_DEINIT_ID	110
5.1.2.2	ADC_DISABLEGROUPNOTIFICATION_ID	110
5.1.2.3	ADC_DISABLEHARDWARETRIGGER_ID	110
5.1.2.4	ADC_E_ALREADY_INITIALIZED	110
5.1.2.5	ADC_E_BUFFER_UNINIT	110
5.1.2.6	ADC_E_BUSY	110
5.1.2.7	ADC_E_IDLE	111
5.1.2.8	ADC_E_NOT_DISENGAGED	111
5.1.2.9	ADC_E_NOTIF_CAPABILITY	111
5.1.2.10	ADC_E_PARAM_CONFIG	111
5.1.2.11	ADC_E_PARAM_GROUP	111
5.1.2.12	ADC_E_PARAM_POINTER	111
5.1.2.13	ADC_E_PERIPHERAL_NOT_PREPARED	112
5.1.2.14	ADC_E_POWER_STATE_NOT_SUPPORTED	112
5.1.2.15	ADC_E_TRANSITION_NOT_POSSIBLE	112
5.1.2.16	ADC_E_UNINIT	112
5.1.2.17	ADC_E_WRONG_CONV_MODE	112
5.1.2.18	ADC_E_WRONG_TRIGG_SRC	112
5.1.2.19	ADC_ENABLEGROUPNOTIFICATION_ID	113
5.1.2.20	ADC_ENABLEHARDWARETRIGGER_ID	113
5.1.2.21	ADC_GETCURRENTPOWERSTATE_ID	113
5.1.2.22	ADC_GETGROUPSTATUS_ID	113
5.1.2.23	ADC_GETSTREAMLASTPOINTER_ID	113
5.1.2.24	ADC_GETTARGETPOWERSTATE_ID	113
5.1.2.25	ADC_GETVERSIONINFO_ID	114
5.1.2.26	ADC_INIT_ID	114
5.1.2.27	ADC_INSTANCE	114

5.1.2.28	ADC_PREPAREPOWERSTATE_ID	114
5.1.2.29	ADC_READGROUP_ID	114
5.1.2.30	ADC_SETPOWERSTATE_ID	114
5.1.2.31	ADC_SETUPRESULTBUFFER_ID	115
5.1.2.32	ADC_STARTGROUPCONVERSION_ID	115
5.1.2.33	ADC_STOPGROUPCONVERSION_ID	115
5.1.3	Function Documentation	115
5.1.3.1	Adc_DeInit()	115
5.1.3.2	Adc_DisableGroupNotification()	115
5.1.3.3	Adc_DisableHardwareTrigger()	116
5.1.3.4	Adc_EnableGroupNotification()	116
5.1.3.5	Adc_EnableHardwareTrigger()	116
5.1.3.6	Adc_GetGroupStatus()	117
5.1.3.7	Adc_GetStreamLastPointer()	117
5.1.3.8	Adc_GetVersionInfo()	118
5.1.3.9	Adc_Init()	118
5.1.3.10	Adc_ReadGroup()	118
5.1.3.11	Adc_SetupResultBuffer()	119
5.1.3.12	Adc_StartGroupConversion()	119
5.1.3.13	Adc_StopGroupConversion()	119
5.1.4	Variable Documentation	120
5.1.4.1	Adc_GenerateConfigPC	120
5.2	Can.h File Reference	120
5.2.1	Detailed Description	122
5.2.2	Macro Definition Documentation	122
5.2.2.1	CAN_E_DATALOST	123
5.2.2.2	CAN_E_ECC_ERROR	123
5.2.2.3	CAN_E_ICOM_CONFIG_INVALID	123
5.2.2.4	CAN_E_INIT_FAILED	123
5.2.2.5	CAN_E_INVALID_CONTROLLER	123
5.2.2.6	CAN_E_INVALID_CONTROLLER_MODE	123

5.2.2.7	CAN_E_PARAM_BAUDRATE	124
5.2.2.8	CAN_E_PARAM_CONTROLLER	124
5.2.2.9	CAN_E_PARAM_DATA_LENGTH	124
5.2.2.10	CAN_E_PARAM_HANDLE	124
5.2.2.11	CAN_E_PARAM_POINTER	124
5.2.2.12	CAN_E_TRANSITION	124
5.2.2.13	CAN_E_UNINIT	125
5.2.2.14	CAN_INSTANCE_ID	125
5.2.2.15	CAN_SID_CBK_CHECK_WAKEUP	125
5.2.2.16	CAN_SID_CHANGE_BAUDRATE	125
5.2.2.17	CAN_SID_CHECK_BAUDRATE	125
5.2.2.18	CAN_SID_DEINIT	125
5.2.2.19	CAN_SID_DISABLE_CONTROLLER_INTERRUPTS	126
5.2.2.20	CAN_SID_ENABLE_CONTROLLER_INTERRUPTS	126
5.2.2.21	CAN_SID_GET_VERSION_INFO	126
5.2.2.22	CAN_SID_GETCONTROLLERERRORSTATE	126
5.2.2.23	CAN_SID_GETCONTROLLERMODE	126
5.2.2.24	CAN_SID_GetControllerRxErrorCounter	126
5.2.2.25	CAN_SID_GetControllerTxErrorCounter	127
5.2.2.26	CAN_SID_INIT	127
5.2.2.27	CAN_SID_MAIN_FUNCTION_BUS_OFF	127
5.2.2.28	CAN_SID_MAIN_FUNCTION_MODE	127
5.2.2.29	CAN_SID_MAIN_FUNCTION_READ	127
5.2.2.30	CAN_SID_MAIN_FUNCTION_WAKEUP	127
5.2.2.31	CAN_SID_MAIN_FUNCTION_WRITE	128
5.2.2.32	CAN_SID_SET_BAUDRATE	128
5.2.2.33	CAN_SID_SET_CONTROLLER_MODE	128
5.2.2.34	CAN_SID_WRITE	128
5.2.3	Enumeration Type Documentation	128
5.2.3.1	Can_HandleType	128
5.2.3.2	Can_ModeType	129

5.2.3.3	Can_OverflowModeType	129
5.2.3.4	Can_PduldType	129
5.2.3.5	Can_StatusType	129
5.2.4	Function Documentation	130
5.2.4.1	Can_CheckWakeup()	130
5.2.4.2	Can_DeInit()	130
5.2.4.3	Can_DisableControllerInterrupts()	131
5.2.4.4	Can_EnableControllerInterrupts()	131
5.2.4.5	Can_GetControllerErrorState()	132
5.2.4.6	Can_GetControllerMode()	132
5.2.4.7	Can_GetControllerRxErrorCounter()	132
5.2.4.8	Can_GetControllerTxErrorCounter()	133
5.2.4.9	Can_GetVersionInfo()	133
5.2.4.10	Can_Init()	134
5.2.4.11	Can_MainFunction_BusOff()	134
5.2.4.12	Can_MainFunction_Mode()	135
5.2.4.13	Can_MainFunction_Read()	135
5.2.4.14	Can_MainFunction_Wakeup()	136
5.2.4.15	Can_MainFunction_Write()	136
5.2.4.16	Can_SetBaudrate()	136
5.2.4.17	Can_SetControllerMode()	137
5.2.4.18	Can_Write()	137
5.2.5	Variable Documentation	138
5.2.5.1	Can_GenerateConfigPB	138
5.3	Can_GeneralTypes.h File Reference	138
5.3.1	Detailed Description	139
5.3.2	Macro Definition Documentation	139
5.3.2.1	CAN_BUSY	139
5.3.2.2	CAN_CANCTRL_MAX_PAYLOAD12_U8	139
5.3.2.3	CAN_CANCTRL_MAX_PAYLOAD16_U8	139
5.3.2.4	CAN_CANCTRL_MAX_PAYLOAD20_U8	139

5.3.2.5	CAN_CANCTRL_MAX_PAYLOAD24_U8	140
5.3.2.6	CAN_CANCTRL_MAX_PAYLOAD32_U8	140
5.3.2.7	CAN_CANCTRL_MAX_PAYLOAD48_U8	140
5.3.2.8	CAN_CANCTRL_MAX_PAYLOAD64_U8	140
5.3.2.9	CAN_CANCTRL_MAX_PAYLOAD8_U8	140
5.3.2.10	CAN_EXTEND_FD_ID_DESCRIPTOR	140
5.3.2.11	CAN_EXTEND_ID_DESCRIPTOR	141
5.3.2.12	CAN_FD_ID_DESCRIPTOR	141
5.3.2.13	CAN_FD_ID_EXTEND_DEF_MASK	141
5.3.2.14	CAN_ID_EXTENDED_MASK_U32	141
5.3.2.15	CAN_ID_EXTENDEDDIFF_MASK_U32	141
5.3.2.16	CAN_ID_STANDARD_MASK_U32	141
5.3.2.17	CAN_IDE_ID_DESCRIPTOR	142
5.3.3	Typedef Documentation	142
5.3.3.1	Can_HwHandleType	142
5.3.3.2	Can_IdType	142
5.3.4	Enumeration Type Documentation	142
5.3.4.1	Can_ControllerStateType	142
5.3.4.2	Can_ErrorStateType	142
5.4	CDD_Acmp.h File Reference	143
5.4.1	Detailed Description	144
5.4.2	Macro Definition Documentation	144
5.4.2.1	ACMP_DEINIT_ID	144
5.4.2.2	ACMP_E_INVALID_CHANNEL	144
5.4.2.3	ACMP_E_INVALID_POINTER	144
5.4.2.4	ACMP_E_STATE_TRANSITION	144
5.4.2.5	ACMP_E_UNINIT	144
5.4.2.6	ACMP_GET_OUTPUTDATA_ID	145
5.4.2.7	ACMP_GET_POLLINGDATA_ID	145
5.4.2.8	ACMP_GET_VERSIONINFO_ID	145
5.4.2.9	ACMP_INIT_ID	145

5.4.2.10	ACMP_INSTANCE_ID	145
5.4.2.11	ACMP_MODULE_INIT	145
5.4.2.12	ACMP_MODULE_UNINIT	146
5.4.3	Function Documentation	146
5.4.3.1	Acmp_Deinit()	146
5.4.3.2	Acmp_GetOutputData()	146
5.4.3.3	Acmp_GetPollingData()	147
5.4.3.4	Acmp_Init()	147
5.5	CDD_Crc.h File Reference	148
5.5.1	Detailed Description	149
5.5.2	Macro Definition Documentation	149
5.5.2.1	CRC_CALCULATECRC_ID	149
5.5.2.2	CRC_E_PARAM_POINTER	149
5.5.2.3	CRC_E_UNINIT	149
5.5.2.4	CRC_GETCONFIG_ID	150
5.5.2.5	CRC_GetCRCRESULT_ID	150
5.5.2.6	CRC_GETVERSIONINFO_ID	150
5.5.2.7	CRC_INIT_ID	150
5.5.3	Function Documentation	150
5.5.3.1	Crc_CalculateCRC()	150
5.5.3.2	Crc_CalculateCRC16()	151
5.5.3.3	Crc_CalculateCRC16ARC()	151
5.5.3.4	Crc_CalculateCRC32()	152
5.5.3.5	Crc_CalculateCRC32P4()	152
5.5.3.6	Crc_CalculateCRC64()	153
5.5.3.7	Crc_CalculateCRC8()	154
5.5.3.8	Crc_CalculateCRC8H2F()	154
5.5.3.9	Crc_Deinit()	155
5.5.3.10	Crc_GetConfig()	155
5.5.3.11	Crc_GetCrcResult()	156
5.5.3.12	Crc_GetVersionInfo()	156

5.5.3.13	Crc_Init()	157
5.6	CDD_Crc_Types.h File Reference	157
5.6.1	Detailed Description	158
5.6.2	Macro Definition Documentation	158
5.6.2.1	CRC_CALCULATECRC_ID	158
5.6.2.2	CRC_CRC16_POLY	158
5.6.2.3	CRC_CRC16ARC_POLY	158
5.6.2.4	CRC_CRC32_POLY	158
5.6.2.5	CRC_CRC32P4_POLY	158
5.6.2.6	CRC_E_PARAM_POINTER	159
5.6.2.7	CRC_E_UNINIT	159
5.6.2.8	CRC_GETCONFIG_ID	159
5.6.2.9	CRC_GetCRCRESULT_ID	159
5.6.2.10	CRC_GETVERSIONINFO_ID	159
5.6.2.11	CRC_INIT_ID	160
5.7	CDD_I2c.h File Reference	160
5.7.1	Detailed Description	161
5.7.2	Macro Definition Documentation	161
5.7.2.1	I2C_ASYNCTRANSMIT_ID	161
5.7.2.2	I2C_DEINIT_ID	161
5.7.2.3	I2C_E_ALREADY_INIT	162
5.7.2.4	I2C_E_BUSY	162
5.7.2.5	I2C_E_INVALID_CHANNEL	162
5.7.2.6	I2C_E_INVALID_POINTER	162
5.7.2.7	I2C_E_STATUES	162
5.7.2.8	I2C_E_UNINIT	162
5.7.2.9	I2C_GETSTATUS_ID	163
5.7.2.10	I2C_GETVERSION_INFO_ID	163
5.7.2.11	I2C_INIT_ID	163
5.7.2.12	I2C_INIT_STATE	163
5.7.2.13	I2C_UNINIT_STATE	163

5.7.3	Enumeration Type Documentation	163
5.7.3.1	l2c_StatusType	163
5.7.4	Function Documentation	164
5.7.4.1	l2c_AbortTransmit()	164
5.7.4.2	l2c_AsyncTransmit()	164
5.7.4.3	l2c_DeInit()	165
5.7.4.4	l2c_GetBaudRate()	165
5.7.4.5	l2c_GetStatus()	166
5.7.4.6	l2c_GetVersionInfo()	166
5.7.4.7	l2c_Init()	167
5.7.4.8	l2c_SetBaudRate()	167
5.7.4.9	l2c_SyncTransmit()	168
5.7.5	Variable Documentation	168
5.7.5.1	l2c_GenerateConfigPC	168
5.8	CDD_Mcl.h File Reference	168
5.8.1	Detailed Description	169
5.8.2	Macro Definition Documentation	169
5.8.2.1	MCL_DEINIT_ID	169
5.8.2.2	MCL_E_PARAM_POINTER	169
5.8.2.3	MCL_GET_VERSION_INFO_ID	169
5.8.2.4	MCL_INIT_ID	170
5.8.2.5	MCL_INSTANCE_ID	170
5.8.3	Function Documentation	170
5.8.3.1	Mcl_DeInit()	170
5.8.3.2	Mcl_GetVersionInfo()	170
5.8.3.3	Mcl_Init()	171
5.9	CDD_Sent.h File Reference	171
5.9.1	Detailed Description	172
5.9.2	Macro Definition Documentation	172
5.9.2.1	SENT_DEINIT_ID	172
5.9.2.2	SENT_E_INVALID_CHANNEL	172

5.9.2.3	SENT_E_INVALID_POINTER	173
5.9.2.4	SENT_E_STATE_TRANSITION	173
5.9.2.5	SENT_E_UNINIT	173
5.9.2.6	SENT_ENABLE_CHANNEL_ID	173
5.9.2.7	SENT_GET_CHANNELSTAUTS_ID	173
5.9.2.8	SENT_GET_VERSIONINFO_ID	173
5.9.2.9	SENT_INIT_ID	174
5.9.2.10	SENT_INIT_TX_CTRL_ID	174
5.9.2.11	SENT_INSTANCE_ID	174
5.9.3	Function Documentation	174
5.9.3.1	Sent_Deinit()	174
5.9.3.2	Sent_EnableChannel()	175
5.9.3.3	Sent_GetChannelStatus()	175
5.9.3.4	Sent_Init()	176
5.9.3.5	Sent_InitChannelTxCtrl()	176
5.10	CDD_Uart.h File Reference	177
5.10.1	Detailed Description	178
5.10.2	Macro Definition Documentation	178
5.10.2.1	UART_ABORTTRANSMIT_ID	178
5.10.2.2	UART_ASYNCTRANSMIT_ID	178
5.10.2.3	UART_DEINIT_ID	178
5.10.2.4	UART_E_BUSY_TRANSMIT	178
5.10.2.5	UART_E_INVALID_CHANNEL	178
5.10.2.6	UART_E_INVALID_POINTER	179
5.10.2.7	UART_E_STATE_TRANSITION	179
5.10.2.8	UART_E_TRANSMIT_LENGTH	179
5.10.2.9	UART_E_UNINIT	179
5.10.2.10	UART_GET_STATUS_ID	179
5.10.2.11	UART_GET_VERSIONINFO_ID	179
5.10.2.12	UART_INIT_ID	180
5.10.2.13	UART_INSTANCE_ID	180

5.10.3 Enumeration Type Documentation	180
5.10.3.1 Uart_DirType	180
5.10.4 Function Documentation	180
5.10.4.1 Uart_AbortTransmit()	180
5.10.4.2 Uart_AsyncTransmit()	181
5.10.4.3 Uart_DelInit()	181
5.10.4.4 Uart_GetStatus()	182
5.10.4.5 Uart_GetVersionInfo()	182
5.10.4.6 Uart_Init()	183
5.10.5 Variable Documentation	183
5.10.5.1 Uart_GenerateConfigPC	183
5.11 Crypto.h File Reference	183
5.11.1 Detailed Description	185
5.11.2 Macro Definition Documentation	185
5.11.2.1 CRYPTO_E_INIT_FAILED	185
5.11.2.2 CRYPTO_E_PARAM_HANDLE	185
5.11.2.3 CRYPTO_E_PARAM_POINTER	185
5.11.2.4 CRYPTO_E_PARAM_VALUE	185
5.11.2.5 CRYPTO_E_UNINIT	185
5.11.2.6 CRYPTO_FUNCTION_BOOTDEFINE	186
5.11.2.7 CRYPTO_FUNCTION_BOOTFAILURE	186
5.11.2.8 CRYPTO_FUNCTION_BOOTOK	186
5.11.2.9 CRYPTO_FUNCTION_CANCELJOB	186
5.11.2.10 CRYPTO_FUNCTION_CERTIFICATEPARSE	186
5.11.2.11 CRYPTO_FUNCTION_CERTIFICATEVERIFY	186
5.11.2.12 CRYPTO_FUNCTION_DEBUGAUTH	187
5.11.2.13 CRYPTO_FUNCTION_DEBUGCHAL	187
5.11.2.14 CRYPTO_FUNCTION_GETID	187
5.11.2.15 CRYPTO_FUNCTION_GETSTATUS	187
5.11.2.16 CRYPTO_FUNCTION_GETVERSIONINFO	187
5.11.2.17 CRYPTO_FUNCTION_INIT	187

5.11.2.18 CRYPTO_FUNCTION_KEYCOPY	188
5.11.2.19 CRYPTO_FUNCTION_KEYDERIVE	188
5.11.2.20 CRYPTO_FUNCTION_KEYELEMENTCOPY	188
5.11.2.21 CRYPTO_FUNCTION_KEYELEMENTGET	188
5.11.2.22 CRYPTO_FUNCTION_KEYELEMENTIDSGET	188
5.11.2.23 CRYPTO_FUNCTION_KEYELEMENTSET	188
5.11.2.24 CRYPTO_FUNCTION_KEYEXCHANGEALCPUBVAL	189
5.11.2.25 CRYPTO_FUNCTION_KEYEXCHANGEALCSECRET	189
5.11.2.26 CRYPTO_FUNCTION_KEYGENERATE	189
5.11.2.27 CRYPTO_FUNCTION_KEYVALIDSET	189
5.11.2.28 CRYPTO_FUNCTION_MPCOMPRESSION	189
5.11.2.29 CRYPTO_FUNCTION_PROCESSJOB	189
5.11.2.30 CRYPTO_FUNCTION_RANDOMSEED	190
5.11.3 Function Documentation	190
5.11.3.1 Crypto_CancelJob()	190
5.11.3.2 Crypto_GetVersionInfo()	190
5.11.3.3 Crypto_Init()	191
5.11.3.4 Crypto_KeyElementGet()	191
5.11.3.5 Crypto_KeyElementIdsGet()	192
5.11.3.6 Crypto_KeyElementSet()	192
5.11.3.7 Crypto_KeySetValid()	193
5.11.3.8 Crypto_MainFunction()	193
5.11.3.9 Crypto_ProcessJob()	194
5.12 Crypto_Types.h File Reference	194
5.12.1 Detailed Description	195
5.12.2 Macro Definition Documentation	195
5.12.2.1 CRYPTO_HW_NOT_SUPPORTED	196
5.12.2.2 CRYPTO_M1SIZE_U32	196
5.12.2.3 CRYPTO_M2SIZE_U32	196
5.12.2.4 CRYPTO_M3SIZE_U32	196
5.12.2.5 CRYPTO_M4SIZE_U32	196

5.12.2.6	CRYPTO_M5SIZE_U32	196
5.12.2.7	CRYPTO_PARAM_LEN_ALIGN	197
5.12.2.8	CRYPTO_PARAM_LEN_IN	197
5.12.2.9	CRYPTO_PARAM_LEN_LARGER	197
5.12.2.10	CRYPTO_PARAM_LEN_MASK	197
5.12.2.11	CRYPTO_SerViceNum	197
5.12.2.12	CRYPTO_SIZE_OUT_U32	198
5.12.3	Typedef Documentation	198
5.12.3.1	Crypto_QueueType	198
5.12.4	Enumeration Type Documentation	198
5.12.4.1	Crypto_DriverStateType	198
5.12.4.2	CryptoKeyElementFormat	198
5.12.4.3	CryptoKeyElementReadAccess	199
5.12.4.4	CryptoKeyElementWriteAccess	199
5.13	Csm_Types.h File Reference	200
5.13.1	Detailed Description	202
5.13.2	Macro Definition Documentation	202
5.13.2.1	CRYPTO_CSE_KE_SHE_M1_OFFSET	202
5.13.2.2	CRYPTO_CSE_KE_SHE_M2_OFFSET	202
5.13.2.3	CRYPTO_CSE_KE_SHE_M3_OFFSET	203
5.13.2.4	CRYPTO_CSE_KE_SHE_M4_OFFSET	203
5.13.2.5	CRYPTO_CSE_KE_SHE_M5_OFFSET	203
5.13.2.6	CRYPTO_KE_CERTIFICATE_CURRENT_TIME	203
5.13.2.7	CRYPTO_KE_CERTIFICATE_DATA	203
5.13.2.8	CRYPTO_KE_CERTIFICATE_EXTENSIONS	203
5.13.2.9	CRYPTO_KE_CERTIFICATE_ISSUER	204
5.13.2.10	CRYPTO_KE_CERTIFICATE_PARSING_FORMAT	204
5.13.2.11	CRYPTO_KE_CERTIFICATE_SERIALNUMBER	204
5.13.2.12	CRYPTO_KE_CERTIFICATE_SIGNATURE	204
5.13.2.13	CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM	204
5.13.2.14	CRYPTO_KE_CERTIFICATE_SUBJECT	204

5.13.2.15 CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY	205
5.13.2.16 CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER	205
5.13.2.17 CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE	205
5.13.2.18 CRYPTO_KE_CERTIFICATE_VERSION	205
5.13.2.19 CRYPTO_KE_CIPHER_2NDKEY	205
5.13.2.20 CRYPTO_KE_CIPHER_IV	205
5.13.2.21 CRYPTO_KE_CIPHER_KEY	206
5.13.2.22 CRYPTO_KE_CIPHER_PROOF	206
5.13.2.23 CRYPTO_KE_KEYDERIVATION_ALGORITHM	206
5.13.2.24 CRYPTO_KE_KEYDERIVATION_ITERATIONS	206
5.13.2.25 CRYPTO_KE_KEYDERIVATION_PASSWORD	206
5.13.2.26 CRYPTO_KE_KEYDERIVATION_SALT	206
5.13.2.27 CRYPTO_KE_KEYEXCHANGE_ALGORITHM	207
5.13.2.28 CRYPTO_KE_KEYEXCHANGE_BASE	207
5.13.2.29 CRYPTO_KE_KEYEXCHANGE_OWNPUBKEY	207
5.13.2.30 CRYPTO_KE_KEYEXCHANGE_PRIVKEY	207
5.13.2.31 CRYPTO_KE_KEYGENERATE_ALGORITHM	207
5.13.2.32 CRYPTO_KE_KEYGENERATE_KEY	207
5.13.2.33 CRYPTO_KE_KEYGENERATE_SEED	208
5.13.2.34 CRYPTO_KE_MAC_KEY	208
5.13.2.35 CRYPTO_KE_MAC_PROOF	208
5.13.2.36 CRYPTO_KE_RANDOM_ALGORITHM	208
5.13.2.37 CRYPTO_KE_RANDOM_SEED_STATE	208
5.13.2.38 CRYPTO_KE_SIGNATURE_KEY	208
5.13.2.39 CRYPTO_KEY_MATERIAL	209
5.13.2.40 CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE	209
5.13.3 Enumeration Type Documentation	209
5.13.3.1 Crypto_AlgorithmFamilyType	209
5.13.3.2 Crypto_AlgorithmModeType	210
5.13.3.3 Crypto_InputOutputRedirectionConfigType	210
5.13.3.4 Crypto_JobStateType	211

5.13.3.5	Crypto_OperationModeType	211
5.13.3.6	Crypto_ProcessingType	211
5.13.3.7	Crypto_ServiceInfoType	212
5.13.3.8	Crypto_VerifyResultType	212
5.14	Dio.h File Reference	213
5.14.1	Detailed Description	213
5.14.2	Function Documentation	213
5.14.2.1	Dio_FlipChannel()	213
5.14.2.2	Dio_GetVersionInfo()	214
5.14.2.3	Dio_MaskedWritePort()	214
5.14.2.4	Dio_ReadChannel()	215
5.14.2.5	Dio_ReadChannelGroup()	215
5.14.2.6	Dio_ReadPort()	216
5.14.2.7	Dio_WriteChannel()	216
5.14.2.8	Dio_WriteChannelGroup()	216
5.14.2.9	Dio_WritePort()	217
5.15	Dio_GeneralTypes.h File Reference	217
5.15.1	Detailed Description	219
5.15.2	Macro Definition Documentation	219
5.15.2.1	DIO_E_PARAM_CONFIG	219
5.15.2.2	DIO_E_PARAM_INVALID_CHANNEL_ID	219
5.15.2.3	DIO_E_PARAM_INVALID_GROUP	219
5.15.2.4	DIO_E_PARAM_INVALID_PORT_ID	219
5.15.2.5	DIO_E_PARAM_LEVEL	220
5.15.2.6	DIO_E_PARAM_POINTER	220
5.15.2.7	DIO_FLIPCHANNEL_ID	220
5.15.2.8	DIO_GETVERSIONINFO_ID	220
5.15.2.9	DIO_INSTANCE_ID	220
5.15.2.10	DIO_MASKEDWRITEPORT_ID	221
5.15.2.11	DIO_READCHANNEL_ID	221
5.15.2.12	DIO_READCHANNELGROUP_ID	221

5.15.2.13 DIO_READPORT_ID	221
5.15.2.14 DIO_WRITECHANNEL_ID	221
5.15.2.15 DIO_WRITECHANNELGROUP_ID	222
5.15.2.16 DIO_WRITEPORT_ID	222
5.15.3 Typedef Documentation	222
5.15.3.1 Dio_ChannelType	222
5.15.3.2 Dio_LevelType	222
5.15.3.3 Dio_PortLevelType	222
5.15.3.4 Dio_PortType	223
5.16 Eep.h File Reference	223
5.16.1 Function Documentation	223
5.16.1.1 Eep_Cancel()	223
5.16.1.2 Eep_Compare()	223
5.16.1.3 Eep_Erase()	224
5.16.1.4 Eep_GetJobResult()	224
5.16.1.5 Eep_GetStatus()	224
5.16.1.6 Eep_GetVersionInfo()	224
5.16.1.7 Eep_Init()	224
5.16.1.8 Eep_Read()	224
5.16.1.9 Eep_SetMode()	224
5.16.1.10 Eep_Write()	225
5.17 Fee.h File Reference	225
5.17.1 Detailed Description	225
5.17.2 Function Documentation	225
5.17.2.1 Fee_Cancel()	226
5.17.2.2 Fee_EraseImmediateBlock()	226
5.17.2.3 Fee_GetJobResult()	227
5.17.2.4 Fee_GetStatus()	227
5.17.2.5 Fee_GetVersionInfo()	228
5.17.2.6 Fee_Init()	228
5.17.2.7 Fee_InvalidateBlock()	229

5.17.2.8	Fee_JobEndNotification()	229
5.17.2.9	Fee_JobErrorNotification()	229
5.17.2.10	Fee_Read()	230
5.17.2.11	Fee_SetMode()	231
5.17.2.12	Fee_Write()	231
5.18	Fee_GeneralTypes.h File Reference	232
5.18.1	Detailed Description	233
5.18.2	Macro Definition Documentation	233
5.18.2.1	FEE_BLOCKHEAD_OVERHEAD	233
5.18.2.2	FEE_CANCEL_ID	233
5.18.2.3	FEE_CLUSTERHEAD_OVERHEAD	233
5.18.2.4	FEE_E_BUSY	233
5.18.2.5	FEE_E_BUSY_INTERNAL	234
5.18.2.6	FEE_E_CANCEL_API	234
5.18.2.7	FEE_E_CLUSTER_GROUP_IDX	234
5.18.2.8	FEE_E_FOREIGN_BLOCKS_OVF	234
5.18.2.9	FEE_E_INIT_FAILED	234
5.18.2.10	FEE_E_INVALID_BLOCK_LEN	234
5.18.2.11	FEE_E_INVALID_BLOCK_NO	235
5.18.2.12	FEE_E_INVALID_BLOCK_OFS	235
5.18.2.13	FEE_E_INVALID_CANCEL	235
5.18.2.14	FEE_E_PARAM_POINTER	235
5.18.2.15	FEE_E_UNINIT	235
5.18.2.16	FEE_ERASEIMMEDIATEBLOCK_ID	235
5.18.2.17	FEE_FLAG_OVERHEAD	236
5.18.2.18	FEE_GETJOBRESULT_ID	236
5.18.2.19	FEE_GETSTATUS_ID	236
5.18.2.20	FEE_GETVERSIONINFO_ID	236
5.18.2.21	FEE_INIT_ID	236
5.18.2.22	FEE_INSTANCE_ID	236
5.18.2.23	FEE_INVALIDATEBLOCK_ID	237

5.18.2.24 FEE_JOBENDNOTIFICATION_ID	237
5.18.2.25 FEE_JOBERRORNOTIFICATION_ID	237
5.18.2.26 FEE_MAINFUNCTION_ID	237
5.18.2.27 FEE_READ_ID	237
5.18.2.28 FEE_SETMODE_ID	237
5.18.2.29 FEE_WRITE_ID	237
5.18.3 Enumeration Type Documentation	237
5.18.3.1 Fee_BlockAssignmentType	237
5.19 Fee_InternalTypes.h File Reference	238
5.19.1 Detailed Description	239
5.19.2 Macro Definition Documentation	239
5.19.2.1 FEE_BLOCK_INVALID_VALUE	239
5.19.3 Enumeration Type Documentation	239
5.19.3.1 Fee_BlockStatusType	239
5.19.3.2 Fee_ClusterStatusType	239
5.19.3.3 Fee_JobType	240
5.20 Fls.h File Reference	240
5.20.1 Detailed Description	241
5.20.2 Function Documentation	241
5.20.2.1 Fls_BlankCheck()	241
5.20.2.2 Fls_Cancel()	242
5.20.2.3 Fls_Compare()	242
5.20.2.4 Fls_Erase()	243
5.20.2.5 Fls_GetJobResult()	243
5.20.2.6 Fls_GetStatus()	244
5.20.2.7 Fls_GetVersionInfo()	244
5.20.2.8 Fls_Init()	245
5.20.2.9 Fls_Read()	245
5.20.2.10 Fls_SetMode()	245
5.20.2.11 Fls_Write()	246
5.21 Fls_GeneralTypes.h File Reference	246

5.21.1 Macro Definition Documentation	247
5.21.1.1 FLS_BLANK_CHECK_ID	248
5.21.1.2 FLS_CANCEL_ID	248
5.21.1.3 FLS_COMPARE_ID	248
5.21.1.4 FLS_E_BUSY	248
5.21.1.5 FLS_E_COMPARE_FAILED	248
5.21.1.6 FLS_E_ERASE_FAILED	248
5.21.1.7 FLS_E_PARAM_ADDRESS	249
5.21.1.8 FLS_E_PARAM_CONFIG	249
5.21.1.9 FLS_E_PARAM_DATA	249
5.21.1.10 FLS_E_PARAM_LENGTH	249
5.21.1.11 FLS_E_PARAM_POINTER	249
5.21.1.12 FLS_E_READ_FAILED	249
5.21.1.13 FLS_E_TIMEOUT	250
5.21.1.14 FLS_E_UNEXPECTED_FLASH_ID	250
5.21.1.15 FLS_E_UNINIT	250
5.21.1.16 FLS_E_VERIFY_ERASE_FAILED	250
5.21.1.17 FLS_E_VERIFY_WRITE_FAILED	250
5.21.1.18 FLS_E_WRITE_FAILED	250
5.21.1.19 FLS_ERASE_ID	251
5.21.1.20 FLS_GETJOBRESULT_ID	251
5.21.1.21 FLS_GETSTATUS_ID	251
5.21.1.22 FLS_GETVERSIONINFO_ID	251
5.21.1.23 FLS_INIT_ID	251
5.21.1.24 FLS_INSTANCE_ID	251
5.21.1.25 FLS_MAINFUNCTION_ID	252
5.21.1.26 FLS_PAGE_WRITE_ASYNC	252
5.21.1.27 FLS_READ_ID	252
5.21.1.28 FLS_SECTOR_ERASE_ASYNC	252
5.21.1.29 FLS_SETMODE_ID	252
5.21.1.30 FLS_WRITE_ID	252

5.21.2	Typedef Documentation	252
5.21.2.1	Fls_AcCallbackPtrType	253
5.21.2.2	Fls_AcErasePtrType	253
5.21.2.3	Fls_AcWritePtrType	253
5.21.2.4	Fls_AddressType	253
5.21.2.5	Fls_JobEndNotificationPtrType	253
5.21.2.6	Fls_JobErrorNotificationPtrType	253
5.21.2.7	Fls_LengthType	253
5.21.3	Enumeration Type Documentation	253
5.21.3.1	Fls_JobType	253
5.21.3.2	Fls_PDType	254
5.22	FlsTst.h File Reference	254
5.22.1	Detailed Description	257
5.22.2	Macro Definition Documentation	257
5.22.2.1	FLSTST_ABORT_SVCID	257
5.22.2.2	FLSTST_DEINIT_SVCID	257
5.22.2.3	FLSTST_E_ALREADY_INITIALIZED	257
5.22.2.4	FLSTST_E_FLSTST_FAILURE	258
5.22.2.5	FLSTST_E_INIT_FAILED	258
5.22.2.6	FLSTST_E_PARAM_INVALID	258
5.22.2.7	FLSTST_E_PARAM_POINTER	258
5.22.2.8	FLSTST_E_STATE_FAILURE	258
5.22.2.9	FLSTST_E_UNINIT	259
5.22.2.10	FLSTST_GET_CURRENT_STATE_SVCID	259
5.22.2.11	FLSTST_GET_ERROR_DETAILS_SVCID	259
5.22.2.12	FLSTST_GET_TEST_RESULT_BGND_SVCID	259
5.22.2.13	FLSTST_GET_TEST_RESULT_FGND_SVCID	259
5.22.2.14	FLSTST_GET_TEST_SIGNATURE_BGND_SVCID	260
5.22.2.15	FLSTST_GET_TEST_SIGNATURE_FGND_SVCID	260
5.22.2.16	FLSTST_GET_VERSION_INFO_SVCID	260
5.22.2.17	FLSTST_INIT_SVCID	260

5.22.2.18 FLSTST_MAIN_FUNCTION_SVCID	260
5.22.2.19 FLSTST_RESUME_SVCID	261
5.22.2.20 FLSTST_START_FGND_SVCID	261
5.22.2.21 FLSTST_SUSPEND_SVCID	261
5.22.2.22 FLSTST_TEST_COMPLETED_NOTIFICATION_SVCID	261
5.22.2.23 FLSTST_TEST_ECC_SVCID	261
5.22.3 Typedef Documentation	261
5.22.3.1 FlsTst_BlockIdFgndType	262
5.22.3.2 FlsTst_NotificationType	262
5.22.4 Enumeration Type Documentation	262
5.22.4.1 FlsTst_StateType	262
5.22.4.2 FlsTst_TestAlgorithmType	262
5.22.4.3 FlsTst_TestResultFgndType	263
5.22.4.4 FlsTst_TestResultType	263
5.22.5 Function Documentation	263
5.22.5.1 FlsTst_Abort()	263
5.22.5.2 FlsTst_DeInit()	264
5.22.5.3 FlsTst_GetCurrentState()	264
5.22.5.4 FlsTst_GetErrorDetails()	265
5.22.5.5 FlsTst_GetTestResultBgnd()	265
5.22.5.6 FlsTst_GetTestResultFgnd()	266
5.22.5.7 FlsTst_GetTestSignatureBgnd()	266
5.22.5.8 FlsTst_GetTestSignatureFgnd()	267
5.22.5.9 FlsTst_GetVersionInfo()	267
5.22.5.10 FlsTst_Init()	268
5.22.5.11 FlsTst_MainFunction()	268
5.22.5.12 FlsTst_Resume()	269
5.22.5.13 FlsTst_StartFgnd()	269
5.22.5.14 FlsTst_Suspend()	270
5.22.5.15 FlsTst_TestCompletedNotification()	270
5.22.5.16 FlsTst_TestEcc()	270

5.22.6 Variable Documentation	271
5.22.6.1 FlsTst_GenerateConfigPC	271
5.22.6.2 FlsTst_GlobalCommonVar	271
5.23 Gpt.h File Reference	271
5.23.1 Macro Definition Documentation	273
5.23.1.1 GPT_CHECK_WAKEUP_ID	273
5.23.1.2 GPT_DEINIT_ID	273
5.23.1.3 GPT_DISABLE_NOTIFICATION_ID	273
5.23.1.4 GPT_DISABLE_WAKEUP_ID	273
5.23.1.5 GPT_E_ALREADY_INITIALIZED	273
5.23.1.6 GPT_E_BUSY	273
5.23.1.7 GPT_E_INIT_FAILED	274
5.23.1.8 GPT_E_MODE	274
5.23.1.9 GPT_E_PARAM_CHANNEL	274
5.23.1.10 GPT_E_PARAM_MODE	274
5.23.1.11 GPT_E_PARAM_POINTER	274
5.23.1.12 GPT_E_PARAM_PREDEF_TIMER	274
5.23.1.13 GPT_E_PARAM_VALUE	275
5.23.1.14 GPT_E_UNINIT	275
5.23.1.15 GPT_ENABLE_NOTIFICATION_ID	275
5.23.1.16 GPT_ENABLE_WAKEUP_ID	275
5.23.1.17 GPT_GET_PREDEF_TIMER_VALUE_ID	275
5.23.1.18 GPT_GET_TIME_ELAPSED_ID	275
5.23.1.19 GPT_GET_TIME_REMAINING_ID	276
5.23.1.20 GPT_GET_VERSION_INFO_ID	276
5.23.1.21 GPT_INIT_ID	276
5.23.1.22 GPT_INSTANCE_ID	276
5.23.1.23 GPT_MODULE_ID	276
5.23.1.24 GPT_SET_MODE_ID	276
5.23.1.25 GPT_START_TIMER_ID	277
5.23.1.26 GPT_STOP_TIMER_ID	277

5.23.2	Function Documentation	277
5.23.2.1	Gpt_CheckWakeup()	277
5.23.2.2	Gpt_DeInit()	277
5.23.2.3	Gpt_DisableNotification()	278
5.23.2.4	Gpt_DisableWakeup()	278
5.23.2.5	Gpt_EnableNotification()	278
5.23.2.6	Gpt_EnableWakeup()	279
5.23.2.7	Gpt_GetPredefTimerValue()	279
5.23.2.8	Gpt_GetTimeElapsed()	279
5.23.2.9	Gpt_GetTimeRemaining()	280
5.23.2.10	Gpt_GetVersionInfo()	280
5.23.2.11	Gpt_Init()	281
5.23.2.12	Gpt_SetMode()	281
5.23.2.13	Gpt_StartTimer()	281
5.23.2.14	Gpt_StopTimer()	282
5.23.3	Variable Documentation	282
5.23.3.1	Gpt_GenerateConfigPC	282
5.24	Icu.h File Reference	282
5.24.1	Detailed Description	286
5.24.2	Macro Definition Documentation	286
5.24.2.1	ICU_CHECKWAKEUP_ID	286
5.24.2.2	ICU_DEINIT_ID	286
5.24.2.3	ICU_DISABLEEDGECOUNT_ID	286
5.24.2.4	ICU_DISABLEEDGEDETECTION_ID	286
5.24.2.5	ICU_DISABLENOTIFICATION_ID	287
5.24.2.6	ICU_DISABLEWAKEUP_ID	287
5.24.2.7	ICU_E_ALREADY_INITIALIZED	287
5.24.2.8	ICU_E_INIT_FAILED	287
5.24.2.9	ICU_E_NOT_STARTED	287
5.24.2.10	ICU_E_PARAM_ACTIVATION	288
5.24.2.11	ICU_E_PARAM_BUFFER_SIZE	288

5.24.2.12	ICU_E_PARAM_CHANNEL	288
5.24.2.13	ICU_E_PARAM_MODE	288
5.24.2.14	ICU_E_PARAM_NOTIFY_INTERVAL	288
5.24.2.15	ICU_E_PARAM_POINTER	289
5.24.2.16	ICU_E_PARAM_VINFO	289
5.24.2.17	ICU_E_UNINIT	289
5.24.2.18	ICU_ENABLEEDGECOUNT_ID	289
5.24.2.19	ICU_ENABLEEDGEDETECTION_ID	289
5.24.2.20	ICU_ENABLENOTIFICATION_ID	290
5.24.2.21	ICU_ENABLEWAKEUP_ID	290
5.24.2.22	ICU_GETDUTYCYCLEVALUES_ID	290
5.24.2.23	ICU_GETEDGENUMBERS_ID	290
5.24.2.24	ICU_GETINPUTSTATE_ID	290
5.24.2.25	ICU_GETTIMEELAPSED_ID	291
5.24.2.26	ICU_GETTIMESTAMPINDEX_ID	291
5.24.2.27	ICU_GETVERSIONINFO_ID	291
5.24.2.28	ICU_INIT_ID	291
5.24.2.29	ICU_RESETEGECOUNT_ID	291
5.24.2.30	ICU_SETACTIVATIONCONDITION_ID	292
5.24.2.31	ICU_SETMODE_ID	292
5.24.2.32	ICU_STARTSIGNALMEASUREMENT_ID	292
5.24.2.33	ICU_STARTTIMESTAMP_ID	292
5.24.2.34	ICU_STOPSIGNALMEASUREMENT_ID	292
5.24.2.35	ICU_STOPTIMESTAMP_ID	293
5.24.3	Typedef Documentation	293
5.24.3.1	Icu_ChannelType	293
5.24.3.2	Icu_EdgeNumberType	293
5.24.3.3	Icu_IndexType	293
5.24.3.4	Icu_ValueType	293
5.24.4	Enumeration Type Documentation	293
5.24.4.1	Icu_ActivationType	293

5.24.4.2	Icu_HwSelect	294
5.24.4.3	Icu_InputStateType	294
5.24.4.4	Icu_MeasurementModeType	294
5.24.4.5	Icu_ModeType	295
5.24.4.6	Icu_SignalMeasurementPropertyType	295
5.24.4.7	Icu_TimestampBufferType	295
5.24.5	Function Documentation	296
5.24.5.1	Icu_CheckWakeup()	296
5.24.5.2	Icu_DeInit()	296
5.24.5.3	Icu_DisableEdgeCount()	296
5.24.5.4	Icu_DisableEdgeDetection()	297
5.24.5.5	Icu_DisableNotification()	297
5.24.5.6	Icu_DisableWakeup()	297
5.24.5.7	Icu_EnableEdgeCount()	298
5.24.5.8	Icu_EnableEdgeDetection()	298
5.24.5.9	Icu_EnableNotification()	298
5.24.5.10	Icu_EnableWakeup()	299
5.24.5.11	Icu_GetDutyCycleValues()	299
5.24.5.12	Icu_GetEdgeNumbers()	300
5.24.5.13	Icu_GetInputState()	300
5.24.5.14	Icu_GetTimeElapsed()	300
5.24.5.15	Icu_GetTimestampIndex()	301
5.24.5.16	Icu_GetVersionInfo()	301
5.24.5.17	Icu_Init()	301
5.24.5.18	Icu_ResetEdgeCount()	302
5.24.5.19	Icu_SetActivationCondition()	302
5.24.5.20	Icu_SetMode()	302
5.24.5.21	Icu_StartSignalMeasurement()	303
5.24.5.22	Icu_StartTimestamp()	303
5.24.5.23	Icu_StopSignalMeasurement()	304
5.24.5.24	Icu_StopTimestamp()	304

5.24.6	Variable Documentation	304
5.24.6.1	Icu_GenerateConfigPC	304
5.25	Lin.h File Reference	304
5.25.1	Detailed Description	306
5.25.2	Macro Definition Documentation	306
5.25.2.1	LIN_CH_OPERATIONAL	306
5.25.2.2	LIN_CH_SLEEP_PENDING	306
5.25.2.3	LIN_CH_SLEEP_STATE	306
5.25.2.4	LIN_CHECK_WAKEUP_ID	306
5.25.2.5	LIN_E_INVALID_CHANNEL	306
5.25.2.6	LIN_E_INVALID_POINTER	307
5.25.2.7	LIN_E_PARAM_POINTER	307
5.25.2.8	LIN_E_STATE_TRANSITION	307
5.25.2.9	LIN_E_TIMEOUT	307
5.25.2.10	LIN_E_UNINIT	307
5.25.2.11	LIN_GETSTATUS_ID	307
5.25.2.12	LIN_GETVERSIONINFO_ID	308
5.25.2.13	LIN_GOTOSLEEP_ID	308
5.25.2.14	LIN_GOTOSLEEPINTERNAL_ID	308
5.25.2.15	LIN_INIT_ID	308
5.25.2.16	LIN_INSTANCE	308
5.25.2.17	LIN_SENDFRAME_ID	308
5.25.2.18	LIN_STATE_INIT	309
5.25.2.19	LIN_STATE_UNINIT	309
5.25.2.20	LIN_WAKEUP_ID	309
5.25.2.21	LIN_WAKEUPINTERNAL_ID	309
5.25.3	Function Documentation	309
5.25.3.1	Lin_CheckWakeup()	309
5.25.3.2	Lin_GetStatus()	310
5.25.3.3	Lin_GetVersionInfo()	310
5.25.3.4	Lin_GoToSleep()	311

5.25.3.5	Lin_GoToSleepInternal()	311
5.25.3.6	Lin_Init()	312
5.25.3.7	Lin_SendFrame()	312
5.25.3.8	Lin_Wakeup()	313
5.25.3.9	Lin_WakeupInternal()	313
5.25.4	Variable Documentation	314
5.25.4.1	Lin_GenerateConfigPC	314
5.26	Lin_GeneralTypes.h File Reference	314
5.26.1	Typedef Documentation	314
5.26.1.1	Lin_FrameDType	314
5.26.1.2	Lin_FramePidType	314
5.26.2	Enumeration Type Documentation	314
5.26.2.1	Lin_SlaveErrorType	314
5.27	Mcu.h File Reference	315
5.27.1	Detailed Description	316
5.27.2	Macro Definition Documentation	316
5.27.2.1	MCU_DISTRIBUTEPLLCLOCK_ID	316
5.27.2.2	MCU_E_ALREADY_INITIALIZED	317
5.27.2.3	MCU_E_CMU_INDEX_OUT_OF_RANGE	317
5.27.2.4	MCU_E_INIT_FAILED	317
5.27.2.5	MCU_E_PARAM_CLOCK	317
5.27.2.6	MCU_E_PARAM_CONFIG	317
5.27.2.7	MCU_E_PARAM_MODE	317
5.27.2.8	MCU_E_PARAM_POINTER	318
5.27.2.9	MCU_E_PARAM_RAMSECTION	318
5.27.2.10	MCU_E_PLL_NOT_LOCKED	318
5.27.2.11	MCU_E_UNINIT	318
5.27.2.12	MCU_GETPLLSTATUS_ID	318
5.27.2.13	MCU_GETTRAMSTATE_ID	318
5.27.2.14	MCU_GETRESETRAWVALUE_ID	319
5.27.2.15	MCU_GETRESETREASON_ID	319

5.27.2.16	MCU_GETVERSIONINFO_ID	319
5.27.2.17	MCU_INIT_ID	319
5.27.2.18	MCU_INITCLOCK_ID	319
5.27.2.19	MCU_INITRAMSECTION_ID	319
5.27.2.20	MCU_INSTANCE_ID	320
5.27.2.21	MCU_PERFORMRESET_ID	320
5.27.2.22	MCU_SETMODE_ID	320
5.27.3	Function Documentation	320
5.27.3.1	Mcu_DistributePllClock()	320
5.27.3.2	Mcu_GetPllStatus()	321
5.27.3.3	Mcu_GetRamState()	321
5.27.3.4	Mcu_GetResetRawValue()	322
5.27.3.5	Mcu_GetResetReason()	322
5.27.3.6	Mcu_GetVersionInfo()	322
5.27.3.7	Mcu_Init()	323
5.27.3.8	Mcu_InitClock()	323
5.27.3.9	Mcu_InitRamSection()	324
5.27.3.10	Mcu_PerformReset()	324
5.27.3.11	Mcu_SetMode()	325
5.27.4	Variable Documentation	325
5.27.4.1	Mcu_GenerateConfigPC	325
5.28	Ocu.h File Reference	325
5.28.1	Detailed Description	327
5.28.2	Macro Definition Documentation	327
5.28.2.1	OCU_DEINIT_ID	327
5.28.2.2	OCU_DISABLENOTIFICATION_ID	328
5.28.2.3	OCU_E_ALREADY_INITIALIZED	328
5.28.2.4	OCU_E_BUSY	328
5.28.2.5	OCU_E_INIT_FAILED	328
5.28.2.6	OCU_E_NO_VALID_NOTIF	328
5.28.2.7	OCU_E_PARAM_INVALID_ACTION	328

5.28.2.8	OCU_E_PARAM_INVALID_CHANNEL	329
5.28.2.9	OCU_E_PARAM_INVALID_STATE	329
5.28.2.10	OCU_E_PARAM_INVALID_VALUE	329
5.28.2.11	OCU_E_PARAM_NO_PIN	329
5.28.2.12	OCU_E_PARAM_POINTER	329
5.28.2.13	OCU_E_UNINIT	329
5.28.2.14	OCU_ENABLENOTIFICATION_ID	330
5.28.2.15	OCU_GETCOUNTER_ID	330
5.28.2.16	OCU_GETVERSIONINFO_ID	330
5.28.2.17	OCU_INIT_ID	330
5.28.2.18	OCU_SETABSOLUTETHRESHOLD_ID	330
5.28.2.19	OCU_SETPINACTION_ID	330
5.28.2.20	OCU_SETPINSTATE_ID	331
5.28.2.21	OCU_SETRELATIVETHRESHOLD_ID	331
5.28.2.22	OCU_STARTCHANNEL_ID	331
5.28.2.23	OCU_STOPCHANNEL_ID	331
5.28.3	Typedef Documentation	331
5.28.3.1	Ocu_ChannelType	331
5.28.3.2	Ocu_ValueType	331
5.28.4	Enumeration Type Documentation	331
5.28.4.1	Ocu_PinActionType	331
5.28.4.2	Ocu_PinStateType	332
5.28.4.3	Ocu_ReturnType	332
5.28.5	Function Documentation	332
5.28.5.1	Ocu_DeInit()	332
5.28.5.2	Ocu_DisableNotification()	333
5.28.5.3	Ocu_EnableNotification()	333
5.28.5.4	Ocu_GetCounter()	333
5.28.5.5	Ocu_GetVersionInfo()	334
5.28.5.6	Ocu_Init()	334
5.28.5.7	Ocu_SetAbsoluteThreshold()	334

5.28.5.8	Ocu_SetPinAction()	335
5.28.5.9	Ocu_SetPinState()	335
5.28.5.10	Ocu_SetRelativeThreshold()	336
5.28.5.11	Ocu_StartChannel()	336
5.28.5.12	Ocu_StopChannel()	336
5.28.6	Variable Documentation	337
5.28.6.1	Ocu_GenerateConfigPC	337
5.29	Oslf.h File Reference	337
5.29.1	Detailed Description	337
5.29.2	Macro Definition Documentation	338
5.29.2.1	OS_PLATFORM	338
5.29.2.2	OSIF_BAREMETAL	338
5.29.2.3	OSIF_OS	338
5.29.2.4	OSIF_SW_MAJOR_VERSION	338
5.29.2.5	OSIF_SW_MINOR_VERSION	338
5.29.2.6	OSIF_SW_PATCH_VERSION	339
5.29.3	Function Documentation	339
5.29.3.1	Oslf_Deinit()	339
5.29.3.2	Oslf_GetCoreId()	339
5.29.3.3	Oslf_Init()	340
5.30	Oslf_Critical.h File Reference	340
5.30.1	Detailed Description	341
5.30.2	Macro Definition Documentation	341
5.30.2.1	CMU_HAL_ID1	341
5.30.2.2	CRYPTO_MCAL_ID	341
5.30.2.3	DMA_HAL_ID1	342
5.30.2.4	DMA_HAL_ID2	342
5.30.2.5	FLS_HAL_ID1	342
5.30.2.6	FLSTST_HAL_ID1	342
5.30.2.7	OSIF_CRITICAL_SW_MAJOR_VERSION	342
5.30.2.8	OSIF_CRITICAL_SW_MINOR_VERSION	343

5.30.2.9 OSIF_CRITICAL_SW_PATCH_VERSION	343
5.30.2.10 OSIF_ENTER_CRITICAL	343
5.30.2.11 OSIF_ENTER_CRITICAL_PROTOTYPES	343
5.30.2.12 OSIF_EXIT_CRITICAL	343
5.30.2.13 OSIF_EXIT_CRITICAL_PROTOTYPES	343
5.30.2.14 UART_HAL_ID1	344
5.30.2.15 WDG_HAL_CS_ID1	344
5.30.2.16 WDG_HAL_CS_ID2	344
5.30.2.17 WDG_HAL_CS_ID3	344
5.30.3 Function Documentation	344
5.30.3.1 OSIF_ENTER_CRITICAL_PROTOTYPES() [1/10]	344
5.30.3.2 OSIF_ENTER_CRITICAL_PROTOTYPES() [2/10]	345
5.30.3.3 OSIF_ENTER_CRITICAL_PROTOTYPES() [3/10]	345
5.30.3.4 OSIF_ENTER_CRITICAL_PROTOTYPES() [4/10]	345
5.30.3.5 OSIF_ENTER_CRITICAL_PROTOTYPES() [5/10]	345
5.30.3.6 OSIF_ENTER_CRITICAL_PROTOTYPES() [6/10]	345
5.30.3.7 OSIF_ENTER_CRITICAL_PROTOTYPES() [7/10]	345
5.30.3.8 OSIF_ENTER_CRITICAL_PROTOTYPES() [8/10]	345
5.30.3.9 OSIF_ENTER_CRITICAL_PROTOTYPES() [9/10]	346
5.30.3.10 OSIF_ENTER_CRITICAL_PROTOTYPES() [10/10]	346
5.30.3.11 OSIF_EXIT_CRITICAL_PROTOTYPES() [1/10]	346
5.30.3.12 OSIF_EXIT_CRITICAL_PROTOTYPES() [2/10]	346
5.30.3.13 OSIF_EXIT_CRITICAL_PROTOTYPES() [3/10]	346
5.30.3.14 OSIF_EXIT_CRITICAL_PROTOTYPES() [4/10]	346
5.30.3.15 OSIF_EXIT_CRITICAL_PROTOTYPES() [5/10]	346
5.30.3.16 OSIF_EXIT_CRITICAL_PROTOTYPES() [6/10]	347
5.30.3.17 OSIF_EXIT_CRITICAL_PROTOTYPES() [7/10]	347
5.30.3.18 OSIF_EXIT_CRITICAL_PROTOTYPES() [8/10]	347
5.30.3.19 OSIF_EXIT_CRITICAL_PROTOTYPES() [9/10]	347
5.30.3.20 OSIF_EXIT_CRITICAL_PROTOTYPES() [10/10]	347
5.31 Oslf_Irq.h File Reference	347

5.31.1 Detailed Description	348
5.31.2 Macro Definition Documentation	348
5.31.2.1 ISR	348
5.31.2.2 OSIF_IRQ_SW_MAJOR_VERSION	348
5.31.2.3 OSIF_IRQ_SW_MINOR_VERSION	348
5.31.2.4 OSIF_IRQ_SW_PATCH_VERSION	349
5.31.3 Function Documentation	349
5.31.3.1 Oslf_ResumeAllInterrupts()	349
5.31.3.2 Oslf_SuspendAllInterrupts()	349
5.31.4 Variable Documentation	349
5.31.4.1 Oslf_CriticalNesting	350
5.31.4.2 Oslf_PriMaskValue	350
5.32 Oslf_Time.h File Reference	350
5.32.1 Detailed Description	350
5.32.2 Macro Definition Documentation	350
5.32.2.1 OSIF_TIME_SW_MAJOR_VERSION	350
5.32.2.2 OSIF_TIME_SW_MINOR_VERSION	351
5.32.2.3 OSIF_TIME_SW_PATCH_VERSION	351
5.32.3 Function Documentation	351
5.32.3.1 Oslf_GetCounter()	351
5.32.3.2 Oslf_GetElapsed()	351
5.32.3.3 Oslf_MicrosToTicks()	352
5.32.3.4 Oslf_UDelay()	352
5.33 Port.h File Reference	352
5.33.1 Detailed Description	353
5.33.2 Function Documentation	353
5.33.2.1 Port_GetVersionInfo()	353
5.33.2.2 Port_Init()	353
5.33.2.3 Port_RefreshPortDirection()	354
5.33.2.4 Port_SetPinDirection()	354
5.33.2.5 Port_SetPinMode()	355

5.34 Port_GeneralTypes.h File Reference	355
5.34.1 Detailed Description	357
5.34.2 Macro Definition Documentation	357
5.34.2.1 GPIO_FUN0	357
5.34.2.2 GPIO_FUN1	357
5.34.2.3 GPIO_FUN2	357
5.34.2.4 GPIO_FUN3	357
5.34.2.5 GPIO_FUN4	358
5.34.2.6 GPIO_FUN5	358
5.34.2.7 GPIO_FUN6	358
5.34.2.8 GPIO_FUN7	358
5.34.2.9 PORT_E_DIRECTION_UNCHANGEABLE	358
5.34.2.10 PORT_E_INIT_FAILED	358
5.34.2.11 PORT_E_MODE_UNCHANGEABLE	359
5.34.2.12 PORT_E_PARAM_INVALID_MODE	359
5.34.2.13 PORT_E_PARAM_PIN	359
5.34.2.14 PORT_E_PARAM_POINTER	359
5.34.2.15 PORT_E_UNINIT	359
5.34.2.16 PORT_GETVERSIONINFO_ID	360
5.34.2.17 PORT_INIT_ID	360
5.34.2.18 PORT_INSTANCE_ID	360
5.34.2.19 PORT_REFRESHPINDIRECTION_ID	360
5.34.2.20 PORT_SETPINDIRECTION_ID	360
5.34.2.21 PORT_SETPINMODE_ID	361
5.34.3 Typedef Documentation	361
5.34.3.1 Port_PinModeType	361
5.34.3.2 Port_PinType	361
5.34.4 Enumeration Type Documentation	361
5.34.4.1 Port_PinDirectionType	361
5.35 Pwm.h File Reference	362
5.35.1 Detailed Description	363

5.35.2	Macro Definition Documentation	363
5.35.2.1	PWM_DUTY_CYCLE_MAX	363
5.35.2.2	PWM_PERIOD_MAX	363
5.35.3	Typedef Documentation	363
5.35.3.1	Pwm_ChannelType	363
5.35.3.2	Pwm_PeriodType	363
5.35.4	Enumeration Type Documentation	363
5.35.4.1	Pwm_ChannelClassType	363
5.35.4.2	Pwm_EdgeNotificationType	364
5.35.4.3	Pwm_OutputStateType	364
5.35.4.4	Pwm_ServiceIdType	364
5.35.5	Function Documentation	365
5.35.5.1	Pwm_DeInit()	365
5.35.5.2	Pwm_DisableNotification()	365
5.35.5.3	Pwm_EnableNotification()	366
5.35.5.4	Pwm_GetOutputState()	366
5.35.5.5	Pwm_GetVersionInfo()	366
5.35.5.6	Pwm_Init()	367
5.35.5.7	Pwm_SetDutyCycle()	367
5.35.5.8	Pwm_SetOutputTolIdle()	367
5.35.5.9	Pwm_SetPeriodAndDuty()	368
5.35.6	Variable Documentation	368
5.35.6.1	Pwm_GenerateConfigPC	368
5.36	Spi.h File Reference	368
5.36.1	Detailed Description	370
5.36.2	Macro Definition Documentation	370
5.36.2.1	SPI_ALL_HW_MAP	370
5.36.2.2	SPI_ASYNCTRANSMIT_ID	370
5.36.2.3	SPI_CANCEL_ID	371
5.36.2.4	SPI_DEINIT_ID	371
5.36.2.5	SPI_E_ALREADY_INITIALIZED	371

5.36.2.6	SPI_E_PARAM_CHANNEL	371
5.36.2.7	SPI_E_PARAM_JOB	371
5.36.2.8	SPI_E_PARAM_LENGTH	371
5.36.2.9	SPI_E_PARAM_POINTER	372
5.36.2.10	SPI_E_PARAM_SEQ	372
5.36.2.11	SPI_E_PARAM_UNIT	372
5.36.2.12	SPI_E_SEQ_IN_PROCESS	372
5.36.2.13	SPI_E_SEQ_PENDING	372
5.36.2.14	SPI_E_UNINIT	373
5.36.2.15	SPI_GETHWUNITSTATUS_ID	373
5.36.2.16	SPI_GETJOBRESULT_ID	373
5.36.2.17	SPI_GETSEQUENCERESULT_ID	373
5.36.2.18	SPI_GETSTATUS_ID	373
5.36.2.19	SPI_GETVERSIONINFO_ID	373
5.36.2.20	SPI_INIT_ID	374
5.36.2.21	SPI_INSTANCE	374
5.36.2.22	SPI_MAINFUNCTIONHANDLING_ID	374
5.36.2.23	SPI_READIB_ID	374
5.36.2.24	SPI_SETASYNCMODE_ID	374
5.36.2.25	SPI_SETUPEB_ID	374
5.36.2.26	SPI_SYNCTRANSMIT_ID	375
5.36.2.27	SPI_WRITEIB_ID	375
5.36.3	Function Documentation	375
5.36.3.1	Spi_AsyncTransmit()	375
5.36.3.2	Spi_Cancel()	375
5.36.3.3	Spi_DeInit()	376
5.36.3.4	Spi_GetHWUnitStatus()	376
5.36.3.5	Spi_GetJobResult()	377
5.36.3.6	Spi_GetSequenceResult()	377
5.36.3.7	Spi_GetStatus()	378
5.36.3.8	Spi_GetVersionInfo()	379

5.36.3.9 Spi_Init()	379
5.36.3.10 Spi_MainFunction_Handling()	379
5.36.3.11 Spi_ReadIB()	380
5.36.3.12 Spi_SetAsyncMode()	380
5.36.3.13 Spi_SetupEB()	381
5.36.3.14 Spi_SyncTransmit()	382
5.36.3.15 Spi_WriteIB()	382
5.37 Wdg.h File Reference	383
5.37.1 Detailed Description	383
5.37.2 Enumeration Type Documentation	383
5.37.2.1 Wdg_ServiceIdType	383
5.37.3 Function Documentation	384
5.37.3.1 Wdg_GetVersionInfo()	384
5.37.3.2 Wdg_Init()	384
5.37.3.3 Wdg_SetMode()	385
5.37.3.4 Wdg_SetTriggerCondition()	385
5.38 Wdg_GeneralTypes.h File Reference	386
5.38.1 Detailed Description	386
5.38.2 Macro Definition Documentation	387
5.38.2.1 WDG_E_DISABLE_REJECTED	387
5.38.2.2 WDG_E_DRIVER_STATE	387
5.38.2.3 WDG_E_PARAM_CONFIG	387
5.38.2.4 WDG_E_PARAM_MODE	387
5.38.2.5 WDG_E_PARAM_POINTER	387
5.38.2.6 WDG_E_PARAM_TIMEOUT	388
5.38.3 Enumeration Type Documentation	388
5.38.3.1 WdgIf_ModeType	388

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

_Crypto_QueueType	8
Acmp_ConfigType	9
Can_BitrateConfigType	10
Can_ConfigType	11
Can_ControllerBaudrateConfigType	12
Can_ControllerDescriptorType	13
Can_ControllerFdConfigType	16
Can_FilterControlType	18
Can_HardwareObjectType	19
Can_HwType	21
Can_PduType	22
Crc_DmaConfig	23
Crypto_AlgorithmInfoType	24
Crypto_JobInfoType	26
Crypto_JobPrimitiveInfoType	27
Crypto_JobPrimitiveInputOutputType	28
Crypto_JobRedirectionInfoType	32
Crypto_JobType	35
Crypto_Key	37
Crypto_ObjectType	38
Crypto_PrimitiveInfoType	40
Crypto_PrimitiveType	42
CryptoKeyElement	43
Dio_ChannelGroupType	45
Dio_ConfigType	46
Fee_BlockHeaderType	47
Fee_BlockInfoType	50
Fee_ClusterGroupType_Struct	
Fee_ClusterGroupType	52
Fee_ConfigType	57
Fee_ClusterHeaderType	54
Fee_ClusterInfoType	56
Fee_ClusterType	56
Fee_ConfigType_Struct	
Fee_BlockType	51
Fee_JobInfoType	58
Fee_JobPendingInfoType	62
Fls_ConfigType	64

FlsTst_BlockConfigType	68
FlsTst_CommonVariableType	69
FlsTst_ConfigType	72
FlsTst_ErrorDetailsType	73
FlsTst_TestResultBgndType	74
FlsTst_TestSignatureBgndType	75
FlsTst_TestSignatureFgndType	76
I2c_CHConfigType	76
I2c_ConfigType	77
Icu_ChannelConfigType	78
Icu_ConfigType	81
Icu_DutyCycleType	81
Icu_IpConfigType	82
Icu_PwmModuleConfigType	83
Lin_ConfigType	84
Mcu_ConfigType	85
Ocu_ChannelConfigType	87
Ocu_ConfigType	89
Ocu_IpConfigType	89
Ocu_ModuleConfigType	90
Port_ConfigType	91
Port_DigitalFilterConfigType	93
Port_PinConfigType	95
Port_UnUsedPinConfigType	97
Pwm_ChannelConfigType	98
Pwm_ConfigType	101
Pwm_IpConfigType	102
Pwm_ModuleConfigType	103
Sent_ConfigType	105
Uart_ConfigType	106
Wdg_ConfigType	107

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

_Crypto_QueueType	
Queue Item type	8
Acmp_ConfigType	
ACMP module configuration structure	9
Can_BitrateConfigType	
CAN Bitrate config satus	10
Can_ConfigType	
.	11
Can_ControllerBaudrateConfigType	
.	12
Can_ControllerDescriptorType	
This is used to Define Controller information	13
Can_ControllerFdConfigType	
CANFD controller information Define	16
Can_FilterControlType	
.	18
Can_HardwareObjectType	
CAN Hardware object Define	19
Can_HwType	
.	21
Can_PduType	
.	22
Crc_DmaConfig	
CRC Dma configuration structure	23
Crypto_AlgorithmInfoType	
Structure which determines the exact algorithm. Note, not every algorithm needs to specify all fields. AUTOSAR shall only allow valid combinations	24
Crypto_JobInfoType	
Structure which contains job information (job ID and job priority)	26
Crypto_JobPrimitiveInfoType	
Structure which contains further information, which depends on the job and the crypto primitive	27
Crypto_JobPrimitiveInputOutputType	
Structure which contains input and output information depending on the job and the crypto primitive . .	28
Crypto_JobRedirectionInfoType	
Structure which holds the identifiers of the keys and key elements which shall be used as input and output for a job and a bit structure which indicates which buffers shall be redirected to those key elements	32
Crypto_JobType	
Structure which contains Job further information, which depends on the job and the crypto primitive . .	35
Crypto_Key	
.	37
Crypto_ObjectType	
Crypto driver object	38
Crypto_PrimitiveInfoType	
Structure which contains basic information about the crypto primitive	40

Crypto_PrimitiveType	
This structure defines configuratgion of a primitive	42
CryptoKeyElement	43
Dio_ChannelGroupType	
Type of a DIO channel group representation	45
Dio_ConfigType	
Type of a DIO configuration structure	46
Fee_BlockHeaderType	
Fee block header configuration structure	47
Fee_BlockInfoType	
Runtime Fee block infomation structure	50
Fee_BlockType	
Fee Configuration type is a stub type, not used, but required by ASR 4.2.2.	51
Fee_ClusterGroupType	
Fee cluster group configuration structure	52
Fee_ClusterHeaderType	
Fee cluster header configuration structure	54
Fee_ClusterInfoType	
Runtime Fee Cluster infomation structure	56
Fee_ClusterType	
Fee cluster configuration structure	56
Fee_ConfigType	
Fee cluster group configuration structure	57
Fee_JobInfoType	
Fee Job Information structure	58
Fee_JobPendingInfoType	
Fee Pending Job Infomation structure	62
Fls_ConfigType	64
FlsTst_BlockConfigType	
Configuration data for a Flash block	68
FlsTst_CommonVariableType	
Initialization data for the Flash Test	69
FlsTst_ConfigType	
Initialization data for the Flash Test	72
FlsTst_ErrorDetailsType	
Test error information	73
FlsTst_TestResultBgndType	
Test result of Flash test in background mode	74
FlsTst_TestSignatureBgndType	
Test result signature of Flash test in background mode	75
FlsTst_TestSignatureFgndType	
Test result signature of Flash test in foreground mode	76
I2c_CHConfigType	76
I2c_ConfigType	77
Icu_ChannelConfigType	
Definition of ICU channel configuration	78
Icu_ConfigType	
Definition of all ICU configurations	81
Icu_DutyCycleType	
Type which shall contain the values, needed for calculating duty cycles	81
Icu_IpConfigType	
Definition of ICU hardware module set configuration	82
Icu_PwmModuleConfigType	
Definition of PWM module configuration	83
Lin_ConfigType	84
Mcu_ConfigType	
A structure to hold the MCU driver configuration	85
Ocu_ChannelConfigType	87
Ocu_ConfigType	89
Ocu_IpConfigType	89
Ocu_ModuleConfigType	90

Port_ConfigType	
All port module configure type	91
Port_DigitalFilterConfigType	
Port digital filter config type	93
Port_PinConfigType	
Each used Pin configure type	95
Port_UnUsedPinConfigType	
Each unused Pin configure type	97
Pwm_ChannelConfigType	98
Pwm_ConfigType	101
Pwm_IpConfigType	102
Pwm_ModuleConfigType	103
Sent_ConfigType	105
Uart_ConfigType	
UART module configuration structure	106
Wdg_ConfigType	
Defines the configuration structure for WDG Driver	107

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Adc.h	This file provides all mcal adc api	108
Can.h	This file provides extern Mcal Can api	120
Can_GeneralTypes.h	This file provides Types used in CAN module	138
CDD_Acmp.h	This file provides Uart function extern	143
CDD_Crc.h	This file provides extern Crc CDD API	148
CDD_Crc_Types.h	This file provides extern Crc CDD API	157
CDD_I2c.h	This file include cdd I2c function	160
CDD_Mcl.h	Middleware common layer includ dma which is common module and so on	168
CDD_Sent.h	This file provides Uart function extern	171
CDD_Uart.h	This file provides Uart function extern	177
Crypto.h	Header file of Crypto MCAL driver	183
Crypto_Types.h	Header file of definitions Crypto types	194
Csm_Types.h	Header file of CSM. It should defined outside Crypto MCAL driver	200
Dio.h	This file provides extern Dio Mcal API	213
Dio_GeneralTypes.h	This file provides extern Dio module structure and macro	217
Eep.h	223
Fee.h	This file provides Fee api interface	225
Fee_GeneralTypes.h	This file provides Fee structure and enum and macro information	232
Fee_InternalTypes.h	This file provides Fee internal structure and enum and macro information	238

Fls.h	This file provides mcal fls api	240
Fls_GeneralTypes.h	246
FlsTst.h	This file is header of MCAL FlsTst	254
Gpt.h	271
Icu.h	This file provides extern Mcal Icu api	282
Lin.h	This file provides extern Mcal lin api	304
Lin_GeneralTypes.h	314
Mcu.h	Mcu driver header file	315
Ocu.h	This file provides extern Mcal Ocu api	325
Oslf.h	This file provides extern Oslf API	337
Oslf_Critical.h	This file provides extern Oslf API	340
Oslf_Irq.h	This file provides extern Oslf Irq API	347
Oslf_Time.h	This file provides Oslf Time API	350
Port.h	This file provides extern Port Mcal API	352
Port_GeneralTypes.h	This file provides extern Port module structure and macro	355
Pwm.h	This file provides all mcal Pwm api	362
Spi.h	Spi driver mcal api header file	368
Wdg.h	Header file of WDG MCAL driver	383
Wdg_GeneralTypes.h	This file provides extern Wdg module structure and macro	386

Chapter 4

Class Documentation

4.1 `_Crypto_QueueType` Struct Reference

Queue Item type.

```
#include <Crypto_Types.h>
```

Public Attributes

- `struct _Crypto_QueueType * Next`
- `Crypto_JobType * JobPtr`

4.1.1 Detailed Description

Queue Item type.

Definition at line 155 of file `Crypto_Types.h`.

4.1.2 Member Data Documentation

4.1.2.1 `JobPtr`

```
Crypto_JobType* _Crypto_QueueType::JobPtr
```

Pointer to Crypto job

Definition at line 158 of file `Crypto_Types.h`.

4.1.2.2 Next

```
struct _Crypto_QueueType* _Crypto_QueueType::Next
```

Index of next item in queue

Definition at line 157 of file Crypto_Types.h.

The documentation for this struct was generated from the following file:

- [Crypto_Types.h](#)

4.2 Acmp_ConfigType Struct Reference

ACMP module configuration structure.

```
#include <CDD_Acmp.h>
```

Public Attributes

- const uint8 [MaxChannelNum](#)
- const Acmp_HwConfigType * [HwConfigPtr](#)

4.2.1 Detailed Description

ACMP module configuration structure.

Definition at line 86 of file CDD_Acmp.h.

4.2.2 Member Data Documentation

4.2.2.1 HwConfigPtr

```
const Acmp_HwConfigType* Acmp_ConfigType::HwConfigPtr
```

Pointer to the hardware configuration structure for ACMP

Definition at line 89 of file CDD_Acmp.h.

4.2.2.2 MaxChannelNum

```
const uint8 Acmp_ConfigType::MaxChannelNum
```

Maximum number of ACMP channels

Definition at line 88 of file CDD_Acmp.h.

The documentation for this struct was generated from the following file:

- [CDD_Acmp.h](#)

4.3 Can_BitrateConfigType Struct Reference

CAN Bitrate config satus.

```
#include <Can.h>
```

Public Attributes

- uint8 [PRESC](#)
- uint8 [SEG_1](#)
- uint8 [SEG_2](#)
- uint8 [SJW](#)

4.3.1 Detailed Description

CAN Bitrate config satus.

Definition at line 141 of file Can.h.

4.3.2 Member Data Documentation

4.3.2.1 PRESC

```
uint8 Can_BitrateConfigType::PRESC
```

Definition at line 143 of file Can.h.

4.3.2.2 SEG_1

```
uint8 Can_BitrateConfigType::SEG_1
```

Definition at line 144 of file Can.h.

4.3.2.3 SEG_2

```
uint8 Can_BitrateConfigType::SEG_2
```

Definition at line 145 of file Can.h.

4.3.2.4 SJW

```
uint8 Can_BitrateConfigType::SJW
```

Definition at line 146 of file Can.h.

The documentation for this struct was generated from the following file:

- [Can.h](#)

4.4 Can_ConfigType Struct Reference

```
#include <Can.h>
```

Public Attributes

- uint8 [ControlerNumber](#)
- uint32 [HohNumber](#)
- const [Can_ControllerDescriptorType](#) * [ControllerDescriptorsPtr](#)
- const [Can_HardwareObjectType](#) * [CanHOHPtr](#)

4.4.1 Detailed Description

Definition at line 223 of file Can.h.

4.4.2 Member Data Documentation

4.4.2.1 CanHOHPtr

```
const Can\_HardwareObjectType* Can_ConfigType::CanHOHPtr
```

Definition at line 228 of file Can.h.

4.4.2.2 ControllerNumber

```
uint8 Can_ConfigType::ControllerNumber
```

Definition at line 225 of file Can.h.

4.4.2.3 ControllerDescriptorsPtr

```
const Can_ControllerDescriptorType* Can_ConfigType::ControllerDescriptorsPtr
```

Definition at line 227 of file Can.h.

4.4.2.4 HohNumber

```
uint32 Can_ConfigType::HohNumber
```

Definition at line 226 of file Can.h.

The documentation for this struct was generated from the following file:

- [Can.h](#)

4.5 Can_ControllerBaudrateConfigType Struct Reference

```
#include <Can.h>
```

Public Attributes

- const [Can_BitrateConfigType](#) ControllerBitrate
- const [Can_ControllerFdConfigType](#) ControllerFDConfig
- const uint16 ControllerBaudRateConfigID
- uint16 ControllerBaudRateKbps

4.5.1 Detailed Description

Definition at line 161 of file Can.h.

4.5.2 Member Data Documentation

4.5.2.1 ControllerBaudRateConfigID

```
const uint16 Can_ControllerBaudrateConfigType::ControllerBaudRateConfigID
```

Definition at line 166 of file Can.h.

4.5.2.2 ControllerBaudRateKbps

```
uint16 Can_ControllerBaudrateConfigType::ControllerBaudRateKbps
```

Definition at line 168 of file Can.h.

4.5.2.3 ControllerBitrate

```
const Can_BitrateConfigType Can_ControllerBaudrateConfigType::ControllerBitrate
```

Definition at line 163 of file Can.h.

4.5.2.4 ControllerFDConfig

```
const Can_ControllerFdConfigType Can_ControllerBaudrateConfigType::ControllerFDConfig
```

Definition at line 164 of file Can.h.

The documentation for this struct was generated from the following file:

- [Can.h](#)

4.6 Can_ControllerDescriptorType Struct Reference

This is used to Define Controller information.

```
#include <Can.h>
```

Public Attributes

- uint16 [ControlerID](#)
- uint16 [Channel](#)
- boolean [CanControllerActivation](#)
- boolean [AutoBusOffRecover](#)
- boolean [TimeStampEn](#)
- boolean [RxInterrupt](#)
- boolean [TxInterrupt](#)
- boolean [BusoffInterrupt](#)
- boolean [WakeUpInterrupt](#)
- boolean [WakeUpEn](#)
- boolean [MemEccEn](#)
- [Can_OverflowModeType](#) [OverflowMode](#)
- uint8 [MaxBaudRateCount](#)
- uint8 [DefaultBaudRateIndex](#)
- const [Can_ControllerBaudrateConfigType](#) * [ControllerBaudrateConfigsPtr](#)
- const uint32 [ECUMWakeupSourceId](#)

4.6.1 Detailed Description

This is used to Define Controller information.

Definition at line 175 of file Can.h.

4.6.2 Member Data Documentation

4.6.2.1 AutoBusOffRecover

```
boolean Can_ControllerDescriptorType::AutoBusOffRecover
```

Definition at line 180 of file Can.h.

4.6.2.2 BusoffInterrupt

```
boolean Can_ControllerDescriptorType::BusoffInterrupt
```

Definition at line 185 of file Can.h.

4.6.2.3 CanControllerActivation

```
boolean Can_ControllerDescriptorType::CanControllerActivation
```

Definition at line 179 of file Can.h.

4.6.2.4 Channel

```
uint16 Can_ControllerDescriptorType::Channel
```

Definition at line 178 of file Can.h.

4.6.2.5 ControlerID

```
uint16 Can_ControllerDescriptorType::ControlerID
```

Definition at line 177 of file Can.h.

4.6.2.6 ControllerBaudrateConfigsPtr

```
const Can_ControllerBaudrateConfigType* Can_ControllerDescriptorType::ControllerBaudrateConfigsPtr
```

Definition at line 194 of file Can.h.

4.6.2.7 DefaultBaudRateIndex

```
uint8 Can_ControllerDescriptorType::DefaultBaudRateIndex
```

Definition at line 193 of file Can.h.

4.6.2.8 ECUMWakeupSourceId

```
const uint32 Can_ControllerDescriptorType::ECUMWakeupSourceId
```

Definition at line 195 of file Can.h.

4.6.2.9 MaxBaudRateCount

```
uint8 Can_ControllerDescriptorType::MaxBaudRateCount
```

Definition at line 192 of file Can.h.

4.6.2.10 MemEccEn

```
boolean Can_ControllerDescriptorType::MemEccEn
```

Definition at line 188 of file Can.h.

4.6.2.11 OverflowMode

```
Can_OverflowModeType Can_ControllerDescriptorType::OverflowMode
```

Definition at line 191 of file Can.h.

4.6.2.12 RxInterrupt

```
boolean Can_ControllerDescriptorType::RxInterrupt
```

Definition at line 183 of file Can.h.

4.6.2.13 TimeStampEn

```
boolean Can_ControllerDescriptorType::TimeStampEn
```

Definition at line 182 of file Can.h.

4.6.2.14 TxInterrupt

```
boolean Can_ControllerDescriptorType::TxInterrupt
```

Definition at line 184 of file Can.h.

4.6.2.15 WakeUpEn

```
boolean Can_ControllerDescriptorType::WakeUpEn
```

Definition at line 187 of file Can.h.

4.6.2.16 WakeUpInterrupt

```
boolean Can_ControllerDescriptorType::WakeUpInterrupt
```

Definition at line 186 of file Can.h.

The documentation for this struct was generated from the following file:

- [Can.h](#)

4.7 Can_ControllerFdConfigType Struct Reference

CANFD controller information Define.

```
#include <Can.h>
```

Public Attributes

- const [Can_BitrateConfigType](#) CanFdBitrate
- const uint32 [CanContrTrcvDelayCompensation](#)
- boolean [CanFdEnable](#)
- boolean [CanFdIsoMode](#)
- boolean [CanBrsEn](#)

4.7.1 Detailed Description

CANFD controller information Define.

Definition at line 152 of file Can.h.

4.7.2 Member Data Documentation

4.7.2.1 CanBrsEn

```
boolean Can_ControllerFdConfigType::CanBrsEn
```

Definition at line 158 of file Can.h.

4.7.2.2 CanContrTrcvDelayCompensation

```
const uint32 Can_ControllerFdConfigType::CanContrTrcvDelayCompensation
```

Definition at line 155 of file Can.h.

4.7.2.3 CanFdBitrate

```
const Can\_BitrateConfigType Can_ControllerFdConfigType::CanFdBitrate
```

Definition at line 154 of file Can.h.

4.7.2.4 CanFdEnable

```
boolean Can_ControllerFdConfigType::CanFdEnable
```

Definition at line 156 of file Can.h.

4.7.2.5 CanFdIsoMode

```
boolean Can_ControllerFdConfigType::CanFdIsoMode
```

Definition at line 157 of file Can.h.

The documentation for this struct was generated from the following file:

- [Can.h](#)

4.8 Can_FilterControlType Struct Reference

```
#include <Can.h>
```

Public Attributes

- uint32 [Code](#)
- uint32 [Mask](#)
- [Can_PduIdType](#) [IdType](#)

4.8.1 Detailed Description

Definition at line 198 of file Can.h.

4.8.2 Member Data Documentation

4.8.2.1 Code

```
uint32 Can_FilterControlType::Code
```

Definition at line 200 of file Can.h.

4.8.2.2 IdType

```
Can\_PduIdType Can_FilterControlType::IdType
```

Definition at line 202 of file Can.h.

4.8.2.3 Mask

```
uint32 Can_FilterControlType::Mask
```

Definition at line 201 of file Can.h.

The documentation for this struct was generated from the following file:

- [Can.h](#)

4.9 Can_HardwareObjectType Struct Reference

CAN Hardware object Define.

```
#include <Can.h>
```

Public Attributes

- const uint8 [CanObjectIndex](#)
- const [Can_HandleType](#) CanHandle
- const [Can_PduIdType](#) CanIdType
- const [Can_ModeType](#) CanRwMode
- const uint8 [CanControllerRef](#)
- const boolean [CanHwObjectUsesPolling](#)
- const uint8 [FilterNumber](#)
- const [Can_FilterControlType](#) * [CanFilterListPtr](#)
- const uint32 [CanMainFunctionRWPeriodRef](#)
- const uint8 [CanFdPaddingValue](#)
- const uint8 [CanHWObjectCount](#)

4.9.1 Detailed Description

CAN Hardware object Define.

Definition at line 208 of file Can.h.

4.9.2 Member Data Documentation

4.9.2.1 CanControllerRef

```
const uint8 Can_HardwareObjectType::CanControllerRef
```

Definition at line 214 of file Can.h.

4.9.2.2 CanFdPaddingValue

```
const uint8 Can_HardwareObjectType::CanFdPaddingValue
```

Definition at line 219 of file Can.h.

4.9.2.3 CanFilterListPtr

```
const Can_FilterControlType* Can_HardwareObjectType::CanFilterListPtr
```

Definition at line 217 of file Can.h.

4.9.2.4 CanHandle

```
const Can_HandleType Can_HardwareObjectType::CanHandle
```

Definition at line 211 of file Can.h.

4.9.2.5 CanHWObjectCount

```
const uint8 Can_HardwareObjectType::CanHWObjectCount
```

Definition at line 220 of file Can.h.

4.9.2.6 CanHwObjectUsesPolling

```
const boolean Can_HardwareObjectType::CanHwObjectUsesPolling
```

Definition at line 215 of file Can.h.

4.9.2.7 CanIdType

```
const Can_PduIdType Can_HardwareObjectType::CanIdType
```

Definition at line 212 of file Can.h.

4.9.2.8 CanMainFunctionRWPeriodRef

```
const uint32 Can_HardwareObjectType::CanMainFunctionRWPeriodRef
```

Definition at line 218 of file Can.h.

4.9.2.9 CanObjectIndex

```
const uint8 Can_HardwareObjectType::CanObjectIndex
```

Definition at line 210 of file Can.h.

4.9.2.10 CanRwMode

```
const Can\_ModeType Can_HardwareObjectType::CanRwMode
```

Definition at line 213 of file Can.h.

4.9.2.11 FilterNumber

```
const uint8 Can_HardwareObjectType::FilterNumber
```

Definition at line 216 of file Can.h.

The documentation for this struct was generated from the following file:

- [Can.h](#)

4.10 Can_HwType Struct Reference

```
#include <Can_GeneralTypes.h>
```

Public Attributes

- [Can_IdType](#) CanId
- [Can_HwHandleType](#) Hoh
- uint8 [ControllerId](#)

4.10.1 Detailed Description

Definition at line 109 of file Can_GeneralTypes.h.

4.10.2 Member Data Documentation

4.10.2.1 CanId

[Can_IdType](#) Can_HwType::CanId

Definition at line 111 of file Can_GeneralTypes.h.

4.10.2.2 ControllerId

uint8 Can_HwType::ControllerId

Definition at line 113 of file Can_GeneralTypes.h.

4.10.2.3 Hoh

[Can_HwHandleType](#) Can_HwType::Hoh

Definition at line 112 of file Can_GeneralTypes.h.

The documentation for this struct was generated from the following file:

- [Can_GeneralTypes.h](#)

4.11 Can_PduType Struct Reference

```
#include <Can_GeneralTypes.h>
```

Public Attributes

- [Can_IdType](#) id
- PduIdType [swPduHandle](#)
- uint8 [length](#)
- uint8 * [sdu](#)

4.11.1 Detailed Description

Definition at line 97 of file Can_GeneralTypes.h.

4.11.2 Member Data Documentation

4.11.2.1 id

[Can_IdType](#) Can_PduType::id

Definition at line 99 of file Can_GeneralTypes.h.

4.11.2.2 length

uint8 Can_PduType::length

Definition at line 101 of file Can_GeneralTypes.h.

4.11.2.3 sdu

uint8* Can_PduType::sdu

Definition at line 102 of file Can_GeneralTypes.h.

4.11.2.4 swPduHandle

PduIdType Can_PduType::swPduHandle

Definition at line 100 of file Can_GeneralTypes.h.

The documentation for this struct was generated from the following file:

- [Can_GeneralTypes.h](#)

4.12 Crc_DmaConfig Struct Reference

CRC Dma configuration structure.

```
#include <CDD_Crc_Types.h>
```

Public Attributes

- boolean [DmaUsed](#)
- uint8 [DmaChannel](#)
- void * [Params](#)

4.12.1 Detailed Description

CRC Dma configuration structure.

This structure holds the dma configuration settings for the crc

Definition at line 92 of file CDD_Crc_Types.h.

4.12.2 Member Data Documentation

4.12.2.1 DmaChannel

```
uint8 Crc_DmaConfig::DmaChannel
```

dma channel used for crc

Definition at line 95 of file CDD_Crc_Types.h.

4.12.2.2 DmaUsed

```
boolean Crc_DmaConfig::DmaUsed
```

whether dma used

Definition at line 94 of file CDD_Crc_Types.h.

4.12.2.3 Params

```
void* Crc_DmaConfig::Params
```

params pass to callback which register to dma

Definition at line 96 of file CDD_Crc_Types.h.

The documentation for this struct was generated from the following file:

- [CDD_Crc_Types.h](#)

4.13 Crypto_AlgorithmInfoType Struct Reference

Structure which determines the exact algorithm. Note, not every algorithm needs to specify all fields. AUTOSAR shall only allow valid combinations.

```
#include <Csm_Types.h>
```

Public Attributes

- [Crypto_AlgorithmFamilyType](#) family
- [Crypto_AlgorithmFamilyType](#) secondaryFamily
- uint32 keyLength
- [Crypto_AlgorithmModeType](#) mode

4.13.1 Detailed Description

Structure which determines the exact algorithm. Note, not every algorithm needs to specify all fields. AUTOSAR shall only allow valid combinations.

Definition at line 257 of file Csm_Types.h.

4.13.2 Member Data Documentation

4.13.2.1 family

[Crypto_AlgorithmFamilyType](#) Crypto_AlgorithmInfoType::family

The family of the algorithm

Definition at line 259 of file Csm_Types.h.

4.13.2.2 keyLength

uint32 Crypto_AlgorithmInfoType::keyLength

The key length in bits to be used with that algorithm

Definition at line 261 of file Csm_Types.h.

4.13.2.3 mode

[Crypto_AlgorithmModeType](#) Crypto_AlgorithmInfoType::mode

The operation mode to be used with that algorithm.

Definition at line 262 of file Csm_Types.h.

4.13.2.4 secondaryFamily

`Crypto_AlgorithmFamilyType` `Crypto_AlgorithmInfoType::secondaryFamily`

The secondary family of the algorithm

Definition at line 260 of file `Csm_Types.h`.

The documentation for this struct was generated from the following file:

- [Csm_Types.h](#)

4.14 Crypto_JobInfoType Struct Reference

Structure which contains job information (job ID and job priority).

```
#include <Csm_Types.h>
```

Public Attributes

- const uint32 [JobId](#)
- const uint32 [JobPriority](#)

4.14.1 Detailed Description

Structure which contains job information (job ID and job priority).

Definition at line 102 of file `Csm_Types.h`.

4.14.2 Member Data Documentation

4.14.2.1 JobId

```
const uint32 Crypto_JobInfoType::JobId
```

Identifier of job

Definition at line 104 of file `Csm_Types.h`.

4.14.2.2 JobPriority

```
const uint32 Crypto_JobInfoType::JobPriority
```

Specifies the importance of the job (the higher, the more important).

Definition at line 105 of file Csm_Types.h.

The documentation for this struct was generated from the following file:

- [Csm_Types.h](#)

4.15 Crypto_JobPrimitiveInfoType Struct Reference

Structure which contains further information, which depends on the job and the crypto primitive.

```
#include <Csm_Types.h>
```

Public Attributes

- const uint32 [callbackId](#)
- const [Crypto_PrimitiveInfoType](#) * [primitiveInfo](#)
- const uint32 [crylfKeyId](#)
- [Crypto_ProcessingType](#) [processingType](#)
- const boolean [callbackUpdateNotification](#)

4.15.1 Detailed Description

Structure which contains further information, which depends on the job and the crypto primitive.

Definition at line 287 of file Csm_Types.h.

4.15.2 Member Data Documentation

4.15.2.1 callbackId

```
const uint32 Crypto_JobPrimitiveInfoType::callbackId
```

Pointer to the input data.

Definition at line 289 of file Csm_Types.h.

4.15.2.2 callbackUpdateNotification

```
const boolean Crypto_JobPrimitiveInfoType::callbackUpdateNotification
```

Pointer to the input data.

Definition at line 293 of file Csm_Types.h.

4.15.2.3 cryIfKeyId

```
const uint32 Crypto_JobPrimitiveInfoType::cryIfKeyId
```

Pointer to the input data.

Definition at line 291 of file Csm_Types.h.

4.15.2.4 primitiveInfo

```
const Crypto\_PrimitiveInfoType* Crypto_JobPrimitiveInfoType::primitiveInfo
```

Pointer to the input data.

Definition at line 290 of file Csm_Types.h.

4.15.2.5 processingType

```
Crypto\_ProcessingType Crypto_JobPrimitiveInfoType::processingType
```

Pointer to the input data.

Definition at line 292 of file Csm_Types.h.

The documentation for this struct was generated from the following file:

- [Csm_Types.h](#)

4.16 Crypto_JobPrimitiveInputOutputType Struct Reference

Structure which contains input and output information depending on the job and the crypto primitive.

```
#include <Csm_Types.h>
```

Public Attributes

- uint8 * [inputPtr](#)
- uint32 [inputLength](#)
- uint8 * [secondaryInputPtr](#)
- uint32 [secondaryInputLength](#)
- uint8 * [tertiaryInputPtr](#)
- uint32 [tertiaryInputLength](#)
- uint8 * [outputPtr](#)
- uint32 * [outputLengthPtr](#)
- uint8 * [secondaryOutputPtr](#)
- uint32 * [secondaryOutputLengthPtr](#)
- uint64 [input64](#)
- [Crypto_VerifyResultType](#) * [verifyPtr](#)
- uint64 * [output64Ptr](#)
- [Crypto_OperationModeType](#) [mode](#)
- uint32 [cryIfKeyId](#)
- uint32 [targetCryIfKeyId](#)

4.16.1 Detailed Description

Structure which contains input and output information depending on the job and the crypto primitive.

Definition at line 234 of file `Csm_Types.h`.

4.16.2 Member Data Documentation

4.16.2.1 cryIfKeyId

```
uint32 Crypto_JobPrimitiveInputOutputType::cryIfKeyId
```

Holds the CryIf key id for key operation services..

Definition at line 250 of file `Csm_Types.h`.

4.16.2.2 input64

```
uint64 Crypto_JobPrimitiveInputOutputType::input64
```

Versatile input parameter.

Definition at line 246 of file `Csm_Types.h`.

4.16.2.3 inputLength

```
uint32 Crypto_JobPrimitiveInputOutputType::inputLength
```

Contains the input length in bytes.

Definition at line 237 of file Csm_Types.h.

4.16.2.4 inputPtr

```
uint8* Crypto_JobPrimitiveInputOutputType::inputPtr
```

Pointer to the input data.

Definition at line 236 of file Csm_Types.h.

4.16.2.5 mode

```
Crypto_OperationModeType Crypto_JobPrimitiveInputOutputType::mode
```

Indicator of the mode(s)/operation(s) to be performed.

Definition at line 249 of file Csm_Types.h.

4.16.2.6 output64Ptr

```
uint64* Crypto_JobPrimitiveInputOutputType::output64Ptr
```

Output pointer to a memory location holding an uint64.

Definition at line 248 of file Csm_Types.h.

4.16.2.7 outputLengthPtr

```
uint32* Crypto_JobPrimitiveInputOutputType::outputLengthPtr
```

Holds a pointer to a memory location containing the output length in bytes.

Definition at line 243 of file Csm_Types.h.

4.16.2.8 outputPtr

```
uint8* Crypto_JobPrimitiveInputOutputType::outputPtr
```

Pointer to the output data.

Definition at line 242 of file Csm_Types.h.

4.16.2.9 secondaryInputLength

```
uint32 Crypto_JobPrimitiveInputOutputType::secondaryInputLength
```

Contains the secondary input length in bytes.

Definition at line 239 of file Csm_Types.h.

4.16.2.10 secondaryInputPtr

```
uint8* Crypto_JobPrimitiveInputOutputType::secondaryInputPtr
```

Pointer to the secondary input data (for MacVerify, SignatureVerify).

Definition at line 238 of file Csm_Types.h.

4.16.2.11 secondaryOutputLengthPtr

```
uint32* Crypto_JobPrimitiveInputOutputType::secondaryOutputLengthPtr
```

Holds a pointer to a memory location containing the secondary output length in bytes.

Definition at line 245 of file Csm_Types.h.

4.16.2.12 secondaryOutputPtr

```
uint8* Crypto_JobPrimitiveInputOutputType::secondaryOutputPtr
```

Pointer to the secondary output data.

Definition at line 244 of file Csm_Types.h.

4.16.2.13 targetCryIfKeyId

```
uint32 Crypto_JobPrimitiveInputOutputType::targetCryIfKeyId
```

Holds the target CryIf key id for key operation services..

Definition at line 251 of file Csm_Types.h.

4.16.2.14 tertiaryInputLength

```
uint32 Crypto_JobPrimitiveInputOutputType::tertiaryInputLength
```

Contains the tertiary input length in bytes.

Definition at line 241 of file Csm_Types.h.

4.16.2.15 tertiaryInputPtr

```
uint8* Crypto_JobPrimitiveInputOutputType::tertiaryInputPtr
```

Pointer to the tertiary input data (for MacVerify, SignatureVerify).

Definition at line 240 of file Csm_Types.h.

4.16.2.16 verifyPtr

```
Crypto_VerifyResultType* Crypto_JobPrimitiveInputOutputType::verifyPtr
```

Output pointer to a memory location holding a Crypto_VerifyResultType.

Definition at line 247 of file Csm_Types.h.

The documentation for this struct was generated from the following file:

- [Csm_Types.h](#)

4.17 Crypto_JobRedirectionInfoType Struct Reference

Structure which holds the identifiers of the keys and key elements which shall be used as input and output for a job and a bit structure which indicates which buffers shall be redirected to those key elements.

```
#include <Csm_Types.h>
```

Public Attributes

- uint8 [redirectionConfig](#)
- uint32 [inputKeyId](#)
- uint32 [inputKeyElementId](#)
- uint32 [secondaryInputKeyId](#)
- uint32 [secondaryInputKeyElementId](#)
- uint32 [tertiaryInputKeyId](#)
- uint32 [tertiaryInputKeyElementId](#)
- uint32 [outputKeyId](#)
- uint32 [outputKeyElementId](#)
- uint32 [secondaryOutputKeyId](#)
- uint32 [secondaryOutputKeyElementId](#)

4.17.1 Detailed Description

Structure which holds the identifiers of the keys and key elements which shall be used as input and output for a job and a bit structure which indicates which buffers shall be redirected to those key elements.

Definition at line 311 of file `Csm_Types.h`.

4.17.2 Member Data Documentation

4.17.2.1 inputKeyElementId

```
uint32 Crypto_JobRedirectionInfoType::inputKeyElementId
```

Identifier of the key element which shall be used as input

Definition at line 315 of file `Csm_Types.h`.

4.17.2.2 inputKeyId

```
uint32 Crypto_JobRedirectionInfoType::inputKeyId
```

Identifier of the key which shall be used as input

Definition at line 314 of file `Csm_Types.h`.

4.17.2.3 outputKeyElementId

```
uint32 Crypto_JobRedirectionInfoType::outputKeyElementId
```

Identifier of the key element which shall be used as output

Definition at line 321 of file `Csm_Types.h`.

4.17.2.4 outputKeyId

```
uint32 Crypto_JobRedirectionInfoType::outputKeyId
```

Identifier of the key which shall be used as output

Definition at line 320 of file Csm_Types.h.

4.17.2.5 redirectionConfig

```
uint8 Crypto_JobRedirectionInfoType::redirectionConfig
```

Bit structure which indicates which buffer shall be redirected to a key element. Values from Crypto_InputOutput↔RedirectionConfigType can be used and combined with unary OR operation.

Definition at line 313 of file Csm_Types.h.

4.17.2.6 secondaryInputKeyElementId

```
uint32 Crypto_JobRedirectionInfoType::secondaryInputKeyElementId
```

Identifier of the key element which shall be used as secondary input

Definition at line 317 of file Csm_Types.h.

4.17.2.7 secondaryInputKeyId

```
uint32 Crypto_JobRedirectionInfoType::secondaryInputKeyId
```

Identifier of the key which shall be used as secondary input

Definition at line 316 of file Csm_Types.h.

4.17.2.8 secondaryOutputKeyElementId

```
uint32 Crypto_JobRedirectionInfoType::secondaryOutputKeyElementId
```

Identifier of the key element which shall be used as secondary output

Definition at line 323 of file Csm_Types.h.

4.17.2.9 secondaryOutputKeyId

```
uint32 Crypto_JobRedirectionInfoType::secondaryOutputKeyId
```

Identifier of the key which shall be used as secondary output

Definition at line 322 of file Csm_Types.h.

4.17.2.10 tertiaryInputKeyElementId

```
uint32 Crypto_JobRedirectionInfoType::tertiaryInputKeyElementId
```

Identifier of the key element which shall be used as tertiary input

Definition at line 319 of file Csm_Types.h.

4.17.2.11 tertiaryInputKeyId

```
uint32 Crypto_JobRedirectionInfoType::tertiaryInputKeyId
```

Identifier of the key which shall be used as tertiary input

Definition at line 318 of file Csm_Types.h.

The documentation for this struct was generated from the following file:

- [Csm_Types.h](#)

4.18 Crypto_JobType Struct Reference

Structure which contains Job further information, which depends on the job and the crypto primitive.

```
#include <Csm_Types.h>
```

Public Attributes

- const uint32 [jobId](#)
- [Crypto_JobStateType](#) [jobState](#)
- [Crypto_JobPrimitiveInputOutputType](#) [jobPrimitiveInputOutput](#)
- const [Crypto_JobPrimitiveInfoType](#) * [jobPrimitiveInfo](#)
- const [Crypto_JobInfoType](#) * [jobInfo](#)
- const [Crypto_JobRedirectionInfoType](#) * [jobRedirectionInfoRef](#)
- uint32 [cryptoKeyId](#)

4.18.1 Detailed Description

Structure which contains Job further information, which depends on the job and the crypto primitive.

Definition at line 329 of file Csm_Types.h.

4.18.2 Member Data Documentation

4.18.2.1 cryptoKeyId

```
uint32 Crypto_JobType::cryptoKeyId
```

Definition at line 337 of file Csm_Types.h.

4.18.2.2 jobId

```
const uint32 Crypto_JobType::jobId
```

Identifier for the job structure.

Definition at line 331 of file Csm_Types.h.

4.18.2.3 jobInfo

```
const Crypto_JobInfoType* Crypto_JobType::jobInfo
```

Pointer to a structure containing further information which depends on the job and the crypto primitive.

Definition at line 335 of file Csm_Types.h.

4.18.2.4 jobPrimitiveInfo

```
const Crypto_JobPrimitiveInfoType* Crypto_JobType::jobPrimitiveInfo
```

Pointer to a structure containing further information which depends on the job and the crypto primitive.

Definition at line 334 of file Csm_Types.h.

4.18.2.5 jobPrimitiveInputOutput

`Crypto_JobPrimitiveInputOutputType` `Crypto_JobType::jobPrimitiveInputOutput`

Structure containing input and output information depending on the job and the crypto primitive.

Definition at line 333 of file `Csm_Types.h`.

4.18.2.6 jobRedirectionInfoRef

`const Crypto_JobRedirectionInfoType*` `Crypto_JobType::jobRedirectionInfoRef`

Pointer to a structure containing further information on the usage of keys as input and output for jobs.

Definition at line 336 of file `Csm_Types.h`.

4.18.2.7 jobState

`Crypto_JobStateType` `Crypto_JobType::jobState`

Determines the current job state.

Definition at line 332 of file `Csm_Types.h`.

The documentation for this struct was generated from the following file:

- [Csm_Types.h](#)

4.19 Crypto_Key Struct Reference

```
#include <Crypto_Types.h>
```

Public Attributes

- `const uint32` [KeyId](#)
- `boolean` [KeyValid](#)
- `const uint32` [Count](#)
- [CryptoKeyElement](#) `**const` [KeyTypePtr](#)

4.19.1 Detailed Description

Definition at line 144 of file `Crypto_Types.h`.

4.19.2 Member Data Documentation

4.19.2.1 Count

```
const uint32 Crypto_Key::Count
```

Definition at line 148 of file `Crypto_Types.h`.

4.19.2.2 KeyId

```
const uint32 Crypto_Key::KeyId
```

Definition at line 146 of file `Crypto_Types.h`.

4.19.2.3 KeyTypePtr

```
CryptoKeyElement** const Crypto_Key::KeyTypePtr
```

Definition at line 149 of file `Crypto_Types.h`.

4.19.2.4 KeyValid

```
boolean Crypto_Key::KeyValid
```

Definition at line 147 of file `Crypto_Types.h`.

The documentation for this struct was generated from the following file:

- [Crypto_Types.h](#)

4.20 Crypto_ObjectType Struct Reference

Crypto driver object.

```
#include <Crypto_Types.h>
```

Public Attributes

- const uint32 [DriverObjectId](#)
- [Crypto_QueueType](#) * [QueuedJobs](#)
- const uint32 [QueueSize](#)
- [Crypto_QueueType](#) * [HeadOfFree](#)
- [Crypto_QueueType](#) * [HeadOfJobs](#)
- const [Crypto_PrimitiveType](#) *const [PrimitivesPtr](#)
- const uint32 [PrimitivesCount](#)

4.20.1 Detailed Description

Crypto driver object.

Definition at line 164 of file [Crypto_Types.h](#).

4.20.2 Member Data Documentation

4.20.2.1 DriverObjectId

```
const uint32 Crypto_ObjectType::DriverObjectId
```

ID of Crypto object

Definition at line 166 of file [Crypto_Types.h](#).

4.20.2.2 HeadOfFree

```
Crypto\_QueueType* Crypto_ObjectType::HeadOfFree
```

Head of free queue.

Definition at line 169 of file [Crypto_Types.h](#).

4.20.2.3 HeadOfJobs

```
Crypto\_QueueType* Crypto_ObjectType::HeadOfJobs
```

Head of jobs queue.

Definition at line 170 of file [Crypto_Types.h](#).

4.20.2.4 PrimitivesCount

```
const uint32 Crypto_ObjectType::PrimitivesCount
```

Count of primitive in the above array.

Definition at line 172 of file Crypto_Types.h.

4.20.2.5 PrimitivesPtr

```
const Crypto\_PrimitiveType* const Crypto_ObjectType::PrimitivesPtr
```

Pointer to crypto object 's primitive array.

Definition at line 171 of file Crypto_Types.h.

4.20.2.6 QueuedJobs

```
Crypto\_QueueType* Crypto_ObjectType::QueuedJobs
```

Start pointer of Job queue.

Definition at line 167 of file Crypto_Types.h.

4.20.2.7 QueueSize

```
const uint32 Crypto_ObjectType::QueueSize
```

Item count in Job queue.

Definition at line 168 of file Crypto_Types.h.

The documentation for this struct was generated from the following file:

- [Crypto_Types.h](#)

4.21 Crypto_PrimitiveInfoType Struct Reference

Structure which contains basic information about the crypto primitive.

```
#include <Csm_Types.h>
```

Public Attributes

- uint32 [resultLength](#)
- [Crypto_ServiceInfoType](#) service
- [Crypto_AlgorithmInfoType](#) algorithm

4.21.1 Detailed Description

Structure which contains basic information about the crypto primitive.

Definition at line 268 of file Csm_Types.h.

4.21.2 Member Data Documentation

4.21.2.1 algorithm

[Crypto_AlgorithmInfoType](#) `Crypto_PrimitiveInfoType::algorithm`

Contains the information of the used algorithm

Definition at line 272 of file Csm_Types.h.

4.21.2.2 resultLength

uint32 `Crypto_PrimitiveInfoType::resultLength`

Contains the result length in bytes.

Definition at line 270 of file Csm_Types.h.

4.21.2.3 service

[Crypto_ServiceInfoType](#) `Crypto_PrimitiveInfoType::service`

Contains the enum of the used service, e.g. Encrypt

Definition at line 271 of file Csm_Types.h.

The documentation for this struct was generated from the following file:

- [Csm_Types.h](#)

4.22 Crypto_PrimitiveType Struct Reference

This structure defines configuratgion of a primitive.

```
#include <Crypto_Types.h>
```

Public Attributes

- const [Crypto_ServiceInfoType Service](#)
Define the crypto service.
- const [Crypto_AlgorithmFamilyType Family](#)
Defines the algorithm family.
- const [Crypto_AlgorithmModeType Mode](#)
Defines the algorithm mode.
- [Crypto_AlgorithmFamilyType SecondaryFamily](#)

4.22.1 Detailed Description

This structure defines configuratgion of a primitive.

Definition at line 122 of file `Crypto_Types.h`.

4.22.2 Member Data Documentation

4.22.2.1 Family

```
const Crypto\_AlgorithmFamilyType Crypto_PrimitiveType::Family
```

Defines the algorithm family.

Definition at line 125 of file `Crypto_Types.h`.

4.22.2.2 Mode

```
const Crypto\_AlgorithmModeType Crypto_PrimitiveType::Mode
```

Defines the algorithm mode.

Definition at line 126 of file `Crypto_Types.h`.

4.22.2.3 SecondaryFamily

`Crypto_AlgorithmFamilyType` `Crypto_PrimitiveType::SecondaryFamily`

Definition at line 127 of file `Crypto_Types.h`.

4.22.2.4 Service

`const Crypto_ServiceInfoType` `Crypto_PrimitiveType::Service`

Define the crypto service.

Definition at line 124 of file `Crypto_Types.h`.

The documentation for this struct was generated from the following file:

- [Crypto_Types.h](#)

4.23 CryptoKeyElement Struct Reference

```
#include <Crypto_Types.h>
```

Public Attributes

- `const uint32` `Id`
- `const uint32` `Uniqueld`
- `const boolean` `AllowPartialAccess`
- `const CryptoKeyElementFormat` `KeyFormatType`
- `const boolean` `Persist`
- `const CryptoKeyElementReadAccess` `ReadAccess`
- `const uint32` `MaxSize`
- `uint32` `ActualSize`
- `const CryptoKeyElementWriteAccess` `WriteAccess`
- `uint8 *` `ValuePtr`

4.23.1 Detailed Description

Definition at line 130 of file `Crypto_Types.h`.

4.23.2 Member Data Documentation

4.23.2.1 ActualSize

```
uint32 CryptoKeyElement::ActualSize
```

Definition at line 139 of file Crypto_Types.h.

4.23.2.2 AllowPartialAccess

```
const boolean CryptoKeyElement::AllowPartialAccess
```

Definition at line 134 of file Crypto_Types.h.

4.23.2.3 Id

```
const uint32 CryptoKeyElement::Id
```

Definition at line 132 of file Crypto_Types.h.

4.23.2.4 KeyFormatType

```
const CryptoKeyElementFormat CryptoKeyElement::KeyFormatType
```

Definition at line 135 of file Crypto_Types.h.

4.23.2.5 MaxSize

```
const uint32 CryptoKeyElement::MaxSize
```

Definition at line 138 of file Crypto_Types.h.

4.23.2.6 Persist

```
const boolean CryptoKeyElement::Persist
```

Definition at line 136 of file Crypto_Types.h.

4.23.2.7 ReadAccess

```
const CryptoKeyElementReadAccess CryptoKeyElement::ReadAccess
```

Definition at line 137 of file `Crypto_Types.h`.

4.23.2.8 UniqueId

```
const uint32 CryptoKeyElement::UniqueId
```

Definition at line 133 of file `Crypto_Types.h`.

4.23.2.9 ValuePtr

```
uint8* CryptoKeyElement::ValuePtr
```

Definition at line 141 of file `Crypto_Types.h`.

4.23.2.10 WriteAccess

```
const CryptoKeyElementWriteAccess CryptoKeyElement::WriteAccess
```

Definition at line 140 of file `Crypto_Types.h`.

The documentation for this struct was generated from the following file:

- [Crypto_Types.h](#)

4.24 Dio_ChannelGroupType Struct Reference

Type of a DIO channel group representation.

```
#include <Dio_GeneralTypes.h>
```

Public Attributes

- [Dio_PortType](#) Port
- [uint8](#) Offset
- [Dio_PortLevelType](#) Mask

4.24.1 Detailed Description

Type of a DIO channel group representation.

Definition at line 121 of file Dio_GeneralTypes.h.

4.24.2 Member Data Documentation

4.24.2.1 Mask

```
Dio_PortLevelType Dio_ChannelGroupType::Mask
```

Group mask.

Definition at line 125 of file Dio_GeneralTypes.h.

4.24.2.2 Offset

```
uint8 Dio_ChannelGroupType::Offset
```

Bit offset within the port.

Definition at line 124 of file Dio_GeneralTypes.h.

4.24.2.3 Port

```
Dio_PortType Dio_ChannelGroupType::Port
```

Port identifier.

Definition at line 123 of file Dio_GeneralTypes.h.

The documentation for this struct was generated from the following file:

- [Dio_GeneralTypes.h](#)

4.25 Dio_ConfigType Struct Reference

Type of a DIO configuration structure.

```
#include <Dio_GeneralTypes.h>
```

Public Attributes

- uint8 [NumChannelGroups](#)
- const [Dio_ChannelGroupType](#) * [ChannelGroupList](#)

4.25.1 Detailed Description

Type of a DIO configuration structure.

Definition at line 129 of file [Dio_GeneralTypes.h](#).

4.25.2 Member Data Documentation

4.25.2.1 ChannelGroupList

```
const Dio\_ChannelGroupType* Dio_ConfigType::ChannelGroupList
```

Pointer to list of channel groups in configuration

Definition at line 132 of file [Dio_GeneralTypes.h](#).

4.25.2.2 NumChannelGroups

```
uint8 Dio_ConfigType::NumChannelGroups
```

Number of channel groups in configuration

Definition at line 131 of file [Dio_GeneralTypes.h](#).

The documentation for this struct was generated from the following file:

- [Dio_GeneralTypes.h](#)

4.26 Fee_BlockHeaderType Struct Reference

Fee block header configuration structure .

```
#include <Fee_InternalTypes.h>
```

Public Attributes

- uint16 [BlockId](#)
- uint16 [Length](#)
- uint32 [TargetAddr](#)
- uint32 [CheckSum](#)
- uint16 [ImmediateBlock](#)
- uint16 [Assigned](#)
- uint64 [Valid](#)
- uint64 [Invalid](#)

4.26.1 Detailed Description

Fee block header configuration structure .

Definition at line 145 of file Fee_InternalTypes.h.

4.26.2 Member Data Documentation

4.26.2.1 Assigned

```
uint16 Fee_BlockHeaderType::Assigned
```

indicate the assigned module

Definition at line 152 of file Fee_InternalTypes.h.

4.26.2.2 BlockId

```
uint16 Fee_BlockHeaderType::BlockId
```

ID of Fee cluster

Definition at line 147 of file Fee_InternalTypes.h.

4.26.2.3 CheckSum

```
uint32 Fee_BlockHeaderType::CheckSum
```

checksum value to check header whether right

Definition at line 150 of file Fee_InternalTypes.h.

4.26.2.4 ImmediateBlock

```
uint16 Fee_BlockHeaderType::ImmediateBlock
```

this block whether is immediate block

Definition at line 151 of file Fee_InternalTypes.h.

4.26.2.5 Invalid

```
uint64 Fee_BlockHeaderType::Invalid
```

indicate the header whether invalid

Definition at line 154 of file Fee_InternalTypes.h.

4.26.2.6 Length

```
uint16 Fee_BlockHeaderType::Length
```

Size of Fee block in bytes

Definition at line 148 of file Fee_InternalTypes.h.

4.26.2.7 TargetAddr

```
uint32 Fee_BlockHeaderType::TargetAddr
```

Address of Fee block in flash

Definition at line 149 of file Fee_InternalTypes.h.

4.26.2.8 Valid

```
uint64 Fee_BlockHeaderType::Valid
```

indicate the header whether valid

Definition at line 153 of file Fee_InternalTypes.h.

The documentation for this struct was generated from the following file:

- [Fee_InternalTypes.h](#)

4.27 Fee_BlockInfoType Struct Reference

runtime Fee block information structure

```
#include <Fee_InternalTypes.h>
```

Public Attributes

- [Fee_BlockHeaderType](#) Header
- [Fee_BlockStatusType](#) Status
- uint32 [HeadAddr](#)

4.27.1 Detailed Description

runtime Fee block information structure

Definition at line 171 of file `Fee_InternalTypes.h`.

4.27.2 Member Data Documentation

4.27.2.1 HeadAddr

```
uint32 Fee_BlockInfoType::HeadAddr
```

indicate current used block header address

Definition at line 175 of file `Fee_InternalTypes.h`.

4.27.2.2 Header

```
Fee\_BlockHeaderType Fee_BlockInfoType::Header
```

indicate current used block header information

Definition at line 173 of file `Fee_InternalTypes.h`.

4.27.2.3 Status

```
Fee\_BlockStatusType Fee_BlockInfoType::Status
```

indicate block current status

Definition at line 174 of file `Fee_InternalTypes.h`.

The documentation for this struct was generated from the following file:

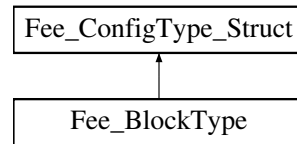
- [Fee_InternalTypes.h](#)

4.28 Fee_BlockType Struct Reference

Fee Configuration type is a stub type, not used, but required by ASR 4.2.2. .

```
#include <Fee_GeneralTypes.h>
```

Inheritance diagram for Fee_BlockType:



Public Attributes

- uint16 [BlockNumber](#)
Fee block number.
- uint16 [BlockSize](#)
Size of Fee block in bytes.
- uint8 [ClusterGrp](#)
Index of cluster group the Fee block belongs to.
- boolean [ImmediateData](#)
TRUE if immediate data block.
- [Fee_BlockAssignmentType](#) [BlockAssign](#)
specifies which project uses this block

4.28.1 Detailed Description

Fee Configuration type is a stub type, not used, but required by ASR 4.2.2. .

Definition at line 148 of file Fee_GeneralTypes.h.

4.28.2 Member Data Documentation

4.28.2.1 BlockAssign

```
Fee\_BlockAssignmentType Fee_BlockType::BlockAssign
```

specifies which project uses this block

Definition at line 154 of file Fee_GeneralTypes.h.

4.28.2.2 BlockNumber

```
uint16 Fee_BlockType::BlockNumber
```

Fee block number.

Definition at line 150 of file Fee_GeneralTypes.h.

4.28.2.3 BlockSize

```
uint16 Fee_BlockType::BlockSize
```

Size of Fee block in bytes.

Definition at line 151 of file Fee_GeneralTypes.h.

4.28.2.4 ClusterGrp

```
uint8 Fee_BlockType::ClusterGrp
```

Index of cluster group the Fee block belongs to.

Definition at line 152 of file Fee_GeneralTypes.h.

4.28.2.5 ImmediateData

```
boolean Fee_BlockType::ImmediateData
```

TRUE if immediate data block.

Definition at line 153 of file Fee_GeneralTypes.h.

The documentation for this struct was generated from the following file:

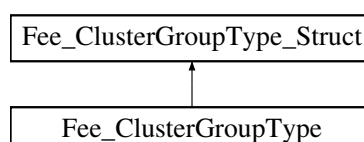
- [Fee_GeneralTypes.h](#)

4.29 Fee_ClusterGroupType Struct Reference

Fee cluster group configuration structure .

```
#include <Fee_GeneralTypes.h>
```

Inheritance diagram for Fee_ClusterGroupType:



Public Attributes

- uint16 [ClusterCount](#)
Number of clusters in cluster group.
- uint16 [ReservedSize](#)
Size of reserved area in the given cluster group (memory occupied by immediate blocks)
- const [Fee_ClusterType](#) * [ClusterPtr](#)
- const [Fee_BlockType](#) * [BlockPtr](#)
Pointer to array of Fee cluster configurations.

4.29.1 Detailed Description

Fee cluster group configuration structure .

Definition at line 162 of file `Fee_GeneralTypes.h`.

4.29.2 Member Data Documentation

4.29.2.1 BlockPtr

```
const Fee\_BlockType* Fee_ClusterGroupType::BlockPtr
```

Pointer to array of Fee cluster configurations.

<

Definition at line 170 of file `Fee_GeneralTypes.h`.

4.29.2.2 ClusterCount

```
uint16 Fee_ClusterGroupType::ClusterCount
```

Number of clusters in cluster group.

Definition at line 164 of file `Fee_GeneralTypes.h`.

4.29.2.3 ClusterPtr

```
const Fee\_ClusterType* Fee_ClusterGroupType::ClusterPtr
```

Definition at line 169 of file `Fee_GeneralTypes.h`.

4.29.2.4 ReservedSize

```
uint16 Fee_ClusterGroupType::ReservedSize
```

Size of reserved area in the given cluster group (memory occupied by immediate blocks)

Definition at line 168 of file Fee_GeneralTypes.h.

The documentation for this struct was generated from the following file:

- [Fee_GeneralTypes.h](#)

4.30 Fee_ClusterHeaderType Struct Reference

Fee cluster header configuration structure .

```
#include <Fee_InternalTypes.h>
```

Public Attributes

- uint32 [ClusterId](#)
- uint32 [StartAddr](#)
- uint32 [Length](#)
- uint32 [Checksum](#)
- uint64 [Valid](#)
- uint64 [Invalid](#)

4.30.1 Detailed Description

Fee cluster header configuration structure .

Definition at line 131 of file Fee_InternalTypes.h.

4.30.2 Member Data Documentation

4.30.2.1 CheckSum

```
uint32 Fee_ClusterHeaderType::Checksum
```

checksum value to check header whether right

Definition at line 136 of file Fee_InternalTypes.h.

4.30.2.2 ClusterId

```
uint32 Fee_ClusterHeaderType::ClusterId
```

ID of Fee cluster

Definition at line 133 of file Fee_InternalTypes.h.

4.30.2.3 Invalid

```
uint64 Fee_ClusterHeaderType::Invalid
```

indicate the header whether invalid

Definition at line 138 of file Fee_InternalTypes.h.

4.30.2.4 Length

```
uint32 Fee_ClusterHeaderType::Length
```

Size of Fee cluster in bytes

Definition at line 135 of file Fee_InternalTypes.h.

4.30.2.5 StartAddr

```
uint32 Fee_ClusterHeaderType::StartAddr
```

Address of Fee cluster in flash

Definition at line 134 of file Fee_InternalTypes.h.

4.30.2.6 Valid

```
uint64 Fee_ClusterHeaderType::Valid
```

indicate the header whether valid

Definition at line 137 of file Fee_InternalTypes.h.

The documentation for this struct was generated from the following file:

- [Fee_InternalTypes.h](#)

4.31 Fee_ClusterInfoType Struct Reference

runtime Fee Cluster information structure

```
#include <Fee_InternalTypes.h>
```

Public Attributes

- [Fee_ClusterHeaderType](#) Header
- [Fee_ClusterStatusType](#) Status

4.31.1 Detailed Description

runtime Fee Cluster information structure

Definition at line 161 of file `Fee_InternalTypes.h`.

4.31.2 Member Data Documentation

4.31.2.1 Header

```
Fee\_ClusterHeaderType Fee_ClusterInfoType::Header
```

indicate current used cluster header information

Definition at line 163 of file `Fee_InternalTypes.h`.

4.31.2.2 Status

```
Fee\_ClusterStatusType Fee_ClusterInfoType::Status
```

indicate cluster current status

Definition at line 164 of file `Fee_InternalTypes.h`.

The documentation for this struct was generated from the following file:

- [Fee_InternalTypes.h](#)

4.32 Fee_ClusterType Struct Reference

Fee cluster configuration structure .

```
#include <Fee_GeneralTypes.h>
```

Public Attributes

- uint32 [StartAddr](#)
Address of Fee cluster in flash.
- uint32 [Length](#)
Size of Fee cluster in bytes.

4.32.1 Detailed Description

Fee cluster configuration structure .

Definition at line 137 of file Fee_GeneralTypes.h.

4.32.2 Member Data Documentation

4.32.2.1 Length

```
uint32 Fee_ClusterType::Length
```

Size of Fee cluster in bytes.

Definition at line 140 of file Fee_GeneralTypes.h.

4.32.2.2 StartAddr

```
uint32 Fee_ClusterType::StartAddr
```

Address of Fee cluster in flash.

Definition at line 139 of file Fee_GeneralTypes.h.

The documentation for this struct was generated from the following file:

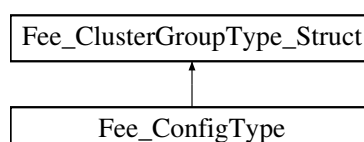
- [Fee_GeneralTypes.h](#)

4.33 Fee_ConfigType Struct Reference

Fee cluster group configuration structure .

```
#include <Fee_GeneralTypes.h>
```

Inheritance diagram for Fee_ConfigType:



Public Attributes

- uint16 [GrpCount](#)
Number of group in Fee config.
- const [Fee_ClusterGroupType](#) * [ClusterGrpPtr](#)

4.33.1 Detailed Description

Fee cluster group configuration structure .

Definition at line 178 of file [Fee_GeneralTypes.h](#).

4.33.2 Member Data Documentation

4.33.2.1 ClusterGrpPtr

```
const Fee\_ClusterGroupType* Fee_ConfigType::ClusterGrpPtr
```

Definition at line 181 of file [Fee_GeneralTypes.h](#).

4.33.2.2 GrpCount

```
uint16 Fee_ConfigType::GrpCount
```

Number of group in Fee config.

Definition at line 180 of file [Fee_GeneralTypes.h](#).

The documentation for this struct was generated from the following file:

- [Fee_GeneralTypes.h](#)

4.34 Fee_JobInfoType Struct Reference

Fee Job Infomation structure .

```
#include <Fee_InternalTypes.h>
```


Public Attributes

- [Fee_JobType](#) State
- [Fee_JobType](#) OldState
- MemIf_JobResultType [Result](#)
- MemIf_StatusType [ModuleStatus](#)
- MemIf_ModeType [Mode](#)
- boolean [BufferValid](#)
- uint16 [BlockIdx](#)
- uint16 [ClusterIdx](#)
- uint16 [GroupIdx](#)
- uint8 * [DataPtr](#)
- const uint8 * [SrcDataPtr](#)
- uint32 [DataLength](#)
- uint32 [DataOffset](#)

4.34.1 Detailed Description

Fee Job Information structure .

Definition at line 182 of file `Fee_InternalTypes.h`.

4.34.2 Member Data Documentation

4.34.2.1 BlockIdx

```
uint16 Fee_JobInfoType::BlockIdx
```

Fee block index. Used by all Fee jobs

Definition at line 190 of file `Fee_InternalTypes.h`.

4.34.2.2 BufferValid

```
boolean Fee_JobInfoType::BufferValid
```

data buffer whether valid ,used by swap and scan

Definition at line 189 of file `Fee_InternalTypes.h`.

4.34.2.3 ClusterIdx

```
uint16 Fee_JobInfoType::ClusterIdx
```

Fee cluster index. Used by all Fee jobs

Definition at line 191 of file `Fee_InternalTypes.h`.

4.34.2.4 DataLength

```
uint32 Fee_JobInfoType::DataLength
```

Size of Fee block data in bytesm, used by Fee_read and swap

Definition at line 195 of file Fee_InternalTypes.h.

4.34.2.5 DataOffset

```
uint32 Fee_JobInfoType::DataOffset
```

data offset in Fee block ,used by Fee_read

Definition at line 196 of file Fee_InternalTypes.h.

4.34.2.6 DataPtr

```
uint8* Fee_JobInfoType::DataPtr
```

tmp data buffer for data copy

Definition at line 193 of file Fee_InternalTypes.h.

4.34.2.7 GroupIdx

```
uint16 Fee_JobInfoType::GroupIdx
```

Fee group index. Used by all Fee jobs

Definition at line 192 of file Fee_InternalTypes.h.

4.34.2.8 Mode

```
MemIf_ModeType Fee_JobInfoType::Mode
```

current module mode .used by fee setmode Jobs

Definition at line 188 of file Fee_InternalTypes.h.

4.34.2.9 ModuleStatus

```
MemIf_StatusType Fee_JobInfoType::ModuleStatus
```

current module status .used by all Fee Jobs

Definition at line 187 of file Fee_InternalTypes.h.

4.34.2.10 OldState

```
Fee_JobType Fee_JobInfoType::OldState
```

save the last job state

Definition at line 185 of file Fee_InternalTypes.h.

4.34.2.11 Result

```
MemIf_JobResultType Fee_JobInfoType::Result
```

Result of last Fee module job

Definition at line 186 of file Fee_InternalTypes.h.

4.34.2.12 SrcDataPtr

```
const uint8* Fee_JobInfoType::SrcDataPtr
```

data buffer to write through by uplayer

Definition at line 194 of file Fee_InternalTypes.h.

4.34.2.13 State

```
Fee_JobType Fee_JobInfoType::State
```

next job need to execute

Definition at line 184 of file Fee_InternalTypes.h.

The documentation for this struct was generated from the following file:

- [Fee_InternalTypes.h](#)

4.35 Fee_JobPendingInfoType Struct Reference

Fee Pending Job Information structure .

```
#include <Fee_InternalTypes.h>
```

Public Attributes

- boolean [PendingValid](#)
- uint16 [PendingJob](#)
- MemIf_ModeType [Mode](#)
- uint8 * [RdDataPtr](#)
- const uint8 * [WrDataPtr](#)
- uint16 [BlockNumber](#)
- uint16 [BlockOffset](#)
- uint16 [Length](#)

4.35.1 Detailed Description

Fee Pending Job Information structure .

Definition at line 203 of file `Fee_InternalTypes.h`.

4.35.2 Member Data Documentation

4.35.2.1 BlockNumber

```
uint16 Fee_JobPendingInfoType::BlockNumber
```

pending job block number

Definition at line 210 of file `Fee_InternalTypes.h`.

4.35.2.2 BlockOffset

```
uint16 Fee_JobPendingInfoType::BlockOffset
```

pending job block data offset

Definition at line 211 of file `Fee_InternalTypes.h`.

4.35.2.3 Length

```
uint16 Fee_JobPendingInfoType::Length
```

pending job block data size

Definition at line 212 of file Fee_InternalTypes.h.

4.35.2.4 Mode

```
MemIf_ModeType Fee_JobPendingInfoType::Mode
```

save setmode pending mode

Definition at line 207 of file Fee_InternalTypes.h.

4.35.2.5 PendingJob

```
uint16 Fee_JobPendingInfoType::PendingJob
```

pending job

Definition at line 206 of file Fee_InternalTypes.h.

4.35.2.6 PendingValid

```
boolean Fee_JobPendingInfoType::PendingValid
```

pending job valid flag

Definition at line 205 of file Fee_InternalTypes.h.

4.35.2.7 RdDataPtr

```
uint8* Fee_JobPendingInfoType::RdDataPtr
```

pending read data buffer

Definition at line 208 of file Fee_InternalTypes.h.

4.35.2.8 WrDataPtr

```
const uint8* Fee_JobPendingInfoType::WrDataPtr
```

pending write data buffer

Definition at line 209 of file Fee_InternalTypes.h.

The documentation for this struct was generated from the following file:

- [Fee_InternalTypes.h](#)

4.36 Fls_ConfigType Struct Reference

```
#include <Fls_GeneralTypes.h>
```

Public Attributes

- [Fls_AcErasePtrType](#) PErasePtr
- [Fls_AcWritePtrType](#) PWritePtr
- [Fls_AcCallbackPtrType](#) PCallBackPtr
- [Fls_JobEndNotificationPtrType](#) JobEndNotificationPtr
- [Fls_JobErrorNotificationPtrType](#) JobErrorNotificationPtr
- [MemIf_ModeType](#) DefaultMode
- [Fls_LengthType](#) MaxReadFastMode
- [Fls_LengthType](#) MaxReadNormalMode
- [Fls_LengthType](#) MaxWriteFastMode
- [Fls_LengthType](#) MaxWriteNormalMode
- const uint16 [SectorCount](#)
- const [Fls_AddressType](#) * [SectorEndAddr](#)
- const [Fls_AddressType](#) * [SectorStartAddr](#)
- const [Fls_LengthType](#) * [SectorSize](#)
- const [Fls_LengthType](#) * [SectorPageSize](#)
- const [Fls_PDType](#) * [PDType](#)
- const uint8 * [SectorFlags](#)
- const uint32 [EraseTimeout](#)
- const uint32 [WriteTimeout](#)

4.36.1 Detailed Description

Definition at line 148 of file Fls_GeneralTypes.h.

4.36.2 Member Data Documentation

4.36.2.1 DefaultMode

```
MemIf_ModeType Fls_ConfigType::DefaultMode
```

Definition at line 162 of file Fls_GeneralTypes.h.

4.36.2.2 EraseTimeout

```
const uint32 Fls_ConfigType::EraseTimeout
```

Definition at line 186 of file Fls_GeneralTypes.h.

4.36.2.3 JobEndNotificationPtr

```
Fls_JobEndNotificationPtrType Fls_ConfigType::JobEndNotificationPtr
```

Definition at line 158 of file Fls_GeneralTypes.h.

4.36.2.4 JobErrorNotificationPtr

```
Fls_JobErrorNotificationPtrType Fls_ConfigType::JobErrorNotificationPtr
```

Definition at line 160 of file Fls_GeneralTypes.h.

4.36.2.5 MaxReadFastMode

```
Fls_LengthType Fls_ConfigType::MaxReadFastMode
```

Definition at line 164 of file Fls_GeneralTypes.h.

4.36.2.6 MaxReadNormalMode

```
Fls_LengthType Fls_ConfigType::MaxReadNormalMode
```

Definition at line 166 of file Fls_GeneralTypes.h.

4.36.2.7 MaxWriteFastMode

`Fls_LengthType Fls_ConfigType::MaxWriteFastMode`

Definition at line 168 of file `Fls_GeneralTypes.h`.

4.36.2.8 MaxWriteNormalMode

`Fls_LengthType Fls_ConfigType::MaxWriteNormalMode`

Definition at line 170 of file `Fls_GeneralTypes.h`.

4.36.2.9 PCallbackPtr

`Fls_AcCallbackPtrType Fls_ConfigType::PCallbackPtr`

Definition at line 156 of file `Fls_GeneralTypes.h`.

4.36.2.10 PDType

`const Fls_PDType* Fls_ConfigType::PDType`

Definition at line 182 of file `Fls_GeneralTypes.h`.

4.36.2.11 PErasePtr

`Fls_AcErasePtrType Fls_ConfigType::PErasePtr`

Definition at line 152 of file `Fls_GeneralTypes.h`.

4.36.2.12 PWritePtr

`Fls_AcWritePtrType Fls_ConfigType::PWritePtr`

Definition at line 154 of file `Fls_GeneralTypes.h`.

4.36.2.13 SectorCount

```
const uint16 Fls_ConfigType::SectorCount
```

Definition at line 172 of file Fls_GeneralTypes.h.

4.36.2.14 SectorEndAddr

```
const Fls_AddressType* Fls_ConfigType::SectorEndAddr
```

Definition at line 174 of file Fls_GeneralTypes.h.

4.36.2.15 SectorFlags

```
const uint8* Fls_ConfigType::SectorFlags
```

Definition at line 184 of file Fls_GeneralTypes.h.

4.36.2.16 SectorPageSize

```
const Fls_LengthType* Fls_ConfigType::SectorPageSize
```

Definition at line 180 of file Fls_GeneralTypes.h.

4.36.2.17 SectorSize

```
const Fls_LengthType* Fls_ConfigType::SectorSize
```

Definition at line 178 of file Fls_GeneralTypes.h.

4.36.2.18 SectorStartAddr

```
const Fls_AddressType* Fls_ConfigType::SectorStartAddr
```

Definition at line 176 of file Fls_GeneralTypes.h.

4.36.2.19 WriteTimeout

```
const uint32 Fls_ConfigType::WriteTimeout
```

Definition at line 188 of file Fls_GeneralTypes.h.

The documentation for this struct was generated from the following file:

- [Fls_GeneralTypes.h](#)

4.37 FlsTst_BlockConfigType Struct Reference

Configuration data for a Flash block.

```
#include <FlsTst.h>
```

Public Attributes

- uint32 [BlockIndex](#)
- uint32 [BlockBaseAddress](#)
- uint32 [BlockSize](#)
- uint32 [NumberOfTestedCells](#)
- uint32 [SignatureAddress](#)
- [FlsTst_TestAlgorithmType](#) [TestAlgorithm](#)

4.37.1 Detailed Description

Configuration data for a Flash block.

Definition at line 183 of file FlsTst.h.

4.37.2 Member Data Documentation

4.37.2.1 BlockBaseAddress

```
uint32 FlsTst_BlockConfigType::BlockBaseAddress
```

Definition at line 188 of file FlsTst.h.

4.37.2.2 BlockIndex

```
uint32 FlsTst_BlockConfigType::BlockIndex
```

Definition at line 186 of file FlsTst.h.

4.37.2.3 BlockSize

```
uint32 FlsTst_BlockConfigType::BlockSize
```

Definition at line 190 of file FlsTst.h.

4.37.2.4 NumberOfTestedCells

```
uint32 FlsTst_BlockConfigType::NumberOfTestedCells
```

Definition at line 192 of file FlsTst.h.

4.37.2.5 SignatureAddress

```
uint32 FlsTst_BlockConfigType::SignatureAddress
```

Definition at line 194 of file FlsTst.h.

4.37.2.6 TestAlgorithm

```
FlsTst_TestAlgorithmType FlsTst_BlockConfigType::TestAlgorithm
```

Definition at line 196 of file FlsTst.h.

The documentation for this struct was generated from the following file:

- [FlsTst.h](#)

4.38 FlsTst_CommonVariableType Struct Reference

Initialization data for the Flash Test.

```
#include <FlsTst.h>
```

Public Attributes

- uint32 [BgndBlockCellLength](#)
- uint32 [BgndBlockCellStartId](#)
- uint32 [BgndBlockCellNumber](#)
- uint32 [BgndBlockCellSignatureValue](#)
- uint8 [BgndTestAbortOrSuspendFlag](#)
- uint32 [TestIntervalId](#)
- uint32 [FgndLastSignature](#)
- uint32 [BgndLastSignature](#)
- uint32 [BgndBlockIndex](#)
- boolean [TestCompletion](#)
- [FlsTst_TestResultFgndType](#) [FgndLastResult](#)
- [FlsTst_TestResultType](#) [BgndOverallResult](#)

4.38.1 Detailed Description

Initialization data for the Flash Test.

Definition at line 217 of file FlsTst.h.

4.38.2 Member Data Documentation

4.38.2.1 BgndBlockCellLength

```
uint32 FlsTst_CommonVariableType::BgndBlockCellLength
```

Definition at line 220 of file FlsTst.h.

4.38.2.2 BgndBlockCellNumber

```
uint32 FlsTst_CommonVariableType::BgndBlockCellNumber
```

Definition at line 224 of file FlsTst.h.

4.38.2.3 BgndBlockCellSignatureValue

```
uint32 FlsTst_CommonVariableType::BgndBlockCellSignatureValue
```

Definition at line 226 of file FlsTst.h.

4.38.2.4 BgndBlockCellStartId

```
uint32 FlsTst_CommonVariableType::BgndBlockCellStartId
```

Definition at line 222 of file FlsTst.h.

4.38.2.5 BgndBlockIndex

```
uint32 FlsTst_CommonVariableType::BgndBlockIndex
```

Definition at line 236 of file FlsTst.h.

4.38.2.6 BgndLastSignature

```
uint32 FlsTst_CommonVariableType::BgndLastSignature
```

Definition at line 234 of file FlsTst.h.

4.38.2.7 BgndOverallResult

```
FlsTst_TestResultType FlsTst_CommonVariableType::BgndOverallResult
```

Definition at line 243 of file FlsTst.h.

4.38.2.8 BgndTestAbortOrSuspendFlag

```
uint8 FlsTst_CommonVariableType::BgndTestAbortOrSuspendFlag
```

Definition at line 228 of file FlsTst.h.

4.38.2.9 FgndLastResult

```
FlsTst_TestResultFgndType FlsTst_CommonVariableType::FgndLastResult
```

Definition at line 241 of file FlsTst.h.

4.38.2.10 FgndLastSignature

```
uint32 FlsTst_CommonVariableType::FgndLastSignature
```

Definition at line 232 of file FlsTst.h.

4.38.2.11 TestCompletion

```
boolean FlsTst_CommonVariableType::TestCompletion
```

Definition at line 238 of file FlsTst.h.

4.38.2.12 TestIntervalId

```
uint32 FlsTst_CommonVariableType::TestIntervalId
```

Definition at line 230 of file FlsTst.h.

The documentation for this struct was generated from the following file:

- [FlsTst.h](#)

4.39 FlsTst_ConfigType Struct Reference

Initialization data for the Flash Test.

```
#include <FlsTst.h>
```

Public Attributes

- uint32 [BgndBlockMaxNumber](#)
- uint32 [FgndBlockMaxNumber](#)
- const [FlsTst_BlockConfigType](#) * [BgndBlockConfig](#)
- const [FlsTst_BlockConfigType](#) * [FgndBlockConfig](#)
- [FlsTst_NotificationType](#) [TestCompletedNotification](#)

4.39.1 Detailed Description

Initialization data for the Flash Test.

Definition at line 200 of file FlsTst.h.

4.39.2 Member Data Documentation

4.39.2.1 BgndBlockConfig

```
const FlsTst\_BlockConfigType* FlsTst_ConfigType::BgndBlockConfig
```

Definition at line 207 of file FlsTst.h.

4.39.2.2 BgndBlockMaxNumber

```
uint32 FlsTst_ConfigType::BgndBlockMaxNumber
```

Definition at line 203 of file FlsTst.h.

4.39.2.3 FgndBlockConfig

```
const FlsTst_BlockConfigType* FlsTst_ConfigType::FgndBlockConfig
```

Definition at line 209 of file FlsTst.h.

4.39.2.4 FgndBlockMaxNumber

```
uint32 FlsTst_ConfigType::FgndBlockMaxNumber
```

Definition at line 205 of file FlsTst.h.

4.39.2.5 TestCompletedNotification

```
FlsTst_NotificationType FlsTst_ConfigType::TestCompletedNotification
```

Definition at line 212 of file FlsTst.h.

The documentation for this struct was generated from the following file:

- [FlsTst.h](#)

4.40 FlsTst_ErrorDetailsType Struct Reference

Test error information.

```
#include <FlsTst.h>
```

Public Attributes

- FlsTst_EccErrorStateType [EccStatus](#)
- uint32 [EccFaultAddress](#)

4.40.1 Detailed Description

Test error information.

Definition at line 174 of file FlsTst.h.

4.40.2 Member Data Documentation

4.40.2.1 EccFaultAddress

```
uint32 FlsTst_ErrorDetailsType::EccFaultAddress
```

Definition at line 179 of file FlsTst.h.

4.40.2.2 EccStatus

```
FlsTst_EccErrorStateType FlsTst_ErrorDetailsType::EccStatus
```

Definition at line 177 of file FlsTst.h.

The documentation for this struct was generated from the following file:

- [FlsTst.h](#)

4.41 FlsTst_TestResultBgndType Struct Reference

Test result of Flash test in background mode.

```
#include <FlsTst.h>
```

Public Attributes

- uint32 [TestIntervalId](#)
- [FlsTst_TestResultType](#) TestResultBgnd

4.41.1 Detailed Description

Test result of Flash test in background mode.

Definition at line 149 of file FlsTst.h.

4.41.2 Member Data Documentation

4.41.2.1 TestIntervalId

```
uint32 FlsTst_TestResultBgndType::TestIntervalId
```

Definition at line 152 of file FlsTst.h.

4.41.2.2 TestResultBgnd

[FlsTst_TestResultType](#) `FlsTst_TestResultBgndType::TestResultBgnd`

Definition at line 154 of file FlsTst.h.

The documentation for this struct was generated from the following file:

- [FlsTst.h](#)

4.42 FlsTst_TestSignatureBgndType Struct Reference

Test result signature of Flash test in background mode.

```
#include <FlsTst.h>
```

Public Attributes

- uint32 [TestIntervalId](#)
- uint32 [TestSignatureValue](#)

4.42.1 Detailed Description

Test result signature of Flash test in background mode.

Definition at line 158 of file FlsTst.h.

4.42.2 Member Data Documentation

4.42.2.1 TestIntervalId

`uint32 FlsTst_TestSignatureBgndType::TestIntervalId`

Definition at line 161 of file FlsTst.h.

4.42.2.2 TestSignatureValue

`uint32 FlsTst_TestSignatureBgndType::TestSignatureValue`

Definition at line 163 of file FlsTst.h.

The documentation for this struct was generated from the following file:

- [FlsTst.h](#)

4.43 FlsTst_TestSignatureFgndType Struct Reference

Test result signature of Flash test in foreground mode.

```
#include <FlsTst.h>
```

Public Attributes

- uint32 [TestSignatureValue](#)

4.43.1 Detailed Description

Test result signature of Flash test in foreground mode.

Definition at line 167 of file FlsTst.h.

4.43.2 Member Data Documentation

4.43.2.1 TestSignatureValue

```
uint32 FlsTst_TestSignatureFgndType::TestSignatureValue
```

Definition at line 170 of file FlsTst.h.

The documentation for this struct was generated from the following file:

- [FlsTst.h](#)

4.44 I2c_CHConfigType Struct Reference

```
#include <CDD_I2c.h>
```

Public Attributes

- uint8 [HwChannel](#)
- const I2c_Hal_ChannelConfigType * [ChConfigPtr](#)

4.44.1 Detailed Description

Definition at line 89 of file CDD_I2c.h.

4.44.2 Member Data Documentation

4.44.2.1 ChConfigPtr

```
const I2c_Hal_ChannelConfigType* I2c_CHConfigType::ChConfigPtr
```

Definition at line 92 of file CDD_I2c.h.

4.44.2.2 HwChannel

```
uint8 I2c_CHConfigType::HwChannel
```

The I2C HwChannel number

Definition at line 91 of file CDD_I2c.h.

The documentation for this struct was generated from the following file:

- [CDD_I2c.h](#)

4.45 I2c_ConfigType Struct Reference

```
#include <CDD_I2c.h>
```

Public Attributes

- uint8 [MaxChannelNum](#)
- const [I2c_CHConfigType](#) * [ChannelConfigPtr](#)

4.45.1 Detailed Description

Definition at line 96 of file CDD_I2c.h.

4.45.2 Member Data Documentation

4.45.2.1 ChannelConfigPtr

```
const I2c_CHConfigType* I2c_ConfigType::ChannelConfigPtr
```

Definition at line 99 of file CDD_I2c.h.

4.45.2.2 MaxChannelNum

```
uint8 I2c_ConfigType::MaxChannelNum
```

Definition at line 98 of file CDD_I2c.h.

The documentation for this struct was generated from the following file:

- [CDD_I2c.h](#)

4.46 Icu_ChannelConfigType Struct Reference

Definition of ICU channel configuration.

```
#include <Icu.h>
```

Public Attributes

- const [Icu_ChannelType](#) LogicId
- const uint32 [Channel](#)
- const uint8 [ModuleId](#)
- const [Icu_MeasurementModeType](#) MeasurementMode
- const [Icu_ActivationType](#) DefaultStartEdge
- const [Icu_NotifyType](#) ChannelNotification
- const [Icu_HwSelect](#) HwSelect
- const [Icu_SignalMeasurementPropertyType](#) SignalMeasurementPropertyType
- const [Icu_TimestampBufferType](#) TimestampBufferType
- const boolean [WakeupCapable](#)
- const [EcuM_WakeupSourceType](#) WakeupSource

4.46.1 Detailed Description

Definition of ICU channel configuration.

Definition at line 224 of file Icu.h.

4.46.2 Member Data Documentation

4.46.2.1 Channel

```
const uint32 Icu_ChannelConfigType::Channel
```

hardware channel id

Definition at line 227 of file Icu.h.

4.46.2.2 ChannelNotification

```
const Icu_NotifyType Icu_ChannelConfigType::ChannelNotification
```

The address of the channel event callback function

Definition at line 231 of file Icu.h.

4.46.2.3 DefaultStartEdge

```
const Icu_ActivationType Icu_ChannelConfigType::DefaultStartEdge
```

Default activation edge

Definition at line 230 of file Icu.h.

4.46.2.4 HwSelect

```
const Icu_HwSelect Icu_ChannelConfigType::HwSelect
```

Hardware type used

Definition at line 232 of file Icu.h.

4.46.2.5 LogicId

```
const Icu_ChannelType Icu_ChannelConfigType::LogicId
```

Logic channel id

Definition at line 226 of file Icu.h.

4.46.2.6 MeasurementMode

```
const Icu_MeasurementModeType Icu_ChannelConfigType::MeasurementMode
```

Channel measurement mode configured

Definition at line 229 of file Icu.h.

4.46.2.7 ModuleId

```
const uint8 Icu_ChannelConfigType::ModuleId
```

hardware module id

Definition at line 228 of file Icu.h.

4.46.2.8 SignalMeasurementPropertyType

```
const Icu_SignalMeasurementPropertyType Icu_ChannelConfigType::SignalMeasurementPropertyType
```

The measurement property type

Definition at line 233 of file Icu.h.

4.46.2.9 TimestampBufferType

```
const Icu_TimestampBufferType Icu_ChannelConfigType::TimestampBufferType
```

The timestamp measurement property type

Definition at line 234 of file Icu.h.

4.46.2.10 WakeupCapable

```
const boolean Icu_ChannelConfigType::WakeupCapable
```

Does the channel have wake-up capability

Definition at line 235 of file Icu.h.

4.46.2.11 WakeupSource

```
const EcuM_WakeupSourceType Icu_ChannelConfigType::WakeupSource
```

Selected wake-up source

Definition at line 236 of file Icu.h.

The documentation for this struct was generated from the following file:

- [Icu.h](#)

4.47 Icu_ConfigType Struct Reference

Definition of all ICU configurations.

```
#include <Icu.h>
```

Public Attributes

- const [Icu_ChannelConfigType](#) * [ChannelConfigPtr](#)
- const [Icu_IpConfigType](#) * [IpConfigPtr](#)

4.47.1 Detailed Description

Definition of all ICU configurations.

Definition at line 240 of file Icu.h.

4.47.2 Member Data Documentation

4.47.2.1 ChannelConfigPtr

```
const Icu\_ChannelConfigType* Icu_ConfigType::ChannelConfigPtr
```

Address of channel configuration array

Definition at line 242 of file Icu.h.

4.47.2.2 IpConfigPtr

```
const Icu\_IpConfigType* Icu_ConfigType::IpConfigPtr
```

Address of channel configuration array

Definition at line 243 of file Icu.h.

The documentation for this struct was generated from the following file:

- [Icu.h](#)

4.48 Icu_DutyCycleType Struct Reference

Type which shall contain the values, needed for calculating duty cycles.

```
#include <Icu.h>
```

Public Attributes

- [Icu_ValueType](#) [ActiveTime](#)
- [Icu_ValueType](#) [PeriodTime](#)

4.48.1 Detailed Description

Type which shall contain the values, needed for calculating duty cycles.

Definition at line 200 of file Icu.h.

4.48.2 Member Data Documentation

4.48.2.1 ActiveTime

```
Icu\_ValueType Icu_DutyCycleType::ActiveTime
```

This shall be the coherent active-time

Definition at line 202 of file Icu.h.

4.48.2.2 PeriodTime

```
Icu\_ValueType Icu_DutyCycleType::PeriodTime
```

This shall be the coherent period-time

Definition at line 203 of file Icu.h.

The documentation for this struct was generated from the following file:

- [Icu.h](#)

4.49 Icu_IpConfigType Struct Reference

Definition of ICU hardware module set configuration.

```
#include <Icu.h>
```

Public Attributes

- const uint8 [PwmModuleCount](#)
- const [Icu_PwmModuleConfigType](#) * [PwmModuleConfigPtr](#)

4.49.1 Detailed Description

Definition of ICU hardware module set configuration.

Definition at line 216 of file Icu.h.

4.49.2 Member Data Documentation

4.49.2.1 PwmModuleConfigPtr

```
const Icu_PwmModuleConfigType* Icu_IpConfigType::PwmModuleConfigPtr
```

Address of PWM module configuration array

Definition at line 219 of file Icu.h.

4.49.2.2 PwmModuleCount

```
const uint8 Icu_IpConfigType::PwmModuleCount
```

Number of PWM modules used in ICU

Definition at line 218 of file Icu.h.

The documentation for this struct was generated from the following file:

- [Icu.h](#)

4.50 Icu_PwmModuleConfigType Struct Reference

Definition of PWM module configuration.

```
#include <Icu.h>
```

Public Attributes

- const Pwm_Hal_InstanceType [Instance](#)
- const Pwm_Hal_ClockSourceType [ClockSource](#)
- const uint16 [Prescaler](#)

4.50.1 Detailed Description

Definition of PWM module configuration.

Definition at line 208 of file Icu.h.

4.50.2 Member Data Documentation

4.50.2.1 ClockSource

```
const Pwm_Hal_ClockSourceType Icu_PwmModuleConfigType::ClockSource
```

PWM module clock source

Definition at line 211 of file Icu.h.

4.50.2.2 Instance

```
const Pwm_Hal_InstanceType Icu_PwmModuleConfigType::Instance
```

PWM module instance

Definition at line 210 of file Icu.h.

4.50.2.3 Prescaler

```
const uint16 Icu_PwmModuleConfigType::Prescaler
```

PWM module clock Prescaler

Definition at line 212 of file Icu.h.

The documentation for this struct was generated from the following file:

- [Icu.h](#)

4.51 Lin_ConfigType Struct Reference

```
#include <Lin.h>
```

Public Attributes

- const Lin_HwConfigType * [Lin_HwConfigPtr](#)

4.51.1 Detailed Description

Definition at line 90 of file Lin.h.

4.51.2 Member Data Documentation

4.51.2.1 Lin_HwConfigPtr

```
const Lin_HwConfigType* Lin_ConfigType::Lin_HwConfigPtr
```

Definition at line 92 of file Lin.h.

The documentation for this struct was generated from the following file:

- [Lin.h](#)

4.52 Mcu_ConfigType Struct Reference

A structure to hold the MCU driver configuration.

```
#include <Mcu.h>
```

Public Attributes

- const Mcu_DemConfigType * [Mcu_DemConfigPtr](#)
- Mcu_RamSectionType [Mcu_RamConfigNum](#)
- Mcu_ModeType [Mcu_ModeConfigNum](#)
- Mcu_ClockType [Mcu_ClkConfigNum](#)
- const Mcu_RamConfigType(* [Mcu_RamConfigPtr](#))[MCU_RAMCONFIG_MAX_NUM]
- const Mcu_ModeConfigType(* [Mcu_ModeConfigPtr](#))[MCU_MODECONFIG_MAX_NUM]
- const Mcu_ClockConfigType(* [Mcu_ClockConfigPtr](#))[MCU_CLOCKCONFIG_MAX_NUM]
- const Mcu_CommonConfigType * [Mcu_CommonConfigPtr](#)

4.52.1 Detailed Description

A structure to hold the MCU driver configuration.

A pointer to such a structure is provided to the MCU initialization routines for configuration.

Definition at line 91 of file Mcu.h.

4.52.2 Member Data Documentation

4.52.2.1 Mcu_ClkConfigNum

```
Mcu_ClockType Mcu_ConfigType::Mcu_ClkConfigNum
```

Definition at line 99 of file Mcu.h.

4.52.2.2 Mcu_ClockConfigPtr

```
const Mcu_ClockConfigType(* Mcu_ConfigType::Mcu_ClockConfigPtr) [MCU_CLOCKCONFIG_MAX_NUM]
```

Definition at line 106 of file Mcu.h.

4.52.2.3 Mcu_CommonConfigPtr

```
const Mcu_CommonConfigType* Mcu_ConfigType::Mcu_CommonConfigPtr
```

Definition at line 108 of file Mcu.h.

4.52.2.4 Mcu_DemConfigPtr

```
const Mcu_DemConfigType* Mcu_ConfigType::Mcu_DemConfigPtr
```

Definition at line 94 of file Mcu.h.

4.52.2.5 Mcu_ModeConfigNum

```
Mcu_ModeType Mcu_ConfigType::Mcu_ModeConfigNum
```

Definition at line 97 of file Mcu.h.

4.52.2.6 Mcu_ModeConfigPtr

```
const Mcu_ModeConfigType(* Mcu_ConfigType::Mcu_ModeConfigPtr) [MCU_MODECONFIG_MAX_NUM]
```

Definition at line 104 of file Mcu.h.

4.52.2.7 Mcu_RamConfigNum

```
Mcu_RamSectionType Mcu_ConfigType::Mcu_RamConfigNum
```

Definition at line 96 of file Mcu.h.

4.52.2.8 Mcu_RamConfigPtr

```
const Mcu_RamConfigType (* Mcu_ConfigType::Mcu_RamConfigPtr) [MCU_RAMCONFIG_MAX_NUM]
```

Definition at line 102 of file Mcu.h.

The documentation for this struct was generated from the following file:

- [Mcu.h](#)

4.53 Ocu_ChannelConfigType Struct Reference

```
#include <Ocu.h>
```

Public Attributes

- const [Ocu_ChannelType](#) LogicId
- const Pwm_Hal_InstanceType [PwmModulesId](#)
- const Pwm_Hal_ChannelType [Channel](#)
- const [Ocu_ValueType](#) ChannelDefaultThreshold
- const boolean [ChannelOutputPinUsed](#)
- const [Ocu_PinActionType](#) ChannelPinAction
- const Pwm_Hal_OutputLevelType [ChannelInitLevel](#)
- const Ocu_NotifyType [ChannelNotification](#)

4.53.1 Detailed Description

Definition at line 106 of file Ocu.h.

4.53.2 Member Data Documentation

4.53.2.1 Channel

```
const Pwm_Hal_ChannelType Ocu_ChannelConfigType::Channel
```

Definition at line 110 of file Ocu.h.

4.53.2.2 ChannelDefaultThreshold

```
const Ocu\_ValueType Ocu_ChannelConfigType::ChannelDefaultThreshold
```

Definition at line 111 of file Ocu.h.

4.53.2.3 ChannelInitLevel

```
const Pwm_Hal_OutputLevelType Ocu_ChannelConfigType::ChannelInitLevel
```

Definition at line 114 of file Ocu.h.

4.53.2.4 ChannelNotification

```
const Ocu_NotifyType Ocu_ChannelConfigType::ChannelNotification
```

Definition at line 115 of file Ocu.h.

4.53.2.5 ChannelOutputPinUsed

```
const boolean Ocu_ChannelConfigType::ChannelOutputPinUsed
```

Definition at line 112 of file Ocu.h.

4.53.2.6 ChannelPinAction

```
const Ocu_PinActionType Ocu_ChannelConfigType::ChannelPinAction
```

Definition at line 113 of file Ocu.h.

4.53.2.7 LogicId

```
const Ocu_ChannelType Ocu_ChannelConfigType::LogicId
```

Definition at line 108 of file Ocu.h.

4.53.2.8 PwmModulesId

```
const Pwm_Hal_InstanceType Ocu_ChannelConfigType::PwmModulesId
```

Definition at line 109 of file Ocu.h.

The documentation for this struct was generated from the following file:

- [Ocu.h](#)

4.54 Ocu_ConfigType Struct Reference

```
#include <Ocu.h>
```

Public Attributes

- const [Ocu_ChannelType](#) OcuChannelCount
- const [Ocu_ChannelConfigType](#) * OcuChannelConfig
- const [Ocu_IpConfigType](#) * IpConfig

4.54.1 Detailed Description

Definition at line 132 of file Ocu.h.

4.54.2 Member Data Documentation

4.54.2.1 IpConfig

```
const Ocu\_IpConfigType* Ocu_ConfigType::IpConfig
```

Definition at line 136 of file Ocu.h.

4.54.2.2 OcuChannelConfig

```
const Ocu\_ChannelConfigType* Ocu_ConfigType::OcuChannelConfig
```

Definition at line 135 of file Ocu.h.

4.54.2.3 OcuChannelCount

```
const Ocu\_ChannelType Ocu_ConfigType::OcuChannelCount
```

Definition at line 134 of file Ocu.h.

The documentation for this struct was generated from the following file:

- [Ocu.h](#)

4.55 Ocu_IpConfigType Struct Reference

```
#include <Ocu.h>
```

Public Attributes

- const uint16 [OcuModuleCount](#)
- const [Ocu_ModuleConfigType](#) * [OcuModuleConfig](#)

4.55.1 Detailed Description

Definition at line 126 of file Ocu.h.

4.55.2 Member Data Documentation

4.55.2.1 OcuModuleConfig

```
const Ocu\_ModuleConfigType* Ocu_IpConfigType::OcuModuleConfig
```

Definition at line 129 of file Ocu.h.

4.55.2.2 OcuModuleCount

```
const uint16 Ocu_IpConfigType::OcuModuleCount
```

Definition at line 128 of file Ocu.h.

The documentation for this struct was generated from the following file:

- [Ocu.h](#)

4.56 Ocu_ModuleConfigType Struct Reference

```
#include <Ocu.h>
```

Public Attributes

- const Pwm_Hal_InstanceType [ModuleId](#)
- const Pwm_Hal_ClockSourceType [ClockSource](#)
- const uint16 [Prescaler](#)
- const uint16 [ModuleMaxValue](#)

4.56.1 Detailed Description

Definition at line 118 of file Ocu.h.

4.56.2 Member Data Documentation

4.56.2.1 ClockSource

```
const Pwm_Hal_ClockSourceType Ocu_ModuleConfigType::ClockSource
```

Definition at line 121 of file Ocu.h.

4.56.2.2 ModuleId

```
const Pwm_Hal_InstanceType Ocu_ModuleConfigType::ModuleId
```

Definition at line 120 of file Ocu.h.

4.56.2.3 ModuleMaxValue

```
const uint16 Ocu_ModuleConfigType::ModuleMaxValue
```

Definition at line 123 of file Ocu.h.

4.56.2.4 Prescaler

```
const uint16 Ocu_ModuleConfigType::Prescaler
```

Definition at line 122 of file Ocu.h.

The documentation for this struct was generated from the following file:

- [Ocu.h](#)

4.57 Port_ConfigType Struct Reference

all port module configure type

```
#include <Port_GeneralTypes.h>
```

Public Attributes

- uint16 [NumPins](#)
- uint16 [NumUnusedPins](#)
- const uint16 * [UnusedPads](#)
- const [Port_UnUsedPinConfigType](#) * [UnusedPadConfig](#)
- const [Port_PinConfigType](#) * [UsedPadConfig](#)
- const uint8 [NumDigitalFilterPorts](#)
- const [Port_DigitalFilterConfigType](#) * [DigitalFilterConfig](#)

4.57.1 Detailed Description

all port module configure type

Definition at line 156 of file Port_GeneralTypes.h.

4.57.2 Member Data Documentation

4.57.2.1 DigitalFilterConfig

```
const Port\_DigitalFilterConfigType* Port_ConfigType::DigitalFilterConfig
```

Digital filter ports configuration

Definition at line 164 of file Port_GeneralTypes.h.

4.57.2.2 NumDigitalFilterPorts

```
const uint8 Port_ConfigType::NumDigitalFilterPorts
```

Number of configured digital filter ports

Definition at line 163 of file Port_GeneralTypes.h.

4.57.2.3 NumPins

```
uint16 Port_ConfigType::NumPins
```

Number of used pads (to be configured)

Definition at line 158 of file Port_GeneralTypes.h.

4.57.2.4 NumUnusedPins

```
uint16 Port_ConfigType::NumUnusedPins
```

Number of unused pads

Definition at line 159 of file Port_GeneralTypes.h.

4.57.2.5 UnusedPadConfig

```
const Port_UnUsedPinConfigType* Port_ConfigType::UnusedPadConfig
```

Unused pad configuration

Definition at line 161 of file Port_GeneralTypes.h.

4.57.2.6 UnusedPads

```
const uint16* Port_ConfigType::UnusedPads
```

Unused pad id's array

Definition at line 160 of file Port_GeneralTypes.h.

4.57.2.7 UsedPadConfig

```
const Port_PinConfigType* Port_ConfigType::UsedPadConfig
```

Used pads data configuration

Definition at line 162 of file Port_GeneralTypes.h.

The documentation for this struct was generated from the following file:

- [Port_GeneralTypes.h](#)

4.58 Port_DigitalFilterConfigType Struct Reference

Port digital filter config type.

```
#include <Port_GeneralTypes.h>
```

Public Attributes

- uint8 [PortID](#)
- uint8 [Clock](#)
- uint8 [Width](#)
- uint32 [PinMask](#)

4.58.1 Detailed Description

Port digital filter config type.

Definition at line 127 of file Port_GeneralTypes.h.

4.58.2 Member Data Documentation

4.58.2.1 Clock

```
uint8 Port_DigitalFilterConfigType::Clock
```

Definition at line 130 of file Port_GeneralTypes.h.

4.58.2.2 PinMask

```
uint32 Port_DigitalFilterConfigType::PinMask
```

Definition at line 132 of file Port_GeneralTypes.h.

4.58.2.3 PortID

```
uint8 Port_DigitalFilterConfigType::PortID
```

Definition at line 129 of file Port_GeneralTypes.h.

4.58.2.4 Width

```
uint8 Port_DigitalFilterConfigType::Width
```

Definition at line 131 of file Port_GeneralTypes.h.

The documentation for this struct was generated from the following file:

- [Port_GeneralTypes.h](#)

4.59 Port_PinConfigType Struct Reference

each used Pin configure type

```
#include <Port_GeneralTypes.h>
```

Public Attributes

- uint32 [PCR](#)
- uint16 [Pin](#)
- uint8 [DataOut](#)
- [Port_PinDirectionType](#) [Direction](#)
- boolean [GPIOMode](#)
- boolean [DirChangeable](#)
- boolean [ModeChangeable](#)

4.59.1 Detailed Description

each used Pin configure type

Definition at line 136 of file Port_GeneralTypes.h.

4.59.2 Member Data Documentation

4.59.2.1 DataOut

```
uint8 Port_PinConfigType::DataOut
```

Pad Data Output

Definition at line 140 of file Port_GeneralTypes.h.

4.59.2.2 DirChangeable

```
boolean Port_PinConfigType::DirChangeable
```

Direction changeability

Definition at line 143 of file Port_GeneralTypes.h.

4.59.2.3 Direction

`Port_PinDirectionType` `Port_PinConfigType::Direction`

Pad Data Direction

Definition at line 141 of file `Port_GeneralTypes.h`.

4.59.2.4 GPIOMode

`boolean` `Port_PinConfigType::GPIOMode`

GPIO initial mode

Definition at line 142 of file `Port_GeneralTypes.h`.

4.59.2.5 ModeChangeable

`boolean` `Port_PinConfigType::ModeChangeable`

Mode changeability

Definition at line 144 of file `Port_GeneralTypes.h`.

4.59.2.6 PCR

`uint32` `Port_PinConfigType::PCR`

Pad Control Register

Definition at line 138 of file `Port_GeneralTypes.h`.

4.59.2.7 Pin

`uint16` `Port_PinConfigType::Pin`

Pin Defined on PORT

Definition at line 139 of file `Port_GeneralTypes.h`.

The documentation for this struct was generated from the following file:

- [Port_GeneralTypes.h](#)

4.60 Port_UnUsedPinConfigType Struct Reference

each unused Pin configure type

```
#include <Port_GeneralTypes.h>
```

Public Attributes

- uint32 [PCR](#)
- uint8 [DataOut](#)
- [Port_PinDirectionType](#) [Direction](#)

4.60.1 Detailed Description

each unused Pin configure type

Definition at line 148 of file Port_GeneralTypes.h.

4.60.2 Member Data Documentation

4.60.2.1 DataOut

```
uint8 Port_UnUsedPinConfigType::DataOut
```

Pad Data Output

Definition at line 151 of file Port_GeneralTypes.h.

4.60.2.2 Direction

```
Port\_PinDirectionType Port_UnUsedPinConfigType::Direction
```

Pad Data Direction

Definition at line 152 of file Port_GeneralTypes.h.

4.60.2.3 PCR

```
uint32 Port_UnUsedPinConfigType::PCR
```

Pad Control Register

Definition at line 150 of file Port_GeneralTypes.h.

The documentation for this struct was generated from the following file:

- [Port_GeneralTypes.h](#)

4.61 Pwm_ChannelConfigType Struct Reference

```
#include <Pwm.h>
```

Public Attributes

- const uint8 [LogicId](#)
- const Pwm_Hal_InstanceType [PwmModulesId](#)
- const Pwm_Hal_ChannelType [Channel](#)
- const Pwm_Hal_OutputChnModeType [ChannelMode](#)
- const uint16 [PwmDefaultDutycycle](#)
- const Pwm_Hal_OutputLevelType [IdleState](#)
- const Pwm_Hal_OutputLevelModeType [Polarity](#)
- const [Pwm_ChannelClassType](#) [ChannelClass](#)
- const Pwm_NotifyType [ChannelNotification](#)
- const boolean [EnableInterrupt](#)
- const boolean [EnableCombaineComple](#)
- const boolean [EnableMatchTrigger](#)
- boolean [EnableDeadTime](#)
- uint16 [DeadTimeValue](#)
- Pwm_Hal_DeadTimePscType [TimePsc](#)

4.61.1 Detailed Description

Definition at line 110 of file Pwm.h.

4.61.2 Member Data Documentation

4.61.2.1 Channel

```
const Pwm_Hal_ChannelType Pwm_ChannelConfigType::Channel
```

Definition at line 114 of file Pwm.h.

4.61.2.2 ChannelClass

```
const Pwm\_ChannelClassType Pwm_ChannelConfigType::ChannelClass
```

Definition at line 119 of file Pwm.h.

4.61.2.3 ChannelMode

```
const Pwm_Hal_OutputChnModeType Pwm_ChannelConfigType::ChannelMode
```

Definition at line 115 of file Pwm.h.

4.61.2.4 ChannelNotification

```
const Pwm_NotifyType Pwm_ChannelConfigType::ChannelNotification
```

Definition at line 120 of file Pwm.h.

4.61.2.5 DeadTimeValue

```
uint16 Pwm_ChannelConfigType::DeadTimeValue
```

Definition at line 125 of file Pwm.h.

4.61.2.6 EnableCombaineComple

```
const boolean Pwm_ChannelConfigType::EnableCombaineComple
```

Definition at line 122 of file Pwm.h.

4.61.2.7 EnableDeadTime

```
boolean Pwm_ChannelConfigType::EnableDeadTime
```

Definition at line 124 of file Pwm.h.

4.61.2.8 EnableInterrupt

```
const boolean Pwm_ChannelConfigType::EnableInterrupt
```

Definition at line 121 of file Pwm.h.

4.61.2.9 EnableMatchTrigger

```
const boolean Pwm_ChannelConfigType::EnableMatchTrigger
```

Definition at line 123 of file Pwm.h.

4.61.2.10 IdleState

```
const Pwm_Hal_OutputLevelType Pwm_ChannelConfigType::IdleState
```

Definition at line 117 of file Pwm.h.

4.61.2.11 LogicId

```
const uint8 Pwm_ChannelConfigType::LogicId
```

Definition at line 112 of file Pwm.h.

4.61.2.12 Polarity

```
const Pwm_Hal_OutputLevelModeType Pwm_ChannelConfigType::Polarity
```

Definition at line 118 of file Pwm.h.

4.61.2.13 PwmDefaultDutycycle

```
const uint16 Pwm_ChannelConfigType::PwmDefaultDutycycle
```

Definition at line 116 of file Pwm.h.

4.61.2.14 PwmModulesId

```
const Pwm_Hal_InstanceType Pwm_ChannelConfigType::PwmModulesId
```

Definition at line 113 of file Pwm.h.

4.61.2.15 TimePsc

```
Pwm_Hal_DeadTimePscType Pwm_ChannelConfigType::TimePsc
```

Definition at line 126 of file Pwm.h.

The documentation for this struct was generated from the following file:

- [Pwm.h](#)

4.62 Pwm_ConfigType Struct Reference

```
#include <Pwm.h>
```

Public Attributes

- const uint8 [PwmChannelCount](#)
- const [Pwm_ChannelConfigType](#) * [ChannelCfg](#)
- const [Pwm_IpConfigType](#) * [IpConfigPtr](#)

4.62.1 Detailed Description

Definition at line 155 of file Pwm.h.

4.62.2 Member Data Documentation

4.62.2.1 ChannelCfg

```
const Pwm\_ChannelConfigType* Pwm_ConfigType::ChannelCfg
```

Definition at line 158 of file Pwm.h.

4.62.2.2 IpConfigPtr

```
const Pwm\_IpConfigType* Pwm_ConfigType::IpConfigPtr
```

Definition at line 159 of file Pwm.h.

4.62.2.3 PwmChannelCount

```
const uint8 Pwm_ConfigType::PwmChannelCount
```

Definition at line 157 of file Pwm.h.

The documentation for this struct was generated from the following file:

- [Pwm.h](#)

4.63 Pwm_IpConfigType Struct Reference

```
#include <Pwm.h>
```

Public Attributes

- const uint8 [PwmModuleCount](#)
- const [Pwm_ModuleConfigType](#) * [ModuleCfg](#)

4.63.1 Detailed Description

Definition at line 147 of file Pwm.h.

4.63.2 Member Data Documentation

4.63.2.1 ModuleCfg

```
const Pwm\_ModuleConfigType* Pwm_IpConfigType::ModuleCfg
```

Definition at line 150 of file Pwm.h.

4.63.2.2 PwmModuleCount

```
const uint8 Pwm_IpConfigType::PwmModuleCount
```

Definition at line 149 of file Pwm.h.

The documentation for this struct was generated from the following file:

- [Pwm.h](#)

4.64 Pwm_ModuleConfigType Struct Reference

```
#include <Pwm.h>
```

Public Attributes

- const Pwm_Hal_InstanceType [Instance](#)
- const Pwm_Hal_ClockSourceType [ClockSource](#)
- const uint16 [Prescaler](#)
- const [Pwm_PeriodType](#) [Period](#)
- const Pwm_Hal_CountModeType [CountMode](#)
- const boolean [EnableInterrupt](#)
- const boolean [EnableOverflowEvent](#)
- const Pwm_NotifyType [OverflowNotification](#)
- boolean [EnableInitTrigger](#)
- boolean [EnableMaxTrigger](#)
- uint8 [TriggerRatio](#)

4.64.1 Detailed Description

Definition at line 131 of file Pwm.h.

4.64.2 Member Data Documentation

4.64.2.1 ClockSource

```
const Pwm_Hal_ClockSourceType Pwm_ModuleConfigType::ClockSource
```

Definition at line 134 of file Pwm.h.

4.64.2.2 CountMode

```
const Pwm_Hal_CountModeType Pwm_ModuleConfigType::CountMode
```

Definition at line 137 of file Pwm.h.

4.64.2.3 EnableInitTrigger

```
boolean Pwm_ModuleConfigType::EnableInitTrigger
```

Definition at line 141 of file Pwm.h.

4.64.2.4 EnableInterrupt

```
const boolean Pwm_ModuleConfigType::EnableInterrupt
```

Definition at line 138 of file Pwm.h.

4.64.2.5 EnableMaxTrigger

```
boolean Pwm_ModuleConfigType::EnableMaxTrigger
```

Definition at line 142 of file Pwm.h.

4.64.2.6 EnableOverflowEvent

```
const boolean Pwm_ModuleConfigType::EnableOverflowEvent
```

Definition at line 139 of file Pwm.h.

4.64.2.7 Instance

```
const Pwm_Hal_InstanceType Pwm_ModuleConfigType::Instance
```

Definition at line 133 of file Pwm.h.

4.64.2.8 OverflowNotification

```
const Pwm_NotifyType Pwm_ModuleConfigType::OverflowNotification
```

Definition at line 140 of file Pwm.h.

4.64.2.9 Period

```
const Pwm\_PeriodType Pwm_ModuleConfigType::Period
```

Definition at line 136 of file Pwm.h.

4.64.2.10 Prescaler

```
const uint16 Pwm_ModuleConfigType::Prescaler
```

Definition at line 135 of file Pwm.h.

4.64.2.11 TriggerRatio

```
uint8 Pwm_ModuleConfigType::TriggerRatio
```

Definition at line 143 of file Pwm.h.

The documentation for this struct was generated from the following file:

- [Pwm.h](#)

4.65 Sent_ConfigType Struct Reference

```
#include <CDD_Sent.h>
```

Public Attributes

- const uint8 [MaxChannelNum](#)
- const Sent_ModuleConfigType * [ModuleConfig](#)
- const Sent_HwConfigType * [HwConfigPtr](#)

4.65.1 Detailed Description

Definition at line 81 of file CDD_Sent.h.

4.65.2 Member Data Documentation

4.65.2.1 HwConfigPtr

```
const Sent_HwConfigType* Sent_ConfigType::HwConfigPtr
```

Definition at line 85 of file CDD_Sent.h.

4.65.2.2 MaxChannelNum

```
const uint8 Sent_ConfigType::MaxChannelNum
```

Definition at line 83 of file CDD_Sent.h.

4.65.2.3 ModuleConfig

```
const Sent_ModuleConfigType* Sent_ConfigType::ModuleConfig
```

Definition at line 84 of file CDD_Sent.h.

The documentation for this struct was generated from the following file:

- [CDD_Sent.h](#)

4.66 Uart_ConfigType Struct Reference

UART module configuration structure.

```
#include <CDD_Uart.h>
```

Public Attributes

- const uint8 [MaxChannelNum](#)
- const Uart_HwConfigType * [HwConfigPtr](#)

4.66.1 Detailed Description

UART module configuration structure.

Definition at line 85 of file CDD_Uart.h.

4.66.2 Member Data Documentation

4.66.2.1 HwConfigPtr

```
const Uart_HwConfigType* Uart_ConfigType::HwConfigPtr
```

Definition at line 88 of file CDD_Uart.h.

4.66.2.2 MaxChannelNum

```
const uint8 Uart_ConfigType::MaxChannelNum
```

Definition at line 87 of file CDD_Uart.h.

The documentation for this struct was generated from the following file:

- [CDD_Uart.h](#)

4.67 Wdg_ConfigType Struct Reference

Defines the configuration structure for WDG Driver.

```
#include <Wdg_GeneralTypes.h>
```

Public Attributes

- [WdgIf_ModeType](#) DefaultMode
- const Wdg_HalConfigType * [ModeSetting](#) [3]

4.67.1 Detailed Description

Defines the configuration structure for WDG Driver.

Definition at line 101 of file Wdg_GeneralTypes.h.

4.67.2 Member Data Documentation

4.67.2.1 DefaultMode

```
WdgIf\_ModeType Wdg_ConfigType::DefaultMode
```

Default WDG Mode

Definition at line 103 of file Wdg_GeneralTypes.h.

4.67.2.2 ModeSetting

```
const Wdg_HalConfigType* Wdg_ConfigType::ModeSetting[3]
```

Mode settings for WDGIF_OFF_MODE/WDGIF_SLOW_MODE and WDGIF_FAST_MODE

Definition at line 104 of file Wdg_GeneralTypes.h.

The documentation for this struct was generated from the following file:

- [Wdg_GeneralTypes.h](#)

Chapter 5

File Documentation

5.1 Adc.h File Reference

This file provides all mcal adc api.

```
#include "Adc_Ipw_Types.h"
```

Macros

- #define [ADC_INSTANCE](#) ((uint8)0U)
- #define [ADC_E_UNINIT](#) ((uint8)0x0AU)
- #define [ADC_E_BUSY](#) ((uint8)0x0BU)
- #define [ADC_E_IDLE](#) ((uint8)0x0CU)
- #define [ADC_E_ALREADY_INITIALIZED](#) ((uint8)0x0DU)
- #define [ADC_E_PARAM_CONFIG](#) ((uint8)0x0EU)
- #define [ADC_E_PARAM_POINTER](#) ((uint8)0x14U)
- #define [ADC_E_PARAM_GROUP](#) ((uint8)0x15U)
- #define [ADC_E_WRONG_CONV_MODE](#) ((uint8)0x16U)
- #define [ADC_E_WRONG_TRIGG_SRC](#) ((uint8)0x17U)
- #define [ADC_E_NOTIF_CAPABILITY](#) ((uint8)0x18U)
- #define [ADC_E_BUFFER_UNINIT](#) ((uint8)0x19U)
- #define [ADC_E_NOT_DISENGAGED](#) ((uint8)0x1AU)
- #define [ADC_E_POWER_STATE_NOT_SUPPORTED](#) ((uint8)0x1BU)
- #define [ADC_E_TRANSITION_NOT_POSSIBLE](#) ((uint8)0x1CU)
- #define [ADC_E_PERIPHERAL_NOT_PREPARED](#) ((uint8)0x1DU)
- #define [ADC_INIT_ID](#) ((uint8)0x00U)
- #define [ADC_DEINIT_ID](#) ((uint8)0x01U)
- #define [ADC_STARTGROUPCONVERSION_ID](#) ((uint8)0x02U)
- #define [ADC_STOPGROUPCONVERSION_ID](#) ((uint8)0x03U)
- #define [ADC_READGROUP_ID](#) ((uint8)0x04U)
- #define [ADC_ENABLEHARDWARETRIGGER_ID](#) ((uint8)0x05U)
- #define [ADC_DISABLEHARDWARETRIGGER_ID](#) ((uint8)0x06U)
- #define [ADC_ENABLEGROUPNOTIFICATION_ID](#) ((uint8)0x07U)
- #define [ADC_DISABLEGROUPNOTIFICATION_ID](#) ((uint8)0x08U)
- #define [ADC_GETGROUPSTATUS_ID](#) ((uint8)0x09U)
- #define [ADC_GETVERSIONINFO_ID](#) ((uint8)0x0AU)
- #define [ADC_GETSTREAMLASTPOINTER_ID](#) ((uint8)0x0BU)
- #define [ADC_SETUPRESULTBUFFER_ID](#) ((uint8)0x0CU)
- #define [ADC_SETPOWERSTATE_ID](#) ((uint8)0x10U)
- #define [ADC_GETCURRENTPOWERSTATE_ID](#) ((uint8)0x11U)
- #define [ADC_GETTARGETPOWERSTATE_ID](#) ((uint8)0x12U)
- #define [ADC_PREPAREPOWERSTATE_ID](#) ((uint8)0x13U)

Functions

- void [Adc_Init](#) (const Adc_ConfigType *ConfigPtr)
: Initializes the ADC hardware units and driver.
- Std_ReturnType [Adc_SetupResultBuffer](#) (Adc_GroupType Group, Adc_ValueGroupType *DataBufferPtr)
: Initializes ADC driver with the group specific result buffer start address where the conversion results will be stored. The application has to ensure that the application buffer, where DataBufferPtr points to, can hold all the conversion results of the specified group. The initialization with Adc_SetupResultBuffer is required after reset, before a group conversion can be started.
- void [Adc_DeInit](#) (void)
: Returns all ADC HW Units to a State comparable to their power on reset State.
- void [Adc_StartGroupConversion](#) (Adc_GroupType Group)
: Starts the conversion of all channels of the requested ADC Channel group.
- void [Adc_StopGroupConversion](#) (Adc_GroupType Group)
: Stops the conversion of the requested ADC Channel group.
- Std_ReturnType [Adc_ReadGroup](#) (Adc_GroupType Group, Adc_ValueGroupType *DataBufferPtr)
: Reads the group conversion result of the last completed conversion round of the requested group and stores the Channel values starting at the DataBufferPtr address. The group Channel values are stored in ascending Channel number order (in contrast to the storage layout of the result buffer if streaming access is configured).
- void [Adc_EnableHardwareTrigger](#) (Adc_GroupType Group)
: Enables the hardware trigger for the requested ADC Channel group.
- void [Adc_DisableHardwareTrigger](#) (Adc_GroupType Group)
: Disables the hardware trigger for the requested ADC Channel group.
- void [Adc_EnableGroupNotification](#) (Adc_GroupType Group)
: Enables the notification mechanism for the requested ADC Channel group.
- void [Adc_DisableGroupNotification](#) (Adc_GroupType Group)
: Disables the notification mechanism for the requested ADC Channel group.
- Adc_StatusType [Adc_GetGroupStatus](#) (Adc_GroupType Group)
: Returns the conversion Status of the requested ADC Channel group.
- Adc_StreamNumSampleType [Adc_GetStreamLastPointer](#) (Adc_GroupType Group, Adc_ValueGroupType **PtrToSamplePtr)
: Returns the number of valid samples per Channel, stored in the result buffer. Reads a pointer, pointing to a position in the group result buffer. With the pointer position, the results of all group channels of the last completed conversion round can be accessed. With the pointer and the return value, all valid group conversion results can be accessed (the user has to take the layout of the result buffer into account).
- void [Adc_GetVersionInfo](#) (Std_VersionInfoType *versioninfo)
: Returns the version information of this module.

Variables

- const Adc_ConfigType [Adc_GenerateConfigPC](#)

5.1.1 Detailed Description

This file provides all mcal adc api.

5.1.2 Macro Definition Documentation

5.1.2.1 ADC_DEINIT_ID

```
#define ADC_DEINIT_ID ((uint8)0x01U)
```

Definition at line 97 of file Adc.h.

5.1.2.2 ADC_DISABLEGROUPNOTIFICATION_ID

```
#define ADC_DISABLEGROUPNOTIFICATION_ID ((uint8)0x08U)
```

Definition at line 111 of file Adc.h.

5.1.2.3 ADC_DISABLEHARDWARETRIGGER_ID

```
#define ADC_DISABLEHARDWARETRIGGER_ID ((uint8)0x06U)
```

Definition at line 107 of file Adc.h.

5.1.2.4 ADC_E_ALREADY_INITIALIZED

```
#define ADC_E_ALREADY_INITIALIZED ((uint8)0x0DU)
```

Definition at line 68 of file Adc.h.

5.1.2.5 ADC_E_BUFFER_UNINIT

```
#define ADC_E_BUFFER_UNINIT ((uint8)0x19U)
```

Definition at line 82 of file Adc.h.

5.1.2.6 ADC_E_BUSY

```
#define ADC_E_BUSY ((uint8)0x0BU)
```

Definition at line 64 of file Adc.h.

5.1.2.7 ADC_E_IDLE

```
#define ADC_E_IDLE ((uint8)0x0CU)
```

Definition at line 66 of file Adc.h.

5.1.2.8 ADC_E_NOT_DISENGAGED

```
#define ADC_E_NOT_DISENGAGED ((uint8)0x1AU)
```

Definition at line 84 of file Adc.h.

5.1.2.9 ADC_E_NOTIF_CAPABILITY

```
#define ADC_E_NOTIF_CAPABILITY ((uint8)0x18U)
```

Definition at line 80 of file Adc.h.

5.1.2.10 ADC_E_PARAM_CONFIG

```
#define ADC_E_PARAM_CONFIG ((uint8)0x0EU)
```

Definition at line 70 of file Adc.h.

5.1.2.11 ADC_E_PARAM_GROUP

```
#define ADC_E_PARAM_GROUP ((uint8)0x15U)
```

Definition at line 74 of file Adc.h.

5.1.2.12 ADC_E_PARAM_POINTER

```
#define ADC_E_PARAM_POINTER ((uint8)0x14U)
```

Definition at line 72 of file Adc.h.

5.1.2.13 ADC_E_PERIPHERAL_NOT_PREPARED

```
#define ADC_E_PERIPHERAL_NOT_PREPARED ((uint8)0x1DU)
```

Definition at line 90 of file Adc.h.

5.1.2.14 ADC_E_POWER_STATE_NOT_SUPPORTED

```
#define ADC_E_POWER_STATE_NOT_SUPPORTED ((uint8)0x1BU)
```

Definition at line 86 of file Adc.h.

5.1.2.15 ADC_E_TRANSITION_NOT_POSSIBLE

```
#define ADC_E_TRANSITION_NOT_POSSIBLE ((uint8)0x1CU)
```

Definition at line 88 of file Adc.h.

5.1.2.16 ADC_E_UNINIT

```
#define ADC_E_UNINIT ((uint8)0x0AU)
```

Development errors. The following errors shall be detectable by the ADC module depending on its configuration (development / production mode). All error codes

Definition at line 62 of file Adc.h.

5.1.2.17 ADC_E_WRONG_CONV_MODE

```
#define ADC_E_WRONG_CONV_MODE ((uint8)0x16U)
```

Definition at line 76 of file Adc.h.

5.1.2.18 ADC_E_WRONG_TRIGG_SRC

```
#define ADC_E_WRONG_TRIGG_SRC ((uint8)0x17U)
```

Definition at line 78 of file Adc.h.

5.1.2.19 ADC_ENABLEGROUPNOTIFICATION_ID

```
#define ADC_ENABLEGROUPNOTIFICATION_ID ((uint8)0x07U)
```

Definition at line 109 of file Adc.h.

5.1.2.20 ADC_ENABLEHARDWARETRIGGER_ID

```
#define ADC_ENABLEHARDWARETRIGGER_ID ((uint8)0x05U)
```

Definition at line 105 of file Adc.h.

5.1.2.21 ADC_GETCURRENTPOWERSTATE_ID

```
#define ADC_GETCURRENTPOWERSTATE_ID ((uint8)0x11U)
```

Definition at line 123 of file Adc.h.

5.1.2.22 ADC_GETGROUPSTATUS_ID

```
#define ADC_GETGROUPSTATUS_ID ((uint8)0x09U)
```

Definition at line 113 of file Adc.h.

5.1.2.23 ADC_GETSTREAMLASTPOINTER_ID

```
#define ADC_GETSTREAMLASTPOINTER_ID ((uint8)0x0BU)
```

Definition at line 117 of file Adc.h.

5.1.2.24 ADC_GETTARGETPOWERSTATE_ID

```
#define ADC_GETTARGETPOWERSTATE_ID ((uint8)0x12U)
```

Definition at line 125 of file Adc.h.

5.1.2.25 ADC_GETVERSIONINFO_ID

```
#define ADC_GETVERSIONINFO_ID ((uint8)0x0AU)
```

Definition at line 115 of file Adc.h.

5.1.2.26 ADC_INIT_ID

```
#define ADC_INIT_ID ((uint8)0x00U)
```

Definition at line 95 of file Adc.h.

5.1.2.27 ADC_INSTANCE

```
#define ADC_INSTANCE ((uint8)0U)
```

Definition at line 52 of file Adc.h.

5.1.2.28 ADC_PREPAREPOWERSTATE_ID

```
#define ADC_PREPAREPOWERSTATE_ID ((uint8)0x13U)
```

Definition at line 127 of file Adc.h.

5.1.2.29 ADC_READGROUP_ID

```
#define ADC_READGROUP_ID ((uint8)0x04U)
```

Definition at line 103 of file Adc.h.

5.1.2.30 ADC_SETPOWERSTATE_ID

```
#define ADC_SETPOWERSTATE_ID ((uint8)0x10U)
```

Definition at line 121 of file Adc.h.

5.1.2.31 ADC_SETUPRESULTBUFFER_ID

```
#define ADC_SETUPRESULTBUFFER_ID ((uint8)0x0CU)
```

Definition at line 119 of file Adc.h.

5.1.2.32 ADC_STARTGROUPCONVERSION_ID

```
#define ADC_STARTGROUPCONVERSION_ID ((uint8)0x02U)
```

Definition at line 99 of file Adc.h.

5.1.2.33 ADC_STOPGROUPCONVERSION_ID

```
#define ADC_STOPGROUPCONVERSION_ID ((uint8)0x03U)
```

Definition at line 101 of file Adc.h.

5.1.3 Function Documentation

5.1.3.1 Adc_DeInit()

```
void Adc_DeInit (  
    void )
```

: Returns all ADC HW Units to a State comparable to their power on reset State.

Note

Function ID: DES_ADC_API_003

Service ID: 0x01

Parameters

in		
----	--	--

5.1.3.2 Adc_DisableGroupNotification()

```
void Adc_DisableGroupNotification (  
    Adc_GroupType Group )
```

: Disables the notification mechanism for the requested ADC Channel group.

Note

Function ID: DES_ADC_API_010
Service ID: 0x08

Parameters

in		
----	--	--

5.1.3.3 Adc_DisableHardwareTrigger()

```
void Adc_DisableHardwareTrigger (
    Adc_GroupType Group )
```

: Disables the hardware trigger for the requested ADC Channel group.

Note

Function ID: DES_ADC_API_008
Service ID: 0x06

Parameters

in		
----	--	--

5.1.3.4 Adc_EnableGroupNotification()

```
void Adc_EnableGroupNotification (
    Adc_GroupType Group )
```

: Enables the notification mechanism for the requested ADC Channel group.

Note

Function ID: DES_ADC_API_009
Service ID: 0x07

Parameters

in		
----	--	--

5.1.3.5 Adc_EnableHardwareTrigger()

```
void Adc_EnableHardwareTrigger (
    Adc_GroupType Group )
```

: Enables the hardware trigger for the requested ADC Channel group.

Note

Function ID: DES_ADC_API_007
Service ID: 0x05

Parameters

in		
----	--	--

5.1.3.6 Adc_GetGroupStatus()

```
Adc_StatusType Adc_GetGroupStatus (
    Adc_GroupType Group )
```

: Returns the conversion Status of the requested ADC Channel group.

Note

Function ID: DES_ADC_API_011
Service ID: 0x09

Parameters

in		
----	--	--

5.1.3.7 Adc_GetStreamLastPointer()

```
Adc_StreamNumSampleType Adc_GetStreamLastPointer (
    Adc_GroupType Group,
    Adc_ValueGroupType ** PtrToSamplePtr )
```

: Returns the number of valid samples per Channel, stored in the result buffer. Reads a pointer, pointing to a position in the group result buffer. With the pointer position, the results of all group channels of the last completed conversion round can be accessed. With the pointer and the return value, all valid group conversion results can be accessed (the user has to take the layout of the result buffer into account).

Note

Function ID: DES_ADC_API_012
Service ID: 0x0b

Parameters

in		
----	--	--

5.1.3.8 Adc_GetVersionInfo()

```
void Adc_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

: Returns the version information of this module.

Note

Function ID: DES_ADC_API_013
Service ID: 0x0a

Parameters

in		
----	--	--

5.1.3.9 Adc_Init()

```
void Adc_Init (
    const Adc_ConfigType * ConfigPtr )
```

: Initializes the ADC hardware units and driver.

Note

Function ID: DES_ADC_API_001
Service ID: 0x00

Parameters

in		
----	--	--

5.1.3.10 Adc_ReadGroup()

```
Std_ReturnType Adc_ReadGroup (
    Adc_GroupType Group,
    Adc_ValueGroupType * DataBufferPtr )
```

: Reads the group conversion result of the last completed conversion round of the requested group and stores the Channel values starting at the DataBufferPtr address. The group Channel values are stored in ascending Channel number order (in contrast to the storage layout of the result buffer if streaming access is configured).

Note

Function ID: DES_ADC_API_006
Service ID: 0x04

Parameters

in		
----	--	--

5.1.3.11 Adc_SetupResultBuffer()

```
Std_ReturnType Adc_SetupResultBuffer (
    Adc_GroupType Group,
    Adc_ValueGroupType * DataBufferPtr )
```

: Initializes ADC driver with the group specific result buffer start address where the conversion results will be stored. The application has to ensure that the application buffer, where DataBufferPtr points to, can hold all the conversion results of the specified group. The initialization with Adc_SetupResultBuffer is required after reset, before a group conversion can be started.

Note

Function ID: DES_ADC_API_002
Service ID: 0x0c

Parameters

in		
----	--	--

5.1.3.12 Adc_StartGroupConversion()

```
void Adc_StartGroupConversion (
    Adc_GroupType Group )
```

: Starts the conversion of all channels of the requested ADC Channel group.

Note

Function ID: DES_ADC_API_004
Service ID: 0x02

Parameters

in		
----	--	--

5.1.3.13 Adc_StopGroupConversion()

```
void Adc_StopGroupConversion (
    Adc_GroupType Group )
```

: Stops the conversion of the requested ADC Channel group.

Note

Function ID: DES_ADC_API_005
 Service ID: 0x03

Parameters

in		
----	--	--

5.1.4 Variable Documentation**5.1.4.1 Adc_GenerateConfigPC**

```
const Adc_ConfigType Adc_GenerateConfigPC
```

5.2 Can.h File Reference

This file provides extern Mcal Can api.

```
#include "Det.h"
#include "Can_Cfg.h"
```

Classes

- struct [Can_BitrateConfigType](#)
CAN Bitrate config satus.
- struct [Can_ControllerFdConfigType](#)
CANFD controller information Define.
- struct [Can_ControllerBaudrateConfigType](#)
- struct [Can_ControllerDescriptorType](#)
This is used to Define Controller information.
- struct [Can_FilterControlType](#)
- struct [Can_HardwareObjectType](#)
CAN Hardware object Define.
- struct [Can_ConfigType](#)

Macros

- #define [CAN_INSTANCE_ID](#) ((uint8)0x00U)
- #define [CAN_E_DATA_LOST](#) ((uint8)0x01U)
- #define [CAN_E_ECC_ERROR](#) ((uint8)0x02U)
- #define [CAN_E_PARAM_POINTER](#) ((uint8)0x01U)
- #define [CAN_E_PARAM_HANDLE](#) ((uint8)0x02U)
- #define [CAN_E_PARAM_DATA_LENGTH](#) ((uint8)0x03U)
- #define [CAN_E_PARAM_CONTROLLER](#) ((uint8)0x04U)
- #define [CAN_E_UNINIT](#) ((uint8)0x05U)
- #define [CAN_E_TRANSITION](#) ((uint8)0x06U)
- #define [CAN_E_PARAM_BAUDRATE](#) ((uint8)0x07U)
- #define [CAN_E_ICOM_CONFIG_INVALID](#) ((uint8)0x08U)
- #define [CAN_E_INIT_FAILED](#) ((uint8)0x09U)
- #define [CAN_E_INVALID_CONTROLLER](#) ((uint8)0x0AU)
- #define [CAN_E_INVALID_CONTROLLER_MODE](#) ((uint8)0x0BU)
- #define [CAN_SID_INIT](#) ((uint8)0x00U)
- #define [CAN_SID_MAIN_FUNCTION_WRITE](#) ((uint8)0x01U)
- #define [CAN_SID_SET_CONTROLLER_MODE](#) ((uint8)0x03U)
- #define [CAN_SID_DISABLE_CONTROLLER_INTERRUPTS](#) ((uint8)0x04U)
- #define [CAN_SID_ENABLE_CONTROLLER_INTERRUPTS](#) ((uint8)0x05U)
- #define [CAN_SID_WRITE](#) ((uint8)0x06U)
- #define [CAN_SID_GET_VERSION_INFO](#) ((uint8)0x07U)
- #define [CAN_SID_MAIN_FUNCTION_READ](#) ((uint8)0x08U)
- #define [CAN_SID_MAIN_FUNCTION_BUS_OFF](#) ((uint8)0x09U)
- #define [CAN_SID_MAIN_FUNCTION_WAKEUP](#) ((uint8)0x0AU)
- #define [CAN_SID_Cbk_CHECK_WAKEUP](#) ((uint8)0x0BU)
- #define [CAN_SID_MAIN_FUNCTION_MODE](#) ((uint8)0x0CU)
- #define [CAN_SID_CHANGE_BAUDRATE](#) ((uint8)0x0DU)
- #define [CAN_SID_CHECK_BAUDRATE](#) ((uint8)0x0EU)
- #define [CAN_SID_SET_BAUDRATE](#) ((uint8)0x0FU)
- #define [CAN_SID_DEINIT](#) ((uint8)0x10U)
- #define [CAN_SID_GETCONTROLLERERRORSTATE](#) ((uint8)0x11U)
- #define [CAN_SID_GETCONTROLLERMODE](#) ((uint8)0x12U)
- #define [CAN_SID_GetControllerRxErrorCounter](#) ((uint8)0x30U)
- #define [CAN_SID_GetControllerTxErrorCounter](#) ((uint8)0x31U)

Enumerations

- enum [Can_PduIdType](#) { [CAN_STANDARD](#) = (uint8)0x00U, [CAN_EXTENDED](#), [CAN_MIXED](#) }
CAN ID Type.
- enum [Can_HandleType](#) { [CAN_BASIC](#) = (uint8)0x00U, [CAN_FULL](#) }
CAN Handle Type.
- enum [Can_ModeType](#) { [CAN_TRANSMIT](#) = (uint8)0x00U, [CAN_RECEIVE](#) }
CAN Mode Type.
- enum [Can_OverflowModeType](#) { [CAN_RECV_OVER_WRITE](#) = (uint8)0x00U, [CAN_RECV_DISCARD](#) }
CAN Recive TYPE.
- enum [Can_StatusType](#) { [CAN_UNINIT](#) = (uint8)0x00U, [CAN_READY](#) }
CAN Bus Status TYPE.

Functions

- void [Can_GetVersionInfo](#) (Std_VersionInfoType *versioninfo)
This function returns the version information of this module.
- void [Can_Init](#) (const [Can_ConfigType](#) *ConfigPtr)
This function initializes the module.
- void [Can_DeInit](#) (void)
This function de-initializes the module.
- Std_ReturnType [Can_GetControllerMode](#) (uint8 Controller, [Can_ControllerStateType](#) *ControllerModePtr)
Reports about the current status of the requested CAN controller.
- Std_ReturnType [Can_GetControllerErrorState](#) (uint8 ControllerId, [Can_ErrorStateType](#) *ErrorStatePtr)
This service obtains the error state of the CAN controller.
- Std_ReturnType [Can_SetControllerMode](#) (uint8 Controller, [Can_ControllerStateType](#) Transition)
This service obtains the error state of the CAN controller.
- void [Can_DisableControllerInterrupts](#) (uint8 Controller)
This function disables all interrupts for this CAN controller.
- void [Can_EnableControllerInterrupts](#) (uint8 Controller)
This function enables all allowed interrupts.
- Std_ReturnType [Can_Write](#) ([Can_HwHandleType](#) Hth, const [Can_PduType](#) *PduInfo)
This function is called by CanIf to pass a CAN message to CanDrv for transmission.
- void [Can_MainFunction_Write](#) (void)
This function performs the polling of TX confirmation when CAN_TX_PROCESSING is set to POLLING.
- void [Can_MainFunction_Read](#) (void)
This function performs the polling of RX indications when CAN_RX_PROCESSING is set to POLLING.
- void [Can_MainFunction_BusOff](#) (void)
This function performs the polling of bus-off events that are configured statically as 'to be polled'.
- void [Can_MainFunction_Wakeup](#) (void)
This function performs the polling of wake-up events that are configured statically as 'to be polled'.
- void [Can_MainFunction_Mode](#) (void)
This function performs the polling of CAN controller mode transitions.
- Std_ReturnType [Can_CheckWakeup](#) (uint8 Controller)
This function checks if a wakeup has occurred for the given controller.
- Std_ReturnType [Can_SetBaudrate](#) (uint8 Controller, const uint16 BaudRateConfigID)
This service shall set the baud rate configuration of the CAN controller. Depending on necessary baud rate modifications the controller might have to reset.
- Std_ReturnType [Can_GetControllerTxErrorCounter](#) (uint8 ControllerId, uint8 *TxErrorCounterPtr)
Returns the Tx error counter for a CAN controller.
- Std_ReturnType [Can_GetControllerRxErrorCounter](#) (uint8 ControllerId, uint8 *RxErrorCounterPtr)
Returns the Rx error counter for a CAN controller.

Variables

- const [Can_ConfigType](#) [Can_GenerateConfigPB](#)

5.2.1 Detailed Description

This file provides extern Mcal Can api.

5.2.2 Macro Definition Documentation

5.2.2.1 CAN_E_DATALOST

```
#define CAN_E_DATALOST ((uint8)0x01U)
```

Definition at line 56 of file Can.h.

5.2.2.2 CAN_E_ECC_ERROR

```
#define CAN_E_ECC_ERROR ((uint8)0x02U)
```

Definition at line 57 of file Can.h.

5.2.2.3 CAN_E_ICOM_CONFIG_INVALID

```
#define CAN_E_ICOM_CONFIG_INVALID ((uint8)0x08U)
```

Definition at line 65 of file Can.h.

5.2.2.4 CAN_E_INIT_FAILED

```
#define CAN_E_INIT_FAILED ((uint8)0x09U)
```

Definition at line 66 of file Can.h.

5.2.2.5 CAN_E_INVALID_CONTROLLER

```
#define CAN_E_INVALID_CONTROLLER ((uint8)0x0AU)
```

Definition at line 67 of file Can.h.

5.2.2.6 CAN_E_INVALID_CONTROLLER_MODE

```
#define CAN_E_INVALID_CONTROLLER_MODE ((uint8)0x0BU)
```

Definition at line 68 of file Can.h.

5.2.2.7 CAN_E_PARAM_BAUDRATE

```
#define CAN_E_PARAM_BAUDRATE ((uint8)0x07U)
```

Definition at line 64 of file Can.h.

5.2.2.8 CAN_E_PARAM_CONTROLLER

```
#define CAN_E_PARAM_CONTROLLER ((uint8)0x04U)
```

Definition at line 61 of file Can.h.

5.2.2.9 CAN_E_PARAM_DATA_LENGTH

```
#define CAN_E_PARAM_DATA_LENGTH ((uint8)0x03U)
```

Definition at line 60 of file Can.h.

5.2.2.10 CAN_E_PARAM_HANDLE

```
#define CAN_E_PARAM_HANDLE ((uint8)0x02U)
```

Definition at line 59 of file Can.h.

5.2.2.11 CAN_E_PARAM_POINTER

```
#define CAN_E_PARAM_POINTER ((uint8)0x01U)
```

Definition at line 58 of file Can.h.

5.2.2.12 CAN_E_TRANSITION

```
#define CAN_E_TRANSITION ((uint8)0x06U)
```

Definition at line 63 of file Can.h.

5.2.2.13 CAN_E_UNINIT

```
#define CAN_E_UNINIT ((uint8)0x05U)
```

Definition at line 62 of file Can.h.

5.2.2.14 CAN_INSTANCE_ID

```
#define CAN_INSTANCE_ID ((uint8)0x00U)
```

Definition at line 54 of file Can.h.

5.2.2.15 CAN_SID_CBK_CHECK_WAKEUP

```
#define CAN_SID_CBK_CHECK_WAKEUP ((uint8)0x0BU)
```

Definition at line 80 of file Can.h.

5.2.2.16 CAN_SID_CHANGE_BAUDRATE

```
#define CAN_SID_CHANGE_BAUDRATE ((uint8)0x0DU)
```

Definition at line 82 of file Can.h.

5.2.2.17 CAN_SID_CHECK_BAUDRATE

```
#define CAN_SID_CHECK_BAUDRATE ((uint8)0x0EU)
```

Definition at line 83 of file Can.h.

5.2.2.18 CAN_SID_DEINIT

```
#define CAN_SID_DEINIT ((uint8)0x10U)
```

Definition at line 85 of file Can.h.

5.2.2.19 CAN_SID_DISABLE_CONTROLLER_INTERRUPTS

```
#define CAN_SID_DISABLE_CONTROLLER_INTERRUPTS ((uint8)0x04U)
```

Definition at line 73 of file Can.h.

5.2.2.20 CAN_SID_ENABLE_CONTROLLER_INTERRUPTS

```
#define CAN_SID_ENABLE_CONTROLLER_INTERRUPTS ((uint8)0x05U)
```

Definition at line 74 of file Can.h.

5.2.2.21 CAN_SID_GET_VERSION_INFO

```
#define CAN_SID_GET_VERSION_INFO ((uint8)0x07U)
```

Definition at line 76 of file Can.h.

5.2.2.22 CAN_SID_GETCONTROLLERERRORSTATE

```
#define CAN_SID_GETCONTROLLERERRORSTATE ((uint8)0x11U)
```

Definition at line 86 of file Can.h.

5.2.2.23 CAN_SID_GETCONTROLLERMODE

```
#define CAN_SID_GETCONTROLLERMODE ((uint8)0x12U)
```

Definition at line 87 of file Can.h.

5.2.2.24 CAN_SID_GetControllerRxErrorCounter

```
#define CAN_SID_GetControllerRxErrorCounter ((uint8)0x30U)
```

Definition at line 88 of file Can.h.

5.2.2.25 CAN_SID_GetControllerTxErrorCounter

```
#define CAN_SID_GetControllerTxErrorCounter ((uint8)0x31U)
```

Definition at line 89 of file Can.h.

5.2.2.26 CAN_SID_INIT

```
#define CAN_SID_INIT ((uint8)0x00U)
```

Definition at line 70 of file Can.h.

5.2.2.27 CAN_SID_MAIN_FUNCTION_BUS_OFF

```
#define CAN_SID_MAIN_FUNCTION_BUS_OFF ((uint8)0x09U)
```

Definition at line 78 of file Can.h.

5.2.2.28 CAN_SID_MAIN_FUNCTION_MODE

```
#define CAN_SID_MAIN_FUNCTION_MODE ((uint8)0x0CU)
```

Definition at line 81 of file Can.h.

5.2.2.29 CAN_SID_MAIN_FUNCTION_READ

```
#define CAN_SID_MAIN_FUNCTION_READ ((uint8)0x08U)
```

Definition at line 77 of file Can.h.

5.2.2.30 CAN_SID_MAIN_FUNCTION_WAKEUP

```
#define CAN_SID_MAIN_FUNCTION_WAKEUP ((uint8)0x0AU)
```

Definition at line 79 of file Can.h.

5.2.2.31 CAN_SID_MAIN_FUNCTION_WRITE

```
#define CAN_SID_MAIN_FUNCTION_WRITE ((uint8)0x01U)
```

Definition at line 71 of file Can.h.

5.2.2.32 CAN_SID_SET_BAUDRATE

```
#define CAN_SID_SET_BAUDRATE ((uint8)0x0FU)
```

Definition at line 84 of file Can.h.

5.2.2.33 CAN_SID_SET_CONTROLLER_MODE

```
#define CAN_SID_SET_CONTROLLER_MODE ((uint8)0x03U)
```

Definition at line 72 of file Can.h.

5.2.2.34 CAN_SID_WRITE

```
#define CAN_SID_WRITE ((uint8)0x06U)
```

Definition at line 75 of file Can.h.

5.2.3 Enumeration Type Documentation

5.2.3.1 Can_HandleType

```
enum Can_HandleType
```

CAN Handle Type.

Enumerator

CAN_BASIC	
CAN_FULL	

Definition at line 104 of file Can.h.

5.2.3.2 Can_ModeType

enum [Can_ModeType](#)

CAN Mode Type.

Enumerator

CAN_TRANSMIT	
CAN_RECEIVE	

Definition at line 113 of file Can.h.

5.2.3.3 Can_OverflowModeType

enum [Can_OverflowModeType](#)

CAN Recive TYPE.

Enumerator

CAN_RECV_OVER_WRITE	
CAN_RECV_DISCARD	

Definition at line 122 of file Can.h.

5.2.3.4 Can_PduIdType

enum [Can_PduIdType](#)

CAN ID Type.

Enumerator

CAN_STANDARD	
CAN_EXTENDED	
CAN_MIXED	

Definition at line 94 of file Can.h.

5.2.3.5 Can_StatusType

enum [Can_StatusType](#)

CAN Bus Status TYPE.

Enumerator

CAN_UNINIT	
CAN_READY	

Definition at line 131 of file Can.h.

5.2.4 Function Documentation

5.2.4.1 Can_CheckWakeup()

```
Std_ReturnType Can_CheckWakeup (
    uint8 Controller )
```

This function checks if a wakeup has occurred for the given controller.

Note

Function ID:[DES_CAN_API_011]

Parameters

in	<i>controller</i>	Controller to be checked for a wakeup.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

5.2.4.2 Can_DeInit()

```
void Can_DeInit (
    void )
```

This function de-initializes the module.

Note

Function ID:[DES_CAN_API_002]

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.2.4.3 Can_DisableControllerInterrupts()

```
void Can_DisableControllerInterrupts (
    uint8 Controller )
```

This function disables all interrupts for this CAN controller.

Note

Function ID:[DES_CAN_API_008]

Parameters

in	<i>Controller</i>	CAN controller ID.
out	<i>None</i>	

Returns

None

5.2.4.4 Can_EnableControllerInterrupts()

```
void Can_EnableControllerInterrupts (
    uint8 Controller )
```

This function enables all allowed interrupts.

Note

Function ID:[DES_CAN_API_009]

Parameters

in	<i>Controller</i>	CAN controller ID.
out	<i>None</i>	

Returns

None

5.2.4.5 Can_GetControllerErrorState()

```
Std_ReturnType Can_GetControllerErrorState (
    uint8 ControllerId,
    Can_ErrorStateType * ErrorStatePtr )
```

This service obtains the error state of the CAN controller.

Note

Function ID:[DES_CAN_API_005]

Parameters

in	<i>Controller</i>	CAN controller ID.
in, out	<i>None</i>	
out	<i>ErrorStatePtr</i>	Pointer to a memory location.

Returns

Std_ReturnType

5.2.4.6 Can_GetControllerMode()

```
Std_ReturnType Can_GetControllerMode (
    uint8 Controller,
    Can_ControllerStateType * ControllerModePtr )
```

Reports about the current status of the requested CAN controller.

Note

Function ID:[DES_CAN_API_004]

Parameters

in	<i>Controller</i>	CAN controller for which the status shall be requested.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

5.2.4.7 Can_GetControllerRxErrorCounter()

```
Std_ReturnType Can_GetControllerRxErrorCounter (
    uint8 ControllerId,
    uint8 * RxErrorCounterPtr )
```

Returns the Rx error counter for a CAN controller.

Note

Function ID:[DES_CAN_API_018]

Parameters

in	<i>controller</i>	Controller to be checked for a wakeup.
in	<i>BaudRateConfigID</i>	references a baud rate configuration by ID.
out	<i>RxErrorCounterPtr</i>	Pointer to a memory location, where the current Rx error counter of the CAN controller will be stored.

Returns

Std_ReturnType

5.2.4.8 Can_GetControllerTxErrorCounter()

```
Std_ReturnType Can_GetControllerTxErrorCounter (
    uint8 ControllerId,
    uint8 * TxErrorCounterPtr )
```

Returns the Tx error counter for a CAN controller.

Note

Function ID:[DES_CAN_API_017]

Parameters

in	<i>controller</i>	Controller to be checked for a wakeup.
in	<i>BaudRateConfigID</i>	references a baud rate configuration by ID.
out	<i>TxErrorCounterPtr</i>	Pointer to a memory location, where the current Tx error counter of the CAN controller will be stored.

Returns

Std_ReturnType

5.2.4.9 Can_GetVersionInfo()

```
void Can_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

This function returns the version information of this module.

Note

Function ID:[DES_CAN_API_003]

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>versioninfo</i>	Pointer to location to store version info

Returns

None

5.2.4.10 Can_Init()

```
void Can_Init (
    const Can_ConfigType * ConfigPtr )
```

This function initializes the module.

Note

Function ID:[DES_CAN_API_001]

Parameters

in	<i>ConfigPtr</i>	Pointer to driver configuration.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.2.4.11 Can_MainFunction_BusOff()

```
void Can_MainFunction_BusOff (
    void )
```

This function performs the polling of bus-off events that are configured statically as 'to be polled'.

Note

Function ID:[DES_CAN_API_014]

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.2.4.12 Can_MainFunction_Mode()

```
void Can_MainFunction_Mode (  
    void )
```

This function performs the polling of CAN controller mode transitions.

Note

Function ID:[DES_CAN_API_016]

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.2.4.13 Can_MainFunction_Read()

```
void Can_MainFunction_Read (  
    void )
```

This function performs the polling of RX indications when CAN_RX_PROCESSING is set to POLLING.

Note

Function ID:[DES_CAN_API_013]

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.2.4.14 Can_MainFunction_Wakeup()

```
void Can_MainFunction_Wakeup (  
    void )
```

This function performs the polling of wake-up events that are configured statically as 'to be polled'.

Note

Function ID:[DES_CAN_API_027]

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.2.4.15 Can_MainFunction_Write()

```
void Can_MainFunction_Write (  
    void )
```

This function performs the polling of TX confirmation when CAN_TX_PROCESSING is set to POLLING.

Note

Function ID:[DES_CAN_API_012]

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.2.4.16 Can_SetBaudrate()

```
Std_ReturnType Can_SetBaudrate (  
    uint8 Controller,  
    const uint16 BaudRateConfigID )
```

This service shall set the baud rate configuration of the CAN controller. Depending on necessary baud rate modifications the controller might have to reset.

Note

Function ID:[DES_CAN_API_006]

Parameters

in	<i>controller</i>	Controller to be checked for a wakeup.
in	<i>BaudRateConfigID</i>	references a baud rate configuration by ID.
out	<i>None</i>	

Returns

Std_ReturnType

5.2.4.17 Can_SetControllerMode()

```
Std_ReturnType Can_SetControllerMode (
    uint8 Controller,
    Can_ControllerStateType Transition )
```

This service obtains the error state of the CAN controller.

Note

Function ID:[DES_CAN_API_007]

Parameters

in	<i>Controller</i>	CAN controller ID.
in	<i>Transition</i>	Transition value to request new CAN controller state.
out	<i>ErrorStatePtr</i>	Pointer to a memory location.

Returns

Std_ReturnType

5.2.4.18 Can_Write()

```
Std_ReturnType Can_Write (
    Can_HwHandleType Hth,
    const Can_PduType * PduInfo )
```

This function is called by CanIf to pass a CAN message to CanDrv for transmission.

Note

Function ID:[DES_CAN_API_010]

Parameters

in	<i>Hth</i>	information which HW-transmit handle shall be used for transmit.
out	<i>None</i>	

Returns

Std_ReturnType

5.2.5 Variable Documentation

5.2.5.1 Can_GenerateConfigPB

```
const Can_ConfigType Can_GenerateConfigPB
```

5.3 Can_GeneralTypes.h File Reference

This file provides Types used in CAN module.

```
#include "ComStackTypes.h"
#include "Mcal_Version.h"
```

Classes

- struct [Can_PduType](#)
- struct [Can_HwType](#)

Macros

- #define [CAN_CANCTRL_MAX_PAYLOAD8_U8](#) ((uint8)8U)
- #define [CAN_CANCTRL_MAX_PAYLOAD12_U8](#) ((uint8)12U)
- #define [CAN_CANCTRL_MAX_PAYLOAD16_U8](#) ((uint8)16U)
- #define [CAN_CANCTRL_MAX_PAYLOAD20_U8](#) ((uint8)20U)
- #define [CAN_CANCTRL_MAX_PAYLOAD24_U8](#) ((uint8)24U)
- #define [CAN_CANCTRL_MAX_PAYLOAD32_U8](#) ((uint8)32U)
- #define [CAN_CANCTRL_MAX_PAYLOAD48_U8](#) ((uint8)48U)
- #define [CAN_CANCTRL_MAX_PAYLOAD64_U8](#) ((uint8)64U)
- #define [CAN_EXTEND_ID_DESCRIPTOR](#) ((uint32)0x80000000U)
- #define [CAN_EXTEND_FD_ID_DESCRIPTOR](#) ((uint32)0xC0000000U)
- #define [CAN_FD_ID_DESCRIPTOR](#) ((uint32)0x40000000U)
- #define [CAN_IDE_ID_DESCRIPTOR](#) ((uint32)0x80000000U)
- #define [CAN_ID_STANDARD_MASK_U32](#) ((uint32)0x07FF)
- #define [CAN_ID_EXTENDED_MASK_U32](#) ((uint32)0x1FFFFFFFU)
- #define [CAN_ID_EXTENDEDIDIFF_MASK_U32](#) ((uint32)0x1fffc00)
- #define [CAN_FD_ID_EXTEND_DEF_MASK](#) ((uint32)0x1FFF0000)
- #define [CAN_BUSY](#) (0x02U)

Typedefs

- typedef uint32 [Can_IdType](#)
- typedef uint8 [Can_HwHandleType](#)

Enumerations

- enum [Can_ControllerStateType](#) { [CAN_CS_UNINIT](#) = 0x00U, [CAN_CS_STARTED](#), [CAN_CS_STOPPED](#), [CAN_CS_SLEEP](#) }
- enum [Can_ErrorStateType](#) { [CAN_ERRORSTATE_ACTIVE](#) = 0x00U, [CAN_ERRORSTATE_PASSIVE](#), [CAN_ERRORSTATE_BUSOFF](#) }

5.3.1 Detailed Description

This file provides Types used in CAN module.

5.3.2 Macro Definition Documentation

5.3.2.1 CAN_BUSY

```
#define CAN_BUSY (0x02U)
```

Definition at line 74 of file [Can_GeneralTypes.h](#).

5.3.2.2 CAN_CANCTRL_MAX_PAYLOAD12_U8

```
#define CAN_CANCTRL_MAX_PAYLOAD12_U8 ((uint8)12U)
```

Definition at line 53 of file [Can_GeneralTypes.h](#).

5.3.2.3 CAN_CANCTRL_MAX_PAYLOAD16_U8

```
#define CAN_CANCTRL_MAX_PAYLOAD16_U8 ((uint8)16U)
```

Definition at line 54 of file [Can_GeneralTypes.h](#).

5.3.2.4 CAN_CANCTRL_MAX_PAYLOAD20_U8

```
#define CAN_CANCTRL_MAX_PAYLOAD20_U8 ((uint8)20U)
```

Definition at line 55 of file [Can_GeneralTypes.h](#).

5.3.2.5 CAN_CANCTRL_MAX_PAYLOAD24_U8

```
#define CAN_CANCTRL_MAX_PAYLOAD24_U8 ((uint8)24U)
```

Definition at line 56 of file Can_GeneralTypes.h.

5.3.2.6 CAN_CANCTRL_MAX_PAYLOAD32_U8

```
#define CAN_CANCTRL_MAX_PAYLOAD32_U8 ((uint8)32U)
```

Definition at line 57 of file Can_GeneralTypes.h.

5.3.2.7 CAN_CANCTRL_MAX_PAYLOAD48_U8

```
#define CAN_CANCTRL_MAX_PAYLOAD48_U8 ((uint8)48U)
```

Definition at line 58 of file Can_GeneralTypes.h.

5.3.2.8 CAN_CANCTRL_MAX_PAYLOAD64_U8

```
#define CAN_CANCTRL_MAX_PAYLOAD64_U8 ((uint8)64U)
```

Definition at line 59 of file Can_GeneralTypes.h.

5.3.2.9 CAN_CANCTRL_MAX_PAYLOAD8_U8

```
#define CAN_CANCTRL_MAX_PAYLOAD8_U8 ((uint8)8U)
```

Definition at line 52 of file Can_GeneralTypes.h.

5.3.2.10 CAN_EXTEND_FD_ID_DESCRIPTOR

```
#define CAN_EXTEND_FD_ID_DESCRIPTOR ((uint32)0xC0000000U)
```

Definition at line 67 of file Can_GeneralTypes.h.

5.3.2.11 CAN_EXTEND_ID_DESCRIPTOR

```
#define CAN_EXTEND_ID_DESCRIPTOR ((uint32)0x80000000U)
```

00 CAN message with Standard CAN ID 01 CAN FD frame with Standard CAN ID 10 CAN message with Extended CAN ID 11 CAN FD frame with Extended CAN ID

Definition at line 66 of file Can_GeneralTypes.h.

5.3.2.12 CAN_FD_ID_DESCRIPTOR

```
#define CAN_FD_ID_DESCRIPTOR ((uint32)0x40000000U)
```

Definition at line 68 of file Can_GeneralTypes.h.

5.3.2.13 CAN_FD_ID_EXTEND_DEF_MASK

```
#define CAN_FD_ID_EXTEND_DEF_MASK ((uint32)0x1FFF0000)
```

Definition at line 73 of file Can_GeneralTypes.h.

5.3.2.14 CAN_ID_EXTENDED_MASK_U32

```
#define CAN_ID_EXTENDED_MASK_U32 ((uint32)0x1FFFFFFFU)
```

Definition at line 71 of file Can_GeneralTypes.h.

5.3.2.15 CAN_ID_EXTENDEDDIFF_MASK_U32

```
#define CAN_ID_EXTENDEDDIFF_MASK_U32 ((uint32)0x1ffffc00)
```

Definition at line 72 of file Can_GeneralTypes.h.

5.3.2.16 CAN_ID_STANDARD_MASK_U32

```
#define CAN_ID_STANDARD_MASK_U32 ((uint32)0x07FF)
```

Definition at line 70 of file Can_GeneralTypes.h.

5.3.2.17 CAN_IDE_ID_DESCRIPTOR

```
#define CAN_IDE_ID_DESCRIPTOR ((uint32)0x80000000U)
```

Definition at line 69 of file Can_GeneralTypes.h.

5.3.3 Typedef Documentation

5.3.3.1 Can_HwHandleType

```
typedef uint8 Can_HwHandleType
```

Definition at line 105 of file Can_GeneralTypes.h.

5.3.3.2 Can_IdType

```
typedef uint32 Can_IdType
```

Definition at line 94 of file Can_GeneralTypes.h.

5.3.4 Enumeration Type Documentation

5.3.4.1 Can_ControllerStateType

```
enum Can_ControllerStateType
```

Enumerator

CAN_CS_UNINIT	
CAN_CS_STARTED	
CAN_CS_STOPPED	
CAN_CS_SLEEP	

Definition at line 77 of file Can_GeneralTypes.h.

5.3.4.2 Can_ErrorStateType

```
enum Can_ErrorStateType
```

Enumerator

CAN_ERRORSTATE_ACTIVE	
CAN_ERRORSTATE_PASSIVE	
CAN_ERRORSTATE_BUSOFF	

Definition at line 86 of file Can_GeneralTypes.h.

5.4 CDD_Acmp.h File Reference

This file provides Uart function extern.

```
#include "CDD_Acmp_Ipw.h"
#include "CDD_Acmp_Cfg.h"
#include "Det.h"
```

Classes

- struct [Acmp_ConfigType](#)
ACMP module configuration structure.

Macros

- #define [ACMP_INIT_ID](#) (0x00U)
- #define [ACMP_DEINIT_ID](#) (0x01U)
- #define [ACMP_GET_OUTPUTDATA_ID](#) (0x02U)
- #define [ACMP_GET_POLLINGDATA_ID](#) (0x03U)
- #define [ACMP_GET_VERSIONINFO_ID](#) (0x04U)
- #define [ACMP_E_STATE_TRANSITION](#) (0x00U)
- #define [ACMP_E_INVALID_POINTER](#) (0x01U)
- #define [ACMP_E_UNINIT](#) (0x02U)
- #define [ACMP_E_INVALID_CHANNEL](#) (0x03U)
- #define [ACMP_MODULE_INIT](#) (0x00U)
- #define [ACMP_MODULE_UNINIT](#) (0x01U)
- #define [ACMP_INSTANCE_ID](#) 0U

Functions

- void [Acmp_Init](#) (const [Acmp_ConfigType](#) *ConfigPtr)
Initializes the Acmp module.
- void [Acmp_Deinit](#) (void)
Deinitializes the ACMP module.
- uint8 [Acmp_GetOutputData](#) (uint8 Channel)
Get ACMP normal mode output data.
- uint16 [Acmp_GetPollingData](#) (uint8 Channel)
Get polling mode compare data.

5.4.1 Detailed Description

This file provides Uart function extern.

5.4.2 Macro Definition Documentation

5.4.2.1 ACMP_DEINIT_ID

```
#define ACMP_DEINIT_ID (0x01U)
```

Definition at line 63 of file CDD_Acmp.h.

5.4.2.2 ACMP_E_INVALID_CHANNEL

```
#define ACMP_E_INVALID_CHANNEL (0x03U)
```

Definition at line 72 of file CDD_Acmp.h.

5.4.2.3 ACMP_E_INVALID_POINTER

```
#define ACMP_E_INVALID_POINTER (0x01U)
```

Definition at line 70 of file CDD_Acmp.h.

5.4.2.4 ACMP_E_STATE_TRANSITION

```
#define ACMP_E_STATE_TRANSITION (0x00U)
```

Definition at line 69 of file CDD_Acmp.h.

5.4.2.5 ACMP_E_UNINIT

```
#define ACMP_E_UNINIT (0x02U)
```

Definition at line 71 of file CDD_Acmp.h.

5.4.2.6 ACMP_GET_OUTPUTDATA_ID

```
#define ACMP_GET_OUTPUTDATA_ID (0x02U)
```

Definition at line 64 of file CDD_Acmp.h.

5.4.2.7 ACMP_GET_POLLINGDATA_ID

```
#define ACMP_GET_POLLINGDATA_ID (0x03U)
```

Definition at line 65 of file CDD_Acmp.h.

5.4.2.8 ACMP_GET_VERSIONINFO_ID

```
#define ACMP_GET_VERSIONINFO_ID (0x04U)
```

Definition at line 66 of file CDD_Acmp.h.

5.4.2.9 ACMP_INIT_ID

```
#define ACMP_INIT_ID (0x00U)
```

Definition at line 62 of file CDD_Acmp.h.

5.4.2.10 ACMP_INSTANCE_ID

```
#define ACMP_INSTANCE_ID 0U
```

Definition at line 78 of file CDD_Acmp.h.

5.4.2.11 ACMP_MODULE_INIT

```
#define ACMP_MODULE_INIT (0x00U)
```

Definition at line 75 of file CDD_Acmp.h.

5.4.2.12 ACMP_MODULE_UNINIT

```
#define ACMP_MODULE_UNINIT (0x01U)
```

Definition at line 76 of file CDD_Acmp.h.

5.4.3 Function Documentation

5.4.3.1 Acmp_Deinit()

```
void Acmp_Deinit (
    void )
```

Deinitializes the ACMP module.

Note

Function ID: DES_ACMP_API_201
Service ID: None

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.4.3.2 Acmp_GetOutputData()

```
uint8 Acmp_GetOutputData (
    uint8 Channel )
```

Get ACMP normal mode output data.

Note

Function ID: DES_ACMP_API_202
Service ID: None

Parameters

in	<i>Channel</i>	ACMP channel to be addressed.
in, out	<i>None.</i>	
out	<i>None</i>	

Returns

output data.

5.4.3.3 Acmp_GetPollingData()

```
uint16 Acmp_GetPollingData (
    uint8 Channel )
```

Get polling mode compare data.

Note

Function ID: DES_ACMP_API_203
Service ID: None

Parameters

in	<i>Channel</i>	ACMP channel to be addressed.
in, out	<i>None.</i>	
out	<i>None</i>	

Returns

polling mode input channel compare data

5.4.3.4 Acmp_Init()

```
void Acmp_Init (
    const Acmp_ConfigType * ConfigPtr )
```

Initializes the Acmp module.

Note

Function ID: DES_ACMP_API_200
Service ID: None

Parameters

in	<i>ConfigPtr</i>	: Pointer to Acmp_ConfigType
in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.5 CDD_Crc.h File Reference

This file provides extern Crc CDD API.

```
#include "Crc_Hal.h"
#include "CDD_Crc_Cfg.h"
```

Macros

- #define [CRC_E_PARAM_POINTER](#) ((uint8)0x0FU)
Development errors.
- #define [CRC_E_UNINIT](#) ((uint8)0x01)
Det Error value, returned by a function if API service called prior to module initialization.
- #define [CRC_INIT_ID](#) 0x09U
API service ID for CRC_Init function.
- #define [CRC_GETVERSIONINFO_ID](#) 0x04U
API service ID for CRC_GETVERSIONINFO_ID function.
- #define [CRC_CALCULATECRC_ID](#) 0x0AU
API service ID for Crc_CalculateCRC function.
- #define [CRC_GetCRCRESULT_ID](#) 0x0BU
API service ID for Crc_GetCrcResult function.
- #define [CRC_GETCONFIG_ID](#) 0x0CU
API service ID for CRC_GetConfig function.

Functions

- void [Crc_GetVersionInfo](#) (Std_VersionInfoType *VersionInfo)
Service to get the version information of this module.
- void [Crc_Init](#) (uint8 Instance, const Crc_ConfigType *ConfigPtr)
This function initializes the driver.
- void [Crc_Deinit](#) (void)
This function deinitializes the driver.
- uint32 [Crc_CalculateCRC](#) (uint8 Instance, const uint8 *DataPtr, uint32 Length, const [Crc_DmaConfig](#) *DmaConfig)
Appends a block of bytes to the current CRC calculation.
- uint32 [Crc_GetCrcResult](#) (uint8 Instance)
Gets the current result of the CRC calculation.
- Std_ReturnType [Crc_GetConfig](#) (uint8 Instance, Crc_ConfigType *ConfigPtr)
Gets the configuration structure of the CRC module currently.
- uint8 [Crc_CalculateCRC8](#) (const uint8 *Crc_DataPtr, uint32 Crc_Length, uint8 Crc_StartValue8, boolean Crc_Is↵
FirstCall)
CRC8 caculate function.
- uint8 [Crc_CalculateCRC8H2F](#) (const uint8 *Crc_DataPtr, uint32 Crc_Length, uint8 Crc_StartValue8H2F, boolean
Crc_IsFirstCall)
CRC8H2F caculate function.
- uint16 [Crc_CalculateCRC16](#) (const uint8 *Crc_DataPtr, uint32 Crc_Length, uint16 Crc_StartValue16, boolean Crc↵
_IsFirstCall)
CRC16 caculate function.
- uint16 [Crc_CalculateCRC16ARC](#) (const uint8 *Crc_DataPtr, uint32 Crc_Length, uint16 Crc_StartValue16, boolean
Crc_IsFirstCall)
CRC16ARC caculate function.

- uint32 [Crc_CalculateCRC32](#) (const uint8 *Crc_DataPtr, uint32 Crc_Length, uint32 Crc_StartValue32, boolean Crc_IsFirstCall)
CRC32 caculate function.
- uint32 [Crc_CalculateCRC32P4](#) (const uint8 *Crc_DataPtr, uint32 Crc_Length, uint32 Crc_StartValue32, boolean Crc_IsFirstCall)
CRC32P4 caculate function.
- uint64 [Crc_CalculateCRC64](#) (const uint8 *Crc_DataPtr, uint32 Crc_Length, uint64 Crc_StartValue64, boolean Crc_IsFirstCall)
CRC64 caculate function.

5.5.1 Detailed Description

This file provides extern Crc CDD API.

5.5.2 Macro Definition Documentation

5.5.2.1 CRC_CALCULATECRC_ID

```
#define CRC_CALCULATECRC_ID 0x0AU
```

API service ID for Crc_CalculateCRC function.

Definition at line 73 of file CDD_Crc.h.

5.5.2.2 CRC_E_PARAM_POINTER

```
#define CRC_E_PARAM_POINTER ((uint8)0x0FU)
```

Development errors.

Definition at line 62 of file CDD_Crc.h.

5.5.2.3 CRC_E_UNINIT

```
#define CRC_E_UNINIT ((uint8)0x01)
```

Det Error value, returned by a function if API service called prior to module initialization.

Definition at line 65 of file CDD_Crc.h.

5.5.2.4 CRC_GETCONFIG_ID

```
#define CRC_GETCONFIG_ID 0x0CU
```

API service ID for CRC_GetConfig function.

Definition at line 77 of file CDD_Crc.h.

5.5.2.5 CRC_GetCRCRESULT_ID

```
#define CRC_GetCRCRESULT_ID 0x0BU
```

API service ID for Crc_GetCrcResult function.

Definition at line 75 of file CDD_Crc.h.

5.5.2.6 CRC_GETVERSIONINFO_ID

```
#define CRC_GETVERSIONINFO_ID 0x04U
```

API service ID for CRC_GETVERSIONINFO_ID function.

Definition at line 71 of file CDD_Crc.h.

5.5.2.7 CRC_INIT_ID

```
#define CRC_INIT_ID 0x09U
```

API service ID for CRC_Init function.

Definition at line 69 of file CDD_Crc.h.

5.5.3 Function Documentation

5.5.3.1 Crc_CalculateCRC()

```
uint32 Crc_CalculateCRC (
    uint8 Instance,
    const uint8 * DataPtr,
    uint32 Length,
    const Crc_DmaConfig * DmaConfig )
```

Appends a block of bytes to the current CRC calculation.

Note

Function ID : DES_CRC_API_002
Service ID : 0x0A

Parameters

in	<i>Instance</i>	The CRC Instance number
in	<i>DataPtr</i>	The Pointer to the data array
in	<i>Length</i>	Number of the data array
in	<i>DmaConfig</i>	Dma config information of crc include channel ,enable and so on
out	<i>None</i>	

Returns

uint32 if dma not enable return crc calculate result , if enable ,result mean call dma success or error

5.5.3.2 Crc_CalculateCRC16()

```
uint16 Crc_CalculateCRC16 (
    const uint8 * Crc_DataPtr,
    uint32 Crc_Length,
    uint16 Crc_StartValue16,
    boolean Crc_IsFirstCall )
```

CRC16 caculate function.

Note

Function ID : DES_CRC_API_008
Service ID : 0x02

Parameters

in	<i>Crc_DataPtr:The</i>	pointer to data block.
in	<i>Crc_Length</i>	caculate crc data length
in	<i>Crc_StartValue16</i>	The CRC16 start value.
in	<i>Crc_IsFirstCall</i>	check it is first call or not.
out	<i>None</i>	

Returns

CRC16 result.

5.5.3.3 Crc_CalculateCRC16ARC()

```
uint16 Crc_CalculateCRC16ARC (
    const uint8 * Crc_DataPtr,
    uint32 Crc_Length,
    uint16 Crc_StartValue16,
    boolean Crc_IsFirstCall )
```

CRC16ARC caculate function.

Note

Function ID : DES_CRC_API_009 Service ID : 0x08

Parameters

in	<i>Crc_DataPtr:The</i>	pointer to data block.
in	<i>Crc_Length</i>	caculate crc data length
in	<i>Crc_StartValue16</i>	The CRC16ARC start value.
in	<i>Crc_IsFirstCall</i>	check it is first call or not.
out	<i>None</i>	

Returns

CRC16ARC result.

5.5.3.4 Crc_CalculateCRC32()

```
uint32 Crc_CalculateCRC32 (
    const uint8 * Crc_DataPtr,
    uint32 Crc_Length,
    uint32 Crc_StartValue32,
    boolean Crc_IsFirstCall )
```

CRC32 caculate function.

Note

Function ID : DES_CRC_API_010
Service ID : 0x03

Parameters

in	<i>Crc_DataPtr:The</i>	pointer to data block.
in	<i>Crc_Length</i>	caculate crc data length
in	<i>Crc_StartValue32</i>	The CRC32 start value.
in	<i>Crc_IsFirstCall</i>	check it is first call or not.
out	<i>None</i>	

Returns

CRC32 result.

5.5.3.5 Crc_CalculateCRC32P4()

```
uint32 Crc_CalculateCRC32P4 (
    const uint8 * Crc_DataPtr,
```

```
uint32 Crc_Length,
uint32 Crc_StartValue32,
boolean Crc_IsFirstCall )
```

CRC32P4 caculate function.

Note

Function ID : DES_CRC_API_011
Service ID : 0x06

Parameters

in	<i>Crc_DataPtr:The</i>	pointer to data block.
in	<i>Crc_Length</i>	caculate crc data length
in	<i>Crc_StartValue32</i>	The CRC32P4 start value.
in	<i>Crc_IsFirstCall</i>	check it is first call or not.
out	<i>None</i>	

Returns

CRC32P4 result.

5.5.3.6 Crc_CalculateCRC64()

```
uint64 Crc_CalculateCRC64 (
    const uint8 * Crc_DataPtr,
    uint32 Crc_Length,
    uint64 Crc_StartValue64,
    boolean Crc_IsFirstCall )
```

CRC64 caculate function.

Note

Function ID : DES_CRC_API_012
Service ID : 0x07

Parameters

in	<i>Crc_DataPtr:The</i>	pointer to data block.
in	<i>Crc_Length</i>	caculate crc data length
in	<i>Crc_StartValue64</i>	The CRC64 start value.
in	<i>Crc_IsFirstCall</i>	check it is first call or not.
out	<i>None</i>	

Returns

CRC64 result.

5.5.3.7 Crc_CalculateCRC8()

```
uint8 Crc_CalculateCRC8 (
    const uint8 * Crc_DataPtr,
    uint32 Crc_Length,
    uint8 Crc_StartValue8,
    boolean Crc_IsFirstCall )
```

CRC8 caculate function.

Note

Function ID : DES_CRC_API_006
Service ID : 0x01

Parameters

in	<i>Crc_DataPtr:The</i>	pointer to data block.
in	<i>Crc_Length</i>	caculate crc data length
in	<i>Crc_StartValue8</i>	The CRC8 start value
in	<i>Crc_IsFirstCall</i>	check it is first call or not.
out	<i>None</i>	

Returns

Crc8 result.

5.5.3.8 Crc_CalculateCRC8H2F()

```
uint8 Crc_CalculateCRC8H2F (
    const uint8 * Crc_DataPtr,
    uint32 Crc_Length,
    uint8 Crc_StartValue8H2F,
    boolean Crc_IsFirstCall )
```

CRC8H2F caculate function.

Note

Function ID : DES_CRC_API_007
Service ID : 0x05

Parameters

in	<i>Crc_DataPtr:The</i>	pointer to data block.
in	<i>Crc_Length</i>	caculate crc data length
in	<i>Crc_StartValue8H2F</i>	The CRC8H2F start value
in	<i>Crc_IsFirstCall</i>	check it is first call or not.
out	<i>None</i>	

Returns

Crc8H2F result.

5.5.3.9 Crc_Deinit()

```
void Crc_Deinit (
    void )
```

This function deinitializes the driver.

Note

Function ID : DES_CRC_API_013
Service ID : 0x0D

Parameters

in	<i>None</i>	
in	<i>None</i>	

Returns

None

5.5.3.10 Crc_GetConfig()

```
Std_ReturnType Crc_GetConfig (
    uint8 Instance,
    Crc_ConfigType * ConfigPtr )
```

Gets the configuration structure of the CRC module currently.

Note

Function ID : DES_CRC_API_005
Service ID : 0x0C

Parameters

in	<i>Instance</i>	The CRC Instance number
out	<i>ConfigPtr</i>	Pointer to structure of CRC configuration

Returns

The result of execution

- E_OK: Operation was successful

- E_NOT_OK: Operation was not successful

5.5.3.11 Crc_GetCrcResult()

```
uint32 Crc_GetCrcResult (
    uint8 Instance )
```

Gets the current result of the CRC calculation.

Note

Function ID : DES_CRC_API_003
Service ID : 0x0B

Parameters

in	<i>Instance</i>	The CRC Instance number
out	<i>None</i>	

Returns

Result of CRC calculation

5.5.3.12 Crc_GetVersionInfo()

```
void Crc_GetVersionInfo (
    Std_VersionInfoType * VersionInfo )
```

Service to get the version information of this module.

Note

Function ID : DES_CRC_API_004
Service ID : 0x04

Parameters

out	<i>VersionInfo</i>	Pointer to where to store the version information of this module.
-----	--------------------	-------------------------------------------------------------------

Returns

None

5.5.3.13 Crc_Init()

```
void Crc_Init (
    uint8 Instance,
    const Crc_ConfigType * ConfigPtr )
```

This function initializes the driver.

Note

Function ID : DES_CRC_API_001
Service ID : 0x09

Parameters

in	<i>Instance</i>	CRC Hardware Device instance
in	<i>Crc_ConfigPtr</i>	: Pointer to a selected configuration structure.

Returns

None

5.6 CDD_Crc_Types.h File Reference

This file provides extern Crc CDD API.

```
#include "Mcal.h"
```

Classes

- struct [Crc_DmaConfig](#)
CRC Dma configuration structure.

Macros

- #define [CRC_E_PARAM_POINTER](#) ((uint8)0x0FU)
Development errors.
- #define [CRC_E_UNINIT](#) ((uint8)0x01)
Det Error value, returned by a function if API service called prior to module initialization.
- #define [CRC_INIT_ID](#) 0x09U
API service ID for CRC_Init function.
- #define [CRC_GETVERSIONINFO_ID](#) 0x04U
API service ID for CRC_GETVERSIONINFO_ID function.
- #define [CRC_CALCULATECRC_ID](#) 0x0AU
API service ID for Crc_CalculateCRC function.
- #define [CRC_GetCRCRESULT_ID](#) 0x0BU
API service ID for Crc_GetCrcResult function.
- #define [CRC_GETCONFIG_ID](#) 0x0CU
API service ID for CRC_GetConfig function.
- #define [CRC_CRC16_POLY](#) (0x1021U)
- #define [CRC_CRC16ARC_POLY](#) (0x8005U)
- #define [CRC_CRC32_POLY](#) (0x04C11DB7U)
- #define [CRC_CRC32P4_POLY](#) (0xF4ACFB13U)

5.6.1 Detailed Description

This file provides extern Crc CDD API.

5.6.2 Macro Definition Documentation

5.6.2.1 CRC_CALCULATECRC_ID

```
#define CRC_CALCULATECRC_ID 0x0AU
```

API service ID for Crc_CalculateCRC function.

Definition at line 72 of file CDD_Crc_Types.h.

5.6.2.2 CRC_CRC16_POLY

```
#define CRC_CRC16_POLY (0x1021U)
```

Definition at line 78 of file CDD_Crc_Types.h.

5.6.2.3 CRC_CRC16ARC_POLY

```
#define CRC_CRC16ARC_POLY (0x8005U)
```

Definition at line 79 of file CDD_Crc_Types.h.

5.6.2.4 CRC_CRC32_POLY

```
#define CRC_CRC32_POLY (0x04C11DB7U)
```

Definition at line 81 of file CDD_Crc_Types.h.

5.6.2.5 CRC_CRC32P4_POLY

```
#define CRC_CRC32P4_POLY (0xF4ACFB13U)
```

Definition at line 82 of file CDD_Crc_Types.h.

5.6.2.6 CRC_E_PARAM_POINTER

```
#define CRC_E_PARAM_POINTER ((uint8)0x0FU)
```

Development errors.

Definition at line 61 of file CDD_Crc_Types.h.

5.6.2.7 CRC_E_UNINIT

```
#define CRC_E_UNINIT ((uint8)0x01)
```

Det Error value, returned by a function if API service called prior to module initialization.

Definition at line 64 of file CDD_Crc_Types.h.

5.6.2.8 CRC_GETCONFIG_ID

```
#define CRC_GETCONFIG_ID 0x0CU
```

API service ID for CRC_GetConfig function.

Definition at line 76 of file CDD_Crc_Types.h.

5.6.2.9 CRC_GetCRCRESULT_ID

```
#define CRC_GetCRCRESULT_ID 0x0BU
```

API service ID for Crc_GetCrcResult function.

Definition at line 74 of file CDD_Crc_Types.h.

5.6.2.10 CRC_GETVERSIONINFO_ID

```
#define CRC_GETVERSIONINFO_ID 0x04U
```

API service ID for CRC_GETVERSIONINFO_ID function.

Definition at line 70 of file CDD_Crc_Types.h.

5.6.2.11 CRC_INIT_ID

```
#define CRC_INIT_ID 0x09U
```

API service ID for CRC_Init function.

Definition at line 68 of file CDD_Crc_Types.h.

5.7 CDD_I2c.h File Reference

This file include cdd I2c function.

```
#include "CDD_I2c_Ipw.h"  
#include "CDD_I2c_Cfg.h"
```

Classes

- struct [I2c_CHConfigType](#)
- struct [I2c_ConfigType](#)

Macros

- #define [I2C_INIT_ID](#) (0x00U)
- #define [I2C_DEINIT_ID](#) (0x01U)
- #define [I2C_ASYNCTRANSMIT_ID](#) (0x02U)
- #define [I2C_GETSTATUS_ID](#) (0x03U)
- #define [I2C_GETVERSION_INFO_ID](#) (0x04U)
- #define [I2C_E_ALREADY_INIT](#) (0x00U)
- #define [I2C_E_INVALID_POINTER](#) (0x01U)
- #define [I2C_E_UNINIT](#) (0x02U)
- #define [I2C_E_INVALID_CHANNEL](#) (0x03U)
- #define [I2C_E_BUSY](#) (0x04U)
- #define [I2C_E_STATUES](#) (0x05U)
- #define [I2C_UNINIT_STATE](#) (0x00U)
- #define [I2C_INIT_STATE](#) (0x01U)

Enumerations

- enum [I2c_StatusType](#) {
 [I2C_NOT_OK](#) = 0x0U, [I2C_IDLE](#), [I2C_BUSY_TRANSMIT](#), [I2C_TRANSMIT_FINISHED](#),
 [I2C_ERROR_PRESENT](#), [I2C_ABORTED_SUCCESS](#) }

Functions

- void [I2c_Init](#) (const [I2c_ConfigType](#) *ConfigPtr)
Initializes the I2C module.
- void [I2c_DeInit](#) (void)
DeInitializes the I2C module.
- Std_ReturnType [I2c_AsyncTransmit](#) (uint8 Channel, DataTransmitType *DataTransmitPtr)
Starts an asynchronous transmission on the I2C bus.
- Std_ReturnType [I2c_SyncTransmit](#) (uint8 Channel, const DataTransmitType *DataTransmitPtr)
Starts an synchronous transmission on the I2C bus.
- Std_ReturnType [I2c_AbortTransmit](#) (uint8 Channel)
Stop an asynchronous transmission on the I2C bus.
- [I2c_StatusType](#) [I2c_GetStatus](#) (uint8 Channel)
Gets the status of an I2C channel.
- Std_ReturnType [I2c_SetBaudRate](#) (uint8 Channel, uint32 BaudRate)
Set the currently configured baud rate.
- uint32 [I2c_GetBaudRate](#) (uint8 Channel)
Set the currently configured baud rate.
- void [I2c_GetVersionInfo](#) (Std_VersionInfoType *VersionInfo)
the version information of this module.

Variables

- const [I2c_ConfigType](#) [I2c_GenerateConfigPC](#)

5.7.1 Detailed Description

This file include cdd I2c function.

5.7.2 Macro Definition Documentation

5.7.2.1 I2C_ASYNCTRANSMIT_ID

```
#define I2C_ASYNCTRANSMIT_ID (0x02U)
```

Definition at line 61 of file CDD_I2c.h.

5.7.2.2 I2C_DEINIT_ID

```
#define I2C_DEINIT_ID (0x01U)
```

Definition at line 60 of file CDD_I2c.h.

5.7.2.3 I2C_E_ALREADY_INIT

```
#define I2C_E_ALREADY_INIT (0x00U)
```

Definition at line 66 of file CDD_I2c.h.

5.7.2.4 I2C_E_BUSY

```
#define I2C_E_BUSY (0x04U)
```

Definition at line 70 of file CDD_I2c.h.

5.7.2.5 I2C_E_INVALID_CHANNEL

```
#define I2C_E_INVALID_CHANNEL (0x03U)
```

Definition at line 69 of file CDD_I2c.h.

5.7.2.6 I2C_E_INVALID_POINTER

```
#define I2C_E_INVALID_POINTER (0x01U)
```

Definition at line 67 of file CDD_I2c.h.

5.7.2.7 I2C_E_STATUES

```
#define I2C_E_STATUES (0x05U)
```

Definition at line 71 of file CDD_I2c.h.

5.7.2.8 I2C_E_UNINIT

```
#define I2C_E_UNINIT (0x02U)
```

Definition at line 68 of file CDD_I2c.h.

5.7.2.9 I2C_GETSTATUS_ID

```
#define I2C_GETSTATUS_ID (0x03U)
```

Definition at line 62 of file CDD_I2c.h.

5.7.2.10 I2C_GETVERSION_INFO_ID

```
#define I2C_GETVERSION_INFO_ID (0x04U)
```

Definition at line 63 of file CDD_I2c.h.

5.7.2.11 I2C_INIT_ID

```
#define I2C_INIT_ID (0x00U)
```

Definition at line 59 of file CDD_I2c.h.

5.7.2.12 I2C_INIT_STATE

```
#define I2C_INIT_STATE (0x01U)
```

Definition at line 74 of file CDD_I2c.h.

5.7.2.13 I2C_UNINIT_STATE

```
#define I2C_UNINIT_STATE (0x00U)
```

Definition at line 73 of file CDD_I2c.h.

5.7.3 Enumeration Type Documentation

5.7.3.1 I2c_StatusType

```
enum I2c_StatusType
```

Enumerator

I2C_NOT_OK	
I2C_IDLE	
I2C_BUSY_TRANSMIT	
I2C_TRANSMIT_FINISHED	
I2C_ERROR_PRESENT	
I2C_ABORTED_SUCCESS	

Definition at line 77 of file CDD_I2c.h.

5.7.4 Function Documentation**5.7.4.1 I2c_AbortTransmit()**

```
Std_ReturnType I2c_AbortTransmit (
    uint8 Channel )
```

Stop an asynchronous transmission on the I2C bus.

Note

Function ID:DES_I2C_API_007
Service ID: 0x07

Parameters

in	<i>Channel</i>	I2C channel to be addressed.
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Success to Send or receive E_NOT_OK: Otherwise

5.7.4.2 I2c_AsyncTransmit()

```
Std_ReturnType I2c_AsyncTransmit (
    uint8 Channel,
    DataTransmitType * DataTransmitPtr )
```

Starts an asynchronous transmission on the I2C bus.

Note

Function ID:DES_I2C_API_002
Service ID: 0x02

Parameters

in	<i>Channel</i>	I2C channel to be addressed.
in, out	<i>DataTransmitPtr</i>	transmit data used
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Success to Send or receive E_NOT_OK: Otherwise

5.7.4.3 I2c_DeInit()

```
void I2c_DeInit (
    void )
```

DeInitializes the I2C module.

Note

Function ID:DES_I2C_API_001
Service ID: 0x01

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.7.4.4 I2c_GetBaudRate()

```
uint32 I2c_GetBaudRate (
    uint8 Channel )
```

Set the currently configured baud rate.

Note

Function ID:DES_I2C_API_008
Service ID: 0x08

Parameters

in	<i>Channel</i>	I2C channel to be addressed.
in	<i>BaudRate</i>	: I2C channel Set baudRate.
Generated by Wogen		

Returns

Std_ReturnType: E_OK: Success to Send or receive E_NOT_OK: Otherwise

5.7.4.5 I2c_GetStatus()

```
I2c_StatusType I2c_GetStatus (
    uint8 Channel )
```

Gets the status of an I2C channel.

Note

Function ID:DES_I2C_API_003
Service ID: 0x03

Parameters

in	<i>Channel</i>	I2C channel to be addressed.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

I2c_StatusType: I2c channel current status

5.7.4.6 I2c_GetVersionInfo()

```
void I2c_GetVersionInfo (
    Std_VersionInfoType * VersionInfo )
```

the version information of this module.

Note

Function ID:DES_I2C_API_004
Service ID: 0x04

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>versionInfo</i>	Pointer for storing the version information

Returns

None

5.7.4.7 I2c_Init()

```
void I2c_Init (
    const I2c_ConfigType * ConfigPtr )
```

Initializes the I2C module.

Note

Function ID:DES_I2C_API_000
Service ID: 0x00

Parameters

in	<i>ConfigPtr</i>	: Pointer to I2c_ConfigType
in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.7.4.8 I2c_SetBaudRate()

```
Std_ReturnType I2c_SetBaudRate (
    uint8 Channel,
    uint32 BaudRate )
```

Set the currently configured baud rate.

Note

Function ID:DES_I2C_API_005
Service ID: 0x05

Parameters

in	<i>Channel</i>	I2C channel to be addressed.
in	<i>BaudRate</i>	: I2C channel Set baudRate.
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Success to Send or receive E_NOT_OK: Otherwise

5.7.4.9 I2c_SyncTransmit()

```
Std_ReturnType I2c_SyncTransmit (
    uint8 Channel,
    const DataTransmitType * DataTransmitPtr )
```

Starts an synchronous transmission on the I2C bus.

Note

Function ID:DES_I2C_API_006
Service ID: 0x06

Parameters

in	<i>Channel</i>	I2C channel to be addressed.
in, out	<i>DataTransmitPtr</i>	transmit data used
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Success to Send or receive E_NOT_OK: Otherwise

5.7.5 Variable Documentation

5.7.5.1 I2c_GenerateConfigPC

```
const I2c_ConfigType I2c_GenerateConfigPC
```

5.8 CDD_Mcl.h File Reference

meddleware common layer includ dma which is common module and so on

```
#include "Dma_Hal.h"
#include "Mpu_Hal.h"
#include "Pbr_Hal.h"
#include "CDD_Mcl_Cfg.h"
```

Macros

- #define **MCL_INIT_ID** 0x01U
API service ID for MCL_Init function.
- #define **MCL_DEINIT_ID** 0x02U
API service ID for MCL_DeInit function.
- #define **MCL_GET_VERSION_INFO_ID** 0x03U
API service ID for MCL_GetVersionInfo function.
- #define **MCL_E_PARAM_POINTER** ((uint8)0x0AU)
API service is called using an invalid pointer (e.g. the pointer should not be NULL).
- #define **MCL_INSTANCE_ID** ((uint8)0x00)
Instance ID of the MCL driver.

Functions

- void [Mcl_GetVersionInfo](#) (Std_VersionInfoType *VersionInfo)
get the module version info.
- void [Mcl_Init](#) (void)
linital mcl include dma ...
- void [Mcl_DeInit](#) (void)
Deinitial mcl include dma ...

5.8.1 Detailed Description

meddleware common layer includ dma which is common module and so on

5.8.2 Macro Definition Documentation

5.8.2.1 MCL_DEINIT_ID

```
#define MCL_DEINIT_ID 0x02U
```

API service ID for MCL_DeInit function.

Definition at line 65 of file CDD_Mcl.h.

5.8.2.2 MCL_E_PARAM_POINTER

```
#define MCL_E_PARAM_POINTER ((uint8)0x0AU)
```

API service is called using an invalid pointer (e.g. the pointer should not be NULL).

Definition at line 70 of file CDD_Mcl.h.

5.8.2.3 MCL_GET_VERSION_INFO_ID

```
#define MCL_GET_VERSION_INFO_ID 0x03U
```

API service ID for MCL_GetVersionInfo function.

Definition at line 67 of file CDD_Mcl.h.

5.8.2.4 MCL_INIT_ID

```
#define MCL_INIT_ID 0x01U
```

API service ID for MCL_Init function.

Definition at line 63 of file CDD_Mcl.h.

5.8.2.5 MCL_INSTANCE_ID

```
#define MCL_INSTANCE_ID ((uint8)0x00)
```

Instance ID of the MCL driver.

Definition at line 73 of file CDD_Mcl.h.

5.8.3 Function Documentation

5.8.3.1 Mcl_DeInit()

```
void Mcl_DeInit (  
    void )
```

Deinitial mcl include dma ...

Note

Function ID : DES_MCL_API_003
Service ID : 0x02

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None.

5.8.3.2 Mcl_GetVersionInfo()

```
void Mcl_GetVersionInfo (  
    Std_VersionInfoType * VersionInfo )
```

get the module version info.

Note

Function ID : DES_MCL_API_001
Service ID : 0x03

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>VersionInfo</i>	Pointer to where to store the version information of this module.

Returns

None.

5.8.3.3 Mcl_Init()

```
void Mcl_Init (
    void )
```

initial mcl include dma ...

Note

Function ID : DES_MCL_API_002
Service ID : 0x01

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None.

5.9 CDD_Sent.h File Reference

This file provides Uart function extern.

```
#include "CDD_Sent_Ipw.h"
#include "CDD_Sent_Cfg.h"
#include "Det.h"
```

Classes

- struct [Sent_ConfigType](#)

Macros

- #define [SENT_INIT_ID](#) (0x00U)
- #define [SENT_DEINIT_ID](#) (0x01U)
- #define [SENT_ENABLE_CHANNEL_ID](#) (0x02U)
- #define [SENT_GET_CHANNELSTAUTS_ID](#) (0x03U)
- #define [SENT_INIT_TX_CTRL_ID](#) (0x04U)
- #define [SENT_GET_VERSIONINFO_ID](#) (0x05U)
- #define [SENT_E_STATE_TRANSITION](#) (0x00U)
- #define [SENT_E_INVALID_POINTER](#) (0x01U)
- #define [SENT_E_UNINIT](#) (0x02U)
- #define [SENT_E_INVALID_CHANNEL](#) (0x03U)
- #define [SENT_INSTANCE_ID](#) 0U

Functions

- void [Sent_Init](#) (const [Sent_ConfigType](#) *ConfigPtr)
Initializes the Sent module.
- void [Sent_Deinit](#) (void)
Deinitializes the Sent module.
- Std_ReturnType [Sent_EnableChannel](#) (uint8 Channel, boolean Enable)
Enable the specified SENT channel.
- Sent_ChannelStatusType [Sent_GetChannelStatus](#) (uint8 Channel)
Gets the status of the specified SENT channel.
- void [Sent_InitChannelTxCtrl](#) (uint8 Channel, const [Sent_TransmitCtrlType](#) *TxConfig)
Initializes the SENT transmit control for a specific channel.

5.9.1 Detailed Description

This file provides Uart function extern.

5.9.2 Macro Definition Documentation

5.9.2.1 SENT_DEINIT_ID

```
#define SENT_DEINIT_ID (0x01U)
```

Definition at line 63 of file CDD_Sent.h.

5.9.2.2 SENT_E_INVALID_CHANNEL

```
#define SENT_E_INVALID_CHANNEL (0x03U)
```

Definition at line 73 of file CDD_Sent.h.

5.9.2.3 SENT_E_INVALID_POINTER

```
#define SENT_E_INVALID_POINTER (0x01U)
```

Definition at line 71 of file CDD_Sent.h.

5.9.2.4 SENT_E_STATE_TRANSITION

```
#define SENT_E_STATE_TRANSITION (0x00U)
```

Definition at line 70 of file CDD_Sent.h.

5.9.2.5 SENT_E_UNINIT

```
#define SENT_E_UNINIT (0x02U)
```

Definition at line 72 of file CDD_Sent.h.

5.9.2.6 SENT_ENABLE_CHANNEL_ID

```
#define SENT_ENABLE_CHANNEL_ID (0x02U)
```

Definition at line 64 of file CDD_Sent.h.

5.9.2.7 SENT_GET_CHANNELSTAUTS_ID

```
#define SENT_GET_CHANNELSTAUTS_ID (0x03U)
```

Definition at line 65 of file CDD_Sent.h.

5.9.2.8 SENT_GET_VERSIONINFO_ID

```
#define SENT_GET_VERSIONINFO_ID (0x05U)
```

Definition at line 67 of file CDD_Sent.h.

5.9.2.9 SENT_INIT_ID

```
#define SENT_INIT_ID (0x00U)
```

Definition at line 62 of file CDD_Sent.h.

5.9.2.10 SENT_INIT_TX_CTRL_ID

```
#define SENT_INIT_TX_CTRL_ID (0x04U)
```

Definition at line 66 of file CDD_Sent.h.

5.9.2.11 SENT_INSTANCE_ID

```
#define SENT_INSTANCE_ID 0U
```

Definition at line 75 of file CDD_Sent.h.

5.9.3 Function Documentation

5.9.3.1 Sent_Deinit()

```
void Sent_Deinit (  
    void )
```

Deinitializes the Sent module.

Note

Function ID: DES_SENT_API_201
Service ID: NA

Parameters

in	<i>None</i>	
in,out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.9.3.2 Sent_EnableChannel()

```
Std_ReturnType Sent_EnableChannel (
    uint8 Channel,
    boolean Enable )
```

Enable the specified SENT channel.

Note

Function ID: DES_SENT_API_202
Service ID: NA

Parameters

in	<i>Channel</i>	: SENT channel number to be enabled
in	<i>Enable</i>	: enable(TRUE) or disable(FALSE) the channel
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

- E_OK : Channel successfully enabled.
- E_NOT_OK : Invalid channel number or other failure.

5.9.3.3 Sent_GetChannelStatus()

```
Sent_ChannelStatusType Sent_GetChannelStatus (
    uint8 Channel )
```

Gets the status of the specified SENT channel.

Note

Function ID: DES_SENT_API_203
Service ID: NA

Parameters

in	<i>Channel</i>	: SENT channel number whose status is to be retrieved.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Sent_ChannelStatusType

- SENT_CHANNEL_STOP : The channel is stopped.
- SENT_CHANNEL_INITIALIZED : The channel is configured and enabled.

- SENT_CHANNEL_RUNNING : The channel is running but not synchronized.
- SENT_CHANNEL_SYNCHRONIZED : The channel is fully synchronized and frequency is in range.

5.9.3.4 Sent_Init()

```
void Sent_Init (
    const Sent_ConfigType * ConfigPtr )
```

Initializes the Sent module.

Note

Function ID: DES_SENT_API_200
Service ID: NA

Parameters

in	<i>ConfigPtr</i>	: Pointer to Sent_ConfigType
in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.9.3.5 Sent_InitChannelTxCtrl()

```
void Sent_InitChannelTxCtrl (
    uint8 Channel,
    const Sent_TransmitCtrlType * TxConfig )
```

Initializes the SENT transmit control for a specific channel.

Note

Function ID: DES_SENT_API_204
Service ID: 0x01

Parameters

in	<i>Channel</i>	: SENT channel number to be enabled
in	<i>TxConfig</i>	: Pointer to the SENT transmission control configuration
in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.10 CDD_Uart.h File Reference

This file provides Uart function extern.

```
#include "CDD_Uart_Ipw.h"
#include "CDD_Uart_Cfg.h"
#include "Det.h"
```

Classes

- struct [Uart_ConfigType](#)
UART module configuration structure.

Macros

- #define [UART_INIT_ID](#) (0x00U)
- #define [UART_DEINIT_ID](#) (0x01U)
- #define [UART_ASYNCTRANSMIT_ID](#) (0x02U)
- #define [UART_GET_STATUS_ID](#) (0x03U)
- #define [UART_GET_VERSIONINFO_ID](#) (0x04U)
- #define [UART_ABORTTRANSMIT_ID](#) (0x05U)
- #define [UART_E_STATE_TRANSITION](#) (0x00U)
- #define [UART_E_INVALID_POINTER](#) (0x01U)
- #define [UART_E_BUSY_TRANSMIT](#) (0x02U)
- #define [UART_E_UNINIT](#) (0x04U)
- #define [UART_E_INVALID_CHANNEL](#) (0x05U)
- #define [UART_E_TRANSMIT_LENGTH](#) (0x06U)
- #define [UART_INSTANCE_ID](#) 0U

Enumerations

- enum [Uart_DirType](#) { [UART_SEND](#) = 0x00U, [UART_RECEIVE](#) = 0x01U }
Type of UART transfer direction (write or read)

Functions

- void [Uart_Init](#) (const [Uart_ConfigType](#) *ConfigPtr)
Initializes the Uart module.
- void [Uart_DeInit](#) (void)
Deinitializes the Uart module.
- Std_ReturnType [Uart_AsyncTransmit](#) (uint8 Channel, uint8 *Buffer, uint32 BufferSize, [Uart_DirType](#) TransDirection)
Starts an asynchronous transmission on the UART bus.
- Uart_StatusType [Uart_GetStatus](#) (uint8 Channel, uint32 *BytesRemaining, [Uart_DirType](#) TransDirection)
Gets the status of an UART channel.
- Std_ReturnType [Uart_AbortTransmit](#) (uint8 Channel, [Uart_DirType](#) TransDirection)
Abort an asynchronous transmission on the UART bus.
- void [Uart_GetVersionInfo](#) (Std_VersionInfoType *VersionInfo)
the version information of this module.

Variables

- const [Uart_ConfigType](#) [Uart_GenerateConfigPC](#)

5.10.1 Detailed Description

This file provides Uart function extern.

5.10.2 Macro Definition Documentation

5.10.2.1 UART_ABORTTRANSMIT_ID

```
#define UART_ABORTTRANSMIT_ID (0x05U)
```

Definition at line 67 of file CDD_Uart.h.

5.10.2.2 UART_ASYNCTRANSMIT_ID

```
#define UART_ASYNCTRANSMIT_ID (0x02U)
```

Definition at line 64 of file CDD_Uart.h.

5.10.2.3 UART_DEINIT_ID

```
#define UART_DEINIT_ID (0x01U)
```

Definition at line 63 of file CDD_Uart.h.

5.10.2.4 UART_E_BUSY_TRANSMIT

```
#define UART_E_BUSY_TRANSMIT (0x02U)
```

Definition at line 72 of file CDD_Uart.h.

5.10.2.5 UART_E_INVALID_CHANNEL

```
#define UART_E_INVALID_CHANNEL (0x05U)
```

Definition at line 74 of file CDD_Uart.h.

5.10.2.6 UART_E_INVALID_POINTER

```
#define UART_E_INVALID_POINTER (0x01U)
```

Definition at line 71 of file CDD_Uart.h.

5.10.2.7 UART_E_STATE_TRANSITION

```
#define UART_E_STATE_TRANSITION (0x00U)
```

Definition at line 70 of file CDD_Uart.h.

5.10.2.8 UART_E_TRANSMIT_LENGTH

```
#define UART_E_TRANSMIT_LENGTH (0x06U)
```

Definition at line 75 of file CDD_Uart.h.

5.10.2.9 UART_E_UNINIT

```
#define UART_E_UNINIT (0x04U)
```

Definition at line 73 of file CDD_Uart.h.

5.10.2.10 UART_GET_STATUS_ID

```
#define UART_GET_STATUS_ID (0x03U)
```

Definition at line 65 of file CDD_Uart.h.

5.10.2.11 UART_GET_VERSIONINFO_ID

```
#define UART_GET_VERSIONINFO_ID (0x04U)
```

Definition at line 66 of file CDD_Uart.h.

5.10.2.12 UART_INIT_ID

```
#define UART_INIT_ID (0x00U)
```

Definition at line 62 of file CDD_Uart.h.

5.10.2.13 UART_INSTANCE_ID

```
#define UART_INSTANCE_ID 0U
```

Definition at line 77 of file CDD_Uart.h.

5.10.3 Enumeration Type Documentation

5.10.3.1 Uart_DirType

```
enum Uart_DirType
```

Type of UART transfer direction (write or read)

Enumerator

UART_SEND	
UART_RECEIVE	

Definition at line 94 of file CDD_Uart.h.

5.10.4 Function Documentation

5.10.4.1 Uart_AbortTransmit()

```
Std_ReturnType Uart_AbortTransmit (
    uint8 Channel,
    Uart_DirType TransDirection )
```

Abort an asynchronous transmission on the UART bus.

Note

Function ID:[DES_UART_API_204]
Service ID: 0x02

Parameters

in	<i>Channel</i>	UART channel to be addressed.
in, out	<i>DataBufferPtr</i>	Pointer to transmit data.
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Success to Send or receive E_NOT_OK: Otherwise

5.10.4.2 Uart_AsyncTransmit()

```
Std_ReturnType Uart_AsyncTransmit (
    uint8 Channel,
    uint8 * Buffer,
    uint32 BufferSize,
    Uart_DirType TransDirection )
```

Starts an asynchronous transmission on the UART bus.

Note

Function ID:[DES_UART_API_202]

Service ID: 0x02

Parameters

in	<i>Channel</i>	UART channel to be addressed.
in	<i>BufferSize</i>	Transmit data length.
in	<i>TransDirection</i>	Transimit direction
in, out	<i>Buffer</i>	Pointer to transmit data.
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Success to Send or receive E_NOT_OK: Otherwise

5.10.4.3 Uart_DeInit()

```
void Uart_DeInit (
    void )
```

Deinitializes the Uart module.

Note

Function ID:[DES_UART_API_201]

Service ID: 0x01

Parameters

in	<i>ConfigPtr</i>	: Pointer to Uart_ConfigType
in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.10.4.4 Uart_GetStatus()

```
Uart_StatusType Uart_GetStatus (
    uint8 Channel,
    uint32 * BytesRemaining,
    Uart_DirType TransDirection )
```

Gets the status of an UART channel.

Note

Function ID:[DES_UART_API_203]
Service ID: 0x06

Parameters

in	<i>Channel</i>	UART channel to be addressed.
in	<i>TransDirection</i>	Transmit direction
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Uart_Channel_Status_Type: Uart Channel status

5.10.4.5 Uart_GetVersionInfo()

```
void Uart_GetVersionInfo (
    Std_VersionInfoType * VersionInfo )
```

the version information of this module.

Note

Function ID:[DES_UART_API_205]
Service ID: 0x08

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>versionInfo</i>	Pointer for storing the version information

Returns

void

5.10.4.6 Uart_Init()

```
void Uart_Init (
    const Uart\_ConfigType * ConfigPtr )
```

Initializes the Uart module.

Note

Function ID:[DES_UART_API_200]
Service ID: 0x00

Parameters

in	<i>ConfigPtr</i>	: Pointer to Uart_ConfigType
in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.10.5 Variable Documentation**5.10.5.1 Uart_GenerateConfigPC**

```
const Uart\_ConfigType Uart_GenerateConfigPC
```

5.11 Crypto.h File Reference

Header file of Crypto MCAL driver.

Macros

- #define [CRYPTO_FUNCTION_INIT](#) ((uint8)0x01U)
- #define [CRYPTO_FUNCTION_GETVERSIONINFO](#) ((uint8)0x01U)
- #define [CRYPTO_FUNCTION_PROCESSJOB](#) ((uint8)0x03U)
- #define [CRYPTO_FUNCTION_CANCELJOB](#) ((uint8)0x0EU)
- #define [CRYPTO_FUNCTION_KEYELEMENTSET](#) ((uint8)0x04U)
- #define [CRYPTO_FUNCTION_KEYVALIDSET](#) ((uint8)0x05U)
- #define [CRYPTO_FUNCTION_KEYELEMENTGET](#) ((uint8)0x06U)
- #define [CRYPTO_FUNCTION_KEYELEMENTCOPY](#) ((uint8)0x0FU)
- #define [CRYPTO_FUNCTION_KEYCOPY](#) ((uint8)0x10U)
- #define [CRYPTO_FUNCTION_KEYELEMENTIDSGET](#) ((uint8)0x11U)
- #define [CRYPTO_FUNCTION_RANDOMSEED](#) ((uint8)0x0DU)
- #define [CRYPTO_FUNCTION_KEYGENERATE](#) ((uint8)0x07U)
- #define [CRYPTO_FUNCTION_KEYDERIVE](#) ((uint8)0x08U)
- #define [CRYPTO_FUNCTION_KEYEXCHANGEALCPUBVAL](#) ((uint8)0x09U)
- #define [CRYPTO_FUNCTION_KEYEXCHANGEALCSECRET](#) ((uint8)0x0AU)
- #define [CRYPTO_FUNCTION_CERTIFICATEPARSE](#) ((uint8)0x0BU)
- #define [CRYPTO_FUNCTION_CERTIFICATEVERIFY](#) ((uint8)0x12U)
- #define [CRYPTO_FUNCTION_GETID](#) ((uint8)0x85U)
- #define [CRYPTO_FUNCTION_DEBUGCHAL](#) ((uint8)0x86U)
- #define [CRYPTO_FUNCTION_DEBUGAUTH](#) ((uint8)0x87U)
- #define [CRYPTO_FUNCTION_MPCCOMPRESSION](#) ((uint8)0x88U)
- #define [CRYPTO_FUNCTION_GETSTATUS](#) ((uint8)0x84U)
- #define [CRYPTO_FUNCTION_BOOTFAILURE](#) ((uint8)0x82U)
- #define [CRYPTO_FUNCTION_BOOTOK](#) ((uint8)0x83U)
- #define [CRYPTO_FUNCTION_BOOTDEFINE](#) ((uint8)0x89U)
- #define [CRYPTO_E_UNINIT](#) ((uint8)0x00U)
- #define [CRYPTO_E_INIT_FAILED](#) ((uint8)0x01U)
- #define [CRYPTO_E_PARAM_POINTER](#) ((uint8)0x02U)
- #define [CRYPTO_E_PARAM_HANDLE](#) ((uint8)0x04U)
- #define [CRYPTO_E_PARAM_VALUE](#) ((uint8)0x05U)

Functions

- void [Crypto_Init](#) (const [Crypto_ConfigType](#) *ConfigPtr)
Initializes the crypto driver.
- void [Crypto_GetVersionInfo](#) ([Std_VersionInfoType](#) *versioninfo)
Returns the version information of the crypto driver.
- [Std_ReturnType](#) [Crypto_ProcessJob](#) (uint32 objectId, [Crypto_JobType](#) *job)
Performs the crypto primitive, that is configured in the job parameter.
- [Std_ReturnType](#) [Crypto_CancelJob](#) (uint32 objectId, [Crypto_JobInfoType](#) *job)
This interface removes the provided job from the queue and cancels the processing of the job if possible.
- void [Crypto_MainFunction](#) (void)
If asynchronous job processing is configured and there are job queues, the function is called cyclically to process queued jobs.
- [Std_ReturnType](#) [Crypto_KeyElementSet](#) (uint32 cryptoKeyId, uint32 keyElementId, const uint8 *keyPtr, uint32 key↵Length)
Sets the given key element bytes to the key identified by cryptoKeyId.
- [Std_ReturnType](#) [Crypto_KeySetValid](#) (uint32 cryptoKeyId)
Sets the key state of the key identified by cryptoKeyId to valid.
- [Std_ReturnType](#) [Crypto_KeyElementGet](#) (uint32 cryptoKeyId, uint32 keyElementId, uint8 *resultPtr, uint32 *result↵LengthPtr)
This interface shall be used to get a key element of the key identified by the cryptoKeyId and store the key element in the memory location pointed by the result pointer.
- [Std_ReturnType](#) [Crypto_KeyElementIdsGet](#) (uint32 cryptoKeyId, uint32 *keyElementIdsPtr, uint32 *keyElement↵IdsLengthPtr)
Used to retrieve information which key elements are available in a given key.

5.11.1 Detailed Description

Header file of Crypto MCAL driver.

5.11.2 Macro Definition Documentation

5.11.2.1 CRYPTO_E_INIT_FAILED

```
#define CRYPTO_E_INIT_FAILED ((uint8)0x01U)
```

Definition at line 76 of file Crypto.h.

5.11.2.2 CRYPTO_E_PARAM_HANDLE

```
#define CRYPTO_E_PARAM_HANDLE ((uint8)0x04U)
```

Definition at line 78 of file Crypto.h.

5.11.2.3 CRYPTO_E_PARAM_POINTER

```
#define CRYPTO_E_PARAM_POINTER ((uint8)0x02U)
```

Definition at line 77 of file Crypto.h.

5.11.2.4 CRYPTO_E_PARAM_VALUE

```
#define CRYPTO_E_PARAM_VALUE ((uint8)0x05U)
```

Definition at line 79 of file Crypto.h.

5.11.2.5 CRYPTO_E_UNINIT

```
#define CRYPTO_E_UNINIT ((uint8)0x00U)
```

Definition at line 75 of file Crypto.h.

5.11.2.6 CRYPTO_FUNCTION_BOOTDEFINE

```
#define CRYPTO_FUNCTION_BOOTDEFINE ((uint8)0x89U)
```

Definition at line 72 of file Crypto.h.

5.11.2.7 CRYPTO_FUNCTION_BOOTFAILURE

```
#define CRYPTO_FUNCTION_BOOTFAILURE ((uint8)0x82U)
```

Definition at line 70 of file Crypto.h.

5.11.2.8 CRYPTO_FUNCTION_BOOTOK

```
#define CRYPTO_FUNCTION_BOOTOK ((uint8)0x83U)
```

Definition at line 71 of file Crypto.h.

5.11.2.9 CRYPTO_FUNCTION_CANCELJOB

```
#define CRYPTO_FUNCTION_CANCELJOB ((uint8)0x0EU)
```

Definition at line 49 of file Crypto.h.

5.11.2.10 CRYPTO_FUNCTION_CERTIFICATEPARSE

```
#define CRYPTO_FUNCTION_CERTIFICATEPARSE ((uint8)0x0BU)
```

Definition at line 63 of file Crypto.h.

5.11.2.11 CRYPTO_FUNCTION_CERTIFICATEVERIFY

```
#define CRYPTO_FUNCTION_CERTIFICATEVERIFY ((uint8)0x12U)
```

Definition at line 64 of file Crypto.h.

5.11.2.12 CRYPTO_FUNCTION_DEBUGAUTH

```
#define CRYPTO_FUNCTION_DEBUGAUTH ((uint8)0x87U)
```

Definition at line 67 of file Crypto.h.

5.11.2.13 CRYPTO_FUNCTION_DEBUGCHAL

```
#define CRYPTO_FUNCTION_DEBUGCHAL ((uint8)0x86U)
```

Definition at line 66 of file Crypto.h.

5.11.2.14 CRYPTO_FUNCTION_GETID

```
#define CRYPTO_FUNCTION_GETID ((uint8)0x85U)
```

Definition at line 65 of file Crypto.h.

5.11.2.15 CRYPTO_FUNCTION_GETSTATUS

```
#define CRYPTO_FUNCTION_GETSTATUS ((uint8)0x84U)
```

Definition at line 69 of file Crypto.h.

5.11.2.16 CRYPTO_FUNCTION_GETVERSIONINFO

```
#define CRYPTO_FUNCTION_GETVERSIONINFO ((uint8)0x01U)
```

Definition at line 47 of file Crypto.h.

5.11.2.17 CRYPTO_FUNCTION_INIT

```
#define CRYPTO_FUNCTION_INIT ((uint8)0x01U)
```

Definition at line 46 of file Crypto.h.

5.11.2.18 CRYPTO_FUNCTION_KEYCOPY

```
#define CRYPTO_FUNCTION_KEYCOPY ((uint8)0x10U)
```

Definition at line 56 of file Crypto.h.

5.11.2.19 CRYPTO_FUNCTION_KEYDERIVE

```
#define CRYPTO_FUNCTION_KEYDERIVE ((uint8)0x08U)
```

Definition at line 60 of file Crypto.h.

5.11.2.20 CRYPTO_FUNCTION_KEYELEMENTCOPY

```
#define CRYPTO_FUNCTION_KEYELEMENTCOPY ((uint8)0x0FU)
```

Definition at line 55 of file Crypto.h.

5.11.2.21 CRYPTO_FUNCTION_KEYELEMENTGET

```
#define CRYPTO_FUNCTION_KEYELEMENTGET ((uint8)0x06U)
```

Definition at line 52 of file Crypto.h.

5.11.2.22 CRYPTO_FUNCTION_KEYELEMENTIDSGET

```
#define CRYPTO_FUNCTION_KEYELEMENTIDSGET ((uint8)0x11U)
```

Definition at line 57 of file Crypto.h.

5.11.2.23 CRYPTO_FUNCTION_KEYELEMENTSET

```
#define CRYPTO_FUNCTION_KEYELEMENTSET ((uint8)0x04U)
```

Definition at line 50 of file Crypto.h.

5.11.2.24 CRYPTO_FUNCTION_KEYEXCHANGEALCPUBVAL

```
#define CRYPTO_FUNCTION_KEYEXCHANGEALCPUBVAL ((uint8)0x09U)
```

Definition at line 61 of file Crypto.h.

5.11.2.25 CRYPTO_FUNCTION_KEYEXCHANGEALCSECRET

```
#define CRYPTO_FUNCTION_KEYEXCHANGEALCSECRET ((uint8)0x0AU)
```

Definition at line 62 of file Crypto.h.

5.11.2.26 CRYPTO_FUNCTION_KEYGENERATE

```
#define CRYPTO_FUNCTION_KEYGENERATE ((uint8)0x07U)
```

Definition at line 59 of file Crypto.h.

5.11.2.27 CRYPTO_FUNCTION_KEYVALIDSET

```
#define CRYPTO_FUNCTION_KEYVALIDSET ((uint8)0x05U)
```

Definition at line 51 of file Crypto.h.

5.11.2.28 CRYPTO_FUNCTION_MP_COMPRESSION

```
#define CRYPTO_FUNCTION_MP_COMPRESSION ((uint8)0x88U)
```

Definition at line 68 of file Crypto.h.

5.11.2.29 CRYPTO_FUNCTION_PROCESSJOB

```
#define CRYPTO_FUNCTION_PROCESSJOB ((uint8)0x03U)
```

Definition at line 48 of file Crypto.h.

5.11.2.30 CRYPTO_FUNCTION_RANDOMSEED

```
#define CRYPTO_FUNCTION_RANDOMSEED ((uint8)0x0DU)
```

Definition at line 58 of file Crypto.h.

5.11.3 Function Documentation

5.11.3.1 Crypto_CancelJob()

```
Std_ReturnType Crypto_CancelJob (
    uint32 objectId,
    Crypto\_JobInfoType * job )
```

This interface removes the provided job from the queue and cancels the processing of the job if possible.

Note

Function ID : DES_CRYPT0_API_014

Parameters

in	<i>objectId</i>	: Holds the identifier of the Crypto Driver Object.
in	<i>job</i>	: Pointer to the configuration of the job.

Returns

Std_ReturnType

See also

[Crypto_JobType](#)

5.11.3.2 Crypto_GetVersionInfo()

```
void Crypto_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

Returns the version information of the crypto driver.

Note

Function ID : DES_CRYPT0_API_001

Parameters

in	<i>versioninfo</i>	: Pointer to VersionInfo.
----	--------------------	---------------------------

Returns

none

See also

struct Std_VersionInfoType

5.11.3.3 Crypto_Init()

```
void Crypto_Init (
    const Crypto_ConfigType * ConfigPtr )
```

Initializes the crypto driver.

Note

Function ID : DES_CRYPT0_API_000

Parameters

in	<i>ConfigPtr</i>	: Pointer to configuration set.
----	------------------	---------------------------------

Returns

none

See also

Crypto_ConfigType

5.11.3.4 Crypto_KeyElementGet()

```
Std_ReturnType Crypto_KeyElementGet (
    uint32 cryptoKeyId,
    uint32 keyElementId,
    uint8 * resultPtr,
    uint32 * resultLengthPtr )
```

This interface shall be used to get a key element of the key identified by the cryptoKeyId and store the key element in the memory location pointed by the result pointer.

Note

Function ID : DES_CRYPT0_API_006

Parameters

in	<i>cryptoKeyId</i>	: Holds the identifier of the key whose key element shall be set.
in	<i>keyElementId</i>	: Holds the identifier of the key element which shall be set.
out	<i>resultPtr</i>	: Holds the pointer of the buffer for the returned key element.
in, out	<i>resultLengthPtr</i>	: Holds a pointer to a memory location in which the length information is stored.

Returns

Std_ReturnType. E_OK: Request successful. E_NOT_OK: Request failed. CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy. CRYPTO_E_KEY_NOT_AVAILABLE: Request failed, the requested key element is not available. CRYPTO_E_KEY_READ_FAIL: Request failed because read access was denied. CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result. CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element.

5.11.3.5 Crypto_KeyElementIdsGet()

```
Std_ReturnType Crypto_KeyElementIdsGet (
    uint32 cryptoKeyId,
    uint32 * keyElementIdsPtr,
    uint32 * keyElementIdsLengthPtr )
```

Used to retrieve information which key elements are available in a given key.

Note

Function ID : DES_CRYPT0_API_017

Parameters

in	<i>cryptoKeyId</i>	: Holds the identifier of the key whose key element shall be set.
out	<i>keyElementIdsPtr</i>	: Contains the pointer to the array where the ids of the key elements shall be stored.
in, out	<i>keyElementIdsLengthPtr</i>	: Holds a pointer to the memory location in which the number of key elements in the given key is stored.

Returns

Std_ReturnType. E_OK: Request successful. E_NOT_OK: Request failed. CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy. CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result

5.11.3.6 Crypto_KeyElementSet()

```
Std_ReturnType Crypto_KeyElementSet (
    uint32 cryptoKeyId,
    uint32 keyElementId,
    const uint8 * keyPtr,
    uint32 keyLength )
```

Sets the given key element bytes to the key identified by cryptoKeyId.

Note

Function ID : DES_CRYPT0_API_004

Parameters

in	<i>cryptoKeyId</i>	: Holds the identifier of the key whose key element shall be set.
in	<i>key↔ ElementId</i>	: Holds the identifier of the key element which shall be set.
in	<i>keyPtr</i>	: Holds the pointer to the key data which shall be set as key element.
in	<i>keyLength</i>	: Contains the length of the key element in bytes.

Returns

Std_ReturnType. E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypro Driver Object is busy

5.11.3.7 Crypto_KeySetValid()

```
Std_ReturnType Crypto_KeySetValid (
    uint32 cryptoKeyId )
```

Sets the key state of the key identified by cryptoKeyId to valid.

Note

Function ID : DES_CRYPT0_API_005

Parameters

in	<i>crypto↔ KeyId</i>	: Holds the identifier of the key whose key element shall be set.
----	--------------------------	-------------------------------------------------------------------

Returns

Std_ReturnType. E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypro Driver Object is busy

5.11.3.8 Crypto_MainFunction()

```
void Crypto_MainFunction (
    void )
```

If asynchronous job processing is configured and there are job queues, the function is called cyclically to process queued jobs.

Note

Function ID : DES_CRYPT0_API_012

Returns

none

5.11.3.9 Crypto_ProcessJob()

```
Std_ReturnType Crypto_ProcessJob (
    uint32 objectId,
    Crypto_JobType * job )
```

Performs the crypto primitive, that is configured in the job parameter.

Note

Function ID : DES_CRYPT0_API_003

Parameters

in	<i>object↔ Id</i>	: Identification number of driver object.
in, out	<i>job</i>	: Pointer to job.

Returns

Std_ReturnType

See also

[Crypto_JobType](#)

5.12 Crypto_Types.h File Reference

Header file of definitions Crypto types.

```
#include "Csm_Types.h"
```

Classes

- struct [Crypto_PrimitiveType](#)
This structure defines configuratgion of a primitive.
- struct [CryptoKeyElement](#)
- struct [Crypto_Key](#)
- struct [_Crypto_QueueType](#)
Queue Item type.
- struct [Crypto_ObjectType](#)
Crypto driver object.

Macros

- #define `CRYPTO_M1SIZE_U32` ((uint32)0x10U)
- #define `CRYPTO_M2SIZE_U32` ((uint32)0x20U)
- #define `CRYPTO_M3SIZE_U32` ((uint32)0x10U)
- #define `CRYPTO_M4SIZE_U32` ((uint32)0x20U)
- #define `CRYPTO_M5SIZE_U32` ((uint32)0x10U)
- #define `CRYPTO_SIZE_OUT_U32`
- #define `CRYPTO_HW_NOT_SUPPORTED` ((Std_ReturnType)0x99U)
- #define `CRYPTO_PARAM_LEN_IN` 0x10U
Length must be equal to input length.
- #define `CRYPTO_PARAM_LEN_ALIGN` 0x20U
Length must be 16 alignment.
- #define `CRYPTO_PARAM_LEN_LARGER` 0x40U
Length must be 16 alignment.
- #define `CRYPTO_PARAM_LEN_MASK` 0xF0U
Length config mask.
- #define `CRYPTO_SerViceNum` ((uint8)((uint8)CRYPTO_SRV_KEYSETVALID + 1U))

Typedefs

- typedef struct `_Crypto_QueueType` `Crypto_QueueType`
Queue Item type.

Enumerations

- enum `CryptoKeyElementWriteAccess` { `CRYPTO_WA_DENIED` = 0x01U, `CRYPTO_WA_INTERNAL_COPY`, `CRYPTO_WA_ALLOWED`, `CRYPTO_WA_ENCRYPTED` }
This enumerated type define the reading access rights of the key element.
- enum `CryptoKeyElementReadAccess` { `CRYPTO_RA_DENIED` = 0x01U, `CRYPTO_RA_INTERNAL_COPY`, `CRYPTO_RA_ALLOWED`, `CRYPTO_RA_ENCRYPTED` }
This enumerated type define read access right of Key element.
- enum `CryptoKeyElementFormat` { `CRYPTO_KE_FORMAT_BIN_OCTET` = 0x01U, `CRYPTO_KE_FORMAT_BIN_SHEKEYS`, `CRYPTO_KE_FORMAT_BIN_IDENT_PRIVATEKEY_PKCS8`, `CRYPTO_KE_FORMAT_BIN_IDENT_PUBLICKEY`, `CRYPTO_KE_FORMAT_BIN_RSA_PRIVATEKEY`, `CRYPTO_KE_FORMAT_BIN_RSA_PUBLICKEY`, `CRYPTO_KE_FORMAT_BIN_CERT_X509_V3`, `CRYPTO_KE_FORMAT_BIN_CERT_CVC` }
This enumerated type defines the format for the key element.
- enum `Crypto_DriverStateType` { `CRYPTO_UNINIT` = 0x00U, `CRYPTO_IDLE` }

5.12.1 Detailed Description

Header file of definitions Crypto types.

5.12.2 Macro Definition Documentation

5.12.2.1 CRYPTO_HW_NOT_SUPPORTED

```
#define CRYPTO_HW_NOT_SUPPORTED ((Std_ReturnType)0x99U)
```

Definition at line 65 of file Crypto_Types.h.

5.12.2.2 CRYPTO_M1SIZE_U32

```
#define CRYPTO_M1SIZE_U32 ((uint32)0x10U)
```

Definition at line 56 of file Crypto_Types.h.

5.12.2.3 CRYPTO_M2SIZE_U32

```
#define CRYPTO_M2SIZE_U32 ((uint32)0x20U)
```

Definition at line 57 of file Crypto_Types.h.

5.12.2.4 CRYPTO_M3SIZE_U32

```
#define CRYPTO_M3SIZE_U32 ((uint32)0x10U)
```

Definition at line 58 of file Crypto_Types.h.

5.12.2.5 CRYPTO_M4SIZE_U32

```
#define CRYPTO_M4SIZE_U32 ((uint32)0x20U)
```

Definition at line 59 of file Crypto_Types.h.

5.12.2.6 CRYPTO_M5SIZE_U32

```
#define CRYPTO_M5SIZE_U32 ((uint32)0x10U)
```

Definition at line 60 of file Crypto_Types.h.

5.12.2.7 CRYPTO_PARAM_LEN_ALIGN

```
#define CRYPTO_PARAM_LEN_ALIGN 0x20U
```

Length must be 16 alignment.

Definition at line 67 of file Crypto_Types.h.

5.12.2.8 CRYPTO_PARAM_LEN_IN

```
#define CRYPTO_PARAM_LEN_IN 0x10U
```

Length must be equal to input length.

Definition at line 66 of file Crypto_Types.h.

5.12.2.9 CRYPTO_PARAM_LEN_LARGER

```
#define CRYPTO_PARAM_LEN_LARGER 0x40U
```

Length must be 16 alignment.

Definition at line 68 of file Crypto_Types.h.

5.12.2.10 CRYPTO_PARAM_LEN_MASK

```
#define CRYPTO_PARAM_LEN_MASK 0xF0U
```

Length config mask.

Definition at line 69 of file Crypto_Types.h.

5.12.2.11 CRYPTO_SerViceNum

```
#define CRYPTO_SerViceNum ((uint8)((uint8)CRYPTO_SRV_KEYSETVALID + 1U))
```

Definition at line 71 of file Crypto_Types.h.

5.12.2.12 CRYPTO_SIZE_OUT_U32

```
#define CRYPTO_SIZE_OUT_U32
```

Value:

```
((uint32) (CRYPTO_M1SIZE_U32 + CRYPTO_M2SIZE_U32 + \
          CRYPTO_M4SIZE_U32 + CRYPTO_M5SIZE_U32)) CRYPTO_M3SIZE_U32 +
```

Definition at line 62 of file Crypto_Types.h.

5.12.3 Typedef Documentation

5.12.3.1 Crypto_QueueType

```
typedef struct _Crypto_QueueType Crypto_QueueType
```

Queue Item type.

5.12.4 Enumeration Type Documentation

5.12.4.1 Crypto_DriverStateType

```
enum Crypto_DriverStateType
```

Enumerator

CRYPTO_UNINIT	
CRYPTO_IDLE	

Definition at line 113 of file Crypto_Types.h.

5.12.4.2 CryptoKeyElementFormat

```
enum CryptoKeyElementFormat
```

This enumerated type defines the format for the key element.

Enumerator

CRYPTO_KE_FORMAT_BIN_OCTET	Key provided as octet value in binary form.
CRYPTO_KE_FORMAT_BIN_SHEKEYS	Combined input/output keys for SHE operation(M1+M2+M3) and (M4+M5).
CRYPTO_KE_FORMAT_BIN_IDENT_PRIVATEKEY ↔ PKCS8	Not used.
CRYPTO_KE_FORMAT_BIN_IDENT_PUBLICKEY	Not used.
CRYPTO_KE_FORMAT_BIN_RSA_PRIVATEKEY	Not used.
CRYPTO_KE_FORMAT_BIN_RSA_PUBLICKEY	Not used.
CRYPTO_KE_FORMAT_BIN_CERT_X509_V3	Not used.
CRYPTO_KE_FORMAT_BIN_CERT_CVC	Not used.

Definition at line 101 of file Crypto_Types.h.

5.12.4.3 CryptoKeyElementReadAccess

```
enum CryptoKeyElementReadAccess
```

This enumerated type define read access right of Key element.

Enumerator

CRYPTO_RA_DENIED	Key element cannot be read from outside of the crypto driver.
CRYPTO_RA_INTERNAL_COPY	Key element can be copied to another key element in the same crypto driver.
CRYPTO_RA_ALLOWED	Key element can be read as plaintext.
CRYPTO_RA_ENCRYPTED	Key element can be read as encrypted. E.g. SHE Ram-key export.

Definition at line 90 of file Crypto_Types.h.

5.12.4.4 CryptoKeyElementWriteAccess

```
enum CryptoKeyElementWriteAccess
```

This enumerated type define the reading access rights of the key element.

Enumerator

CRYPTO_WA_DENIED	Key element cannot be written from outside of the crypto driver.
CRYPTO_WA_INTERNAL_COPY	Key element can be filled with another key element in the same crypto driver.
CRYPTO_WA_ALLOWED	Key element can be written as plaintext.
CRYPTO_WA_ENCRYPTED	Key element can be written as encrypted. E.g. SHE Ram-key export.

Definition at line 79 of file Crypto_Types.h.

5.13 Csm_Types.h File Reference

Header file of CSM. It should defined outside Crypto MCAL driver.

```
#include "Mcal.h"
```

Classes

- struct [Crypto_JobInfoType](#)
Structure which contains job information (job ID and job priority).
- struct [Crypto_JobPrimitiveInputOutputType](#)
Structure which contains input and output information depending on the job and the crypto primitive.
- struct [Crypto_AlgorithmInfoType](#)
Structure which determines the exact algorithm. Note, not every algorithm needs to specify all fields. AUTOSAR shall only allow valid combinations.
- struct [Crypto_PrimitiveInfoType](#)
Structure which contains basic information about the crypto primitive.
- struct [Crypto_JobPrimitiveInfoType](#)
Structure which contains further information, which depends on the job and the crypto primitive.
- struct [Crypto_JobRedirectionInfoType](#)
Structure which holds the identifiers of the keys and key elements which shall be used as input and output for a job and a bit structure which indicates which buffers shall be redirected to those key elements.
- struct [Crypto_JobType](#)
Structure which contains Job further information, which depends on the job and the crypto primitive.

Macros

- #define [CRYPTO_CSE_KE_SHE_M1_OFFSET](#) ((uint8)0U)
- #define [CRYPTO_CSE_KE_SHE_M2_OFFSET](#) ((uint8)16U)
- #define [CRYPTO_CSE_KE_SHE_M3_OFFSET](#) ((uint8)48U)
- #define [CRYPTO_CSE_KE_SHE_M4_OFFSET](#) ((uint8)64U)
- #define [CRYPTO_CSE_KE_SHE_M5_OFFSET](#) ((uint8)96U)
- #define [CRYPTO_KE_MAC_KEY](#) ((uint32)1U)
- #define [CRYPTO_KE_MAC_PROOF](#) ((uint32)2U)
- #define [CRYPTO_KE_SIGNATURE_KEY](#) ((uint32)1U)
- #define [CRYPTO_KE_RANDOM_SEED_STATE](#) ((uint32)3U)
- #define [CRYPTO_KE_RANDOM_ALGORITHM](#) ((uint32)4U)
- #define [CRYPTO_KE_CIPHER_KEY](#) ((uint32)1U)
- #define [CRYPTO_KE_CIPHER_IV](#) ((uint32)5U)
- #define [CRYPTO_KE_CIPHER_PROOF](#) ((uint32)6U)
- #define [CRYPTO_KE_CIPHER_2NDKEY](#) ((uint32)7U)
- #define [CRYPTO_KE_KEYEXCHANGE_BASE](#) ((uint32)8U)
- #define [CRYPTO_KE_KEYEXCHANGE_PRIVKEY](#) ((uint32)9U)
- #define [CRYPTO_KE_KEYEXCHANGE_OWNPUKEY](#) ((uint32)10U)
- #define [CRYPTO_KE_KEYEXCHANGE_SHAREDVALUE](#) ((uint32)11U)
- #define [CRYPTO_KE_KEYEXCHANGE_ALGORITHM](#) ((uint32)12U)
- #define [CRYPTO_KE_KEYDERIVATION_PASSWORD](#) ((uint32)1U)
- #define [CRYPTO_KE_KEYDERIVATION_SALT](#) ((uint32)13U)
- #define [CRYPTO_KE_KEYDERIVATION_ITERATIONS](#) ((uint32)14U)
- #define [CRYPTO_KE_KEYDERIVATION_ALGORITHM](#) ((uint32)15U)
- #define [CRYPTO_KE_KEYGENERATE_KEY](#) ((uint32)1U)
- #define [CRYPTO_KE_KEYGENERATE_SEED](#) ((uint32)16U)
- #define [CRYPTO_KE_KEYGENERATE_ALGORITHM](#) ((uint32)17U)

- #define CRYPTO_KE_CERTIFICATE_DATA ((uint32)0U)
- #define CRYPTO_KE_CERTIFICATE_PARSING_FORMAT ((uint32)18U)
- #define CRYPTO_KE_CERTIFICATE_CURRENT_TIME ((uint32)19U)
- #define CRYPTO_KE_CERTIFICATE_VERSION ((uint32)20U)
- #define CRYPTO_KE_CERTIFICATE_SERIALNUMBER ((uint32)21U)
- #define CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM ((uint32)22U)
- #define CRYPTO_KE_CERTIFICATE_ISSUER ((uint32)23U)
- #define CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE ((uint32)24U)
- #define CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER ((uint32)25U)
- #define CRYPTO_KE_CERTIFICATE_SUBJECT ((uint32)26U)
- #define CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY ((uint32)1U)
- #define CRYPTO_KE_CERTIFICATE_EXTENSIONS ((uint32)27U)
- #define CRYPTO_KE_CERTIFICATE_SIGNATURE ((uint32)28U)
- #define CRYPTO_KEY_MATERIAL ((uint32)1U)

Enumerations

- enum Crypto_JobStateType { CRYPTO_JOBSTATE_IDLE = 0x00U, CRYPTO_JOBSTATE_ACTIVE = 0x01U }
Enumeration of the current job state.
- enum Crypto_AlgorithmModeType {
CRYPTO_ALGOMODE_NOT_SET = 0x00U, CRYPTO_ALGOMODE_ECB = 0x01U, CRYPTO_ALGOMODE_CBC
= 0x02U, CRYPTO_ALGOMODE_CFB = 0x03U,
CRYPTO_ALGOMODE_OFB = 0x04U, CRYPTO_ALGOMODE_CTR = 0x05U, CRYPTO_ALGOMODE_GCM =
0x06U, CRYPTO_ALGOMODE_XTS = 0x07U,
CRYPTO_ALGOMODE_RSAES_OAEP = 0x08U, CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5 = 0x09U, CRY←
PTO_ALGOMODE_RSASSA_PSS = 0x0AU, CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5 = 0x0BU,
CRYPTO_ALGOMODE_8ROUNDS = 0x0CU, CRYPTO_ALGOMODE_12ROUNDS = 0x0DU, CRYPTO_ALGO←
MODE_20ROUNDS = 0x0EU, CRYPTO_ALGOMODE_HMAC = 0x0FU,
CRYPTO_ALGOMODE_CMAC = 0x10U, CRYPTO_ALGOMODE_GMAC = 0x11U, CRYPTO_ALGOMODE_CT←
RDRBG = 0x12U, CRYPTO_ALGOMODE_SIPHASH_2_4 = 0x13U,
CRYPTO_ALGOMODE_SIPHASH_4_8 = 0x14U, CRYPTO_ALGOMODE_CCM = 0x15U, CRYPTO_ALGOMOD←
E_CUSTOM = 0xffU }
Enumeration of the algorithm mode.
- enum Crypto_VerifyResultType { CRYPTO_E_VER_OK = 0x00U, CRYPTO_E_VER_NOT_OK = 0x01U }
Enumeration of the result type of verification operations.
- enum Crypto_OperationModeType {
CRYPTO_OPERATIONMODE_START = 0x01U, CRYPTO_OPERATIONMODE_UPDATE = 0x02U, CRYPTO_←
OPERATIONMODE_STREAMSTART = 0x03U, CRYPTO_OPERATIONMODE_FINISH = 0x04U,
CRYPTO_OPERATIONMODE_SINGLECALL = 0x07U }
*Enumeration which operation shall be performed. This enumeration is constructed from a bit mask, where the first bit indicates
"Start", the second "Update" and the third "Finish".*
- enum Crypto_ServiceInfoType {
CRYPTO_HASH = 0x00U, CRYPTO_MACGENERATE = 0x01U, CRYPTO_MACVERIFY = 0x02U, CRYPTO_E←
NCRYPT = 0x03U,
CRYPTO_DECRYPT = 0x04U, CRYPTO_AEADENCRYPT = 0x05U, CRYPTO_AEADDECRYPT = 0x06U, CRY←
PTO_SIGNATUREGENERATE = 0x07U,
CRYPTO_SIGNATUREVERIFY = 0x08U, CRYPTO_RANDOMGENERATE = 0x0BU, CRYPTO_RANDOMSEED =
0x0CU, CRYPTO_KEYGENERATE = 0x0DU,
CRYPTO_KEYDERIVE = 0x0EU, CRYPTO_KEYEXCHANGEALCPUBVAL = 0x0FU, CRYPTO_KEYEXCHAN←
GECALCSECRET = 0x10U, CRYPTO_CERTIFICATEPARSE = 0x11U,
CRYPTO_CERTIFICATEVERIFY = 0x12U, CRYPTO_KEYSETVALID = 0x13U }
Enumeration of the kind of the service.
- enum Crypto_AlgorithmFamilyType {
CRYPTO_ALGOFAM_NOT_SET = 0x00U, CRYPTO_ALGOFAM_SHA1 = 0x01U, CRYPTO_ALGOFAM_SHA2_←
224 = 0x02U, CRYPTO_ALGOFAM_SHA2_256 = 0x03U,
CRYPTO_ALGOFAM_SHA2_384 = 0x04U, CRYPTO_ALGOFAM_SHA2_512 = 0x05U, CRYPTO_ALGOFAM_S←
HA2_512_224 = 0x06U, CRYPTO_ALGOFAM_SHA2_512_256 = 0x07U,

```

CRYPTO_ALGOFAM_SHA3_224 = 0x08U, CRYPTO_ALGOFAM_SHA3_256 = 0x09U, CRYPTO_ALGOFAM_SHA3_384 = 0x0AU, CRYPTO_ALGOFAM_SHA3_512 = 0x0BU,
CRYPTO_ALGOFAM_SHAKE128 = 0x0CU, CRYPTO_ALGOFAM_SHAKE256 = 0x0DU, CRYPTO_ALGOFAM_RIPEMD160 = 0x0EU, CRYPTO_ALGOFAM_BLAKE_1_256 = 0x0FU,
CRYPTO_ALGOFAM_BLAKE_1_512 = 0x10U, CRYPTO_ALGOFAM_BLAKE_2s_256 = 0x11U, CRYPTO_ALGOFAM_BLAKE_2s_512 = 0x12U, CRYPTO_ALGOFAM_3DES = 0x13U,
CRYPTO_ALGOFAM_AES = 0x14U, CRYPTO_ALGOFAM_CHACHA = 0x15U, CRYPTO_ALGOFAM_RSA = 0x16U, CRYPTO_ALGOFAM_ED25519 = 0x17U,
CRYPTO_ALGOFAM_BRAINPOOL = 0x18U, CRYPTO_ALGOFAM_ECCNIST = 0x19U, CRYPTO_ALGOFAM_SECURECOUNTER = 0x1AU, CRYPTO_ALGOFAM_RNG = 0x1BU,
CRYPTO_ALGOFAM_SIPHASH = 0x1CU, CRYPTO_ALGOFAM_ECIES = 0x1DU, CRYPTO_ALGOFAM_CUSTOM = 0xFFU }

```

Enumeration of the algorithm family.

- enum `Crypto_ProcessingType` { `CRYPTO_PROCESSING_ASYNC` = 0x00U, `CRYPTO_PROCESSING_SYNC` = 0x01U }

Enumeration of the processing type.

- enum `Crypto_InputOutputRedirectionConfigType` {
`CRYPTO_REDIRECT_CONFIG_PRIMARY_INPUT` = 0x01, `CRYPTO_REDIRECT_CONFIG_SECONDARY_INPUT` = 0x02, `CRYPTO_REDIRECT_CONFIG_TERTIARY_INPUT` = 0x04, `CRYPTO_REDIRECT_CONFIG_PRIMARY_OUTPUT` = 0x10,
`CRYPTO_REDIRECT_CONFIG_SECONDARY_OUTPUT` = 0x20 }

Defines which of the input/output parameters are re-directed to a key element. The values can be combined to define a bit field.

5.13.1 Detailed Description

Header file of CSM. It should be defined outside Crypto MCAL driver.

5.13.2 Macro Definition Documentation

5.13.2.1 CRYPTO_CSE_KE_SHE_M1_OFFSET

```
#define CRYPTO_CSE_KE_SHE_M1_OFFSET ((uint8)0U)
```

Definition at line 55 of file `Csm_Types.h`.

5.13.2.2 CRYPTO_CSE_KE_SHE_M2_OFFSET

```
#define CRYPTO_CSE_KE_SHE_M2_OFFSET ((uint8)16U)
```

Definition at line 56 of file `Csm_Types.h`.

5.13.2.3 CRYPTO_CSE_KE_SHE_M3_OFFSET

```
#define CRYPTO_CSE_KE_SHE_M3_OFFSET ((uint8)48U)
```

Definition at line 57 of file Csm_Types.h.

5.13.2.4 CRYPTO_CSE_KE_SHE_M4_OFFSET

```
#define CRYPTO_CSE_KE_SHE_M4_OFFSET ((uint8)64U)
```

Definition at line 58 of file Csm_Types.h.

5.13.2.5 CRYPTO_CSE_KE_SHE_M5_OFFSET

```
#define CRYPTO_CSE_KE_SHE_M5_OFFSET ((uint8)96U)
```

Definition at line 59 of file Csm_Types.h.

5.13.2.6 CRYPTO_KE_CERTIFICATE_CURRENT_TIME

```
#define CRYPTO_KE_CERTIFICATE_CURRENT_TIME ((uint32)19U)
```

Definition at line 83 of file Csm_Types.h.

5.13.2.7 CRYPTO_KE_CERTIFICATE_DATA

```
#define CRYPTO_KE_CERTIFICATE_DATA ((uint32)0U)
```

Definition at line 81 of file Csm_Types.h.

5.13.2.8 CRYPTO_KE_CERTIFICATE_EXTENSIONS

```
#define CRYPTO_KE_CERTIFICATE_EXTENSIONS ((uint32)27U)
```

Definition at line 92 of file Csm_Types.h.

5.13.2.9 CRYPTO_KE_CERTIFICATE_ISSUER

```
#define CRYPTO_KE_CERTIFICATE_ISSUER ((uint32)23U)
```

Definition at line 87 of file Csm_Types.h.

5.13.2.10 CRYPTO_KE_CERTIFICATE_PARSING_FORMAT

```
#define CRYPTO_KE_CERTIFICATE_PARSING_FORMAT ((uint32)18U)
```

Definition at line 82 of file Csm_Types.h.

5.13.2.11 CRYPTO_KE_CERTIFICATE_SERIALNUMBER

```
#define CRYPTO_KE_CERTIFICATE_SERIALNUMBER ((uint32)21U)
```

Definition at line 85 of file Csm_Types.h.

5.13.2.12 CRYPTO_KE_CERTIFICATE_SIGNATURE

```
#define CRYPTO_KE_CERTIFICATE_SIGNATURE ((uint32)28U)
```

Definition at line 93 of file Csm_Types.h.

5.13.2.13 CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM

```
#define CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM ((uint32)22U)
```

Definition at line 86 of file Csm_Types.h.

5.13.2.14 CRYPTO_KE_CERTIFICATE_SUBJECT

```
#define CRYPTO_KE_CERTIFICATE_SUBJECT ((uint32)26U)
```

Definition at line 90 of file Csm_Types.h.

5.13.2.15 CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY

```
#define CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY ((uint32)1U)
```

Definition at line 91 of file Csm_Types.h.

5.13.2.16 CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER

```
#define CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER ((uint32)25U)
```

Definition at line 89 of file Csm_Types.h.

5.13.2.17 CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE

```
#define CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE ((uint32)24U)
```

Definition at line 88 of file Csm_Types.h.

5.13.2.18 CRYPTO_KE_CERTIFICATE_VERSION

```
#define CRYPTO_KE_CERTIFICATE_VERSION ((uint32)20U)
```

Definition at line 84 of file Csm_Types.h.

5.13.2.19 CRYPTO_KE_CIPHER_2NDKEY

```
#define CRYPTO_KE_CIPHER_2NDKEY ((uint32)7U)
```

Definition at line 68 of file Csm_Types.h.

5.13.2.20 CRYPTO_KE_CIPHER_IV

```
#define CRYPTO_KE_CIPHER_IV ((uint32)5U)
```

Definition at line 66 of file Csm_Types.h.

5.13.2.21 CRYPTO_KE_CIPHER_KEY

```
#define CRYPTO_KE_CIPHER_KEY ((uint32)1U)
```

Definition at line 65 of file Csm_Types.h.

5.13.2.22 CRYPTO_KE_CIPHER_PROOF

```
#define CRYPTO_KE_CIPHER_PROOF ((uint32)6U)
```

Definition at line 67 of file Csm_Types.h.

5.13.2.23 CRYPTO_KE_KEYDERIVATION_ALGORITHM

```
#define CRYPTO_KE_KEYDERIVATION_ALGORITHM ((uint32)15U)
```

Definition at line 77 of file Csm_Types.h.

5.13.2.24 CRYPTO_KE_KEYDERIVATION_ITERATIONS

```
#define CRYPTO_KE_KEYDERIVATION_ITERATIONS ((uint32)14U)
```

Definition at line 76 of file Csm_Types.h.

5.13.2.25 CRYPTO_KE_KEYDERIVATION_PASSWORD

```
#define CRYPTO_KE_KEYDERIVATION_PASSWORD ((uint32)1U)
```

Definition at line 74 of file Csm_Types.h.

5.13.2.26 CRYPTO_KE_KEYDERIVATION_SALT

```
#define CRYPTO_KE_KEYDERIVATION_SALT ((uint32)13U)
```

Definition at line 75 of file Csm_Types.h.

5.13.2.27 CRYPTO_KE_KEYEXCHANGE_ALGORITHM

```
#define CRYPTO_KE_KEYEXCHANGE_ALGORITHM ((uint32)12U)
```

Definition at line 73 of file Csm_Types.h.

5.13.2.28 CRYPTO_KE_KEYEXCHANGE_BASE

```
#define CRYPTO_KE_KEYEXCHANGE_BASE ((uint32)8U)
```

Definition at line 69 of file Csm_Types.h.

5.13.2.29 CRYPTO_KE_KEYEXCHANGE_OWNPUKEY

```
#define CRYPTO_KE_KEYEXCHANGE_OWNPUKEY ((uint32)10U)
```

Definition at line 71 of file Csm_Types.h.

5.13.2.30 CRYPTO_KE_KEYEXCHANGE_PRIVKEY

```
#define CRYPTO_KE_KEYEXCHANGE_PRIVKEY ((uint32)9U)
```

Definition at line 70 of file Csm_Types.h.

5.13.2.31 CRYPTO_KE_KEYGENERATE_ALGORITHM

```
#define CRYPTO_KE_KEYGENERATE_ALGORITHM ((uint32)17U)
```

Definition at line 80 of file Csm_Types.h.

5.13.2.32 CRYPTO_KE_KEYGENERATE_KEY

```
#define CRYPTO_KE_KEYGENERATE_KEY ((uint32)1U)
```

Definition at line 78 of file Csm_Types.h.

5.13.2.33 CRYPTO_KE_KEYGENERATE_SEED

```
#define CRYPTO_KE_KEYGENERATE_SEED ((uint32)16U)
```

Definition at line 79 of file Csm_Types.h.

5.13.2.34 CRYPTO_KE_MAC_KEY

```
#define CRYPTO_KE_MAC_KEY ((uint32)1U)
```

Definition at line 60 of file Csm_Types.h.

5.13.2.35 CRYPTO_KE_MAC_PROOF

```
#define CRYPTO_KE_MAC_PROOF ((uint32)2U)
```

Definition at line 61 of file Csm_Types.h.

5.13.2.36 CRYPTO_KE_RANDOM_ALGORITHM

```
#define CRYPTO_KE_RANDOM_ALGORITHM ((uint32)4U)
```

Definition at line 64 of file Csm_Types.h.

5.13.2.37 CRYPTO_KE_RANDOM_SEED_STATE

```
#define CRYPTO_KE_RANDOM_SEED_STATE ((uint32)3U)
```

Definition at line 63 of file Csm_Types.h.

5.13.2.38 CRYPTO_KE_SIGNATURE_KEY

```
#define CRYPTO_KE_SIGNATURE_KEY ((uint32)1U)
```

Definition at line 62 of file Csm_Types.h.

5.13.2.39 CRYPTO_KEY_MATERIAL

```
#define CRYPTO_KEY_MATERIAL ((uint32)1U)
```

Definition at line 94 of file Csm_Types.h.

5.13.2.40 CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE

```
#define CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE ((uint32)1U)
```

Definition at line 72 of file Csm_Types.h.

5.13.3 Enumeration Type Documentation

5.13.3.1 Crypto_AlgorithmFamilyType

```
enum Crypto_AlgorithmFamilyType
```

Enumeration of the algorithm family.

Enumerator

CRYPTO_ALGOFAM_NOT_SET	Algorithm family is not set
CRYPTO_ALGOFAM_SHA1	
CRYPTO_ALGOFAM_SHA2_224	
CRYPTO_ALGOFAM_SHA2_256	
CRYPTO_ALGOFAM_SHA2_384	
CRYPTO_ALGOFAM_SHA2_512	
CRYPTO_ALGOFAM_SHA2_512_224	
CRYPTO_ALGOFAM_SHA2_512_256	
CRYPTO_ALGOFAM_SHA3_224	
CRYPTO_ALGOFAM_SHA3_256	
CRYPTO_ALGOFAM_SHA3_384	
CRYPTO_ALGOFAM_SHA3_512	
CRYPTO_ALGOFAM_SHAKE128	
CRYPTO_ALGOFAM_SHAKE256	
CRYPTO_ALGOFAM_RIPEMD160	
CRYPTO_ALGOFAM_BLAKE_1_256	
CRYPTO_ALGOFAM_BLAKE_1_512	
CRYPTO_ALGOFAM_BLAKE_2s_256	
CRYPTO_ALGOFAM_BLAKE_2s_512	
CRYPTO_ALGOFAM_3DES	
CRYPTO_ALGOFAM_AES	AES cipher
CRYPTO_ALGOFAM_CHACHA	
CRYPTO_ALGOFAM_RSA	
CRYPTO_ALGOFAM_ED25519	
CRYPTO_ALGOFAM_BRAINPOOL	
CRYPTO_ALGOFAM_ECCNIST	
CRYPTO_ALGOFAM_SECURECOUNTER	
CRYPTO_ALGOFAM_RNG	
CRYPTO_ALGOFAM_SIPHASH	

Definition at line 196 of file Csm_Types.h.

5.13.3.2 Crypto_AlgorithmModeType

```
enum Crypto_AlgorithmModeType
```

Enumeration of the algorithm mode.

Enumerator

CRYPTO_ALGOMODE_NOT_SET	Algorithm key is not set
CRYPTO_ALGOMODE_ECB	Blockmode: Electronic Code Book
CRYPTO_ALGOMODE_CBC	Blockmode: Cipher Block Chaining
CRYPTO_ALGOMODE_CFB	
CRYPTO_ALGOMODE_OFB	
CRYPTO_ALGOMODE_CTR	
CRYPTO_ALGOMODE_GCM	
CRYPTO_ALGOMODE_XTS	
CRYPTO_ALGOMODE_RSAES_OAEP	
CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5	
CRYPTO_ALGOMODE_RSASSA_PSS	
CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5	
CRYPTO_ALGOMODE_8ROUNDS	
CRYPTO_ALGOMODE_12ROUNDS	
CRYPTO_ALGOMODE_20ROUNDS	
CRYPTO_ALGOMODE_HMAC	
CRYPTO_ALGOMODE_CMAC	Cipher-based MAC
CRYPTO_ALGOMODE_GMAC	
CRYPTO_ALGOMODE_CTRDRBG	
CRYPTO_ALGOMODE_SIPHASH_2_4	
CRYPTO_ALGOMODE_SIPHASH_4_8	
CRYPTO_ALGOMODE_CCM	
CRYPTO_ALGOMODE_CUSTOM	

Definition at line 120 of file Csm_Types.h.

5.13.3.3 Crypto_InputOutputRedirectionConfigType

```
enum Crypto_InputOutputRedirectionConfigType
```

Defines which of the input/output parameters are re-directed to a key element. The values can be combined to define a bit field.

Enumerator

CRYPTO_REDIRECT_CONFIG_PRIMARY_INPUT	
CRYPTO_REDIRECT_CONFIG_SECONDARY_INPUT	
CRYPTO_REDIRECT_CONFIG_TERTIARY_INPUT	
CRYPTO_REDIRECT_CONFIG_PRIMARY_OUTPUT	
CRYPTO_REDIRECT_CONFIG_SECONDARY_OUTPUT	

Definition at line 299 of file Csm_Types.h.

5.13.3.4 Crypto_JobStateType

```
enum Crypto_JobStateType
```

Enumeration of the current job state.

Enumerator

CRYPTO_JOBSTATE_IDLE	Job is in the state "idle". This state is reached after Csm_Init() or when the "Finish" state is finished.
CRYPTO_JOBSTATE_ACTIVE	Job is in the state "active". There was already some input or there are intermediate results. This state is reached, when the "update" or "start" operation finishes.

Definition at line 111 of file Csm_Types.h.

5.13.3.5 Crypto_OperationModeType

```
enum Crypto_OperationModeType
```

Enumeration which operation shall be performed. This enumeration is constructed from a bit mask, where the first bit indicates "Start", the second "Update" and the third "Finish".

Enumerator

CRYPTO_OPERATIONMODE_START	Operation Mode is "Start". The job's state shall be reset, i.e. previous input data and intermediate results shall be deleted.
CRYPTO_OPERATIONMODE_UPDATE	Operation Mode is "Update". Used to calculate intermediate results.
CRYPTO_OPERATIONMODE_STREAMSTART	Operation Mode is "Stream Start". Mixture of "Start" and "Update". Used for streaming.
CRYPTO_OPERATIONMODE_FINISH	Operation Mode is "Finish". The calculations shall be finalized.
CRYPTO_OPERATIONMODE_SINGLECALL	Operation Mode is "Single Call". Mixture of "Start", "Update" and "Finish".

Definition at line 159 of file Csm_Types.h.

5.13.3.6 Crypto_ProcessingType

```
enum Crypto_ProcessingType
```

Enumeration of the processing type.

Enumerator

CRYPTO_PROCESSING_ASYNC	Asynchronous job processing
CRYPTO_PROCESSING_SYNC	Synchronous job processing

Definition at line 278 of file Csm_Types.h.

5.13.3.7 Crypto_ServiceInfoType

enum [Crypto_ServiceInfoType](#)

Enumeration of the kind of the service.

Enumerator

CRYPTO_HASH	Hash Service
CRYPTO_MACGENERATE	MacGenerate Service
CRYPTO_MACVERIFY	MacVerify Service
CRYPTO_ENCRYPT	Encrypt Service
CRYPTO_DECRYPT	Decrypt Service
CRYPTO_AEADENCRYPT	AEADEncrypt Service
CRYPTO_AEADDECRYPT	AEADDecrypt Service
CRYPTO_SIGNATUREGENERATE	SignatureGenerate Service
CRYPTO_SIGNATUREVERIFY	SignatureVerify Service
CRYPTO_RANDOMGENERATE	RandomGenerate Service
CRYPTO_RANDOMSEED	RandomSeed Service
CRYPTO_KEYGENERATE	KeyGenerate Service
CRYPTO_KEYDERIVE	KeyDerive Service
CRYPTO_KEYEXCHANGEALCPUBVAL	KeyExchangeCalcPubVal Service
CRYPTO_KEYEXCHANGEALCSECRET	KeyExchangeCalcSecret Service
CRYPTO_CERTIFICATEPARSE	CertificateParse Service
CRYPTO_CERTIFICATEVERIFY	CertificateVerify Service
CRYPTO_KEYSETVALID	KeySetValid Service

Definition at line 171 of file Csm_Types.h.

5.13.3.8 Crypto_VerifyResultType

enum [Crypto_VerifyResultType](#)

Enumeration of the result type of verification operations.

Enumerator

CRYPTO_E_VER_OK	The result of the verification is "true", i.e. the two compared elements are identical. This return code shall be given as value "0"
CRYPTO_E_VER_NOT_OK	The result of the verification is "false", i.e. the two compared elements are not identical. This return code shall be given as value "1".

Definition at line 150 of file Csm_Types.h.

5.14 Dio.h File Reference

This file provides extern Dio Mcal API.

```
#include "Dio_Ipw.h"
```

Functions

- void [Dio_GetVersionInfo](#) (Std_VersionInfoType *VersionInfo)
Service to get the version information of this module.
- [Dio_LevelType Dio_ReadChannel](#) ([Dio_ChannelType](#) ChannelId)
Returns the value of the specified DIO channel.
- void [Dio_WriteChannel](#) ([Dio_ChannelType](#) ChannelId, [Dio_LevelType](#) Level)
Returns the value of the specified DIO channel.
- [Dio_LevelType Dio_FlipChannel](#) ([Dio_ChannelType](#) ChannelId)
Service to flip (change from 1 to 0 or from 0 to 1) the level of a channel and return the level of the channel after flip.
- [Dio_PortLevelType Dio_ReadPort](#) ([Dio_PortType](#) PortId)
Returns the level of all channels of that port.
- void [Dio_WritePort](#) ([Dio_PortType](#) PortId, [Dio_PortLevelType](#) Level)
Service to set a value of the port.
- [Dio_PortLevelType Dio_ReadChannelGroup](#) (const [Dio_ChannelGroupType](#) *ChannelGroupIdPtr)
This Service reads a subset of the adjoining bits of a port.
- void [Dio_WriteChannelGroup](#) (const [Dio_ChannelGroupType](#) *ChannelGroupIdPtr, [Dio_PortLevelType](#) Level)
Service to set a subset of the adjoining bits of a port to a specified level.
- void [Dio_MaskedWritePort](#) ([Dio_PortType](#) PortId, [Dio_PortLevelType](#) Level, [Dio_PortLevelType](#) Mask)
Service to set the value of a given port with required mask.

5.14.1 Detailed Description

This file provides extern Dio Mcal API.

5.14.2 Function Documentation

5.14.2.1 Dio_FlipChannel()

```
Dio\_LevelType Dio_FlipChannel (
    Dio\_ChannelType ChannelId )
```

Service to flip (change from 1 to 0 or from 0 to 1) the level of a channel and return the level of the channel after flip.

Note

Function ID : DES_DIO_API_008
Service ID : 0x11

Parameters

in	<i>Channel↔ Id</i>	ID of DIO channel
----	------------------------	-------------------

Returns

Dio_LevelType

5.14.2.2 Dio_GetVersionInfo()

```
void Dio_GetVersionInfo (
    Std_VersionInfoType * VersionInfo )
```

Service to get the version information of this module.

Note

Function ID : DES_DIO_API_007 Service ID : 0x12 Sync/Async : Synchronous Reentrancy : Reentrant

Parameters

in	<i>VersionInfo</i>	Pointer to where to store the version information of this module.
----	--------------------	-------------------------------------------------------------------

Returns

boolean

5.14.2.3 Dio_MaskedWritePort()

```
void Dio_MaskedWritePort (
    Dio_PortType PortId,
    Dio_PortLevelType Level,
    Dio_PortLevelType Mask )
```

Service to set the value of a given port with required mask.

Note

Function ID : DES_DIO_API_009
Service ID : 0x13

Parameters

in	<i>Port↔ Id</i>	ID of DIO Port.
in	<i>Level</i>	Value to be written
in	<i>Mask</i>	Channels to be masked in the port.

Returns

None

5.14.2.4 Dio_ReadChannel()

```
Dio_LevelType Dio_ReadChannel (
    Dio_ChannelType ChannelId )
```

Returns the value of the specified DIO channel.

Note

Function ID : DES_DIO_API_001
Service ID : 0x00

Parameters

in	<i>ChannelId</i>	ID of DIO channel.
----	------------------	--------------------

Returns

Dio_LevelType

5.14.2.5 Dio_ReadChannelGroup()

```
Dio_PortLevelType Dio_ReadChannelGroup (
    const Dio_ChannelGroupType * ChannelGroupIdPtr )
```

This Service reads a subset of the adjoining bits of a port.

Note

Function ID : DES_DIO_API_005
Service ID : 0x04

Parameters

in	<i>ChannelGroupIdPtr</i>	Pointer to ChannelGroup
----	--------------------------	-------------------------

Returns

Dio_PortLevelType

5.14.2.6 Dio_ReadPort()

```
Dio_PortLevelType Dio_ReadPort (
    Dio_PortType PortId )
```

Returns the level of all channels of that port.

Note

Function ID : DES_DIO_API_003
Service ID : 0x02

Parameters

in	<i>PortId</i>	ID of DIO Port
----	---------------	----------------

Returns

Dio_PortLevelType

5.14.2.7 Dio_WriteChannel()

```
void Dio_WriteChannel (
    Dio_ChannelType ChannelId,
    Dio_LevelType Level )
```

Returns the value of the specified DIO channel.

Note

Function ID : DES_DIO_API_002
Service ID : 0x01

Parameters

in	<i>ChannelId</i>	ID of DIO channel.
in	<i>Level</i>	Value to be written..

Returns

None

5.14.2.8 Dio_WriteChannelGroup()

```
void Dio_WriteChannelGroup (
    const Dio_ChannelGroupType * ChannelGroupIdPtr,
    Dio_PortLevelType Level )
```

Service to set a subset of the adjoining bits of a port to a specified level.

Note

Function ID : DES_DIO_API_006

Service ID : 0x05

Parameters

in	<i>ChannelGroupIdPtr</i>	Pointer to ChannelGroup.
in	<i>Level</i>	Value to be written..

Returns

None

5.14.2.9 Dio_WritePort()

```
void Dio_WritePort (
    Dio_PortType PortId,
    Dio_PortLevelType Level )
```

Service to set a value of the port.

Note

Function ID : DES_DIO_API_004

Service ID : 0x03

Parameters

in	<i>Port↔ Id</i>	ID of DIO Port.
in	<i>Level</i>	Value to be written..

Returns

None

5.15 Dio_GeneralTypes.h File Reference

This file provides extern Dio module structure and macro.

```
#include "Mcal.h"
#include "Device_Register.h"
```

Classes

- struct [Dio_ChannelGroupType](#)
Type of a DIO channel group representation.
- struct [Dio_ConfigType](#)
Type of a DIO configuration structure.

Macros

- #define [DIO_E_PARAM_INVALID_CHANNEL_ID](#) ((uint8)0x0AUL)
Invalid channel name requested.
- #define [DIO_E_PARAM_CONFIG](#) ((uint8)0x10UL)
API service called with "NULL pointer" parameter.
- #define [DIO_E_PARAM_INVALID_PORT_ID](#) ((uint8)0x14UL)
Invalid port name requested.
- #define [DIO_E_PARAM_INVALID_GROUP](#) ((uint8)0x1FUL)
Invalid ChannelGroup id passed.
- #define [DIO_E_PARAM_POINTER](#) ((uint8)0x20UL)
API service called with a NULL pointer.
- #define [DIO_E_PARAM_LEVEL](#) ((uint8)0x21UL)
API service called with invalid channel level value.
- #define [DIO_READCHANNEL_ID](#) ((uint8)0x00UL)
API service ID for [Dio_ReadChannel\(\)](#) function.
- #define [DIO_WRITECHANNEL_ID](#) ((uint8)0x01UL)
API service ID for [Dio_WriteChannel\(\)](#) function.
- #define [DIO_FLIPCHANNEL_ID](#) ((uint8)0x11UL)
API service ID for [Dio_FlipChannel\(\)](#) function.
- #define [DIO_READPORT_ID](#) ((uint8)0x02UL)
API service ID for [Dio_ReadPort\(\)](#) function.
- #define [DIO_WRITEPORT_ID](#) ((uint8)0x03UL)
API service ID for [Dio_WritePort\(\)](#) function.
- #define [DIO_READCHANNELGROUP_ID](#) ((uint8)0x04UL)
API service ID for [Dio_ReadChannel\(\)](#) Group function.
- #define [DIO_WRITECHANNELGROUP_ID](#) ((uint8)0x05UL)
API service ID for [Dio_WriteChannel\(\)](#) Group function.
- #define [DIO_GETVERSIONINFO_ID](#) ((uint8)0x12UL)
API service ID for [DIO Get Version\(\)](#) Info function.
- #define [DIO_MASKEDWRITEPORT_ID](#) ((uint8)0x22UL)
API service ID for [Dio_MaskedWritePort\(\)](#) function.
- #define [DIO_INSTANCE_ID](#) ((uint8)0x00UL)
Instance ID of the Dio driver.

Typedefs

- typedef uint16 [Dio_ChannelType](#)
Type of a DIO channel representation.
- typedef uint8 [Dio_PortType](#)
Type of a DIO port representation.
- typedef uint32 [Dio_PortLevelType](#)
Type of a DIO port levels representation.
- typedef uint8 [Dio_LevelType](#)
Type of a DIO channel levels representation.

5.15.1 Detailed Description

This file provides extern Dio module structure and macro.

5.15.2 Macro Definition Documentation

5.15.2.1 DIO_E_PARAM_CONFIG

```
#define DIO_E_PARAM_CONFIG ((uint8)0x10UL)
```

API service called with "NULL pointer" parameter.

Definition at line 61 of file Dio_GeneralTypes.h.

5.15.2.2 DIO_E_PARAM_INVALID_CHANNEL_ID

```
#define DIO_E_PARAM_INVALID_CHANNEL_ID ((uint8)0x0AUL)
```

Invalid channel name requested.

Definition at line 58 of file Dio_GeneralTypes.h.

5.15.2.3 DIO_E_PARAM_INVALID_GROUP

```
#define DIO_E_PARAM_INVALID_GROUP ((uint8)0x1FUL)
```

Invalid ChannelGroup id passed.

Definition at line 67 of file Dio_GeneralTypes.h.

5.15.2.4 DIO_E_PARAM_INVALID_PORT_ID

```
#define DIO_E_PARAM_INVALID_PORT_ID ((uint8)0x14UL)
```

Invalid port name requested.

Definition at line 64 of file Dio_GeneralTypes.h.

5.15.2.5 DIO_E_PARAM_LEVEL

```
#define DIO_E_PARAM_LEVEL ((uint8)0x21UL)
```

API service called with invalid channel level value.

Definition at line 73 of file Dio_GeneralTypes.h.

5.15.2.6 DIO_E_PARAM_POINTER

```
#define DIO_E_PARAM_POINTER ((uint8)0x20UL)
```

API service called with a NULL pointer.

Definition at line 70 of file Dio_GeneralTypes.h.

5.15.2.7 DIO_FLIPCHANNEL_ID

```
#define DIO_FLIPCHANNEL_ID ((uint8)0x11UL)
```

API service ID for [Dio_FlipChannel\(\)](#) function.

Definition at line 83 of file Dio_GeneralTypes.h.

5.15.2.8 DIO_GETVERSIONINFO_ID

```
#define DIO_GETVERSIONINFO_ID ((uint8)0x12UL)
```

API service ID for DIO Get Version() Info function.

Definition at line 98 of file Dio_GeneralTypes.h.

5.15.2.9 DIO_INSTANCE_ID

```
#define DIO_INSTANCE_ID ((uint8)0x00UL)
```

Instance ID of the Dio driver.

Definition at line 104 of file Dio_GeneralTypes.h.

5.15.2.10 DIO_MASKEDWRITEPORT_ID

```
#define DIO_MASKEDWRITEPORT_ID ((uint8)0x22UL)
```

API service ID for [Dio_MaskedWritePort\(\)](#) function.

Definition at line 101 of file Dio_GeneralTypes.h.

5.15.2.11 DIO_READCHANNEL_ID

```
#define DIO_READCHANNEL_ID ((uint8)0x00UL)
```

API service ID for [Dio_ReadChannel\(\)](#) function.

Definition at line 76 of file Dio_GeneralTypes.h.

5.15.2.12 DIO_READCHANNELGROUP_ID

```
#define DIO_READCHANNELGROUP_ID ((uint8)0x04UL)
```

API service ID for [Dio_ReadChannel\(\)](#) Group function.

Definition at line 92 of file Dio_GeneralTypes.h.

5.15.2.13 DIO_READPORT_ID

```
#define DIO_READPORT_ID ((uint8)0x02UL)
```

API service ID for [Dio_ReadPort\(\)](#) function.

Definition at line 86 of file Dio_GeneralTypes.h.

5.15.2.14 DIO_WRITECHANNEL_ID

```
#define DIO_WRITECHANNEL_ID ((uint8)0x01UL)
```

API service ID for [Dio_WriteChannel\(\)](#) function.

Definition at line 80 of file Dio_GeneralTypes.h.

5.15.2.15 DIO_WRITECHANNELGROUP_ID

```
#define DIO_WRITECHANNELGROUP_ID ((uint8)0x05UL)
```

API service ID for [Dio_WriteChannel\(\)](#) Group function.

Definition at line 95 of file Dio_GeneralTypes.h.

5.15.2.16 DIO_WRITEPORT_ID

```
#define DIO_WRITEPORT_ID ((uint8)0x03UL)
```

API service ID for [Dio_WritePort\(\)](#) function.

Definition at line 89 of file Dio_GeneralTypes.h.

5.15.3 Typedef Documentation

5.15.3.1 Dio_ChannelType

```
typedef uint16 Dio_ChannelType
```

Type of a DIO channel representation.

Definition at line 109 of file Dio_GeneralTypes.h.

5.15.3.2 Dio_LevelType

```
typedef uint8 Dio_LevelType
```

Type of a DIO channel levels representation.

Definition at line 118 of file Dio_GeneralTypes.h.

5.15.3.3 Dio_PortLevelType

```
typedef uint32 Dio_PortLevelType
```

Type of a DIO port levels representation.

Definition at line 115 of file Dio_GeneralTypes.h.

5.15.3.4 Dio_PortType

```
typedef uint8 Dio_PortType
```

Type of a DIO port representation.

Definition at line 112 of file Dio_GeneralTypes.h.

5.16 Eep.h File Reference

```
#include "Mcal.h"  
#include "Eep_Ipw.h"
```

Functions

- void [Eep_Init](#) (const Eep_ConfigType *ConfigPtr)
- void [Eep_SetMode](#) (MemIf_ModeType Mode)
- Std_ReturnType [Eep_Read](#) (Eep_AddressType EepromAddress, uint8 *DataBufferPtr, Eep_LengthType Length)
- Std_ReturnType [Eep_Write](#) (Eep_AddressType EepromAddress, const uint8 *DataBufferPtr, Eep_LengthType Length)
- Std_ReturnType [Eep_Erase](#) (Eep_AddressType EepromAddress, Eep_LengthType Length)
- Std_ReturnType [Eep_Compare](#) (Eep_AddressType EepromAddress, const uint8 *DataBufferPtr, Eep_LengthType Length)
- void [Eep_Cancel](#) (void)
- MemIf_StatusType [Eep_GetStatus](#) (void)
- MemIf_JobResultType [Eep_GetJobResult](#) (void)
- void [Eep_GetVersionInfo](#) (Std_VersionInfoType *VersionInfoPtr)

5.16.1 Function Documentation

5.16.1.1 Eep_Cancel()

```
void Eep_Cancel (  
    void )
```

5.16.1.2 Eep_Compare()

```
Std_ReturnType Eep_Compare (  
    Eep_AddressType EepromAddress,  
    const uint8 * DataBufferPtr,  
    Eep_LengthType Length )
```

5.16.1.3 Eep_Erase()

```
Std_ReturnType Eep_Erase (
    Eep_AddressType EepromAddress,
    Eep_LengthType Length )
```

5.16.1.4 Eep_GetJobResult()

```
MemIf_JobResultType Eep_GetJobResult (
    void )
```

5.16.1.5 Eep_GetStatus()

```
MemIf_StatusType Eep_GetStatus (
    void )
```

5.16.1.6 Eep_GetVersionInfo()

```
void Eep_GetVersionInfo (
    Std_VersionInfoType * VersionInfoPtr )
```

5.16.1.7 Eep_Init()

```
void Eep_Init (
    const Eep_ConfigType * ConfigPtr )
```

5.16.1.8 Eep_Read()

```
Std_ReturnType Eep_Read (
    Eep_AddressType EepromAddress,
    uint8 * DataBufferPtr,
    Eep_LengthType Length )
```

5.16.1.9 Eep_SetMode()

```
void Eep_SetMode (
    MemIf_ModeType Mode )
```

5.16.1.10 Eep_Write()

```
Std_ReturnType Eep_Write (
    Eep_AddressType EepromAddress,
    const uint8 * DataBufferPtr,
    Eep_LengthType Length )
```

5.17 Fee.h File Reference

This file provides Fee api interface.

```
#include "Fee_Cfg.h"
```

Functions

- void [Fee_Init](#) (const [Fee_ConfigType](#) *ConfigPtr)
Initializes the fee Driver module.
- void [Fee_SetMode](#) (MemIf_ModeType Mode)
Set the fee working mode.
- Std_ReturnType [Fee_Read](#) (uint16 BlockNumber, uint16 BlockOffset, uint8 *DataBufferPtr, uint16 Length)
Read the fee block data.
- Std_ReturnType [Fee_Write](#) (uint16 BlockNumber, const uint8 *DataBufferPtr)
Write the fee block data.
- void [Fee_Cancel](#) (void)
Cancel the fee task, call the cancel function of the underlying flash driver.
- MemIf_StatusType [Fee_GetStatus](#) (void)
Get the fee current status.
- MemIf_JobResultType [Fee_GetJobResult](#) (void)
query the result of the last accepted job issued by the upper
- Std_ReturnType [Fee_InvalidateBlock](#) (uint16 BlockNumber)
invalid the fee block data.
- void [Fee_GetVersionInfo](#) (Std_VersionInfoType *VersionInfoPtr)
invalid the fee block data.
- Std_ReturnType [Fee_EraseImmediateBlock](#) (uint16 BlockNumber)
erase the fee immediate block data.
- void [Fee_JobEndNotification](#) (void)
report to this module the successful end of an asynchronous operation.
- void [Fee_JobErrorNotification](#) (void)
report to this module the failure of an asynchronous operation.

5.17.1 Detailed Description

This file provides Fee api interface.

5.17.2 Function Documentation

5.17.2.1 Fee_Cancel()

```
void Fee_Cancel (
    void )
```

Cancel the fee task, call the cancel function of the underlying flash driver.

Note

Function ID : DES_FEE_API_005
Service ID : 0x04

Parameters

in	<i>None</i>	
in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.17.2.2 Fee_EraseImmediateBlock()

```
Std_ReturnType Fee_EraseImmediateBlock (
    uint16 BlockNumber )
```

erase the fee immediate block data.

Note

Function ID : DES_FEE_API_010
Service ID : 0x08

Parameters

in	<i>BlockNumber</i>	the block which will to erase.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

- E_OK: The requested job has been accepted by the module.
- E_NOT_OK: The requested job has not been accepted by the module.

5.17.2.3 Fee_GetJobResult()

```
MemIf_JobResultType Fee_GetJobResult (
    void )
```

query the result of the last accepted job issued by the upper

Note

Function ID : DES_FEE_API_007
Service ID : 0x06

Parameters

in	<i>None</i>	
in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

MemIf_JobResultType MEMIF_JOB_OK: The last job has been finished successfully. MEMIF_JOB_PENDING↔: The last job is waiting for execution or currently being executed. MEMIF_JOB_CANCELED: The last job has been canceled. MEMIF_JOB_FAILED: The last job has not been finished successfully (it failed). MEMIF_BLOCK_IN↔CONSISTENT: The requested block is inconsistent, it may contain corrupted data. MEMIF_BLOCK_INVALID: The requested block has been invalidated, the requested readoperation can not be performed.

5.17.2.4 Fee_GetStatus()

```
MemIf_StatusType Fee_GetStatus (
    void )
```

Get the fee current status.

Note

Function ID : DES_FEE_API_006
Service ID : 0x05

Parameters

in	<i>None</i>	
in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

MemIf_StatusType MEMIF_UNINIT: The FEE module has not been initialized. MEMIF_IDLE: The FEE module is currently idle. MEMIF_BUSY: The FEE module is currently busy. MEMIF_BUSY_INTERNAL: The FEE module is busy with internal management operations.

5.17.2.5 Fee_GetVersionInfo()

```
void Fee_GetVersionInfo (
    Std_VersionInfoType * VersionInfoPtr )
```

invalid the fee block data.

Note

Function ID : DES_FEE_API_009
Service ID : 0x07

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>VersionInfoPtr:the</i>	fee module version

Returns

None

5.17.2.6 Fee_Init()

```
void Fee_Init (
    const Fee_ConfigType * ConfigPtr )
```

Initializes the fee Driver module.

Note

Function ID : DES_FEE_API_001
Service ID : 0x00

Parameters

in	<i>ConfigPtr</i>	Pointer to the selected configuration set.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.17.2.7 Fee_InvalidateBlock()

```
Std_ReturnType Fee_InvalidateBlock (
    uint16 BlockNumber )
```

invalid the fee block data.

Note

Function ID : DES_FEE_API_008
Service ID : 0x07

Parameters

in	<i>BlockNumber</i>	the block which will to invalid.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

invalid success return E_OK or fail return E_NOT_OK,

5.17.2.8 Fee_JobEndNotification()

```
void Fee_JobEndNotification (
    void )
```

report to this module the successful end of an asynchronous operation.

Note

Function ID : DES_FEE_API_011
Service ID : 0x10

Parameters

in	<i>None</i>	
in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.17.2.9 Fee_JobErrorNotification()

```
void Fee_JobErrorNotification (
    void )
```

report to this module the failure of an asynchronous operation.

Note

Function ID : DES_FEE_API_012

Service ID : 0x11

Parameters

in	<i>None</i>	
in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.17.2.10 Fee_Read()

```
Std_ReturnType Fee_Read (
    uint16 BlockNumber,
    uint16 BlockOffset,
    uint8 * DataBufferPtr,
    uint16 Length )
```

Read the fee block data.

Note

Function ID : DES_FEE_API_003

Service ID : 0x02

Parameters

in	<i>BlockNumber</i>	Number of logical block, also denoting start address of that block.
in	<i>BlockOffset</i>	the offset in block will to read
in	<i>DataBufferPtr</i>	read data to store this address.
in	<i>Length</i>	read data length
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

- E_OK: The requested job has been accepted by the module.
- E_NOT_OK: The requested job has not been accepted by the module.

5.17.2.11 Fee_SetMode()

```
void Fee_SetMode (
    MemIf_ModeType Mode )
```

Set the fee working mode.

Note

Function ID : DES_FEE_API_002
Service ID : 0x01

Parameters

in	<i>Mode</i>	fee working mode MEMIF_MODE_SLOW or MEMIF_MODE_FAST.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.17.2.12 Fee_Write()

```
Std_ReturnType Fee_Write (
    uint16 BlockNumber,
    const uint8 * DataBufferPtr )
```

Write the fee block data.

Note

Function ID : DES_FEE_API_004
Service ID : 0x03

Parameters

in	<i>BlockNumber</i>	Number of logical block, also denoting start address of that block.
in	<i>DataBufferPtr</i>	write data from this address.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

- E_OK: The requested job has been accepted by the module.
- E_NOT_OK: The requested job has not been accepted by the module.

5.18 Fee_GeneralTypes.h File Reference

This file provides Fee structure and enum and macro information.

```
#include "MemIf_Types.h"
#include "StandardTypes.h"
#include "Mcal.h"
```

Classes

- struct [Fee_ClusterType](#)
Fee cluster configuration structure .
- struct [Fee_BlockType](#)
Fee Configuration type is a stub type, not used, but required by ASR 4.2.2. .
- struct [Fee_ClusterGroupType](#)
Fee cluster group configuration structure .
- struct [Fee_ConfigType](#)
Fee cluster group configuration structure .

Macros

- #define [FEE_INSTANCE_ID](#) 0U
- #define [FEE_INIT_ID](#) 0x00U
- #define [FEE_SETMODE_ID](#) 0x01U
- #define [FEE_READ_ID](#) 0x02U
- #define [FEE_WRITE_ID](#) 0x03U
- #define [FEE_CANCEL_ID](#) 0x04U
- #define [FEE_GETSTATUS_ID](#) 0x05U
- #define [FEE_GETJOBRESULT_ID](#) 0x06U
- #define [FEE_INVALIDATEBLOCK_ID](#) 0x07U
- #define [FEE_GETVERSIONINFO_ID](#) 0x08U
- #define [FEE_ERASEIMMEDIATEBLOCK_ID](#) 0x09U
- #define [FEE_JOBENDNOTIFICATION_ID](#) 0x10U
- #define [FEE_JOBERRORNOTIFICATION_ID](#) 0x11U
- #define [FEE_MAINFUNCTION_ID](#) 0x12U
- #define [FEE_E_UNINIT](#) 0x01U
- #define [FEE_E_INVALID_BLOCK_NO](#) 0x02U
- #define [FEE_E_INVALID_BLOCK_OFS](#) 0x03U
- #define [FEE_E_PARAM_POINTER](#) 0x04U
- #define [FEE_E_INVALID_BLOCK_LEN](#) 0x05U
- #define [FEE_E_BUSY](#) 0x06U
- #define [FEE_E_BUSY_INTERNAL](#) 0x07U
- #define [FEE_E_INVALID_CANCEL](#) 0x08U
- #define [FEE_E_INIT_FAILED](#) 0x09U
- #define [FEE_E_CANCEL_API](#) 0x0AU
- #define [FEE_E_CLUSTER_GROUP_IDX](#) 0x0BU
- #define [FEE_E_FOREIGN_BLOCKS_OVF](#) 0x0CU
- #define [FEE_BLOCKHEAD_OVERHEAD](#) 16U
Management overhead per logical block header info in bytes.
- #define [FEE_CLUSTERHEAD_OVERHEAD](#) 16U
Management overhead per logical cluster head info in bytes.
- #define [FEE_FLAG_OVERHEAD](#) 8U
Management overhead per valid or invalid in bytes.

Enumerations

- enum `Fee_BlockAssignmentType` { `FEE_PROJECT_SHARED` = 0x01U, `FEE_PROJECT_APP` = 0x02U, `FEE_PROJECT_BOOTLOADER` = 0x03U, `FEE_PROJECT_RESERVED` = 0xFFU }
- Fee block assignment type .*

5.18.1 Detailed Description

This file provides Fee structure and enum and macro information.

5.18.2 Macro Definition Documentation

5.18.2.1 FEE_BLOCKHEAD_OVERHEAD

```
#define FEE_BLOCKHEAD_OVERHEAD 16U
```

Management overhead per logical block header info in bytes.

Definition at line 114 of file `Fee_GeneralTypes.h`.

5.18.2.2 FEE_CANCEL_ID

```
#define FEE_CANCEL_ID 0x04U
```

Definition at line 67 of file `Fee_GeneralTypes.h`.

5.18.2.3 FEE_CLUSTERHEAD_OVERHEAD

```
#define FEE_CLUSTERHEAD_OVERHEAD 16U
```

Management overhead per logical cluster head info in bytes.

Definition at line 117 of file `Fee_GeneralTypes.h`.

5.18.2.4 FEE_E_BUSY

```
#define FEE_E_BUSY 0x06U
```

Definition at line 97 of file `Fee_GeneralTypes.h`.

5.18.2.5 FEE_E_BUSY_INTERNAL

```
#define FEE_E_BUSY_INTERNAL 0x07U
```

Definition at line 99 of file Fee_GeneralTypes.h.

5.18.2.6 FEE_E_CANCEL_API

```
#define FEE_E_CANCEL_API 0x0AU
```

Definition at line 106 of file Fee_GeneralTypes.h.

5.18.2.7 FEE_E_CLUSTER_GROUP_IDX

```
#define FEE_E_CLUSTER_GROUP_IDX 0x0BU
```

Definition at line 108 of file Fee_GeneralTypes.h.

5.18.2.8 FEE_E_FOREIGN_BLOCKS_OVF

```
#define FEE_E_FOREIGN_BLOCKS_OVF 0x0CU
```

Definition at line 111 of file Fee_GeneralTypes.h.

5.18.2.9 FEE_E_INIT_FAILED

```
#define FEE_E_INIT_FAILED 0x09U
```

Definition at line 103 of file Fee_GeneralTypes.h.

5.18.2.10 FEE_E_INVALID_BLOCK_LEN

```
#define FEE_E_INVALID_BLOCK_LEN 0x05U
```

Definition at line 95 of file Fee_GeneralTypes.h.

5.18.2.11 FEE_E_INVALID_BLOCK_NO

```
#define FEE_E_INVALID_BLOCK_NO 0x02U
```

Definition at line 89 of file Fee_GeneralTypes.h.

5.18.2.12 FEE_E_INVALID_BLOCK_OFS

```
#define FEE_E_INVALID_BLOCK_OFS 0x03U
```

Definition at line 91 of file Fee_GeneralTypes.h.

5.18.2.13 FEE_E_INVALID_CANCEL

```
#define FEE_E_INVALID_CANCEL 0x08U
```

Definition at line 101 of file Fee_GeneralTypes.h.

5.18.2.14 FEE_E_PARAM_POINTER

```
#define FEE_E_PARAM_POINTER 0x04U
```

Definition at line 93 of file Fee_GeneralTypes.h.

5.18.2.15 FEE_E_UNINIT

```
#define FEE_E_UNINIT 0x01U
```

Definition at line 87 of file Fee_GeneralTypes.h.

5.18.2.16 FEE_ERASEIMMEDIATEBLOCK_ID

```
#define FEE_ERASEIMMEDIATEBLOCK_ID 0x09U
```

Definition at line 77 of file Fee_GeneralTypes.h.

5.18.2.17 FEE_FLAG_OVERHEAD

```
#define FEE_FLAG_OVERHEAD 8U
```

Management overhead per valid or invalid in bytes.

Definition at line 119 of file Fee_GeneralTypes.h.

5.18.2.18 FEE_GETJOBRESULT_ID

```
#define FEE_GETJOBRESULT_ID 0x06U
```

Definition at line 71 of file Fee_GeneralTypes.h.

5.18.2.19 FEE_GETSTATUS_ID

```
#define FEE_GETSTATUS_ID 0x05U
```

Definition at line 69 of file Fee_GeneralTypes.h.

5.18.2.20 FEE_GETVERSIONINFO_ID

```
#define FEE_GETVERSIONINFO_ID 0x08U
```

Definition at line 75 of file Fee_GeneralTypes.h.

5.18.2.21 FEE_INIT_ID

```
#define FEE_INIT_ID 0x00U
```

Definition at line 59 of file Fee_GeneralTypes.h.

5.18.2.22 FEE_INSTANCE_ID

```
#define FEE_INSTANCE_ID 0U
```

Definition at line 55 of file Fee_GeneralTypes.h.

5.18.2.23 FEE_INVALIDATEBLOCK_ID

```
#define FEE_INVALIDATEBLOCK_ID 0x07U
```

Definition at line 73 of file Fee_GeneralTypes.h.

5.18.2.24 FEE_JOBENDNOTIFICATION_ID

```
#define FEE_JOBENDNOTIFICATION_ID 0x10U
```

Definition at line 79 of file Fee_GeneralTypes.h.

5.18.2.25 FEE_JOBERRORNOTIFICATION_ID

```
#define FEE_JOBERRORNOTIFICATION_ID 0x11U
```

Definition at line 81 of file Fee_GeneralTypes.h.

5.18.2.26 FEE_MAINFUNCTION_ID

```
#define FEE_MAINFUNCTION_ID 0x12U
```

Definition at line 83 of file Fee_GeneralTypes.h.

5.18.2.27 FEE_READ_ID

```
#define FEE_READ_ID 0x02U
```

Definition at line 63 of file Fee_GeneralTypes.h.

5.18.2.28 FEE_SETMODE_ID

```
#define FEE_SETMODE_ID 0x01U
```

Definition at line 61 of file Fee_GeneralTypes.h.

5.18.2.29 FEE_WRITE_ID

```
#define FEE_WRITE_ID 0x03U
```

Definition at line 65 of file Fee_GeneralTypes.h.

5.18.3 Enumeration Type Documentation

5.18.3.1 Fee_BlockAssignmentType

```
enum Fee_BlockAssignmentType
```

Fee block assignment type .

Enumerator

FEE_PROJECT_SHARED	
FEE_PROJECT_APP	block is used for all the projects
FEE_PROJECT_BOOTLOADER	block is used for the application project
FEE_PROJECT_RESERVED	block is used for the bootloader project

Definition at line 125 of file Fee_GeneralTypes.h.

5.19 Fee_InternalTypes.h File Reference

This file provides Fee internal structure and enum and macro information.

```
#include "MemIf_Types.h"
#include "StandardTypes.h"
```

Classes

- struct [Fee_ClusterHeaderType](#)
Fee cluster header configuration structure .
- struct [Fee_BlockHeaderType](#)
Fee block header configuration structure .
- struct [Fee_ClusterInfoType](#)
runtime Fee Cluster information structure
- struct [Fee_BlockInfoType](#)
runtime Fee block information structure
- struct [Fee_JobInfoType](#)
Fee Job Information structure .
- struct [Fee_JobPendingInfoType](#)
Fee Pending Job Information structure .

Macros

- #define [FEE_BLOCK_INVALID_VALUE](#) (0xFFFFU)

Enumerations

- enum [Fee_ClusterStatusType](#) { [FEE_CLUSTER_VALID](#) = 0, [FEE_CLUSTER_INVALID](#), [FEE_CLUSTER_INCONSISTENT](#) }
Status of Fee cluster header .
- enum [Fee_JobType](#) {
[FEE_JOB_READ](#) = 0U, [FEE_JOB_READ_DONE](#), [FEE_JOB_WRITE](#), [FEE_JOB_WRITE_DATA](#),
[FEE_JOB_WRITE_UNALIGNED](#), [FEE_JOB_WRITE_VALIDATE](#), [FEE_JOB_WRITE_DONE](#), [FEE_JOB_INVALID_BLOCK](#),
[FEE_JOB_INVALID_BLOCK_DONE](#), [FEE_JOB_ERASE_IMMEDIATE](#), [FEE_JOB_ERASE_IMMEDIATE_DONE](#), [FEE_JOB_INT_SCAN](#),
[FEE_JOB_INT_SCAN_CLR_HDR_PARSE](#), [FEE_JOB_INT_SCAN_CLR](#), [FEE_JOB_INT_SCAN_CLR_FMT](#), [FEE_JOB_INT_SCAN_CLR_FMT_DONE](#),
[FEE_JOB_INT_SCAN_BLOCK_HDR_PARSE](#), [FEE_JOB_INT_SWAP_BLOCK](#), [FEE_JOB_INT_SWAP_CLR_FMT](#), [FEE_JOB_INT_SWAP_DATA_READ](#),
[FEE_JOB_INT_SWAP_DATA_WRITE](#), [FEE_JOB_INT_SWAP_CLR_VLD_DONE](#), [FEE_JOB_DONE](#), [FEE_JOB_SETMODE](#) }

Type of job currently executed by Fee_MainFunction .

- enum `Fee_BlockStatusType` {
`FEE_BLOCK_VALID = 0, FEE_BLOCK_INVALID, FEE_BLOCK_INCONSISTENT, FEE_BLOCK_HEADER_BLANK,`
`FEE_BLOCK_INCONSISTENT_COPY, FEE_BLOCK_NEVER_WRITTEN }`

Status of Fee block header .

5.19.1 Detailed Description

This file provides Fee internal structure and enum and macro information.

5.19.2 Macro Definition Documentation

5.19.2.1 FEE_BLOCK_INVALID_VALUE

```
#define FEE_BLOCK_INVALID_VALUE (0xFFFFFU)
```

Definition at line 52 of file `Fee_InternalTypes.h`.

5.19.3 Enumeration Type Documentation

5.19.3.1 Fee_BlockStatusType

```
enum Fee_BlockStatusType
```

Status of Fee block header .

Enumerator

<code>FEE_BLOCK_VALID</code>	Fee block is valid
<code>FEE_BLOCK_INVALID</code>	Fee block is invalid(has been invalidated)
<code>FEE_BLOCK_INCONSISTENT</code>	Fee block is inconsistent (contains bogus data)
<code>FEE_BLOCK_HEADER_BLANK</code>	Fee block header is blank,it is used to mark the end of Fee block header list when parsing the memory at initialization
<code>FEE_BLOCK_INCONSISTENT_COPY</code>	FEE data read error during swap (ie data area was allocated)
<code>FEE_BLOCK_NEVER_WRITTEN</code>	FEE block was never written in data flash

Definition at line 116 of file `Fee_InternalTypes.h`.

5.19.3.2 Fee_ClusterStatusType

```
enum Fee_ClusterStatusType
```

Status of Fee cluster header .

Enumerator

FEE_CLUSTER_VALID	Fee cluster is valid
FEE_CLUSTER_INVALID	Fee cluster is invalid
FEE_CLUSTER_INCONSISTENT	Fee cluster is inconsistent (contains bogus data)

Definition at line 59 of file Fee_InternalTypes.h.

5.19.3.3 Fee_JobType

enum [Fee_JobType](#)

Type of job currently executed by Fee_MainFunction .

Enumerator

FEE_JOB_READ	Read Fee block data from flash
FEE_JOB_READ_DONE	finish Read Fee block data from flash
FEE_JOB_WRITE	Write Fee block header to flash
FEE_JOB_WRITE_DATA	Write Fee block data to flash
FEE_JOB_WRITE_UNALIGNED	Write unaligned rest of Fee block data to flash
FEE_JOB_WRITE_VALIDATE	Validate Fee block by writing validation flag to flash
FEE_JOB_WRITE_DONE	Finalize validation of Fee block
FEE_JOB_INVALID_BLOCK	
FEE_JOB_INVALID_BLOCK_DONE	Finalize invalidation of Fee block
FEE_JOB_ERASE_IMMEDIATE	
FEE_JOB_ERASE_IMMEDIATE_DONE	Finalize erase (pre-allocation) of Fee block
FEE_JOB_INT_SCAN	
FEE_JOB_INT_SCAN_CLR_HDR_PARSE	Parse Fee cluster header
FEE_JOB_INT_SCAN_CLR	Scan active cluster of current cluster group
FEE_JOB_INT_SCAN_CLR_FMT	Format first Fee cluster
FEE_JOB_INT_SCAN_CLR_FMT_DONE	Finalize format of first Fee cluster
FEE_JOB_INT_SCAN_BLOCK_HDR_PARSE	Parse Fee block header
FEE_JOB_INT_SWAP_BLOCK	Copy next block from source to target cluster
FEE_JOB_INT_SWAP_CLR_FMT	Format current Fee cluster in current Fee cluster group
FEE_JOB_INT_SWAP_DATA_READ	Read data from source cluster to internal Fee buffer
FEE_JOB_INT_SWAP_DATA_WRITE	Write data from internal Fee buffer to target cluster
FEE_JOB_INT_SWAP_CLR_VLD_DONE	Finalize cluster validation
FEE_JOB_DONE	No more subsequent jobs to schedule
FEE_JOB_SETMODE	

Definition at line 70 of file Fee_InternalTypes.h.

5.20 Fls.h File Reference

This file provides mcal fls api.

```
#include "Fls_Ipw.h"
```

Functions

- void [Fls_Init](#) (const [Fls_ConfigType](#) *ConfigPtr)
Initializes the Flash Driver.
- Std_ReturnType [Fls_Erase](#) ([Fls_AddressType](#) TargetAddress, [Fls_LengthType](#) Length)
Starts an erase job asynchronously. The actual job is performed by the Fls_MainFunction.
- Std_ReturnType [Fls_Write](#) ([Fls_AddressType](#) TargetAddress, const uint8 *SourceAddressPtr, [Fls_LengthType](#) Length)
Starts a write job asynchronously. The actual job is performed by the Fls_MainFunction.
- void [Fls_Cancel](#) (void)
Abort a running job synchronously so that directly after returning from this function a new job can be started.
- MemIf_StatusType [Fls_GetStatus](#) (void)
Returns the FLS module status synchronously.
- MemIf_JobResultType [Fls_GetJobResult](#) (void)
Returns synchronously the result of the last job.
- Std_ReturnType [Fls_Read](#) ([Fls_AddressType](#) SourceAddress, uint8 *TargetAddressPtr, [Fls_LengthType](#) Length)
Reads from flash memory.
- Std_ReturnType [Fls_Compare](#) ([Fls_AddressType](#) SourceAddress, const uint8 *TargetAddressPtr, [Fls_LengthType](#) Length)
Starts a compare job asynchronously. The actual job is performed by the Fls_MainFunction.
- void [Fls_SetMode](#) (MemIf_ModeType Mode)
Reads from flash memory.
- void [Fls_GetVersionInfo](#) (Std_VersionInfoType *VersionInfoPtr)
Returns the version information of this module.
- Std_ReturnType [Fls_BlankCheck](#) ([Fls_AddressType](#) TargetAddress, [Fls_LengthType](#) Length)
Starts a Blank check job asynchronously. The actual job is performed by the Fls_MainFunction.

5.20.1 Detailed Description

This file provides mcal fls api.

5.20.2 Function Documentation

5.20.2.1 Fls_BlankCheck()

```
Std_ReturnType Fls_BlankCheck (
    Fls\_AddressType TargetAddress,
    Fls\_LengthType Length )
```

Starts a Blank check job asynchronously. The actual job is performed by the Fls_MainFunction.

Note

Function ID:[DES_FLS_API_011]

Service ID: 0x0a [in] TargetAddress: Address in flash memory from which the blank check Length: Number of bytes to be checked for erase pattern

Parameters

in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

5.20.2.2 Fls_Cancel()

```
void Fls_Cancel (
    void )
```

Abort a running job synchronously so that directly after returning from this function a new job can be started.

Note

Function ID:[DES_FLS_API_004]
Service ID: 0x03

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.20.2.3 Fls_Compare()

```
Std_ReturnType Fls_Compare (
    Fls_AddressType SourceAddress,
    const uint8 * TargetAddressPtr,
    Fls_LengthType Length )
```

Starts a compare job asynchronously. The actual job is performed by the Fls_MainFunction.

Note

Function ID:[DES_FLS_API_008]
Service ID: 0x08 [in] SourceAddress: Source address in flash memory TargetAddressPtr: Pointer to target data buffer
Length: Number of bytes to read

Parameters

in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

5.20.2.4 Fls_Erase()

```
Std_ReturnType Fls_Erase (
    Fls_AddressType TargetAddress,
    Fls_LengthType Length )
```

Starts an erase job asynchronously. The actual job is performed by the Fls_MainFunction.

Note

Function ID:[DES_FLS_API_002]

Service ID: 0x01 [in] TargetAddress: Target address in flash memory Length: Number of bytes to erase

Parameters

in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

5.20.2.5 Fls_GetJobResult()

```
MemIf_JobResultType Fls_GetJobResult (
    void )
```

Returns synchronously the result of the last job.

Note

Function ID:[DES_FLS_API_006]

Service ID: 0x05

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

MemIf_StatusType

5.20.2.6 Fls_GetStatus()

```
MemIf_StatusType Fls_GetStatus (
    void )
```

Returns the FLS module status synchronously.

Note

Function ID:[DES_FLS_API_005]
Service ID: 0x04

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

MemIf_StatusType

5.20.2.7 Fls_GetVersionInfo()

```
void Fls_GetVersionInfo (
    Std_VersionInfoType * VersionInfoPtr )
```

Returns the version information of this module.

Note

Function ID:[DES_FLS_API_010]
Service ID: 0x10 [in] VersionInfoPtr: Version Info ptr

Parameters

in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.20.2.8 Fls_Init()

```
void Fls_Init (
    const Fls_ConfigType * ConfigPtr )
```

Initializes the Flash Driver.

Note

Function ID:[DES_FLS_API_001]

Service ID: 0x00 [in] ConfigPtr: Pointer to flash driver configuration set

Parameters

in, out	None	
out	None	

Returns

void

5.20.2.9 Fls_Read()

```
Std_ReturnType Fls_Read (
    Fls_AddressType SourceAddress,
    uint8 * TargetAddressPtr,
    Fls_LengthType Length )
```

Reads from flash memory.

Note

Function ID:[DES_FLS_API_007]

Service ID: 0x07 [in] SourceAddress: Source address in flash memory TargetAddressPtr: Pointer to target data buffer

Length: Number of bytes to read

Parameters

in, out	None	
out	None	

Returns

Std_ReturnType

5.20.2.10 Fls_SetMode()

```
void Fls_SetMode (
    MemIf_ModeType Mode )
```

Reads from flash memory.

Note

Function ID:[DES_FLS_API_009]

Service ID: 0x09 [in] Mode: MEMIF_MODE_FAST or MEMIF_MODE_SLOW

Parameters

in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.20.2.11 Fls_Write()

```
Std_ReturnType Fls_Write (
    Fls_AddressType TargetAddress,
    const uint8 * SourceAddressPtr,
    Fls_LengthType Length )
```

Starts a write job asynchronously. The actual job is performed by the Fls_MainFunction.

Note

Function ID:[DES_FLS_API_003]

Service ID: 0x02 [in] TargetAddress: Target address in flash memory SourceAddressPtr Pointer to source data buffer

Length: Number of bytes to write

Parameters

in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

5.21 Fls_GeneralTypes.h File Reference

```
#include "Mcal.h"
#include "MemIf_Types.h"
```

Classes

- struct [Fls_ConfigType](#)

Macros

- `#define FLS_INSTANCE_ID ((uint8)0U)`
- `#define FLS_E_PARAM_CONFIG ((uint8)0x01U)`
- `#define FLS_E_PARAM_ADDRESS ((uint8)0x02U)`
- `#define FLS_E_PARAM_LENGTH ((uint8)0x03U)`
- `#define FLS_E_PARAM_DATA ((uint8)0x04U)`
- `#define FLS_E_UNINIT ((uint8)0x05U)`
- `#define FLS_E_BUSY ((uint8)0x06U)`
- `#define FLS_E_VERIFY_ERASE_FAILED ((uint8)0x07U)`
- `#define FLS_E_VERIFY_WRITE_FAILED ((uint8)0x08U)`
- `#define FLS_E_TIMEOUT ((uint8)0x09U)`
- `#define FLS_E_PARAM_POINTER ((uint8)0x0AU)`
- `#define FLS_E_ERASE_FAILED ((uint8)0x01U)`
- `#define FLS_E_WRITE_FAILED ((uint8)0x02U)`
- `#define FLS_E_READ_FAILED ((uint8)0x03U)`
- `#define FLS_E_COMPARE_FAILED ((uint8)0x04U)`
- `#define FLS_E_UNEXPECTED_FLASH_ID ((uint8)0x05U)`
- `#define FLS_INIT_ID ((uint8)0x00U)`
- `#define FLS_ERASE_ID ((uint8)0x01U)`
- `#define FLS_WRITE_ID ((uint8)0x02U)`
- `#define FLS_CANCEL_ID ((uint8)0x03U)`
- `#define FLS_GETSTATUS_ID ((uint8)0x04U)`
- `#define FLS_GETJOBRESULT_ID ((uint8)0x05U)`
- `#define FLS_MAINFUNCTION_ID ((uint8)0x06U)`
- `#define FLS_READ_ID ((uint8)0x07U)`
- `#define FLS_COMPARE_ID ((uint8)0x08U)`
- `#define FLS_SETMODE_ID ((uint8)0x09U)`
- `#define FLS_GETVERSIONINFO_ID ((uint8)0x0AU)`
- `#define FLS_BLANK_CHECK_ID ((uint8)0x0BU)`
- `#define FLS_SECTOR_ERASE_ASYNC (0x01U)`
- `#define FLS_PAGE_WRITE_ASYNC (0x02U)`

Typedefs

- `typedef uint32 Fls_AddressType`
- `typedef uint32 Fls_LengthType`
- `typedef void(* Fls_AcCallbackPtrType) (void)`
- `typedef void(* Fls_JobEndNotificationPtrType) (void)`
- `typedef void(* Fls_JobErrorNotificationPtrType) (void)`
- `typedef void(* Fls_AcErasePtrType) (Fls_AcCallbackPtrType PCallBackPtr)`
- `typedef void(* Fls_AcWritePtrType) (Fls_AcCallbackPtrType PCallBackPtr)`

Enumerations

- `enum Fls_JobType {`
`FLS_JOB_ERASE = 0x00U, FLS_JOB_WRITE, FLS_JOB_READ, FLS_JOB_COMPARE,`
`FLS_JOB_BLANK_CHECK }`
- `enum Fls_PDType { P_FLASH_FLAG = 0x00U, D_FLASH_FLAG }`

5.21.1 Macro Definition Documentation

5.21.1.1 FLS_BLANK_CHECK_ID

```
#define FLS_BLANK_CHECK_ID ((uint8)0x0AU)
```

Definition at line 96 of file Fls_GeneralTypes.h.

5.21.1.2 FLS_CANCEL_ID

```
#define FLS_CANCEL_ID ((uint8)0x03U)
```

Definition at line 88 of file Fls_GeneralTypes.h.

5.21.1.3 FLS_COMPARE_ID

```
#define FLS_COMPARE_ID ((uint8)0x08U)
```

Definition at line 93 of file Fls_GeneralTypes.h.

5.21.1.4 FLS_E_BUSY

```
#define FLS_E_BUSY ((uint8)0x06U)
```

Definition at line 64 of file Fls_GeneralTypes.h.

5.21.1.5 FLS_E_COMPARE_FAILED

```
#define FLS_E_COMPARE_FAILED ((uint8)0x04U)
```

Definition at line 81 of file Fls_GeneralTypes.h.

5.21.1.6 FLS_E_ERASE_FAILED

```
#define FLS_E_ERASE_FAILED ((uint8)0x01U)
```

Definition at line 75 of file Fls_GeneralTypes.h.

5.21.1.7 FLS_E_PARAM_ADDRESS

```
#define FLS_E_PARAM_ADDRESS ((uint8)0x02U)
```

Definition at line 55 of file Fls_GeneralTypes.h.

5.21.1.8 FLS_E_PARAM_CONFIG

```
#define FLS_E_PARAM_CONFIG ((uint8)0x01U)
```

Definition at line 53 of file Fls_GeneralTypes.h.

5.21.1.9 FLS_E_PARAM_DATA

```
#define FLS_E_PARAM_DATA ((uint8)0x04U)
```

Definition at line 59 of file Fls_GeneralTypes.h.

5.21.1.10 FLS_E_PARAM_LENGTH

```
#define FLS_E_PARAM_LENGTH ((uint8)0x03U)
```

Definition at line 57 of file Fls_GeneralTypes.h.

5.21.1.11 FLS_E_PARAM_POINTER

```
#define FLS_E_PARAM_POINTER ((uint8)0x0AU)
```

Definition at line 72 of file Fls_GeneralTypes.h.

5.21.1.12 FLS_E_READ_FAILED

```
#define FLS_E_READ_FAILED ((uint8)0x03U)
```

Definition at line 79 of file Fls_GeneralTypes.h.

5.21.1.13 FLS_E_TIMEOUT

```
#define FLS_E_TIMEOUT ((uint8)0x09U)
```

Definition at line 70 of file Fls_GeneralTypes.h.

5.21.1.14 FLS_E_UNEXPECTED_FLASH_ID

```
#define FLS_E_UNEXPECTED_FLASH_ID ((uint8)0x05U)
```

Definition at line 83 of file Fls_GeneralTypes.h.

5.21.1.15 FLS_E_UNINIT

```
#define FLS_E_UNINIT ((uint8)0x05U)
```

Definition at line 61 of file Fls_GeneralTypes.h.

5.21.1.16 FLS_E_VERIFY_ERASE_FAILED

```
#define FLS_E_VERIFY_ERASE_FAILED ((uint8)0x07U)
```

Definition at line 66 of file Fls_GeneralTypes.h.

5.21.1.17 FLS_E_VERIFY_WRITE_FAILED

```
#define FLS_E_VERIFY_WRITE_FAILED ((uint8)0x08U)
```

Definition at line 68 of file Fls_GeneralTypes.h.

5.21.1.18 FLS_E_WRITE_FAILED

```
#define FLS_E_WRITE_FAILED ((uint8)0x02U)
```

Definition at line 77 of file Fls_GeneralTypes.h.

5.21.1.19 FLS_ERASE_ID

```
#define FLS_ERASE_ID ((uint8)0x01U)
```

Definition at line 86 of file Fls_GeneralTypes.h.

5.21.1.20 FLS_GETJOBRESULT_ID

```
#define FLS_GETJOBRESULT_ID ((uint8)0x05U)
```

Definition at line 90 of file Fls_GeneralTypes.h.

5.21.1.21 FLS_GETSTATUS_ID

```
#define FLS_GETSTATUS_ID ((uint8)0x04U)
```

Definition at line 89 of file Fls_GeneralTypes.h.

5.21.1.22 FLS_GETVERSIONINFO_ID

```
#define FLS_GETVERSIONINFO_ID ((uint8)0x10U)
```

Definition at line 95 of file Fls_GeneralTypes.h.

5.21.1.23 FLS_INIT_ID

```
#define FLS_INIT_ID ((uint8)0x00U)
```

Definition at line 85 of file Fls_GeneralTypes.h.

5.21.1.24 FLS_INSTANCE_ID

```
#define FLS_INSTANCE_ID ((uint8)0U)
```

Definition at line 50 of file Fls_GeneralTypes.h.

5.21.1.25 FLS_MAINFUNCTION_ID

```
#define FLS_MAINFUNCTION_ID ((uint8)0x06U)
```

Definition at line 91 of file Fls_GeneralTypes.h.

5.21.1.26 FLS_PAGE_WRITE_ASYNC

```
#define FLS_PAGE_WRITE_ASYNC (0x02U)
```

Definition at line 99 of file Fls_GeneralTypes.h.

5.21.1.27 FLS_READ_ID

```
#define FLS_READ_ID ((uint8)0x07U)
```

Definition at line 92 of file Fls_GeneralTypes.h.

5.21.1.28 FLS_SECTOR_ERASE_ASYNC

```
#define FLS_SECTOR_ERASE_ASYNC (0x01U)
```

Definition at line 98 of file Fls_GeneralTypes.h.

5.21.1.29 FLS_SETMODE_ID

```
#define FLS_SETMODE_ID ((uint8)0x09U)
```

Definition at line 94 of file Fls_GeneralTypes.h.

5.21.1.30 FLS_WRITE_ID

```
#define FLS_WRITE_ID ((uint8)0x02U)
```

Definition at line 87 of file Fls_GeneralTypes.h.

5.21.2 Typedef Documentation

5.21.2.1 Fls_AcCallbackPtrType

```
typedef void(* Fls_AcCallbackPtrType) (void)
```

Definition at line 133 of file Fls_GeneralTypes.h.

5.21.2.2 Fls_AcErasePtrType

```
typedef void(* Fls_AcErasePtrType) (Fls_AcCallbackPtrType PCallBackPtr)
```

Definition at line 142 of file Fls_GeneralTypes.h.

5.21.2.3 Fls_AcWritePtrType

```
typedef void(* Fls_AcWritePtrType) (Fls_AcCallbackPtrType PCallBackPtr)
```

Definition at line 145 of file Fls_GeneralTypes.h.

5.21.2.4 Fls_AddressType

```
typedef uint32 Fls_AddressType
```

Definition at line 103 of file Fls_GeneralTypes.h.

5.21.2.5 Fls_JobEndNotificationPtrType

```
typedef void(* Fls_JobEndNotificationPtrType) (void)
```

Definition at line 136 of file Fls_GeneralTypes.h.

5.21.2.6 Fls_JobErrorNotificationPtrType

```
typedef void(* Fls_JobErrorNotificationPtrType) (void)
```

Definition at line 139 of file Fls_GeneralTypes.h.

5.21.2.7 Fls_LengthType

```
typedef uint32 Fls_LengthType
```

Definition at line 106 of file Fls_GeneralTypes.h.

5.21.3 Enumeration Type Documentation

5.21.3.1 Fls_JobType

```
enum Fls_JobType
```

Enumerator

FLS_JOB_ERASE	
FLS_JOB_WRITE	
FLS_JOB_READ	
FLS_JOB_COMPARE	
FLS_JOB_BLANK_CHECK	

Definition at line 109 of file Fls_GeneralTypes.h.

5.21.3.2 Fls_PDType

```
enum Fls_PDType
```

Enumerator

P_FLASH_FLAG	
D_FLASH_FLAG	

Definition at line 124 of file Fls_GeneralTypes.h.

5.22 FlsTst.h File Reference

This file is header of MCAL FlsTst.

```
#include "FlsTst_Cfg.h"
```

Classes

- struct [FlsTst_TestResultBgndType](#)
Test result of Flash test in background mode.
- struct [FlsTst_TestSignatureBgndType](#)
Test result signature of Flash test in background mode.
- struct [FlsTst_TestSignatureFgndType](#)
Test result signature of Flash test in foreground mode.
- struct [FlsTst_ErrorDetailsType](#)
Test error information.
- struct [FlsTst_BlockConfigType](#)
Configuration data for a Flash block.
- struct [FlsTst_ConfigType](#)
Initialization data for the Flash Test.
- struct [FlsTst_CommonVariableType](#)
Initialization data for the Flash Test.

Macros

- #define **FLSTST_INIT_SVCID** ((uint8)0x00U)
Service ID of function: *FlsTst_Init*.
- #define **FLSTST_DEINIT_SVCID** ((uint8)0x01U)
Service ID of function: *FlsTst_DeInit*.
- #define **FLSTST_START_FGND_SVCID** ((uint8)0x02U)
Service ID of function: *FlsTst_StartFgnd*.
- #define **FLSTST_ABORT_SVCID** ((uint8)0x03U)
Service ID of function: *FlsTst_Abort*.
- #define **FLSTST_SUSPEND_SVCID** ((uint8)0x04U)
Service ID of function: *FlsTst_Suspend*.
- #define **FLSTST_RESUME_SVCID** ((uint8)0x05U)
Service ID of function: *FlsTst_Resume*.
- #define **FLSTST_GET_CURRENT_STATE_SVCID** ((uint8)0x06U)
Service ID of function: *FlsTst_GetCurrentState*.
- #define **FLSTST_GET_TEST_RESULT_BGND_SVCID** ((uint8)0x07U)
Service ID of function: *FlsTst_GetTestResultBgnd*.
- #define **FLSTST_GET_VERSION_INFO_SVCID** ((uint8)0x08U)
Service ID of function: *FlsTst_GetVersionInfo*.
- #define **FLSTST_GET_TEST_SIGNATURE_BGND_SVCID** ((uint8)0x09U)
Service ID of function: *FlsTst_GetTestSignatureBgnd*.
- #define **FLSTST_GET_TEST_SIGNATURE_FGND_SVCID** ((uint8)0x0AU)
Service ID of function: *FlsTst_GetTestSignatureFgnd*.
- #define **FLSTST_GET_ERROR_DETAILS_SVCID** ((uint8)0x0BU)
Service ID of function: *FlsTst_GetErrorDetails*.
- #define **FLSTST_TEST_ECC_SVCID** ((uint8)0x0CU)
Service ID of function: *FlsTst_TestEcc*.
- #define **FLSTST_MAIN_FUNCTION_SVCID** ((uint8)0x0DU)
Service ID of function: *FlsTst_MainFunction*.
- #define **FLSTST_TEST_COMPLETED_NOTIFICATION_SVCID** ((uint8)0x0EU)
Service ID of function: *FlsTst_TestCompletedNotification*.
- #define **FLSTST_GET_TEST_RESULT_FGND_SVCID** ((uint8)0x0FU)
Service ID of function: *FlsTst_GetTestResultFgnd*.
- #define **FLSTST_E_STATE_FAILURE** ((uint8)0x01U)
Development error: Failure within *FlsTst* execution state.
- #define **FLSTST_E_PARAM_INVALID** ((uint8)0x02U)
Development error: API parameter out of specified range.
- #define **FLSTST_E_UNINIT** ((uint8)0x03U)
Development error: API service used without module initialization.
- #define **FLSTST_E_ALREADY_INITIALIZED** ((uint8)0x04U)
Development error: *FlsTst* module is already initialized.
- #define **FLSTST_E_INIT_FAILED** ((uint8)0x05U)
Development error: For Variant PB, Configuration pointer is a NULL pointer; For Variant PC, Configuration pointer is NOT a NULL pointer.
- #define **FLSTST_E_PARAM_POINTER** ((uint8)0x06U)
Development error: Pointer is a NULL pointer.
- #define **FLSTST_E_FLSTST_FAILURE** ((uint8)0xF0U)
DEM event ID when test fail.

Typedefs

- typedef void(* [FlsTst_NotificationType](#)) (void)
Define notification type.
- typedef uint32 [FlsTst_BlockIdFgndType](#)
Identification for a Flash block to be tested in foreground mode.

Enumerations

- enum [FlsTst_StateType](#) {
 [FLSTST_UNINIT](#) = 0x00U, [FLSTST_INIT](#) = 0x01U, [FLSTST_RUNNING](#) = 0x02U, [FLSTST_ABORTED](#) = 0x03U,
 [FLSTST_SUSPENDED](#) = 0x04U }
Current status of the FlsTst execution state.
- enum [FlsTst_TestResultFgndType](#) { [FLSTST_NOT_TESTED](#) = 0x00U, [FLSTST_OK](#) = 0x01U, [FLSTST_NOT_OK](#) = 0x02U }
Test result status of foreground flash test.
- enum [FlsTst_TestResultType](#) { [FLSTST_RESULT_NOT_TESTED](#) = 0x00U, [FLSTST_RESULT_OK](#) = 0x01U, [FLSTST_RESULT_NOT_OK](#) = 0x02U }
Test result status of flash test.
- enum [FlsTst_TestAlgorithmType](#) {
 [FLSTST_8BIT_CRC](#) = 0x00U, [FLSTST_16BIT_CRC](#) = 0x01U, [FLSTST_32BIT_CRC](#) = 0x02U, [FLSTST_CHECKSUM](#) = 0x03U,
 [FLSTST_DUPLICATED_MEMORY](#) = 0x04U, [FLSTST_ECC](#) = 0x05U }
Test algorithm used for flash test.

Functions

- void [FlsTst_Init](#) (const [FlsTst_ConfigType](#) *ConfigPtr)
Service for Flash Test initialization.
- void [FlsTst_DeInit](#) (void)
Service for Flash Test De-Initialization.
- Std_ReturnType [FlsTst_StartFgnd](#) ([FlsTst_BlockIdFgndType](#) FgndBlockId)
Service for executing foreground Flash Test.
- void [FlsTst_Abort](#) (void)
Service for aborting the Flash Test.
- void [FlsTst_Suspend](#) (void)
Service for suspending current operation of the Flash Test, until FlsTst_Resume is called.
- void [FlsTst_Resume](#) (void)
Service for continuing the Flash Test at the point it was suspended.
- [FlsTst_StateType](#) [FlsTst_GetCurrentState](#) (void)
Service returns the current Flash Test execution state.
- [FlsTst_TestResultBgndType](#) [FlsTst_GetTestResultBgnd](#) (void)
Service returns the background Flash Test result.
- [FlsTst_TestResultFgndType](#) [FlsTst_GetTestResultFgnd](#) (void)
Service returns the foreground Flash Test result.
- void [FlsTst_GetVersionInfo](#) (Std_VersionInfoType *versioninfo)
Service returns the version information of Flash Test.
- [FlsTst_TestSignatureBgndType](#) [FlsTst_GetTestSignatureBgnd](#) (void)
Service returns the Flash Test result in background mode.
- [FlsTst_TestSignatureFgndType](#) [FlsTst_GetTestSignatureFgnd](#) (void)
Service returns the Flash Test result in foreground mode.
- [FlsTst_ErrorDetailsType](#) [FlsTst_GetErrorDetails](#) (void)

Service returns error details monitored from the Flash module.

- Std_ReturnType [FlsTst_TestEcc](#) (void)

Service executes a test of ECC hardware.

- void [FlsTst_MainFunction](#) (void)

Service for executing the Flash Test in background mode.

- void [FlsTst_TestCompletedNotification](#) (void)

The function shall be called every time when a complete test cycle had been tested.

Variables

- [FlsTst_CommonVariableType](#) [FlsTst_GlobalCommonVar](#)
- const [FlsTst_ConfigType](#) [FlsTst_GenerateConfigPC](#)

5.22.1 Detailed Description

This file is header of MCAL FlsTst.

5.22.2 Macro Definition Documentation

5.22.2.1 FLSTST_ABORT_SVCID

```
#define FLSTST_ABORT_SVCID ((uint8) 0x03U)
```

Service ID of function: [FlsTst_Abort](#).

Definition at line 59 of file [FlsTst.h](#).

5.22.2.2 FLSTST_DEINIT_SVCID

```
#define FLSTST_DEINIT_SVCID ((uint8) 0x01U)
```

Service ID of function: [FlsTst_DeInit](#).

Definition at line 55 of file [FlsTst.h](#).

5.22.2.3 FLSTST_E_ALREADY_INITIALIZED

```
#define FLSTST_E_ALREADY_INITIALIZED ((uint8) 0x04U)
```

Development error: [FlsTst](#) module is already initialized.

Definition at line 92 of file [FlsTst.h](#).

5.22.2.4 FLSTST_E_FLSTST_FAILURE

```
#define FLSTST_E_FLSTST_FAILURE ((uint8)0xF0U)
```

DEM event ID when test fail.

Definition at line 100 of file FlsTst.h.

5.22.2.5 FLSTST_E_INIT_FAILED

```
#define FLSTST_E_INIT_FAILED ((uint8)0x05U)
```

Development error: For Variant PB, Configuration pointer is a NULL pointer; For Variant PC, Configuration pointer is NOT a NULL pointer.

Definition at line 95 of file FlsTst.h.

5.22.2.6 FLSTST_E_PARAM_INVALID

```
#define FLSTST_E_PARAM_INVALID ((uint8)0x02U)
```

Development error: API parameter out of specified range.

Definition at line 88 of file FlsTst.h.

5.22.2.7 FLSTST_E_PARAM_POINTER

```
#define FLSTST_E_PARAM_POINTER ((uint8)0x06U)
```

Development error: Pointer is a NULL pointer.

Definition at line 97 of file FlsTst.h.

5.22.2.8 FLSTST_E_STATE_FAILURE

```
#define FLSTST_E_STATE_FAILURE ((uint8)0x01U)
```

Development error: Failure within FlsTst execution state.

Definition at line 86 of file FlsTst.h.

5.22.2.9 FLSTST_E_UNINIT

```
#define FLSTST_E_UNINIT ((uint8)0x03U)
```

Development error: API service used without module initialization.

Definition at line 90 of file FlsTst.h.

5.22.2.10 FLSTST_GET_CURRENT_STATE_SVCID

```
#define FLSTST_GET_CURRENT_STATE_SVCID ((uint8)0x06U)
```

Service ID of function: FlsTst_GetCurrentState.

Definition at line 65 of file FlsTst.h.

5.22.2.11 FLSTST_GET_ERROR_DETAILS_SVCID

```
#define FLSTST_GET_ERROR_DETAILS_SVCID ((uint8)0x0BU)
```

Service ID of function: FlsTst_GetErrorDetails.

Definition at line 75 of file FlsTst.h.

5.22.2.12 FLSTST_GET_TEST_RESULT_BGND_SVCID

```
#define FLSTST_GET_TEST_RESULT_BGND_SVCID ((uint8)0x07U)
```

Service ID of function: FlsTst_GetTestResultBgnd.

Definition at line 67 of file FlsTst.h.

5.22.2.13 FLSTST_GET_TEST_RESULT_FGND_SVCID

```
#define FLSTST_GET_TEST_RESULT_FGND_SVCID ((uint8)0x0FU)
```

Service ID of function: FlsTst_GetTestResultFgnd.

Definition at line 83 of file FlsTst.h.

5.22.2.14 FLSTST_GET_TEST_SIGNATURE_BGND_SVCID

```
#define FLSTST_GET_TEST_SIGNATURE_BGND_SVCID ((uint8) 0x09U)
```

Service ID of function: FlsTst_GetTestSignatureBgnd.

Definition at line 71 of file FlsTst.h.

5.22.2.15 FLSTST_GET_TEST_SIGNATURE_FGND_SVCID

```
#define FLSTST_GET_TEST_SIGNATURE_FGND_SVCID ((uint8) 0x0AU)
```

Service ID of function: FlsTst_GetTestSignatureFgnd.

Definition at line 73 of file FlsTst.h.

5.22.2.16 FLSTST_GET_VERSION_INFO_SVCID

```
#define FLSTST_GET_VERSION_INFO_SVCID ((uint8) 0x08U)
```

Service ID of function: FlsTst_GetVersionInfo.

Definition at line 69 of file FlsTst.h.

5.22.2.17 FLSTST_INIT_SVCID

```
#define FLSTST_INIT_SVCID ((uint8) 0x00U)
```

Service ID of function: FlsTst_Init.

Definition at line 53 of file FlsTst.h.

5.22.2.18 FLSTST_MAIN_FUNCTION_SVCID

```
#define FLSTST_MAIN_FUNCTION_SVCID ((uint8) 0x0DU)
```

Service ID of function: FlsTst_MainFunction.

Definition at line 79 of file FlsTst.h.

5.22.2.19 FLSTST_RESUME_SVCID

```
#define FLSTST_RESUME_SVCID ((uint8)0x05U)
```

Service ID of function: FlsTst_Resume.

Definition at line 63 of file FlsTst.h.

5.22.2.20 FLSTST_START_FGND_SVCID

```
#define FLSTST_START_FGND_SVCID ((uint8)0x02U)
```

Service ID of function: FlsTst_StartFgnd.

Definition at line 57 of file FlsTst.h.

5.22.2.21 FLSTST_SUSPEND_SVCID

```
#define FLSTST_SUSPEND_SVCID ((uint8)0x04U)
```

Service ID of function: FlsTst_Suspend.

Definition at line 61 of file FlsTst.h.

5.22.2.22 FLSTST_TEST_COMPLETED_NOTIFICATION_SVCID

```
#define FLSTST_TEST_COMPLETED_NOTIFICATION_SVCID ((uint8)0x0EU)
```

Service ID of function: FlsTst_TestCompletedNotification.

Definition at line 81 of file FlsTst.h.

5.22.2.23 FLSTST_TEST_ECC_SVCID

```
#define FLSTST_TEST_ECC_SVCID ((uint8)0x0CU)
```

Service ID of function: FlsTst_TestEcc.

Definition at line 77 of file FlsTst.h.

5.22.3 Typedef Documentation

5.22.3.1 FlsTst_BlockIdFgndType

```
typedef uint32 FlsTst_BlockIdFgndType
```

Identification for a Flash block to be tested in foreground mode.

Definition at line 146 of file FlsTst.h.

5.22.3.2 FlsTst_NotificationType

```
typedef void(* FlsTst_NotificationType) (void)
```

Define notification type.

Definition at line 143 of file FlsTst.h.

5.22.4 Enumeration Type Documentation

5.22.4.1 FlsTst_StateType

```
enum FlsTst_StateType
```

Current status of the FlsTst execution state.

Enumerator

FLSTST_UNINIT	
FLSTST_INIT	
FLSTST_RUNNING	
FLSTST_ABORTED	
FLSTST_SUSPENDED	

Definition at line 106 of file FlsTst.h.

5.22.4.2 FlsTst_TestAlgorithmType

```
enum FlsTst_TestAlgorithmType
```

Test algorithm used for flash test.

Enumerator

FLSTST_8BIT_CRC	
FLSTST_16BIT_CRC	

Enumerator

FLSTST_32BIT_CRC	
FLSTST_CHECKSUM	
FLSTST_DUPLICATED_MEMORY	
FLSTST_ECC	

Definition at line 132 of file FlsTst.h.

5.22.4.3 FlsTst_TestResultFgndType

```
enum FlsTst_TestResultFgndType
```

Test result status of foreground flash test.

Enumerator

FLSTST_NOT_TESTED	
FLSTST_OK	
FLSTST_NOT_OK	

Definition at line 116 of file FlsTst.h.

5.22.4.4 FlsTst_TestResultType

```
enum FlsTst_TestResultType
```

Test result status of flash test.

Enumerator

FLSTST_RESULT_NOT_TESTED	
FLSTST_RESULT_OK	
FLSTST_RESULT_NOT_OK	

Definition at line 124 of file FlsTst.h.

5.22.5 Function Documentation**5.22.5.1 FlsTst_Abort()**

```
void FlsTst_Abort (  
    void )
```

Service for aborting the Flash Test.

Note

Function ID: DES_FLSTST_API_004
Service ID: 0x03

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.22.5.2 FlsTst_DeInit()

```
void FlsTst_DeInit (
    void )
```

Service for Flash Test De-Initialization.

Note

Function ID: DES_FLSTST_API_002
Service ID: 0x01

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.22.5.3 FlsTst_GetCurrentState()

```
FlsTst_StateType FlsTst_GetCurrentState (
    void )
```

Service returns the current Flash Test execution state.

Note

Function ID: DES_FLSTST_API_007
Service ID: 0x06

Parameters

in	:None	
in, out	None	
out	None	

Returns

FlsTst_StateType: FLSTST_UNINIT, the Flash Test is not initialized or not usable; FLSTST_INIT, the Flash Test is initialized and ready to be started; FLSTST_RUNNING, the Flash Test is currently running; FLSTST_ABORTED, the Flash Test is aborted; FLSTST_SUSPENDED, the Flash Test is waiting to be resumed or is waiting to start foreground mode test.

5.22.5.4 FlsTst_GetErrorDetails()

```
FlsTst_ErrorDetailsType FlsTst_GetErrorDetails (  
    void )
```

Service returns error details monitored from the Flash module.

Note

Function ID: DES_FLSTST_API_013
Service ID: 0x0B

Parameters

in	None	
in, out	None	
out	None	

Returns

[FlsTst_ErrorDetailsType](#), pointer to store error information monitored in the Flash Test.

5.22.5.5 FlsTst_GetTestResultBgnd()

```
FlsTst_TestResultBgndType FlsTst_GetTestResultBgnd (  
    void )
```

Service returns the background Flash Test result.

Note

Function ID: DES_FLSTST_API_008
Service ID: 0x07

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

[FlsTst_TestResultBgndType](#), pointer to return test result of background Flash Test.

5.22.5.6 FlsTst_GetTestResultFgnd()

```
FlsTst_TestResultFgndType FlsTst_GetTestResultFgnd (  
    void )
```

Service returns the background Flash Test result.

Note

Function ID: DES_FLSTST_API_009
Service ID: 0x0F

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

FlsTst_TestResultFgndType: 0x00: FLSTST_NOT_TESTED, no result available; 0x01: FLSTST_OK, the last Flash Test has been tested with OK result; 0x02: FLSTST_NOT_OK, the last Flash Test has been tested with NOT_OK result.

5.22.5.7 FlsTst_GetTestSignatureBgnd()

```
FlsTst_TestSignatureBgndType FlsTst_GetTestSignatureBgnd (  
    void )
```

Service returns the Flash Test result in backbround mode.

Note

Function ID: DES_FLSTST_API_011
Service ID: 0x09

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

[FlsTst_TestSignatureBgndType](#), pointer to store test result signature of background Flash Test.

5.22.5.8 FlsTst_GetTestSignatureFgnd()

```
FlsTst_TestSignatureFgndType FlsTst_GetTestSignatureFgnd (
    void )
```

Service returns the Flash Test result in forebround mode.

Note

Function ID: DES_FLSTST_API_012
Service ID: 0x0A

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

[FlsTst_TestSignatureFgndType](#), pointer to store test result signature of foreground Flash Test.

5.22.5.9 FlsTst_GetVersionInfo()

```
void FlsTst_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

Service returns the version information of Flash Test.

Note

Function ID: DES_FLSTST_API_010
Service ID: 0x08

Parameters

in	<i>None</i>	
in, out	<i>versioninfo</i>	pointer to where to store the version information of Flash Test.
Generated by Doxygen	<i>None</i>	

Returns

None

5.22.5.10 FlsTst_Init()

```
void FlsTst_Init (
    const FlsTst_ConfigType * ConfigPtr )
```

Service for Flash Test initialization.

Note

Function ID: DES_FLSTST_API_001
Service ID: 0x00

Parameters

in	<i>ConfigPtr</i>	Pointer to Flash Test configuration set.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.22.5.11 FlsTst_MainFunction()

```
void FlsTst_MainFunction (
    void )
```

Service for executing the Flash Test in background mode.

Note

Function ID: DES_FLSTST_API_015
Service ID: 0x0D

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.22.5.12 FlsTst_Resume()

```
void FlsTst_Resume (
    void )
```

Service for continuing the Flash Test at the point it was suspended.

Note

Function ID: DES_FLSTST_API_006
Service ID: 0x05

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.22.5.13 FlsTst_StartFgnd()

```
Std_ReturnType FlsTst_StartFgnd (
    FlsTst_BlockIdFgndType FgndBlockId )
```

Service for executing foreground Flash Test.

Note

Function ID: DES_FLSTST_API_003
Service ID: 0x02

Parameters

in	<i>Fgnd↔ BlockId</i>	Number of the foreground test to be executed.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Foreground test processed; E_NOT_OK: Foreground test not accepted.

5.22.5.14 FlsTst_Suspend()

```
void FlsTst_Suspend (  
    void )
```

Service for suspending current operation of the Flash Test, until FlsTst_Resume is called.

Note

Function ID: DES_FLSTST_API_005
Service ID: 0x04

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.22.5.15 FlsTst_TestCompletedNotification()

```
void FlsTst_TestCompletedNotification (  
    void )
```

The function shall be called every time when a complete test cycle had been tested.

Note

Function ID: DES_FLSTST_API_016
Service ID: 0x0E

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None.

5.22.5.16 FlsTst_TestEcc()

```
Std_ReturnType FlsTst_TestEcc (  
    void )
```

Service executes a test of ECC hardware.

Note

Function ID: DES_FLSTST_API_014
Service ID: 0x0C

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Flash ECC test pass; E_NOT_OK: Flash ECC test fail or not usable.

5.22.6 Variable Documentation**5.22.6.1 FlsTst_GenerateConfigPC**

```
const FlsTst_ConfigType FlsTst_GenerateConfigPC
```

5.22.6.2 FlsTst_GlobalCommonVar

```
FlsTst_CommonVariableType FlsTst_GlobalCommonVar
```

5.23 Gpt.h File Reference

```
#include "Gpt_Ipw_Types.h"
```

Macros

- #define GPT_E_UNINIT (0x0AU)
Development errors, SRS_Gpt_069.
- #define GPT_E_ALREADY_INITIALIZED (0x0DU)
- #define GPT_E_INIT_FAILED (0x0EU)
- #define GPT_E_PARAM_CHANNEL (0x14U)
- #define GPT_E_PARAM_VALUE (0x15U)
- #define GPT_E_PARAM_POINTER (0x16U)
- #define GPT_E_PARAM_PREDEF_TIMER (0x17U)
- #define GPT_E_PARAM_MODE (0x1FU)
- #define GPT_E_BUSY (0x0BU)
Runtime errors, SRS_GPT_070.
- #define GPT_E_MODE (0x0CU)
- #define GPT_MODULE_ID (100U)

- #define `GPT_INSTANCE_ID` (0U)
- #define `GPT_GET_VERSION_INFO_ID` (0x00U)
- #define `GPT_INIT_ID` (0x01U)
- #define `GPT_DEINIT_ID` (0x02U)
- #define `GPT_GET_TIME_ELAPSED_ID` (0x03U)
- #define `GPT_GET_TIME_REMAINING_ID` (0x04U)
- #define `GPT_START_TIMER_ID` (0x05U)
- #define `GPT_STOP_TIMER_ID` (0x06U)
- #define `GPT_ENABLE_NOTIFICATION_ID` (0x07U)
- #define `GPT_DISABLE_NOTIFICATION_ID` (0x08U)
- #define `GPT_SET_MODE_ID` (0x09U)
- #define `GPT_DISABLE_WAKEUP_ID` (0x0AU)
- #define `GPT_ENABLE_WAKEUP_ID` (0x0BU)
- #define `GPT_CHECK_WAKEUP_ID` (0x0CU)
- #define `GPT_GET_PREDEF_TIMER_VALUE_ID` (0x0DU)

Functions

- void `Gpt_GetVersionInfo` (Std_VersionInfoType *VersionInfoPtr)
: Returns the version information of this module.
- void `Gpt_Init` (const Gpt_ConfigType *ConfigPtr)
: Initializes the GPT driver.
- void `Gpt_DeInit` (void)
: Deinitializes the GPT driver.
- Gpt_ValueType `Gpt_GetTimeElapsed` (Gpt_ChannelType Channel)
: Returns the time already elapsed.
- Gpt_ValueType `Gpt_GetTimeRemaining` (Gpt_ChannelType Channel)
: Returns the time remaining until the target time is reached.
- void `Gpt_StartTimer` (Gpt_ChannelType Channel, Gpt_ValueType Value)
: Starts a timer channel.
- void `Gpt_StopTimer` (Gpt_ChannelType Channel)
: Stops a timer channel.
- void `Gpt_EnableNotification` (Gpt_ChannelType Channel)
: Enables the interrupt notification for a channel (relevant in normal mode).
- void `Gpt_DisableNotification` (Gpt_ChannelType Channel)
: Disables the interrupt notification for a channel (relevant in normal mode).
- void `Gpt_SetMode` (Gpt_ModeType Mode)
: Sets the operation mode of the GPT.
- void `Gpt_DisableWakeup` (Gpt_ChannelType Channel)
: Disables the wakeup interrupt of a channel (relevant in sleep mode).
- void `Gpt_EnableWakeup` (Gpt_ChannelType Channel)
: Enables the wakeup interrupt of a channel (relevant in sleep mode).
- void `Gpt_CheckWakeup` (EcuM_WakeupSourceType WakeupSource)
: Checks if a wakeup capable GPT channel is the source for a wakeup event and calls the ECU state manager service `EcuM_SetWakeupEvent` in case of a valid GPT channel wakeup event.
- Std_ReturnType `Gpt_GetPredefTimerValue` (Gpt_PredDefTimerType PredDefTimer, uint32 *TimeValuePtr)
: Delivers the current value of the desired GPT PredDef Timer.

Variables

- const Gpt_ConfigType `Gpt_GenerateConfigPC`

5.23.1 Macro Definition Documentation

5.23.1.1 GPT_CHECK_WAKEUP_ID

```
#define GPT_CHECK_WAKEUP_ID (0x0CU)
```

Definition at line 82 of file Gpt.h.

5.23.1.2 GPT_DEINIT_ID

```
#define GPT_DEINIT_ID (0x02U)
```

Definition at line 72 of file Gpt.h.

5.23.1.3 GPT_DISABLE_NOTIFICATION_ID

```
#define GPT_DISABLE_NOTIFICATION_ID (0x08U)
```

Definition at line 78 of file Gpt.h.

5.23.1.4 GPT_DISABLE_WAKEUP_ID

```
#define GPT_DISABLE_WAKEUP_ID (0x0AU)
```

Definition at line 80 of file Gpt.h.

5.23.1.5 GPT_E_ALREADY_INITIALIZED

```
#define GPT_E_ALREADY_INITIALIZED (0x0DU)
```

Definition at line 53 of file Gpt.h.

5.23.1.6 GPT_E_BUSY

```
#define GPT_E_BUSY (0x0BU)
```

Runtime errors, SRS_GPT_070.

Definition at line 64 of file Gpt.h.

5.23.1.7 GPT_E_INIT_FAILED

```
#define GPT_E_INIT_FAILED (0x0EU)
```

Definition at line 54 of file Gpt.h.

5.23.1.8 GPT_E_MODE

```
#define GPT_E_MODE (0x0CU)
```

Definition at line 65 of file Gpt.h.

5.23.1.9 GPT_E_PARAM_CHANNEL

```
#define GPT_E_PARAM_CHANNEL (0x14U)
```

Definition at line 55 of file Gpt.h.

5.23.1.10 GPT_E_PARAM_MODE

```
#define GPT_E_PARAM_MODE (0x1FU)
```

Definition at line 59 of file Gpt.h.

5.23.1.11 GPT_E_PARAM_POINTER

```
#define GPT_E_PARAM_POINTER (0x16U)
```

Definition at line 57 of file Gpt.h.

5.23.1.12 GPT_E_PARAM_PREDEF_TIMER

```
#define GPT_E_PARAM_PREDEF_TIMER (0x17U)
```

Definition at line 58 of file Gpt.h.

5.23.1.13 GPT_E_PARAM_VALUE

```
#define GPT_E_PARAM_VALUE (0x15U)
```

Definition at line 56 of file Gpt.h.

5.23.1.14 GPT_E_UNINIT

```
#define GPT_E_UNINIT (0x0AU)
```

Development errors, SRS_Gpt_069.

Definition at line 52 of file Gpt.h.

5.23.1.15 GPT_ENABLE_NOTIFICATION_ID

```
#define GPT_ENABLE_NOTIFICATION_ID (0x07U)
```

Definition at line 77 of file Gpt.h.

5.23.1.16 GPT_ENABLE_WAKEUP_ID

```
#define GPT_ENABLE_WAKEUP_ID (0x0BU)
```

Definition at line 81 of file Gpt.h.

5.23.1.17 GPT_GET_PREDEF_TIMER_VALUE_ID

```
#define GPT_GET_PREDEF_TIMER_VALUE_ID (0x0DU)
```

Definition at line 83 of file Gpt.h.

5.23.1.18 GPT_GET_TIME_ELAPSED_ID

```
#define GPT_GET_TIME_ELAPSED_ID (0x03U)
```

Definition at line 73 of file Gpt.h.

5.23.1.19 GPT_GET_TIME_REMAINING_ID

```
#define GPT_GET_TIME_REMAINING_ID (0x04U)
```

Definition at line 74 of file Gpt.h.

5.23.1.20 GPT_GET_VERSION_INFO_ID

```
#define GPT_GET_VERSION_INFO_ID (0x00U)
```

Definition at line 70 of file Gpt.h.

5.23.1.21 GPT_INIT_ID

```
#define GPT_INIT_ID (0x01U)
```

Definition at line 71 of file Gpt.h.

5.23.1.22 GPT_INSTANCE_ID

```
#define GPT_INSTANCE_ID (0U)
```

Definition at line 68 of file Gpt.h.

5.23.1.23 GPT_MODULE_ID

```
#define GPT_MODULE_ID (100U)
```

Definition at line 67 of file Gpt.h.

5.23.1.24 GPT_SET_MODE_ID

```
#define GPT_SET_MODE_ID (0x09U)
```

Definition at line 79 of file Gpt.h.

5.23.1.25 GPT_START_TIMER_ID

```
#define GPT_START_TIMER_ID (0x05U)
```

Definition at line 75 of file Gpt.h.

5.23.1.26 GPT_STOP_TIMER_ID

```
#define GPT_STOP_TIMER_ID (0x06U)
```

Definition at line 76 of file Gpt.h.

5.23.2 Function Documentation

5.23.2.1 Gpt_CheckWakeup()

```
void Gpt_CheckWakeup (
    EcuM_WakeupSourceType WakeupSource )
```

: Checks if a wakeup capable GPT channel is the source for a wakeup event and calls the ECU state manager service EcuM_SetWakeupEvent in case of a valid GPT channel wakeup event.

Note

Function ID: DES_GPT_API_013
Service ID: 0x0d

Parameters

in		
----	--	--

5.23.2.2 Gpt_DeInit()

```
void Gpt_DeInit (
    void )
```

: Deinitializes the GPT driver.

Note

Function ID: DES_GPT_API_003
Service ID: 0x02

Parameters

in		
----	--	--

5.23.2.3 Gpt_DisableNotification()

```
void Gpt_DisableNotification (
    Gpt_ChannelType Channel )
```

: Disables the interrupt notification for a channel (relevant in normal mode).

Note

Function ID: DES_GPT_API_009
Service ID: 0x08

Parameters

in		
----	--	--

5.23.2.4 Gpt_DisableWakeup()

```
void Gpt_DisableWakeup (
    Gpt_ChannelType Channel )
```

: Disables the wakeup interrupt of a channel (relevant in sleep mode).

Note

Function ID: DES_GPT_API_011
Service ID: 0x0a

Parameters

in		
----	--	--

5.23.2.5 Gpt_EnableNotification()

```
void Gpt_EnableNotification (
    Gpt_ChannelType Channel )
```

: Enables the interrupt notification for a channel (relevant in normal mode).

Note

Function ID: DES_GPT_API_008
Service ID: 0x07

Parameters

in		
----	--	--

5.23.2.6 Gpt_EnableWakeup()

```
void Gpt_EnableWakeup (
    Gpt_ChannelType Channel )
```

: Enables the wakeup interrupt of a channel (relevant in sleep mode).

Note

Function ID: DES_GPT_API_012
Service ID: 0x0b

Parameters

in		
----	--	--

5.23.2.7 Gpt_GetPredefTimerValue()

```
Std_ReturnType Gpt_GetPredefTimerValue (
    Gpt_PredefTimerType PredefTimer,
    uint32 * TimeValuePtr )
```

: Delivers the current value of the desired GPT Predef Timer.

Note

Function ID: DES_GPT_API_014
Service ID: none

Parameters

in		
----	--	--

5.23.2.8 Gpt_GetTimeElapsed()

```
Gpt_ValueType Gpt_GetTimeElapsed (
```

```
Gpt_ChannelType Channel )
```

: Returns the time already elapsed.

Note

Function ID: DES_GPT_API_004
Service ID: 0x03

Parameters

in		
----	--	--

5.23.2.9 Gpt_GetTimeRemaining()

```
Gpt_ValueType Gpt_GetTimeRemaining (
    Gpt_ChannelType Channel )
```

: Returns the time remaining until the target time is reached.

Note

Function ID: DES_GPT_API_005
Service ID: 0x04

Parameters

in		
----	--	--

5.23.2.10 Gpt_GetVersionInfo()

```
void Gpt_GetVersionInfo (
    Std_VersionInfoType * VersionInfoPtr )
```

: Returns the version information of this module.

Note

Function ID: DES_GPT_API_001
Service ID: 0x00

Parameters

in		
----	--	--

5.23.2.11 Gpt_Init()

```
void Gpt_Init (
    const Gpt_ConfigType * ConfigPtr )
```

: Initializes the GPT driver.

Note

Function ID: DES_GPT_API_002
Service ID: 0x01

Parameters

in		
----	--	--

5.23.2.12 Gpt_SetMode()

```
void Gpt_SetMode (
    Gpt_ModeType Mode )
```

: Sets the operation mode of the GPT.

Note

Function ID: DES_GPT_API_010
Service ID: 0x09

Parameters

in		
----	--	--

5.23.2.13 Gpt_StartTimer()

```
void Gpt_StartTimer (
    Gpt_ChannelType Channel,
    Gpt_ValueType Value )
```

: Starts a timer channel.

Note

Function ID: DES_GPT_API_006
Service ID: 0x05

Parameters

in		
----	--	--

5.23.2.14 Gpt_StopTimer()

```
void Gpt_StopTimer (
    Gpt_ChannelType Channel )
```

: Stops a timer channel.

Note

Function ID: DES_GPT_API_007
Service ID: 0x06

Parameters

in		
----	--	--

5.23.3 Variable Documentation

5.23.3.1 Gpt_GenerateConfigPC

```
const Gpt_ConfigType Gpt_GenerateConfigPC
```

5.24 Icu.h File Reference

This file provides extern Mcal Icu api.

```
#include "Icu_Cfg.h"
#include "EcuM.h"
```

Classes

- struct [Icu_DutyCycleType](#)
Type which shall contain the values, needed for calculating duty cycles.
- struct [Icu_PwmModuleConfigType](#)
Definition of PWM module configuration.
- struct [Icu_IpConfigType](#)
Definition of ICU hardware module set configuration.
- struct [Icu_ChannelConfigType](#)
Definition of ICU channel configuration.
- struct [Icu_ConfigType](#)
Definition of all ICU configurations.

Macros

- #define [ICU_E_PARAM_POINTER](#) (0x0AU)
Det Error value, API IS called with invalid pointer.
- #define [ICU_E_PARAM_CHANNEL](#) (0x0BU)
Det Error value, API service used with an invalid channel identifier or channel was not configured for the functionality of the calling API.
- #define [ICU_E_PARAM_ACTIVATION](#) (0x0CU)
Det Error value, API service used with an invalid or not feasible activation.
- #define [ICU_E_INIT_FAILED](#) (0x0DU)
Det Error value, Init function failed.
- #define [ICU_E_UNINIT](#) (0x14U)
Det Error value, API service used without module initialization.
- #define [ICU_E_ALREADY_INITIALIZED](#) (0x17U)
Det Error value, API Icu_Init service is called and when the ICU driver and the Hardware are already initialized.
- #define [ICU_E_PARAM_VINFO](#) (0x19U)
Det Error value, API Icu_GetVersionInfo is called and the parameter versioninfo is invalid (e.g. NULL)
- #define [ICU_E_PARAM_BUFFER_SIZE](#) (0x0EU)
Det Error value, API service used with an invalid buffer size.
- #define [ICU_E_PARAM_NOTIFY_INTERVAL](#) (0x18U)
Det Error value, API Icu_StartTimeStamp is called and the parameter NotifyInterval is invalid (e.g."0", NotifyInterval < 1)
- #define [ICU_E_NOT_STARTED](#) (0x15U)
Det Error value, API service Icu_StopTimestamp called on a channel which was not started or already stopped.
- #define [ICU_E_PARAM_MODE](#) (0x0FU)
Det Error value, API service Icu_SetMode used with an invalid mode.
- #define [ICU_INIT_ID](#) (0x00U)
API service ID for ICU init function.
- #define [ICU_DEINIT_ID](#) (0x01U)
API service ID for ICU deinit function.
- #define [ICU_SETMODE_ID](#) (0x02U)
API service ID for ICU set mode function.
- #define [ICU_ENABLEWAKEUP_ID](#) (0x04U)
API service ID for ICU enable wakeup function.
- #define [ICU_DISABLEWAKEUP_ID](#) (0x03U)
API service ID for ICU disable wakeup function.
- #define [ICU_CHECKWAKEUP_ID](#) (0x15U)
API service ID for ICU check wakeup function.
- #define [ICU_SETACTIVATIONCONDITION_ID](#) (0x05U)
API service ID for ICU set activation condition function.
- #define [ICU_DISABLENOTIFICATION_ID](#) (0x06U)
API service ID for ICU disable notification function.
- #define [ICU_ENABLENOTIFICATION_ID](#) (0x07U)
API service ID for ICU enable notification function.
- #define [ICU_GETINPUTSTATE_ID](#) (0x08U)
API service ID for ICU get input state function.
- #define [ICU_RESETEDEGECOUNT_ID](#) (0x0CU)
API service ID for ICU reset edge count function.
- #define [ICU_ENABLEEDGECOUNT_ID](#) (0x0DU)
API service ID for ICU enable edge count function.
- #define [ICU_ENABLEEDGEDETECTION_ID](#) (0x16U)
API service ID for ICU enable edge detect function.
- #define [ICU_DISABLEEDGEDETECTION_ID](#) (0x17U)
API service ID for ICU disable edge detect function.

- #define `ICU_DISABLEEDGECOUNT_ID` (0x0EU)
API service ID for ICU disable edge count function.
- #define `ICU_GETEDGENUMBERS_ID` (0x0FU)
API service ID for ICU get edge number function.
- #define `ICU_GETVERSIONINFO_ID` (0x12U)
API service ID for ICU get version function.
- #define `ICU_STARTTIMESTAMP_ID` (0x09U)
API service ID for ICU start timestamp function.
- #define `ICU_STOPTIMESTAMP_ID` (0x0aU)
API service ID for ICU stop timestamp function.
- #define `ICU_GETTIMESTAMPINDEX_ID` (0x0bU)
API service ID for ICU get timestamp index function.
- #define `ICU_STARTSIGNALMEASUREMENT_ID` (0x13U)
API service ID for ICU start signal measurement function.
- #define `ICU_STOPSIGNALMEASUREMENT_ID` (0x14U)
API service ID for ICU stop signal measurement function.
- #define `ICU_GETTIMEELAPSED_ID` (0x10U)
API service ID for ICU get time elapsed function.
- #define `ICU_GETDUTYCYCLEVALUES_ID` (0x11U)
API service ID for ICU get duty cycle values function.

Typedefs

- typedef uint8 `Icu_ChannelType`
Hardware options for implementing ICU functions.
- typedef uint16 `Icu_ValueType`
Width of the buffer for timestamp ticks and measured elapsed timeticks.
- typedef uint16 `Icu_IndexType`
The index of the buffer storing timestamps.
- typedef uint32 `Icu_EdgeNumberType`
The type of edge count.

Enumerations

- enum `Icu_ModeType` { `ICU_MODE_NORMAL` = 0U, `ICU_MODE_SLEEP` }
ICU power mode options.
- enum `Icu_InputStateType` { `ICU_ACTIVE` = 0U, `ICU_IDLE` }
ICU channel input states options.
- enum `Icu_MeasurementModeType` { `ICU_MODE_SIGNAL_EDGE_DETECT` = 0x01U, `ICU_MODE_SIGNAL_MEASUREMENT` = 0x02U, `ICU_MODE_TIMESTAMP` = 0x04U, `ICU_MODE_EDGE_COUNTER` = 0x08U }
ICU measurement mode options.
- enum `Icu_ActivationType` { `ICU_FALLING_EDGE` = 0U, `ICU_RISING_EDGE`, `ICU_BOTH_EDGES` }
ICU Channel activation options.
- enum `Icu_SignalMeasurementPropertyType` { `ICU_LOW_TIME` = 0U, `ICU_HIGH_TIME`, `ICU_PERIOD_TIME`, `ICU_DUTY_CYCLE` }
Definition of the measurement property type.
- enum `Icu_TimestampBufferType` { `ICU_LINEAR_BUFFER` = 0U, `ICU_CIRCULAR_BUFFER` }
Definition of the timestamp measurement property type.
- enum `Icu_HwSelect` { `ICU_HW_PWM` = 0U, `ICU_HW_PORT` }
Hardware options for implementing ICU functions.

Functions

- void [Icu_Init](#) (const [Icu_ConfigType](#) *ConfigPtr)
: *Icu_Init: service for ICU initialization*
- void [Icu_DeInit](#) (void)
: *Icu_DeInit: Deinitialize icu module.*
- void [Icu_SetMode](#) ([Icu_ModeType](#) Mode)
: *Icu_SetMode: This function sets the ICU mode.*
- void [Icu_EnableWakeup](#) ([Icu_ChannelType](#) Channel)
: *Icu_EnableWakeup: This function (re-)enables the wakeup capability of the given ICU channel.*
- void [Icu_DisableWakeup](#) ([Icu_ChannelType](#) Channel)
: *Icu_DisableWakeup: This function disables the wakeup capability of a single ICU channel.*
- void [Icu_CheckWakeup](#) ([EcuM_WakeupSourceType](#) WakeupSource)
: *Icu_CheckWakeup: Checks if a wakeup capable ICU channel is the source for a wakeup event and calls the ECU state manager service EcuM_SetWakeupEvent in case of a valid ICU channel wakeup event.*
- void [Icu_SetActivationCondition](#) ([Icu_ChannelType](#) Channel, [Icu_ActivationType](#) Activation)
: *Icu_SetActivationCondition: Set Activation Condition.*
- void [Icu_ResetEdgeCount](#) ([Icu_ChannelType](#) Channel)
: *Icu_ResetEdgeCount: Reset Edge Count.*
- void [Icu_EnableEdgeCount](#) ([Icu_ChannelType](#) Channel)
: *Icu_EnableEdgeCount: enable the counting of edges of the given channel.*
- void [Icu_DisableEdgeCount](#) ([Icu_ChannelType](#) Channel)
: *Icu_DisableEdgeCount: disable the counting of edges of the given channel.*
- [Icu_EdgeNumberType](#) [Icu_GetEdgeNumbers](#) ([Icu_ChannelType](#) Channel)
: *Icu_GetEdgeNumbers: reads the number of counted edges*
- void [Icu_EnableEdgeDetection](#) ([Icu_ChannelType](#) Channel)
: *Icu_EnableEdgeDetection: enables / re-enables the detection of edges of the given channel.*
- void [Icu_DisableEdgeDetection](#) ([Icu_ChannelType](#) Channel)
: *Icu_DisableEdgeDetection: disables the detection of edges of the given channel.*
- void [Icu_EnableNotification](#) ([Icu_ChannelType](#) Channel)
: *Icu_EnableNotification: enables the notification on the given channel.*
- void [Icu_DisableNotification](#) ([Icu_ChannelType](#) Channel)
: *Icu_DisableNotification: disables the notification of a channel.*
- [Icu_InputStateType](#) [Icu_GetInputState](#) ([Icu_ChannelType](#) Channel)
: *Icu_GetInputState: returns the status of the ICU input.*
- void [Icu_StartTimestamp](#) ([Icu_ChannelType](#) Channel, [Icu_ValueType](#) *BufferPtr, uint16 BufferSize, uint16 NotifyInterval)
: *Icu_StartTimestamp: This function starts the capturing of timer values on the edges.*
- void [Icu_StopTimestamp](#) ([Icu_ChannelType](#) Channel)
: *Icu_StopTimestamp: This function stops the timestamp measurement of the given channel.*
- [Icu_IndexType](#) [Icu_GetTimestampIndex](#) ([Icu_ChannelType](#) Channel)
: *Icu_GetTimestampIndex: This function reads the timestamp index of the given channel.*
- void [Icu_StartSignalMeasurement](#) ([Icu_ChannelType](#) Channel)
: *Icu_StartSignalMeasurement: This function starts the measurement of signals.*
- void [Icu_StopSignalMeasurement](#) ([Icu_ChannelType](#) Channel)
: *Icu_StopSignalMeasurement: This function stops the measurement of signals of the given channel.*
- [Icu_ValueType](#) [Icu_GetTimeElapsed](#) ([Icu_ChannelType](#) Channel)
: *Icu_GetTimeElapsed: This function reads the elapsed Signal Low Time for the given channel.*
- void [Icu_GetDutyCycleValues](#) ([Icu_ChannelType](#) Channel, [Icu_DutyCycleType](#) *DutyCycleValues)
: *Icu_GetDutyCycleValues: This function reads the coherent active time and period time for the given ICU Channel.*
- void [Icu_GetVersionInfo](#) ([Std_VersionInfoType](#) *versioninfo)
: *Icu_GetVersionInfo: return the version information of this module.*

Variables

- const [Icu_ConfigType](#) [Icu_GenerateConfigPC](#)

5.24.1 Detailed Description

This file provides extern Mcal Icu api.

5.24.2 Macro Definition Documentation

5.24.2.1 ICU_CHECKWAKEUP_ID

```
#define ICU_CHECKWAKEUP_ID (0x15U)
```

API service ID for ICU check wakeup function.

Definition at line 93 of file Icu.h.

5.24.2.2 ICU_DEINIT_ID

```
#define ICU_DEINIT_ID (0x01U)
```

API service ID for ICU deinit function.

Definition at line 85 of file Icu.h.

5.24.2.3 ICU_DISABLEEDGECOUNT_ID

```
#define ICU_DISABLEEDGECOUNT_ID (0x0EU)
```

API service ID for ICU disable edge count function.

Definition at line 111 of file Icu.h.

5.24.2.4 ICU_DISABLEEDGEDETECTION_ID

```
#define ICU_DISABLEEDGEDETECTION_ID (0x17U)
```

API service ID for ICU disable edge detect function.

Definition at line 109 of file Icu.h.

5.24.2.5 ICU_DISABLENOTIFICATION_ID

```
#define ICU_DISABLENOTIFICATION_ID (0x06U)
```

API service ID for ICU disable notification function.

Definition at line 97 of file Icu.h.

5.24.2.6 ICU_DISABLEWAKEUP_ID

```
#define ICU_DISABLEWAKEUP_ID (0x03U)
```

API service ID for ICU disable wakeup function.

Definition at line 91 of file Icu.h.

5.24.2.7 ICU_E_ALREADY_INITIALIZED

```
#define ICU_E_ALREADY_INITIALIZED (0x17U)
```

Det Error value, API Icu_Init service is called and when the ICU driver and the Hardware are already initialized.

Definition at line 68 of file Icu.h.

5.24.2.8 ICU_E_INIT_FAILED

```
#define ICU_E_INIT_FAILED (0x0DU)
```

Det Error value, Init function failed.

Definition at line 63 of file Icu.h.

5.24.2.9 ICU_E_NOT_STARTED

```
#define ICU_E_NOT_STARTED (0x15U)
```

Det Error value, API service Icu_StopTimestamp called on a channel which was not started or already stopped.

Definition at line 78 of file Icu.h.

5.24.2.10 ICU_E_PARAM_ACTIVATION

```
#define ICU_E_PARAM_ACTIVATION (0x0CU)
```

Det Error value, API service used with an invalid or not feasible activation.

Definition at line 61 of file Icu.h.

5.24.2.11 ICU_E_PARAM_BUFFER_SIZE

```
#define ICU_E_PARAM_BUFFER_SIZE (0x0EU)
```

Det Error value, API service used with an invalid buffer size.

Definition at line 73 of file Icu.h.

5.24.2.12 ICU_E_PARAM_CHANNEL

```
#define ICU_E_PARAM_CHANNEL (0x0BU)
```

Det Error value, API service used with an invalid channel identifier or channel was not configured for the functionality of the calling API.

Definition at line 59 of file Icu.h.

5.24.2.13 ICU_E_PARAM_MODE

```
#define ICU_E_PARAM_MODE (0x0FU)
```

Det Error value, API service Icu_SetMode used with an invalid mode.

Definition at line 80 of file Icu.h.

5.24.2.14 ICU_E_PARAM_NOTIFY_INTERVAL

```
#define ICU_E_PARAM_NOTIFY_INTERVAL (0x18U)
```

Det Error value, API Icu_StartTimeStamp is called and the parameter NotifyInterval is invalid (e.g."0", NotifyInterval < 1)

Definition at line 76 of file Icu.h.

5.24.2.15 ICU_E_PARAM_POINTER

```
#define ICU_E_PARAM_POINTER (0x0AU)
```

Det Error value, API IS called with invalid pointer.

Definition at line 56 of file Icu.h.

5.24.2.16 ICU_E_PARAM_VINFO

```
#define ICU_E_PARAM_VINFO (0x19U)
```

Det Error value, API Icu_GetVersionInfo is called and the parameter versioninfo is invalid (e.g. NULL)

Definition at line 71 of file Icu.h.

5.24.2.17 ICU_E_UNINIT

```
#define ICU_E_UNINIT (0x14U)
```

Det Error value, API service used without module initialization.

Definition at line 65 of file Icu.h.

5.24.2.18 ICU_ENABLEEDGECOUNT_ID

```
#define ICU_ENABLEEDGECOUNT_ID (0x0DU)
```

API service ID for ICU enable edge count function.

Definition at line 105 of file Icu.h.

5.24.2.19 ICU_ENABLEEDGEDETECTION_ID

```
#define ICU_ENABLEEDGEDETECTION_ID (0x16U)
```

API service ID for ICU enable edge detect function.

Definition at line 107 of file Icu.h.

5.24.2.20 ICU_ENABLENOTIFICATION_ID

```
#define ICU_ENABLENOTIFICATION_ID (0X07U)
```

API service ID for ICU enable notification function.

Definition at line 99 of file Icu.h.

5.24.2.21 ICU_ENABLEWAKEUP_ID

```
#define ICU_ENABLEWAKEUP_ID (0x04U)
```

API service ID for ICU enable wakeup function.

Definition at line 89 of file Icu.h.

5.24.2.22 ICU_GETDUTYCYCLEVALUES_ID

```
#define ICU_GETDUTYCYCLEVALUES_ID (0x11U)
```

API service ID for ICU get duty cycle values function.

Definition at line 129 of file Icu.h.

5.24.2.23 ICU_GETEDGENUMBERS_ID

```
#define ICU_GETEDGENUMBERS_ID (0x0FU)
```

API service ID for ICU get edge number function.

Definition at line 113 of file Icu.h.

5.24.2.24 ICU_GETINPUTSTATE_ID

```
#define ICU_GETINPUTSTATE_ID (0x08U)
```

API service ID for ICU get input state function.

Definition at line 101 of file Icu.h.

5.24.2.25 ICU_GETTIMEELAPSED_ID

```
#define ICU_GETTIMEELAPSED_ID (0x10U)
```

API service ID for ICU get time elapsed function.

Definition at line 127 of file Icu.h.

5.24.2.26 ICU_GETTIMESTAMPINDEX_ID

```
#define ICU_GETTIMESTAMPINDEX_ID (0x0bU)
```

API service ID for ICU get timestamp index function.

Definition at line 121 of file Icu.h.

5.24.2.27 ICU_GETVERSIONINFO_ID

```
#define ICU_GETVERSIONINFO_ID (0x12U)
```

API service ID for ICU get version function.

Definition at line 115 of file Icu.h.

5.24.2.28 ICU_INIT_ID

```
#define ICU_INIT_ID (0x00U)
```

API service ID for ICU init function.

Definition at line 83 of file Icu.h.

5.24.2.29 ICU_RESETEGECOUNT_ID

```
#define ICU_RESETEGECOUNT_ID (0x0CU)
```

API service ID for ICU reset edge count function.

Definition at line 103 of file Icu.h.

5.24.2.30 ICU_SETACTIVATIONCONDITION_ID

```
#define ICU_SETACTIVATIONCONDITION_ID (0x05U)
```

API service ID for ICU set activation condition function.

Definition at line 95 of file Icu.h.

5.24.2.31 ICU_SETMODE_ID

```
#define ICU_SETMODE_ID (0x02U)
```

API service ID for ICU set mode function.

Definition at line 87 of file Icu.h.

5.24.2.32 ICU_STARTSIGNALMEASUREMENT_ID

```
#define ICU_STARTSIGNALMEASUREMENT_ID (0x13U)
```

API service ID for ICU start signal measurement function.

Definition at line 123 of file Icu.h.

5.24.2.33 ICU_STARTTIMESTAMP_ID

```
#define ICU_STARTTIMESTAMP_ID (0x09U)
```

API service ID for ICU start timestamp function.

Definition at line 117 of file Icu.h.

5.24.2.34 ICU_STOPSIGNALMEASUREMENT_ID

```
#define ICU_STOPSIGNALMEASUREMENT_ID (0x14U)
```

API service ID for ICU stop signal measurement function.

Definition at line 125 of file Icu.h.

5.24.2.35 ICU_STOPTIMESTAMP_ID

```
#define ICU_STOPTIMESTAMP_ID (0x0aU)
```

API service ID for ICU stop timestamp function.

Definition at line 119 of file Icu.h.

5.24.3 Typedef Documentation

5.24.3.1 Icu_ChannelType

```
typedef uint8 Icu_ChannelType
```

Hardware options for implementing ICU functions.

Definition at line 188 of file Icu.h.

5.24.3.2 Icu_EdgeNumberType

```
typedef uint32 Icu_EdgeNumberType
```

The type of edge count.

Definition at line 197 of file Icu.h.

5.24.3.3 Icu_IndexType

```
typedef uint16 Icu_IndexType
```

The index of the buffer storing timestamps.

Definition at line 194 of file Icu.h.

5.24.3.4 Icu_ValueType

```
typedef uint16 Icu_ValueType
```

Width of the buffer for timestamp ticks and measured elapsed timeticks.

Definition at line 191 of file Icu.h.

5.24.4 Enumeration Type Documentation

5.24.4.1 Icu_ActivationType

```
enum Icu_ActivationType
```

ICU Channel activation options.

Enumerator

ICU_FALLING_EDGE	
ICU_RISING_EDGE	
ICU_BOTH_EDGES	

Definition at line 156 of file Icu.h.

5.24.4.2 Icu_HwSelect

enum Icu_HwSelect

Hardware options for implementing ICU functions.

Enumerator

ICU_HW_PWM	
ICU_HW_PORT	

Definition at line 180 of file Icu.h.

5.24.4.3 Icu_InputStateType

enum Icu_InputStateType

ICU channel input states options.

Enumerator

ICU_ACTIVE	
ICU_IDLE	

Definition at line 140 of file Icu.h.

5.24.4.4 Icu_MeasurementModeType

enum Icu_MeasurementModeType

ICU measurement mode options.

Enumerator

ICU_MODE_SIGNAL_EDGE_DETECT	
ICU_MODE_SIGNAL_MEASUREMENT	
ICU_MODE_TIMESTAMP	
ICU_MODE_EDGE_COUNTER	

Definition at line 147 of file Icu.h.

5.24.4.5 Icu_ModeType

enum [Icu_ModeType](#)

ICU power mode options.

Enumerator

ICU_MODE_NORMAL	
ICU_MODE_SLEEP	

Definition at line 133 of file Icu.h.

5.24.4.6 Icu_SignalMeasurementPropertyType

enum [Icu_SignalMeasurementPropertyType](#)

Definition of the measurement property type.

Enumerator

ICU_LOW_TIME	The channel is configured for reading the elapsed Signal Low Time
ICU_HIGH_TIME	The channel is configured for reading the elapsed Signal High Time
ICU_PERIOD_TIME	The channel is configured for reading the elapsed Signal Period Time
ICU_DUTY_CYCLE	The channel is configured to read values which are needed for calculating the duty cycle (coherent Active and Period Time).

Definition at line 164 of file Icu.h.

5.24.4.7 Icu_TimestampBufferType

enum [Icu_TimestampBufferType](#)

Definition of the timestamp measurement property type.

Enumerator

ICU_LINEAR_BUFFER	The buffer will just be filled once
ICU_CIRCULAR_BUFFER	After reaching the end of the buffer, the driver restarts at the beginning of the buffer

Definition at line 173 of file Icu.h.

5.24.5 Function Documentation

5.24.5.1 Icu_CheckWakeup()

```
void Icu_CheckWakeup (
    EcuM_WakeupSourceType WakeupSource )
```

: Icu_CheckWakeup: Checks if a wakeup capable ICU channel is the source for a wakeup event and calls the ECU state manager service EcuM_SetWakeupEvent in case of a valid ICU channel wakeup event.

Note

Function ID: DES_ICU_API_024
Service ID: 0x15

Parameters

in		
----	--	--

5.24.5.2 Icu_DeInit()

```
void Icu_DeInit (
    void )
```

: Icu_DeInit: Deinitialize icu module.

Note

Function ID: DES_ICU_API_002
Service ID: 0x01

Parameters

in		
----	--	--

5.24.5.3 Icu_DisableEdgeCount()

```
void Icu_DisableEdgeCount (
    Icu_ChannelType Channel )
```

: Icu_DisableEdgeCount: disable the counting of edges of the given channel.

Note

Function ID: DES_ICU_API_006
Service ID: 0x0e

Parameters

in		
----	--	--

5.24.5.4 Icu_DisableEdgeDetection()

```
void Icu_DisableEdgeDetection (
    Icu_ChannelType Channel )
```

: Icu_DisableEdgeDetection: disables the detection of edges of the given channel.

Note

Function ID: DES_ICU_API_017

Service ID: 0x17

Parameters

in		
----	--	--

5.24.5.5 Icu_DisableNotification()

```
void Icu_DisableNotification (
    Icu_ChannelType Channel )
```

: Icu_DisableNotification: disables the notification of a channel.

Note

Function ID: DES_ICU_API_019

Service ID: 0x06

Parameters

in		
----	--	--

5.24.5.6 Icu_DisableWakeup()

```
void Icu_DisableWakeup (
    Icu_ChannelType Channel )
```

: Icu_DisableWakeup: This function disables the wakeup capability of a single ICU channel.

Note

Function ID: DES_ICU_API_023
Service ID: 0x03

Parameters

in		
----	--	--

5.24.5.7 Icu_EnableEdgeCount()

```
void Icu_EnableEdgeCount (
    Icu_ChannelType Channel )
```

: Icu_EnableEdgeCount: enable the counting of edges of the given channel.

Note

Function ID: DES_ICU_API_005
Service ID: 0x0d

Parameters

in		
----	--	--

5.24.5.8 Icu_EnableEdgeDetection()

```
void Icu_EnableEdgeDetection (
    Icu_ChannelType Channel )
```

: Icu_EnableEdgeDetection: enables / re-enables the detection of edges of the given channel.

Note

Function ID: DES_ICU_API_016
Service ID: 0x16

Parameters

in		
----	--	--

5.24.5.9 Icu_EnableNotification()

```
void Icu_EnableNotification (
    Icu_ChannelType Channel )
```


: Icu_EnableNotification: enables the notification on the given channel.

Note

Function ID: DES_ICU_API_018

Service ID: 0x07

Parameters

in		
----	--	--

5.24.5.10 Icu_EnableWakeup()

```
void Icu_EnableWakeup (
    Icu_ChannelType Channel )
```

: Icu_EnableWakeup: This function (re-)enables the wakeup capability of the given ICU channel.

Note

Function ID: DES_ICU_API_022

Service ID: 0x04

Parameters

in		
----	--	--

5.24.5.11 Icu_GetDutyCycleValues()

```
void Icu_GetDutyCycleValues (
    Icu_ChannelType Channel,
    Icu_DutyCycleType * DutyCycleValues )
```

: Icu_GetDutyCycleValues: This function reads the coherent active time and period time for the given ICU Channel.

Note

Function ID: DES_ICU_API_031

Service ID: 0x11

Parameters

in		
----	--	--

5.24.5.12 Icu_GetEdgeNumbers()

```
Icu_EdgeNumberType Icu_GetEdgeNumbers (
    Icu_ChannelType Channel )
```

: Icu_GetEdgeNumbers: reads the number of counted edges

Note

Function ID: DES_ICU_API_007
Service ID: 0x0f

Parameters

in		
----	--	--

5.24.5.13 Icu_GetInputState()

```
Icu_InputStateType Icu_GetInputState (
    Icu_ChannelType Channel )
```

: Icu_GetInputState: returns the status of the ICU input.

Note

Function ID: DES_ICU_API_020
Service ID: 0x08

Parameters

in		
----	--	--

5.24.5.14 Icu_GetTimeElapsed()

```
Icu_ValueType Icu_GetTimeElapsed (
    Icu_ChannelType Channel )
```

: Icu_GetTimeElapsed: This function reads the elapsed Signal Low Time for the given channel.

Note

Function ID: DES_ICU_API_030
Service ID: 0x10

Parameters

in		
----	--	--

5.24.5.15 Icu_GetTimestampIndex()

```
Icu_IndexType Icu_GetTimestampIndex (
    Icu_ChannelType Channel )
```

: Icu_GetTimestampIndex: This function reads the timestamp index of the given channel.

Note

Function ID: DES_ICU_API_027
Service ID: 0x0b

Parameters

in		
----	--	--

5.24.5.16 Icu_GetVersionInfo()

```
void Icu_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

: Icu_GetVersionInfo: return the version information of this module.

Note

Function ID: DES_ICU_API_008
Service ID: 0x12

Parameters

in		
----	--	--

5.24.5.17 Icu_Init()

```
void Icu_Init (
    const Icu_ConfigType * ConfigPtr )
```

: Icu_Init: service for ICU initialization

Note

Function ID: DES_ICU_API_001
Service ID: 0x00

Parameters

in		
----	--	--

5.24.5.18 Icu_ResetEdgeCount()

```
void Icu_ResetEdgeCount (
    Icu_ChannelType Channel )
```

: Icu_ResetEdgeCount: Reset Edge Count.

Note

Function ID: DES_ICU_API_004

Service ID: 0x0c

Parameters

in		
----	--	--

5.24.5.19 Icu_SetActivationCondition()

```
void Icu_SetActivationCondition (
    Icu_ChannelType Channel,
    Icu_ActivationType Activation )
```

: Icu_SetActivationCondition: Set Activation Condition.

Note

Function ID: DES_ICU_API_003

Service ID: 0x05

Parameters

in		
----	--	--

5.24.5.20 Icu_SetMode()

```
void Icu_SetMode (
    Icu_ModeType Mode )
```

: Icu_SetMode: This function sets the ICU mode.

Note

Function ID: DES_ICU_API_021
Service ID: 0x02

Parameters

in		
----	--	--

5.24.5.21 Icu_StartSignalMeasurement()

```
void Icu_StartSignalMeasurement (
    Icu_ChannelType Channel )
```

: Icu_StartSignalMeasurement: This function starts the measurement of signals.

Note

Function ID: DES_ICU_API_028
Service ID: 0x13

Parameters

in		
----	--	--

5.24.5.22 Icu_StartTimestamp()

```
void Icu_StartTimestamp (
    Icu_ChannelType Channel,
    Icu_ValueType * BufferPtr,
    uint16 BufferSize,
    uint16 NotifyInterval )
```

: Icu_StartTimestamp: This function starts the capturing of timer values on the edges.

Note

Function ID: DES_ICU_API_025
Service ID: 0x09

Parameters

in		
----	--	--

5.24.5.23 Icu_StopSignalMeasurement()

```
void Icu_StopSignalMeasurement (
    Icu_ChannelType Channel )
```

: Icu_StopSignalMeasurement: This function stops the measurement of signals of the given channel.

Note

Function ID: DES_ICU_API_029
Service ID: 0x14

Parameters

in		
----	--	--

5.24.5.24 Icu_StopTimestamp()

```
void Icu_StopTimestamp (
    Icu_ChannelType Channel )
```

: Icu_StopTimestamp: This function stops the timestamp measurement of the given channel.

Note

Function ID: DES_ICU_API_026
Service ID: 0x0a

Parameters

in		
----	--	--

5.24.6 Variable Documentation

5.24.6.1 Icu_GenerateConfigPC

```
const Icu_ConfigType Icu_GenerateConfigPC
```

5.25 Lin.h File Reference

This file provides extern Mcal lin api.

```
#include "Mcal.h"
#include "Lin_Ipw.h"
#include "Lin_Cfg.h"
```

Classes

- struct [Lin_ConfigType](#)

Macros

- #define [LIN_INIT_ID](#) (0x00U)
- #define [LIN_GETVERSIONINFO_ID](#) (0x01U)
- #define [LIN_GOTOSLEEP_ID](#) (0x06U)
- #define [LIN_WAKEUP_ID](#) (0x07U)
- #define [LIN_GOTOSLEEPINTERNAL_ID](#) (0x09U)
- #define [LIN_WAKEUPINTERNAL_ID](#) (0x0BU)
- #define [LIN_GETSTATUS_ID](#) (0x08U)
- #define [LIN_CHECK_WAKEUP_ID](#) (0x0AU)
- #define [LIN_SENDFRAME_ID](#) (0x04U)
- #define [LIN_E_UNINIT](#) ((uint8)0x00U)
- #define [LIN_E_INVALID_CHANNEL](#) ((uint8)0x02U)
- #define [LIN_E_INVALID_POINTER](#) ((uint8)0x03U)
- #define [LIN_E_STATE_TRANSITION](#) ((uint8)0x04U)
- #define [LIN_E_PARAM_POINTER](#) ((uint8)0x05U)
- #define [LIN_E_TIMEOUT](#) ((uint8)0x06U)
- #define [LIN_STATE_UNINIT](#) ((uint8)0x01U)
- #define [LIN_STATE_INIT](#) ((uint8)0x02U)
- #define [LIN_CH_SLEEP_PENDING](#) ((uint8)0x01U)
- #define [LIN_CH_OPERATIONAL](#) ((uint8)0x03U)
- #define [LIN_CH_SLEEP_STATE](#) ((uint8)0x04U)
- #define [LIN_INSTANCE](#) 0U

Functions

- void [Lin_Init](#) (const [Lin_ConfigType](#) *Config)
Initializes the LIN module.
- void [Lin_GetVersionInfo](#) (Std_VersionInfoType *versioninfo)
Get Lin module version information.
- Std_ReturnType [Lin_CheckWakeup](#) (uint8 Channel)
This function checks if a wakeup has occurred on the addressed LIN channel.
- Std_ReturnType [Lin_Wakeup](#) (uint8 Channel)
Generates a wake up pulse and sets the channel state to LIN_CH_OPERATIONAL.
- Std_ReturnType [Lin_WakeupInternal](#) (uint8 Channel)
Sets the channel state to LIN_CH_OPERATIONAL without generating a wake up pulse.
- Std_ReturnType [Lin_GoToSleep](#) (uint8 Channel)
The service instructs the driver to transmit a go-to-sleep-command on the addressed LIN channel. Only used for LIN master nodes.
- Std_ReturnType [Lin_GoToSleepInternal](#) (uint8 Channel)
Sets the channel state to LIN_CH_SLEEP_STATE, enables the wake-up detection and optionally sets the LIN hardware unit to reduced power operation mode (if supported by HW).
- Std_ReturnType [Lin_SendFrame](#) (uint8 Channel, const Lin_PduType *PduInfoPtr)
Sends a LIN header and a LIN respnse, if necessary.The direction of the frame response(master response, slave response,slave-to-slave communication)is provided by the PduInfoPtr. Only used master node.
- Lin_StatusType [Lin_GetStatus](#) (uint8 Channel, uint8 **Lin_SduPtr)
Gets the status of the LIN driver, Only used for LIN master nodes.

Variables

- const [Lin_ConfigType](#) [Lin_GenerateConfigPC](#)

5.25.1 Detailed Description

This file provides extern Mcal lin api.

5.25.2 Macro Definition Documentation

5.25.2.1 LIN_CH_OPERATIONAL

```
#define LIN_CH_OPERATIONAL ((uint8)0x03U)
```

Definition at line 82 of file Lin.h.

5.25.2.2 LIN_CH_SLEEP_PENDING

```
#define LIN_CH_SLEEP_PENDING ((uint8)0x01U)
```

Definition at line 81 of file Lin.h.

5.25.2.3 LIN_CH_SLEEP_STATE

```
#define LIN_CH_SLEEP_STATE ((uint8)0x04U)
```

Definition at line 83 of file Lin.h.

5.25.2.4 LIN_CHECK_WAKEUP_ID

```
#define LIN_CHECK_WAKEUP_ID (0x0AU)
```

Definition at line 68 of file Lin.h.

5.25.2.5 LIN_E_INVALID_CHANNEL

```
#define LIN_E_INVALID_CHANNEL ((uint8)0x02U)
```

Definition at line 72 of file Lin.h.

5.25.2.6 LIN_E_INVALID_POINTER

```
#define LIN_E_INVALID_POINTER ((uint8)0x03U)
```

Definition at line 73 of file Lin.h.

5.25.2.7 LIN_E_PARAM_POINTER

```
#define LIN_E_PARAM_POINTER ((uint8)0x05U)
```

Definition at line 75 of file Lin.h.

5.25.2.8 LIN_E_STATE_TRANSITION

```
#define LIN_E_STATE_TRANSITION ((uint8)0x04U)
```

Definition at line 74 of file Lin.h.

5.25.2.9 LIN_E_TIMEOUT

```
#define LIN_E_TIMEOUT ((uint8)0x06U)
```

Definition at line 76 of file Lin.h.

5.25.2.10 LIN_E_UNINIT

```
#define LIN_E_UNINIT ((uint8)0x00U)
```

Definition at line 71 of file Lin.h.

5.25.2.11 LIN_GETSTATUS_ID

```
#define LIN_GETSTATUS_ID (0x08U)
```

Definition at line 67 of file Lin.h.

5.25.2.12 LIN_GETVERSIONINFO_ID

```
#define LIN_GETVERSIONINFO_ID (0x01U)
```

Definition at line 62 of file Lin.h.

5.25.2.13 LIN_GOTOSLEEP_ID

```
#define LIN_GOTOSLEEP_ID (0x06U)
```

Definition at line 63 of file Lin.h.

5.25.2.14 LIN_GOTOSLEEPINTERNAL_ID

```
#define LIN_GOTOSLEEPINTERNAL_ID (0x09U)
```

Definition at line 65 of file Lin.h.

5.25.2.15 LIN_INIT_ID

```
#define LIN_INIT_ID (0x00U)
```

Definition at line 61 of file Lin.h.

5.25.2.16 LIN_INSTANCE

```
#define LIN_INSTANCE 0U
```

Definition at line 85 of file Lin.h.

5.25.2.17 LIN_SENDFRAME_ID

```
#define LIN_SENDFRAME_ID (0x04U)
```

Definition at line 69 of file Lin.h.

5.25.2.18 LIN_STATE_INIT

```
#define LIN_STATE_INIT ((uint8)0x02U)
```

Definition at line 80 of file Lin.h.

5.25.2.19 LIN_STATE_UNINIT

```
#define LIN_STATE_UNINIT ((uint8)0x01U)
```

Definition at line 79 of file Lin.h.

5.25.2.20 LIN_WAKEUP_ID

```
#define LIN_WAKEUP_ID (0x07U)
```

Definition at line 64 of file Lin.h.

5.25.2.21 LIN_WAKEUPINTERNAL_ID

```
#define LIN_WAKEUPINTERNAL_ID (0x0BU)
```

Definition at line 66 of file Lin.h.

5.25.3 Function Documentation

5.25.3.1 Lin_CheckWakeup()

```
Std_ReturnType Lin_CheckWakeup (
    uint8 Channel )
```

This function checks if a wakeup has occurred on the addressed LIN channel.

Note

Function ID:[DES_LIN_API_206]
Service ID: 0x0a

Parameters

in	Channel	LIN channel to be addressed.
in, out	None	
out	None	

Returns

Std_ReturnType: E_OK: No error has occurred during execution of the API E_NOT_OK: An error has occurred during execution of the API

5.25.3.2 Lin_GetStatus()

```
Lin_StatusType Lin_GetStatus (
    uint8 Channel,
    uint8 ** Lin_SduPtr )
```

Gets the status of the LIN driver, Only used for LIN master nodes.

Note

Function ID:[DES_LIN_API_205]
Service ID: 0x08

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
in, out	<i>Lin_SduPtr</i>	Pointer to pointer to a shadow buffer or memory mapped LIN Hardware receive buffer where the current SDU is stored.
out	<i>None</i>	

Returns

Lin_StatusType: Lin Channel status from master node

5.25.3.3 Lin_GetVersionInfo()

```
void Lin_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

Get Lin module version information.

Note

Function ID:[DES_LIN_API_208]
Service ID: 0x01

Parameters

in	<i>None</i>	
in, out	<i>versioninfo</i>	LIN module version information address
out	<i>None</i>	

Returns

void

5.25.3.4 Lin_GoToSleep()

```
Std_ReturnType Lin_GoToSleep (
    uint8 Channel )
```

The service instructs the driver to transmit a go-to-sleep-command on the addressed LIN channel. Only used for LIN master nodes.

Note

Function ID:[DES_LIN_API_203]
Service ID: 0x06

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType: Std_ReturnType: E_OK: Sleep command has been accepted E_NOT_OK: Sleep command has not been accepted, development or production error occurred

5.25.3.5 Lin_GoToSleepInternal()

```
Std_ReturnType Lin_GoToSleepInternal (
    uint8 Channel )
```

Sets the channel state to LIN_CH_SLEEP_STATE, enables the wake-up detection and optionally sets the LIN hardware unit to reduced power operation mode (if supported by HW).

Note

Function ID:[DES_LIN_API_204]
Service ID: 0x09

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Command has been accepted E_NOT_OK: Command has not been accepted, development or production error occurred

5.25.3.6 Lin_Init()

```
void Lin_Init (
    const Lin_ConfigType * Config )
```

Initializes the LIN module.

Note

Function ID:[DES_LIN_API_200]
Service ID: 0x00

Parameters

in	<i>Config</i>	Pointer to LIN driver configuration set.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

void

5.25.3.7 Lin_SendFrame()

```
Std_ReturnType Lin_SendFrame (
    uint8 Channel,
    const Lin_PduType * PduInfoPtr )
```

Sends a LIN header and a LIN response, if necessary. The direction of the frame response (master response, slave response, slave-to-slave communication) is provided by the PduInfoPtr. Only used master node.

Note

Function ID:[DES_LIN_API_207]
Service ID: 0x04

Parameters

in	<i>Channel</i>	LIN channel to be addressed. PduInfoPtr: Pointer to PDU containing the PID, checksum model, response type, DI and SDU data pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType: E_OK: Send command has been accepted E_NOT_OK: Send command has not been accepted

5.25.3.8 Lin_Wakeup()

```
Std_ReturnType Lin_Wakeup (  
    uint8 Channel )
```

Generates a wake up pulse and sets the channel state to LIN_CH_OPERATIONAL.

Note

Function ID:[DES_LIN_API_201]
Service ID: 0x07

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType: E_OK:success generates a wake up pulse E_NOT_OK: failed to generates a wake up pulse

5.25.3.9 Lin_WakeupInternal()

```
Std_ReturnType Lin_WakeupInternal (  
    uint8 Channel )
```

Sets the channel state to LIN_CH_OPERATIONAL without generating a wake up pulse.

Note

Function ID:[DES_LIN_API_202]
Service ID: 0x0b

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType: E_OK:success Sets the channel state to LIN_CH_OPERATIONAL. E_NOT_OK: failed to Sets the channel state to LIN_CH_OPERATIONAL.

5.25.4 Variable Documentation

5.25.4.1 Lin_GenerateConfigPC

```
const Lin_ConfigType Lin_GenerateConfigPC
```

5.26 Lin_GeneralTypes.h File Reference

```
#include "StandardTypes.h"
#include "Lin_Hal_Types.h"
```

Typedefs

- typedef uint8 [Lin_FrameDlType](#)
specification of LIN Driver.
- typedef uint8 [Lin_FramePidType](#)

Enumerations

- enum [Lin_SlaveErrorType](#) {
 [LIN_ERR_HEADER](#) = 0x0U, [LIN_ERR_RESP_STOPBIT](#) = 0x1U, [LIN_ERR_RESP_CHKSUM](#) = 0x2U, [LIN_ERR_RESP_DATABIT](#) = 0x3U,
 [LIN_ERR_NO_RESP](#) = 0x4U, [LIN_ERR_INC_RESP](#) = 0x5U }

5.26.1 Typedef Documentation

5.26.1.1 Lin_FrameDlType

```
typedef uint8 Lin\_FrameDlType
```

specification of LIN Driver.

Definition at line 59 of file Lin_GeneralTypes.h.

5.26.1.2 Lin_FramePidType

```
typedef uint8 Lin\_FramePidType
```

Definition at line 62 of file Lin_GeneralTypes.h.

5.26.2 Enumeration Type Documentation

5.26.2.1 Lin_SlaveErrorType

```
enum Lin\_SlaveErrorType
```


Enumerator

LIN_ERR_HEADER	
LIN_ERR_RESP_STOPBIT	
LIN_ERR_RESP_CHKSUM	
LIN_ERR_RESP_DATABIT	
LIN_ERR_NO_RESP	
LIN_ERR_INC_RESP	

Definition at line 65 of file Lin_GeneralTypes.h.

5.27 Mcu.h File Reference

Mcu driver header file.

```
#include "Mcu_Ipw_Types.h"
```

Classes

- struct [Mcu_ConfigType](#)
A structure to hold the MCU driver configuration.

Macros

- #define [MCU_INSTANCE_ID](#) ((uint8)0x0U)
- #define [MCU_INIT_ID](#) ((uint8)0x00U)
- #define [MCU_INITRAMSECTION_ID](#) ((uint8)0x01U)
- #define [MCU_INITCLOCK_ID](#) ((uint8)0x02U)
- #define [MCU_DISTRIBUTEPLLCLOCK_ID](#) ((uint8)0x03U)
- #define [MCU_GETPLLSTATUS_ID](#) ((uint8)0x04U)
- #define [MCU_GETRESETREASON_ID](#) ((uint8)0x05U)
- #define [MCU_GETRESETRAWVALUE_ID](#) ((uint8)0x06U)
- #define [MCU_PERFORMRESET_ID](#) ((uint8)0x07U)
- #define [MCU_SETMODE_ID](#) ((uint8)0x08U)
- #define [MCU_GETVERSIONINFO_ID](#) ((uint8)0x09U)
- #define [MCU_GETRAMSTATE_ID](#) ((uint8)0x0AU)
- #define [MCU_E_PARAM_CONFIG](#) ((uint8)0x0AU)
- #define [MCU_E_PARAM_CLOCK](#) ((uint8)0x0BU)
- #define [MCU_E_PARAM_MODE](#) ((uint8)0x0CU)
- #define [MCU_E_PARAM_RAMSECTION](#) ((uint8)0x0DU)
- #define [MCU_E_PLL_NOT_LOCKED](#) ((uint8)0x0EU)
- #define [MCU_E_UNINIT](#) ((uint8)0x0FU)
- #define [MCU_E_PARAM_POINTER](#) ((uint8)0x10U)
- #define [MCU_E_INIT_FAILED](#) ((uint8)0x11U)
- #define [MCU_E_ALREADY_INITIALIZED](#) ((uint8)0x13U)
- #define [MCU_E_CMU_INDEX_OUT_OF_RANGE](#) ((uint8)0x22U)

Functions

- void [Mcu_Init](#) (const [Mcu_ConfigType](#) *ConfigPtr)
This service initializes the MCU driver.
- Std_ReturnType [Mcu_InitRamSection](#) ([Mcu_RamSectionType](#) RamSection)
This service initializes the RAM section wise.
- Std_ReturnType [Mcu_InitClock](#) ([Mcu_ClockType](#) ClockSetting)
This service initializes the PLL and other MCU specific clock options.
- Std_ReturnType [Mcu_DistributePllClock](#) (void)
This service activates the PLL clock to the MCU clock distribution.
- [Mcu_PllStatusType](#) [Mcu_GetPllStatus](#) (void)
This service provides the lock status of PLL.
- [Mcu_ResetType](#) [Mcu_GetResetReason](#) (void)
This service reads the reset type from the hardware.
- [Mcu_RawResetType](#) [Mcu_GetResetRawValue](#) (void)
This service reads the reset type from the hardware register.
- void [Mcu_PerformReset](#) (void)
This service performs a microcontroller reset.
- void [Mcu_SetMode](#) ([Mcu_ModeType](#) McuMode)
This service activates the MCU power modes.
- void [Mcu_GetVersionInfo](#) ([Std_VersionInfoType](#) *versioninfo)
This service returns the version information of this module.
- [Mcu_RamStateType](#) [Mcu_GetRamState](#) (void)
This service provides the actual status of the microcontroller Ram.

Variables

- const [Mcu_ConfigType](#) [Mcu_GenerateConfigPC](#)
Mcu Configuration.

5.27.1 Detailed Description

Mcu driver header file.

5.27.2 Macro Definition Documentation

5.27.2.1 MCU_DISTRIBUTEPLLCLOCK_ID

```
#define MCU_DISTRIBUTEPLLCLOCK_ID ((uint8)0x03U)
```

Definition at line 63 of file Mcu.h.

5.27.2.2 MCU_E_ALREADY_INITIALIZED

```
#define MCU_E_ALREADY_INITIALIZED ((uint8)0x13U)
```

Definition at line 80 of file Mcu.h.

5.27.2.3 MCU_E_CMU_INDEX_OUT_OF_RANGE

```
#define MCU_E_CMU_INDEX_OUT_OF_RANGE ((uint8)0x22U)
```

Definition at line 81 of file Mcu.h.

5.27.2.4 MCU_E_INIT_FAILED

```
#define MCU_E_INIT_FAILED ((uint8)0x11U)
```

Definition at line 79 of file Mcu.h.

5.27.2.5 MCU_E_PARAM_CLOCK

```
#define MCU_E_PARAM_CLOCK ((uint8)0x0BU)
```

Definition at line 73 of file Mcu.h.

5.27.2.6 MCU_E_PARAM_CONFIG

```
#define MCU_E_PARAM_CONFIG ((uint8)0x0AU)
```

Definition at line 72 of file Mcu.h.

5.27.2.7 MCU_E_PARAM_MODE

```
#define MCU_E_PARAM_MODE ((uint8)0x0CU)
```

Definition at line 74 of file Mcu.h.

5.27.2.8 MCU_E_PARAM_POINTER

```
#define MCU_E_PARAM_POINTER ((uint8)0x10U)
```

Definition at line 78 of file Mcu.h.

5.27.2.9 MCU_E_PARAM_RAMSECTION

```
#define MCU_E_PARAM_RAMSECTION ((uint8)0x0DU)
```

Definition at line 75 of file Mcu.h.

5.27.2.10 MCU_E_PLL_NOT_LOCKED

```
#define MCU_E_PLL_NOT_LOCKED ((uint8)0x0EU)
```

Definition at line 76 of file Mcu.h.

5.27.2.11 MCU_E_UNINIT

```
#define MCU_E_UNINIT ((uint8)0x0FU)
```

Definition at line 77 of file Mcu.h.

5.27.2.12 MCU_GETPLLSTATUS_ID

```
#define MCU_GETPLLSTATUS_ID ((uint8)0x04U)
```

Definition at line 64 of file Mcu.h.

5.27.2.13 MCU_GETRAMSTATE_ID

```
#define MCU_GETRAMSTATE_ID ((uint8)0x0AU)
```

Definition at line 70 of file Mcu.h.

5.27.2.14 MCU_GETRESETRAWVALUE_ID

```
#define MCU_GETRESETRAWVALUE_ID ((uint8)0x06U)
```

Definition at line 66 of file Mcu.h.

5.27.2.15 MCU_GETRESETREASON_ID

```
#define MCU_GETRESETREASON_ID ((uint8)0x05U)
```

Definition at line 65 of file Mcu.h.

5.27.2.16 MCU_GETVERSIONINFO_ID

```
#define MCU_GETVERSIONINFO_ID ((uint8)0x09U)
```

Definition at line 69 of file Mcu.h.

5.27.2.17 MCU_INIT_ID

```
#define MCU_INIT_ID ((uint8)0x00U)
```

Definition at line 60 of file Mcu.h.

5.27.2.18 MCU_INITCLOCK_ID

```
#define MCU_INITCLOCK_ID ((uint8)0x02U)
```

Definition at line 62 of file Mcu.h.

5.27.2.19 MCU_INITRAMSECTION_ID

```
#define MCU_INITRAMSECTION_ID ((uint8)0x01U)
```

Definition at line 61 of file Mcu.h.

5.27.2.20 MCU_INSTANCE_ID

```
#define MCU_INSTANCE_ID ((uint8)0x0U)
```

Definition at line 58 of file Mcu.h.

5.27.2.21 MCU_PERFORMRESET_ID

```
#define MCU_PERFORMRESET_ID ((uint8)0x07U)
```

Definition at line 67 of file Mcu.h.

5.27.2.22 MCU_SETMODE_ID

```
#define MCU_SETMODE_ID ((uint8)0x08U)
```

Definition at line 68 of file Mcu.h.

5.27.3 Function Documentation

5.27.3.1 Mcu_DistributePllClock()

```
Std_ReturnType Mcu_DistributePllClock (  
    void )
```

This service activates the PLL clock to the MCU clock distribution.

Note

Function ID: [DES_MCU_API_003]
Service ID: 0x03

Parameters

in	<i>None</i>	
in,out	<i>None</i>	
out	<i>None</i>	

Returns

Command has been accepted or not. -E_OK: command has been accepted. -E_NOT_OK: command has not been accepted e.g. due to parameter error.

5.27.3.2 Mcu_GetPllStatus()

```
Mcu_PllStatusType Mcu_GetPllStatus (
    void )
```

This service provides the lock status of PLL.

Note

Function ID: [DES_MCU_API_004]
Service ID: 0x04

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

PLL status. -MCU_PLL_LOCKED: PLL is locked. -MCU_PLL_UNLOCKED: PLL is not locked. -MCU_PLL_STAT↔
US_UNDEFINED: PLL status is undefined.

5.27.3.3 Mcu_GetRamState()

```
Mcu_RamStateType Mcu_GetRamState (
    void )
```

This service provides the actual status of the microcontroller Ram.

Note

Function ID: [DES_MCU_API_010]
Service ID: 0x0a

Parameters

in	<i>None.</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Status of Ram Content -MCU_RAMSTATE_INVALID: Ram content is not valid or unknown (default). -MCU_RAM↔
STATE_VALID: Ram content is valid.

5.27.3.4 Mcu_GetResetRawValue()

```
Mcu_RawResetType Mcu_GetResetRawValue (
    void )
```

This service reads the reset type from the hardware register.

Note

Function ID: [DES_MCU_API_006]
Service ID: 0x06

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Reset raw value.

5.27.3.5 Mcu_GetResetReason()

```
Mcu_ResetType Mcu_GetResetReason (
    void )
```

This service reads the reset type from the hardware.

Note

Function ID: [DES_MCU_API_005]
Service ID: 0x05

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Reset reason.

5.27.3.6 Mcu_GetVersionInfo()

```
void Mcu_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

This service returns the version information of this module.

Note

Function ID: [DES_MCU_API_009]
Service ID: 0x09

Parameters

<i>in</i>	<i>None</i>	
<i>in, out</i>	<i>None</i>	
<i>out</i>	<i>versioninfo</i>	Pointer to where to store the version information of this module.

Returns

None

5.27.3.7 Mcu_Init()

```
void Mcu_Init (
    const Mcu_ConfigType * ConfigPtr )
```

This service initializes the MCU driver.

Note

Function ID: [DES_MCU_API_000]
Service ID: 0x00

Parameters

<i>in</i>	<i>ConfigPtr</i>	Pointer to MCU driver configuration set.
<i>in, out</i>	<i>None</i>	
<i>out</i>	<i>None</i>	

Returns

None

5.27.3.8 Mcu_InitClock()

```
Std_ReturnType Mcu_InitClock (
    Mcu_ClockType ClockSetting )
```

This service initializes the PLL and other MCU specific clock options.

Note

Function ID: [DES_MCU_API_002]
Service ID: 0x02

Parameters

in	<i>ClockSetting</i>	Clock setting.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Command has been accepted or not. -E_OK: command has been accepted. -E_NOT_OK: command has not been accepted e.g. due to parameter error.

5.27.3.9 Mcu_InitRamSection()

```
Std_ReturnType Mcu_InitRamSection (
    Mcu_RamSectionType RamSection )
```

This service initializes the RAM section wise.

Note

Function ID: [DES_MCU_API_001]
Service ID: 0x01

Parameters

in	<i>RamSection</i>	Selects RAM memory section provided in configuration set.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Command has been accepted or not. -E_OK: command has been accepted. -E_NOT_OK: command has not been accepted e.g. due to parameter error.

5.27.3.10 Mcu_PerformReset()

```
void Mcu_PerformReset (
    void )
```

This service performs a microcontroller reset.

Note

Function ID: [DES_MCU_API_007]
Service ID: 0x07

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.27.3.11 Mcu_SetMode()

```
void Mcu_SetMode (
    Mcu_ModeType McuMode )
```

This service activates the MCU power modes.

Note

Function ID: [DES_MCU_API_008]
Service ID: 0x08

Parameters

in	<i>McuMode</i>	Set different MCU power modes in the configuration set.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.27.4 Variable Documentation**5.27.4.1 Mcu_GenerateConfigPC**

```
const Mcu_ConfigType Mcu_GenerateConfigPC
```

Mcu Configuration.

5.28 Ocu.h File Reference

This file provides extern Mcal Ocu api.

```
#include "Ocu_Ipw_Types.h"
#include "Ocu_Cfg.h"
```

Classes

- struct [Ocu_ChannelConfigType](#)
- struct [Ocu_ModuleConfigType](#)
- struct [Ocu_IpConfigType](#)
- struct [Ocu_ConfigType](#)

Macros

- #define [OCU_E_UNINIT](#) (0x02U)
- #define [OCU_E_PARAM_INVALID_CHANNEL](#) (0x03U)
- #define [OCU_E_PARAM_INVALID_STATE](#) (0x04U)
- #define [OCU_E_PARAM_INVALID_ACTION](#) (0x05U)
- #define [OCU_E_NO_VALID_NOTIF](#) (0x06U)
- #define [OCU_E_ALREADY_INITIALIZED](#) (0x07U)
- #define [OCU_E_PARAM_POINTER](#) (0x08U)
- #define [OCU_E_BUSY](#) (0x09U)
- #define [OCU_E_PARAM_NO_PIN](#) (0x0AU)
- #define [OCU_E_INIT_FAILED](#) (0x0BU)
- #define [OCU_E_PARAM_INVALID_VALUE](#) (0x0CU)
- #define [OCU_INIT_ID](#) (0x00U)
- #define [OCU_DEINIT_ID](#) (0x01U)
- #define [OCU_STARTCHANNEL_ID](#) (0x02U)
- #define [OCU_STOPCHANNEL_ID](#) (0x03U)
- #define [OCU_SETPINSTATE_ID](#) (0x04U)
- #define [OCU_SETPINACTION_ID](#) (0x05U)
- #define [OCU_GETCOUNTER_ID](#) (0x06U)
- #define [OCU_SETABSOLUTETHRESHOLD_ID](#) (0x07U)
- #define [OCU_SETRELATIVETHRESHOLD_ID](#) (0x08U)
- #define [OCU_GETVERSIONINFO_ID](#) (0x09U)
- #define [OCU_DISABLENOTIFICATION_ID](#) (0x0AU)
- #define [OCU_ENABLENOTIFICATION_ID](#) (0x0BU)

Typedefs

- typedef uint8 [Ocu_ChannelType](#)
- typedef uint16 [Ocu_ValueType](#)

Enumerations

- enum [Ocu_PinStateType](#) { [OCU_LOW](#) = 0U, [OCU_HIGH](#) }
- enum [Ocu_PinActionType](#) { [OCU_SET_LOW](#) = 0U, [OCU_SET_HIGH](#), [OCU_TOGGLE](#), [OCU_DISABLE](#) }
- enum [Ocu_ReturnType](#) { [OCU_CM_IN_REF_INTERVAL](#) = 0U, [OCU_CM_OUT_REF_INTERVAL](#) }

Functions

- void `Ocu_Init` (const `Ocu_ConfigType` *ConfigPtr)
: *Ocu_Init: Service for OCU initialization.*
- void `Ocu_DeInit` (void)
: *Ocu_DeInit: This function de-initializes the OCU module.*
- void `Ocu_StartChannel` (`Ocu_ChannelType` ChannelNumber)
: *Ocu_StartChannel: The Service to start an OCU channel.*
- void `Ocu_StopChannel` (`Ocu_ChannelType` ChannelNumber)
: *Ocu_StopChannel: Service to stop an OCU channel.*
- void `Ocu_SetPinState` (`Ocu_ChannelType` ChannelNumber, `Ocu_PinStateType` PinState)
: *set immediately the level of the pin associated to an OCU channel.*
- void `Ocu_SetPinAction` (`Ocu_ChannelType` ChannelNumber, `Ocu_PinActionType` PinAction)
: *Ocu_SetPinAction: indicate the driver what shall be done automatically by hardware upon compare match.*
- `Ocu_ValueType` `Ocu_GetCounter` (`Ocu_ChannelType` ChannelNumber)
: *Ocu_GetCounter: Service to read the current value of the counter.*
- `Ocu_ReturnType` `Ocu_SetAbsoluteThreshold` (`Ocu_ChannelType` ChannelNumber, `Ocu_ValueType` Reference↔Value, `Ocu_ValueType` AbsoluteValue)
: *Ocu_SetAbsoluteThreshold: set the value of the channel threshold using an absolute input data.*
- `Ocu_ReturnType` `Ocu_SetRelativeThreshold` (`Ocu_ChannelType` ChannelNumber, `Ocu_ValueType` RelativeValue)
: *Ocu_SetRelativeThreshold: set the value of the channel threshold relative to the current value of the counter.*
- void `Ocu_DisableNotification` (`Ocu_ChannelType` ChannelNumber)
: *Ocu_DisableNotification: used to disable notifications from an OCU channel.*
- void `Ocu_EnableNotification` (`Ocu_ChannelType` ChannelNumber)
: *Ocu_EnableNotification: used to enable notifications from an OCU channel.*
- void `Ocu_GetVersionInfo` (`Std_VersionInfoType` *versioninfo)
: *Ocu_GetVersionInfo: returns the version information of this module.*

Variables

- const `Ocu_ConfigType` `Ocu_GenerateConfigPC`

5.28.1 Detailed Description

This file provides extern Mcal Ocu api.

5.28.2 Macro Definition Documentation

5.28.2.1 OCU_DEINIT_ID

```
#define OCU_DEINIT_ID (0x01U)
```

Definition at line 68 of file Ocu.h.

5.28.2.2 OCU_DISABLENOTIFICATION_ID

```
#define OCU_DISABLENOTIFICATION_ID (0x0AU)
```

Definition at line 77 of file Ocu.h.

5.28.2.3 OCU_E_ALREADY_INITIALIZED

```
#define OCU_E_ALREADY_INITIALIZED (0x07U)
```

Definition at line 60 of file Ocu.h.

5.28.2.4 OCU_E_BUSY

```
#define OCU_E_BUSY (0x09U)
```

Definition at line 62 of file Ocu.h.

5.28.2.5 OCU_E_INIT_FAILED

```
#define OCU_E_INIT_FAILED (0x0BU)
```

Definition at line 64 of file Ocu.h.

5.28.2.6 OCU_E_NO_VALID_NOTIF

```
#define OCU_E_NO_VALID_NOTIF (0x06U)
```

Definition at line 59 of file Ocu.h.

5.28.2.7 OCU_E_PARAM_INVALID_ACTION

```
#define OCU_E_PARAM_INVALID_ACTION (0x05U)
```

Definition at line 58 of file Ocu.h.

5.28.2.8 OCU_E_PARAM_INVALID_CHANNEL

```
#define OCU_E_PARAM_INVALID_CHANNEL (0x03U)
```

Definition at line 56 of file Ocu.h.

5.28.2.9 OCU_E_PARAM_INVALID_STATE

```
#define OCU_E_PARAM_INVALID_STATE (0x04U)
```

Definition at line 57 of file Ocu.h.

5.28.2.10 OCU_E_PARAM_INVALID_VALUE

```
#define OCU_E_PARAM_INVALID_VALUE (0x0CU)
```

Definition at line 65 of file Ocu.h.

5.28.2.11 OCU_E_PARAM_NO_PIN

```
#define OCU_E_PARAM_NO_PIN (0x0AU)
```

Definition at line 63 of file Ocu.h.

5.28.2.12 OCU_E_PARAM_POINTER

```
#define OCU_E_PARAM_POINTER (0x08U)
```

Definition at line 61 of file Ocu.h.

5.28.2.13 OCU_E_UNINIT

```
#define OCU_E_UNINIT (0x02U)
```

Definition at line 55 of file Ocu.h.

5.28.2.14 OCU_ENABLENOTIFICATION_ID

```
#define OCU_ENABLENOTIFICATION_ID (0x0BU)
```

Definition at line 78 of file Ocu.h.

5.28.2.15 OCU_GETCOUNTER_ID

```
#define OCU_GETCOUNTER_ID (0x06U)
```

Definition at line 73 of file Ocu.h.

5.28.2.16 OCU_GETVERSIONINFO_ID

```
#define OCU_GETVERSIONINFO_ID (0x09U)
```

Definition at line 76 of file Ocu.h.

5.28.2.17 OCU_INIT_ID

```
#define OCU_INIT_ID (0x00U)
```

Definition at line 67 of file Ocu.h.

5.28.2.18 OCU_SETABSOLUTETHRESHOLD_ID

```
#define OCU_SETABSOLUTETHRESHOLD_ID (0x07U)
```

Definition at line 74 of file Ocu.h.

5.28.2.19 OCU_SETPINACTION_ID

```
#define OCU_SETPINACTION_ID (0x05U)
```

Definition at line 72 of file Ocu.h.

5.28.2.20 OCU_SETPINSTATE_ID

```
#define OCU_SETPINSTATE_ID (0x04U)
```

Definition at line 71 of file Ocu.h.

5.28.2.21 OCU_SETRELATIVETHRESHOLD_ID

```
#define OCU_SETRELATIVETHRESHOLD_ID (0x08U)
```

Definition at line 75 of file Ocu.h.

5.28.2.22 OCU_STARTCHANNEL_ID

```
#define OCU_STARTCHANNEL_ID (0x02U)
```

Definition at line 69 of file Ocu.h.

5.28.2.23 OCU_STOPCHANNEL_ID

```
#define OCU_STOPCHANNEL_ID (0x03U)
```

Definition at line 70 of file Ocu.h.

5.28.3 Typedef Documentation

5.28.3.1 Ocu_ChannelType

```
typedef uint8 Ocu_ChannelType
```

Definition at line 102 of file Ocu.h.

5.28.3.2 Ocu_ValueType

```
typedef uint16 Ocu_ValueType
```

Definition at line 104 of file Ocu.h.

5.28.4 Enumeration Type Documentation

5.28.4.1 Ocu_PinActionType

```
enum Ocu_PinActionType
```

Enumerator

OCU_SET_LOW	
OCU_SET_HIGH	
OCU_TOGGLE	
OCU_DISABLE	

Definition at line 87 of file Ocu.h.

5.28.4.2 Ocu_PinStateType

```
enum Ocu_PinStateType
```

Enumerator

OCU_LOW	
OCU_HIGH	

Definition at line 81 of file Ocu.h.

5.28.4.3 Ocu_ReturnType

```
enum Ocu_ReturnType
```

Enumerator

OCU_CM_IN_REF_INTERVAL	
OCU_CM_OUT_REF_INTERVAL	

Definition at line 95 of file Ocu.h.

5.28.5 Function Documentation**5.28.5.1 Ocu_DeInit()**

```
void Ocu_DeInit (
    void )
```

: Ocu_DeInit: This function de-initializes the OCU module.

Note

Function ID: DES_OCU_API_002
Service ID: 01

Parameters

in	<i>none</i>	
in, out		

5.28.5.2 Ocu_DisableNotification()

```
void Ocu_DisableNotification (
    Ocu_ChannelType ChannelNumber )
```

: Ocu_DisableNotification: used to disable notifications from an OCU channel.

Note

Function ID: DES_OCU_API_010
Service ID: 10

Parameters

in		
----	--	--

5.28.5.3 Ocu_EnableNotification()

```
void Ocu_EnableNotification (
    Ocu_ChannelType ChannelNumber )
```

: Ocu_EnableNotification: used to enable notifications from an OCU channel.

Note

Function ID: DES_OCU_API_011
Service ID: 11

Parameters

in		
----	--	--

5.28.5.4 Ocu_GetCounter()

```
Ocu_ValueType Ocu_GetCounter (
    Ocu_ChannelType ChannelNumber )
```

: Ocu_GetCounter: Service to read the current value of the counter.

Note

Function ID: DES_OCU_API_007
Service ID: 06

Parameters

in		
----	--	--

5.28.5.5 Ocu_GetVersionInfo()

```
void Ocu_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

: Ocu_GetVersionInfo: returns the version information of this module.

Note

Function ID: DES_OCU_API_012
Service ID: 09

Parameters

in		
----	--	--

5.28.5.6 Ocu_Init()

```
void Ocu_Init (
    const Ocu_ConfigType * ConfigPtr )
```

: Ocu_Init: Service for OCU initialization.

Note

Function ID: DES_OCU_API_001
Service ID: 00

Parameters

in	config	pointer to OCU configure variable
in, out		

5.28.5.7 Ocu_SetAbsoluteThreshold()

```
Ocu_ReturnType Ocu_SetAbsoluteThreshold (
```

```
Ocu_ChannelType ChannelNumber,  
Ocu_ValueType ReferenceValue,  
Ocu_ValueType AbsoluteValue )
```

: Ocu_SetAbsoluteThreshold: set the value of the channel threshold using an absolute input data.

Note

Function ID: DES_OCU_API_008
Service ID: 07

Parameters

in		
----	--	--

5.28.5.8 Ocu_SetPinAction()

```
void Ocu_SetPinAction (  
    Ocu_ChannelType ChannelNumber,  
    Ocu_PinActionType PinAction )
```

: Ocu_SetPinAction: indicate the driver what shall be done automatically by hardware upon compare match.

Note

Function ID: DES_OCU_API_006
Service ID: 05

Parameters

in		
----	--	--

5.28.5.9 Ocu_SetPinState()

```
void Ocu_SetPinState (  
    Ocu_ChannelType ChannelNumber,  
    Ocu_PinStateType PinState )
```

: set immediately the level of the pin associated to an OCU channel.

Note

Function ID: DES_OCU_API_005
Service ID: 04

Parameters

in		
----	--	--

5.28.5.10 Ocu_SetRelativeThreshold()

```
Ocu_ReturnType Ocu_SetRelativeThreshold (
    Ocu_ChannelType ChannelNumber,
    Ocu_ValueType RelativeValue )
```

: Ocu_SetRelativeThreshold: set the value of the channel threshold relative to the current value of the counter.

Note

Function ID: DES_OCU_API_009

Service ID: 08

Parameters

in		
----	--	--

5.28.5.11 Ocu_StartChannel()

```
void Ocu_StartChannel (
    Ocu_ChannelType ChannelNumber )
```

: Ocu_StartChannel: The Service to start an OCU channel.

Note

Function ID: DES_OCU_API_003

Service ID: 02

Parameters

in		
----	--	--

5.28.5.12 Ocu_StopChannel()

```
void Ocu_StopChannel (
    Ocu_ChannelType ChannelNumber )
```

: Ocu_StopChannel: Service to stop an OCU channel.

Note

Function ID: DES_OCU_API_004

Service ID: 03

Parameters

in		
----	--	--

5.28.6 Variable Documentation

5.28.6.1 Ocu_GenerateConfigPC

```
const Ocu_ConfigType Ocu_GenerateConfigPC
```

5.29 Oslf.h File Reference

This file provides extern Oslf API.

```
#include "StandardTypes.h"
```

Macros

- `#define OSIF_SW_MAJOR_VERSION (1U)`
- `#define OSIF_SW_MINOR_VERSION (0U)`
- `#define OSIF_SW_PATCH_VERSION (0U)`
- `#define OSIF_BAREMETAL 0U`
osif used in baremetal
- `#define OSIF_OS 1U`
osif used in os
- `#define OS_PLATFORM OSIF_BAREMETAL`
os use osif need define OS_PLATFORM equal to OSIF_OS, default used to baremetal

Functions

- void `Oslf_Init` (void)
Initialize SysTick.
- void `Oslf_Deinit` (void)
Deinitialize SysTick.
- uint16 `Oslf_GetCoreId` (void)
get current cpu id

5.29.1 Detailed Description

This file provides extern Oslf API.

5.29.2 Macro Definition Documentation

5.29.2.1 OS_PLATFORM

```
#define OS_PLATFORM OSIF_BAREMETAL
```

os use osif need define OS_PLATFORM equal to OSIF_OS, default used to baremetal

Definition at line 67 of file Oslf.h.

5.29.2.2 OSIF_BAREMETAL

```
#define OSIF_BAREMETAL 0U
```

osif usesd in baremetal

Definition at line 60 of file Oslf.h.

5.29.2.3 OSIF_OS

```
#define OSIF_OS 1U
```

osif usesd in os

Definition at line 63 of file Oslf.h.

5.29.2.4 OSIF_SW_MAJOR_VERSION

```
#define OSIF_SW_MAJOR_VERSION (1U)
```

Definition at line 50 of file Oslf.h.

5.29.2.5 OSIF_SW_MINOR_VERSION

```
#define OSIF_SW_MINOR_VERSION (0U)
```

Definition at line 51 of file Oslf.h.

5.29.2.6 OSIF_SW_PATCH_VERSION

```
#define OSIF_SW_PATCH_VERSION (0U)
```

Definition at line 52 of file Oslf.h.

5.29.3 Function Documentation

5.29.3.1 Oslf_Deinit()

```
void OsIf_Deinit (  
    void )
```

Deinitialize Systick.

Note

Function ID: DES_OSIF_API_209

Returns

void

5.29.3.2 Oslf_GetCoreId()

```
uint16 OsIf_GetCoreId (  
    void )
```

get current cpu id

Note

Function ID: DES_OSIF_API_201

Returns

uint16: current cpu id

5.29.3.3 Oslf_Init()

```
void OsIf_Init (
    void )
```

Initialize SysTick.

Note

Function ID: DES_OSIF_API_208

Returns

void

5.30 Oslf_Critical.h File Reference

This file provides extern Oslf API.

```
#include "StandardTypes.h"
#include "Conf_AC784xx.h"
```

Macros

- #define [OSIF_CRITICAL_SW_MAJOR_VERSION](#) (1U)
- #define [OSIF_CRITICAL_SW_MINOR_VERSION](#) (1U)
- #define [OSIF_CRITICAL_SW_PATCH_VERSION](#) (0U)
- #define [WDG_HAL_CS_ID1](#) (1U)
ID1 for Watchdog Critical Section.
- #define [WDG_HAL_CS_ID2](#) (2U)
ID2 for Watchdog Critical Section.
- #define [WDG_HAL_CS_ID3](#) (3U)
ID3 for Watchdog Critical Section.
- #define [DMA_HAL_ID1](#) (4U)
ID1 for DMA Critical Section.
- #define [DMA_HAL_ID2](#) (5U)
ID2 for DMA Critical Section.
- #define [CRYPTO_MCAL_ID](#) (0U)
ID for Crypto Critical Section.
- #define [CMU_HAL_ID1](#) (1U)
ID1 for CMU Critical Section.
- #define [FLS_HAL_ID1](#) (1U)
ID1 for FLS Critical Section.
- #define [UART_HAL_ID1](#) (1U)
ID1 for FLS Critical Section.
- #define [FLSTST_HAL_ID1](#) (1U)
ID1 for FlsTst Critical Section.
- #define [OSIF_ENTER_CRITICAL](#)(AREA_ID) SchM_Enter_##AREA_ID##_ProtectDataArea()
enter ctirical protect area
- #define [OSIF_EXIT_CRITICAL](#)(AREA_ID) SchM_Exit_##AREA_ID##_ProtectDataArea()
exit ctirical protect area
- #define [OSIF_ENTER_CRITICAL_PROTOTYPES](#)(AREA_ID) void SchM_Enter_##AREA_ID##_ProtectData↔Area(void)
- #define [OSIF_EXIT_CRITICAL_PROTOTYPES](#)(AREA_ID) void SchM_Exit_##AREA_ID##_ProtectDataArea(void)

Functions

- [OSIF_ENTER_CRITICAL_PROTOTYPES \(WDG_HAL_CS_ID1\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(WDG_HAL_CS_ID1\)](#)
- [OSIF_ENTER_CRITICAL_PROTOTYPES \(WDG_HAL_CS_ID2\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(WDG_HAL_CS_ID2\)](#)
- [OSIF_ENTER_CRITICAL_PROTOTYPES \(WDG_HAL_CS_ID3\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(WDG_HAL_CS_ID3\)](#)
- [OSIF_ENTER_CRITICAL_PROTOTYPES \(DMA_HAL_ID1\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(DMA_HAL_ID1\)](#)
- [OSIF_ENTER_CRITICAL_PROTOTYPES \(DMA_HAL_ID2\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(DMA_HAL_ID2\)](#)
- [OSIF_ENTER_CRITICAL_PROTOTYPES \(CMU_HAL_ID1\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(CMU_HAL_ID1\)](#)
- [OSIF_ENTER_CRITICAL_PROTOTYPES \(FLS_HAL_ID1\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(FLS_HAL_ID1\)](#)
- [OSIF_ENTER_CRITICAL_PROTOTYPES \(UART_HAL_ID1\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(UART_HAL_ID1\)](#)
- [OSIF_ENTER_CRITICAL_PROTOTYPES \(CRYPTO_MCAL_ID\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(CRYPTO_MCAL_ID\)](#)
- [OSIF_ENTER_CRITICAL_PROTOTYPES \(FLSTST_HAL_ID1\)](#)
- [OSIF_EXIT_CRITICAL_PROTOTYPES \(FLSTST_HAL_ID1\)](#)

5.30.1 Detailed Description

This file provides extern Oslf API.

5.30.2 Macro Definition Documentation

5.30.2.1 CMU_HAL_ID1

```
#define CMU_HAL_ID1 (1U)
```

ID1 for CMU Critical Section.

Definition at line 80 of file Oslf_Critical.h.

5.30.2.2 CRYPTO_MCAL_ID

```
#define CRYPTO_MCAL_ID (0U)
```

ID for Crypto Critical Section.

Definition at line 77 of file Oslf_Critical.h.

5.30.2.3 DMA_HAL_ID1

```
#define DMA_HAL_ID1 (4U)
```

ID1 for DMA Critical Section.

Definition at line 71 of file Oslf_Critical.h.

5.30.2.4 DMA_HAL_ID2

```
#define DMA_HAL_ID2 (5U)
```

ID2 for DMA Critical Section.

Definition at line 74 of file Oslf_Critical.h.

5.30.2.5 FLS_HAL_ID1

```
#define FLS_HAL_ID1 (1U)
```

ID1 for FLS Critical Section.

Definition at line 83 of file Oslf_Critical.h.

5.30.2.6 FLSTST_HAL_ID1

```
#define FLSTST_HAL_ID1 (1U)
```

ID1 for FlsTst Critical Section.

Definition at line 89 of file Oslf_Critical.h.

5.30.2.7 OSIF_CRITICAL_SW_MAJOR_VERSION

```
#define OSIF_CRITICAL_SW_MAJOR_VERSION (1U)
```

Definition at line 51 of file Oslf_Critical.h.

5.30.2.8 OSIF_CRITICAL_SW_MINOR_VERSION

```
#define OSIF_CRITICAL_SW_MINOR_VERSION (1U)
```

Definition at line 52 of file Oslf_Critical.h.

5.30.2.9 OSIF_CRITICAL_SW_PATCH_VERSION

```
#define OSIF_CRITICAL_SW_PATCH_VERSION (0U)
```

Definition at line 53 of file Oslf_Critical.h.

5.30.2.10 OSIF_ENTER_CRITICAL

```
#define OSIF_ENTER_CRITICAL(  
    AREA_ID ) SchM_Enter_##AREA_ID##_ProtectDataArea()
```

enter ctirical protect area

Definition at line 103 of file Oslf_Critical.h.

5.30.2.11 OSIF_ENTER_CRITICAL_PROTOTYPES

```
#define OSIF_ENTER_CRITICAL_PROTOTYPES(  
    AREA_ID ) void SchM_Enter_##AREA_ID##_ProtectDataArea(void)
```

Definition at line 108 of file Oslf_Critical.h.

5.30.2.12 OSIF_EXIT_CRITICAL

```
#define OSIF_EXIT_CRITICAL(  
    AREA_ID ) SchM_Exit_##AREA_ID##_ProtectDataArea()
```

exit ctirical protect area

Definition at line 106 of file Oslf_Critical.h.

5.30.2.13 OSIF_EXIT_CRITICAL_PROTOTYPES

```
#define OSIF_EXIT_CRITICAL_PROTOTYPES(  
    AREA_ID ) void SchM_Exit_##AREA_ID##_ProtectDataArea(void)
```

Definition at line 109 of file Oslf_Critical.h.

5.30.2.14 UART_HAL_ID1

```
#define UART_HAL_ID1 (1U)
```

ID1 for FLS Critical Section.

Definition at line 86 of file Oslf_Critical.h.

5.30.2.15 WDG_HAL_CS_ID1

```
#define WDG_HAL_CS_ID1 (1U)
```

ID1 for Watchdog Critical Section.

Definition at line 62 of file Oslf_Critical.h.

5.30.2.16 WDG_HAL_CS_ID2

```
#define WDG_HAL_CS_ID2 (2U)
```

ID2 for Watchdog Critical Section.

Definition at line 65 of file Oslf_Critical.h.

5.30.2.17 WDG_HAL_CS_ID3

```
#define WDG_HAL_CS_ID3 (3U)
```

ID3 for Watchdog Critical Section.

Definition at line 68 of file Oslf_Critical.h.

5.30.3 Function Documentation

5.30.3.1 OSIF_ENTER_CRITICAL_PROTOTYPES() [1/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (  
    WDG_HAL_CS_ID1 )
```

5.30.3.2 OSIF_ENTER_CRITICAL_PROTOTYPES() [2/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (
    WDG_HAL_CS_ID2 )
```

5.30.3.3 OSIF_ENTER_CRITICAL_PROTOTYPES() [3/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (
    WDG_HAL_CS_ID3 )
```

5.30.3.4 OSIF_ENTER_CRITICAL_PROTOTYPES() [4/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (
    DMA_HAL_ID1 )
```

5.30.3.5 OSIF_ENTER_CRITICAL_PROTOTYPES() [5/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (
    DMA_HAL_ID2 )
```

5.30.3.6 OSIF_ENTER_CRITICAL_PROTOTYPES() [6/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (
    CMU_HAL_ID1 )
```

5.30.3.7 OSIF_ENTER_CRITICAL_PROTOTYPES() [7/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (
    FLS_HAL_ID1 )
```

5.30.3.8 OSIF_ENTER_CRITICAL_PROTOTYPES() [8/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (
    UART_HAL_ID1 )
```

5.30.3.9 OSIF_ENTER_CRITICAL_PROTOTYPES() [9/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (
    CRYPTO_MCAL_ID )
```

5.30.3.10 OSIF_ENTER_CRITICAL_PROTOTYPES() [10/10]

```
OSIF_ENTER_CRITICAL_PROTOTYPES (
    FLSTST_HAL_ID1 )
```

5.30.3.11 OSIF_EXIT_CRITICAL_PROTOTYPES() [1/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    WDG_HAL_CS_ID1 )
```

5.30.3.12 OSIF_EXIT_CRITICAL_PROTOTYPES() [2/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    WDG_HAL_CS_ID2 )
```

5.30.3.13 OSIF_EXIT_CRITICAL_PROTOTYPES() [3/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    WDG_HAL_CS_ID3 )
```

5.30.3.14 OSIF_EXIT_CRITICAL_PROTOTYPES() [4/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    DMA_HAL_ID1 )
```

5.30.3.15 OSIF_EXIT_CRITICAL_PROTOTYPES() [5/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    DMA_HAL_ID2 )
```


5.30.3.16 OSIF_EXIT_CRITICAL_PROTOTYPES() [6/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    CMU_HAL_ID1 )
```

5.30.3.17 OSIF_EXIT_CRITICAL_PROTOTYPES() [7/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    FLS_HAL_ID1 )
```

5.30.3.18 OSIF_EXIT_CRITICAL_PROTOTYPES() [8/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    UART_HAL_ID1 )
```

5.30.3.19 OSIF_EXIT_CRITICAL_PROTOTYPES() [9/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    CRYPTO_MCAL_ID )
```

5.30.3.20 OSIF_EXIT_CRITICAL_PROTOTYPES() [10/10]

```
OSIF_EXIT_CRITICAL_PROTOTYPES (
    FLSTST_HAL_ID1 )
```

5.31 Oslf_Irq.h File Reference

This file provides extern Oslf Irq API.

```
#include "StandardTypes.h"
#include "Device_Register.h"
```

Macros

- `#define OSIF_IRQ_SW_MAJOR_VERSION (1U)`
- `#define OSIF_IRQ_SW_MINOR_VERSION (0U)`
- `#define OSIF_IRQ_SW_PATCH_VERSION (1U)`
- `#define ISR(IsrName) INTERRUPT_FUNC void IsrName(void)`
used to define irq handler

Functions

- LOCAL_INLINE void [Oslf_SuspendAllInterrupts](#) (void)
Resume all interrupts of system.
- LOCAL_INLINE void [Oslf_ResumeAllInterrupts](#) (void)
Resume all interrupts of system.

Variables

- volatile uint32 [Oslf_PriMaskValue](#)
- volatile sint32 [Oslf_CriticalNesting](#)

5.31.1 Detailed Description

This file provides extern Oslf Irq API.

5.31.2 Macro Definition Documentation

5.31.2.1 ISR

```
#define ISR(  
    IsrName ) INTERRUPT_FUNC void IsrName(void)
```

used to define irq handler

Definition at line 61 of file Oslf_Irq.h.

5.31.2.2 OSIF_IRQ_SW_MAJOR_VERSION

```
#define OSIF_IRQ_SW_MAJOR_VERSION (1U)
```

Definition at line 51 of file Oslf_Irq.h.

5.31.2.3 OSIF_IRQ_SW_MINOR_VERSION

```
#define OSIF_IRQ_SW_MINOR_VERSION (0U)
```

Definition at line 52 of file Oslf_Irq.h.

5.31.2.4 OSIF_IRQ_SW_PATCH_VERSION

```
#define OSIF_IRQ_SW_PATCH_VERSION (1U)
```

Definition at line 53 of file Oslf_Irq.h.

5.31.3 Function Documentation

5.31.3.1 Oslf_ResumeAllInterrupts()

```
LOCAL_INLINE void Oslf_ResumeAllInterrupts (  
    void )
```

Resume all interrupts of system.

Note

Function ID: DES_OSIF_API_202

Returns

void

Definition at line 96 of file Oslf_Irq.h.

5.31.3.2 Oslf_SuspendAllInterrupts()

```
LOCAL_INLINE void Oslf_SuspendAllInterrupts (  
    void )
```

Resume all interrupts of system.

Note

Function ID: DES_OSIF_API_203

Returns

void

Definition at line 77 of file Oslf_Irq.h.

5.31.4 Variable Documentation

5.31.4.1 Oslf_CriticalNesting

```
volatile sint32 OsIf_CriticalNesting
```

5.31.4.2 Oslf_PriMaskValue

```
volatile uint32 OsIf_PriMaskValue
```

5.32 Oslf_Time.h File Reference

This file provides Oslf Time API.

```
#include "StandardTypes.h"
```

Macros

- `#define OSIF_TIME_SW_MAJOR_VERSION (1U)`
- `#define OSIF_TIME_SW_MINOR_VERSION (0U)`
- `#define OSIF_TIME_SW_PATCH_VERSION (0U)`

Functions

- `uint32 Oslf_GetCounter (void)`
Get the current value of the counter.
- `uint32 Oslf_GetElapsed (uint32 *const CurrentRef)`
Get the delta time in ticks compared to a reference, and updates the reference.
- `uint32 Oslf_MicrosToTicks (uint32 Micros)`
Converts a value from microsecond units to ticks units.
- `void Oslf_UDelay (uint32 Micros)`
Microseconds delay.

5.32.1 Detailed Description

This file provides Oslf Time API.

5.32.2 Macro Definition Documentation

5.32.2.1 OSIF_TIME_SW_MAJOR_VERSION

```
#define OSIF_TIME_SW_MAJOR_VERSION (1U)
```

Definition at line 50 of file Oslf_Time.h.

5.32.2.2 OSIF_TIME_SW_MINOR_VERSION

```
#define OSIF_TIME_SW_MINOR_VERSION (0U)
```

Definition at line 51 of file Oslf_Time.h.

5.32.2.3 OSIF_TIME_SW_PATCH_VERSION

```
#define OSIF_TIME_SW_PATCH_VERSION (0U)
```

Definition at line 52 of file Oslf_Time.h.

5.32.3 Function Documentation

5.32.3.1 Oslf_GetCounter()

```
uint32 OsIf_GetCounter (  
    void )
```

Get the current value of the counter.

Note

Function ID: DES_OSIF_API_207

Returns

The current value of the counter

5.32.3.2 Oslf_GetElapsed()

```
uint32 OsIf_GetElapsed (  
    uint32 *const CurrentRef )
```

Get the delta time in ticks compared to a reference, and updates the reference.

Note

Function ID: DES_OSIF_API_206

Parameters

<i>in, out</i>	<i>CurrentRef</i>	reference counter value, updated to current counter value
----------------	-------------------	-----------------------------------------------------------

Returns

The elapsed time

5.32.3.3 OsIf_MicrosToTicks()

```
uint32 OsIf_MicrosToTicks (
    uint32 Micros )
```

Converts a value from microsecond units to ticks units.

Note

Function ID: DES_OSIF_API_204

Parameters

in	<i>Micros</i>	microseconds value (multiple of 1000 can be convert to tick)
----	---------------	--------------------------------------------------------------

Returns

uint32: ticks value

5.32.3.4 OsIf_UDelay()

```
void OsIf_UDelay (
    uint32 Micros )
```

Microseconds delay.

Note

Function ID: DES_OSIF_API_205

Parameters

in	<i>Micros</i>	microseconds value
----	---------------	--------------------

Returns

void

5.33 Port.h File Reference

This file provides extern Port Mcal API.

```
#include "Port_Ipw.h"
```

Functions

- void `Port_Init` (const `Port_ConfigType` *ConfigPtr)
Initializes the Port Driver module.
- void `Port_SetPinDirection` (`Port_PinType` Pin, `Port_PinDirectionType` Direction)
Sets the port pin direction.
- void `Port_SetPinMode` (`Port_PinType` Pin, `Port_PinModeType` Mode)
Sets the port pin mode.
- void `Port_GetVersionInfo` (`Std_VersionInfoType` *versioninfo)
Returns the version information of this module.
- void `Port_RefreshPortDirection` (void)
Refreshes port direction.

5.33.1 Detailed Description

This file provides extern Port Mcal API.

5.33.2 Function Documentation

5.33.2.1 Port_GetVersionInfo()

```
void Port_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

Returns the version information of this module.

Note

Function ID : DES_PORT_API_005
Service ID : 0x03

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>versioninfo</i>	Pointer to where to store the version information of this module

Returns

None

5.33.2.2 Port_Init()

```
void Port_Init (
    const Port_ConfigType * ConfigPtr )
```

Initializes the Port Driver module.

Note

Function ID : DES_PORT_API_001
Service ID : 0x00

Parameters

in	<i>ConfigPtr</i>	Pointer to configuration set.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.33.2.3 Port_RefreshPortDirection()

```
void Port_RefreshPortDirection (
    void )
```

Refreshes port direction.

Note

Function ID : DES_PORT_API_004
Service ID : 0x02

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.33.2.4 Port_SetPinDirection()

```
void Port_SetPinDirection (
    Port_PinType Pin,
    Port_PinDirectionType Direction )
```

Sets the port pin direction.

Note

Function ID : DES_PORT_API_002
Service ID : 0x01

Parameters

in	<i>Pin</i>	Port Pin ID number.
in	<i>Direction</i>	Port Pin Direction.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.33.2.5 Port_SetPinMode()

```
void Port_SetPinMode (
    Port_PinType Pin,
    Port_PinModeType Mode )
```

Sets the port pin mode.

Note

Function ID : DES_PORT_API_003
Service ID : 0x04

Parameters

in	<i>Pin</i>	Port Pin ID number.
in	<i>Mode</i>	New Port Pin mode to be set on port pin.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.34 Port_GeneralTypes.h File Reference

This file provides extern Port module structure and macro.

```
#include "Mcal.h"
#include "Device_Register.h"
```

Classes

- struct [Port_DigitalFilterConfigType](#)
Port digital filter config type.

- struct [Port_PinConfigType](#)
each used Pin configure type
- struct [Port_UnUsedPinConfigType](#)
each unused Pin configure type
- struct [Port_ConfigType](#)
all port module configure type

Macros

- `#define GPIO_FUN0 0U /* Disable */`
- `#define GPIO_FUN1 1U /* GPIO */`
- `#define GPIO_FUN2 2U /* reuse mode two */`
- `#define GPIO_FUN3 3U /* reuse mode three */`
- `#define GPIO_FUN4 4U /* reuse mode four */`
- `#define GPIO_FUN5 5U /* reuse mode fix */`
- `#define GPIO_FUN6 6U /* reuse mode six */`
- `#define GPIO_FUN7 7U /* reuse mode seven */`
- `#define PORT_INSTANCE_ID ((uint8)0x0)`
Instance ID of port driver.
- `#define PORT_INIT_ID ((uint8)0x00)`
API service ID for PORT Init function.
- `#define PORT_SETPINDIRECTION_ID ((uint8)0x01)`
API service ID for PORT set pin direction function.
- `#define PORT_REFRESHPINDIRECTION_ID ((uint8)0x02)`
API service ID for PORT refresh pin direction function.
- `#define PORT_GETVERSIONINFO_ID ((uint8)0x03)`
API service ID for PORT get version info function.
- `#define PORT_SETPINMODE_ID ((uint8)0x04)`
API service ID for PORT set pin mode.
- `#define PORT_E_PARAM_PIN ((uint8)0x0A)`
Det Error value, returned by Port_SetPinDirection and Port_PinMode if an wrong PortPin ID is passed.
- `#define PORT_E_DIRECTION_UNCHANGEABLE ((uint8)0x0B)`
- `#define PORT_E_INIT_FAILED ((uint8)0x0C)`
Det Error value, returned by Port_Init function if Port_Init is called with wrong parameter.
- `#define PORT_E_PARAM_INVALID_MODE ((uint8)0x0D)`
Det Error value, returned by Port_SetPinMode function if the passed PortPinMode is invalid.
- `#define PORT_E_MODE_UNCHANGEABLE ((uint8)0x0E)`
Det Error value, returned by Port_SetPinMode function if the passed PortPin have a unchangeable Mode.
- `#define PORT_E_UNINIT ((uint8)0x0F)`
Det Error value, returned by a function if API service called prior to module initialization.
- `#define PORT_E_PARAM_POINTER ((uint8)0x10)`
Det Error value, returned by Port_GetVersionInfo function if API is called with NULL Pointer Parameter.

Typedefs

- typedef uint32 [Port_PinType](#)
Port Pin ID number.
- typedef uint8 [Port_PinModeType](#)
Port Pin mode type.

Enumerations

- enum [Port_PinDirectionType](#) { [PORT_PIN_IN](#), [PORT_PIN_OUT](#) }
pin direction define

5.34.1 Detailed Description

This file provides extern Port module structure and macro.

5.34.2 Macro Definition Documentation

5.34.2.1 GPIO_FUN0

```
#define GPIO_FUN0 0U /* Disable */
```

Definition at line 57 of file Port_GeneralTypes.h.

5.34.2.2 GPIO_FUN1

```
#define GPIO_FUN1 1U /* GPIO */
```

Definition at line 58 of file Port_GeneralTypes.h.

5.34.2.3 GPIO_FUN2

```
#define GPIO_FUN2 2U /* reuse mode two */
```

Definition at line 59 of file Port_GeneralTypes.h.

5.34.2.4 GPIO_FUN3

```
#define GPIO_FUN3 3U /* reuse mode three */
```

Definition at line 60 of file Port_GeneralTypes.h.

5.34.2.5 GPIO_FUN4

```
#define GPIO_FUN4 4U /* reuse mode four */
```

Definition at line 61 of file Port_GeneralTypes.h.

5.34.2.6 GPIO_FUN5

```
#define GPIO_FUN5 5U /* reuse mode fix */
```

Definition at line 62 of file Port_GeneralTypes.h.

5.34.2.7 GPIO_FUN6

```
#define GPIO_FUN6 6U /* reuse mode six */
```

Definition at line 63 of file Port_GeneralTypes.h.

5.34.2.8 GPIO_FUN7

```
#define GPIO_FUN7 7U /* reuse mode seven */
```

Definition at line 64 of file Port_GeneralTypes.h.

5.34.2.9 PORT_E_DIRECTION_UNCHANGEABLE

```
#define PORT_E_DIRECTION_UNCHANGEABLE ((uint8)0x0B)
```

Definition at line 90 of file Port_GeneralTypes.h.

5.34.2.10 PORT_E_INIT_FAILED

```
#define PORT_E_INIT_FAILED ((uint8)0x0C)
```

Det Error value, returned by Port_Init function if Port_Init is called with wrong parameter.

Definition at line 94 of file Port_GeneralTypes.h.

5.34.2.11 PORT_E_MODE_UNCHANGEABLE

```
#define PORT_E_MODE_UNCHANGEABLE ((uint8)0x0E)
```

Det Error value, returned by Port_SetPinMode function if the passed PortPin have a unchangeable Mode.

Definition at line 102 of file Port_GeneralTypes.h.

5.34.2.12 PORT_E_PARAM_INVALID_MODE

```
#define PORT_E_PARAM_INVALID_MODE ((uint8)0x0D)
```

Det Error value, returned by Port_SetPinMode function if the passed PortPinMode is invalid.

Definition at line 98 of file Port_GeneralTypes.h.

5.34.2.13 PORT_E_PARAM_PIN

```
#define PORT_E_PARAM_PIN ((uint8)0x0A)
```

Det Error value, returned by Port_SetPinDirection and Port_PinMode if an wrong PortPin ID is passed.

Definition at line 86 of file Port_GeneralTypes.h.

5.34.2.14 PORT_E_PARAM_POINTER

```
#define PORT_E_PARAM_POINTER ((uint8)0x10)
```

Det Error value, returned by Port_GetVersionInfo function if API is called with NULL Pointer Parameter.

Definition at line 110 of file Port_GeneralTypes.h.

5.34.2.15 PORT_E_UNINIT

```
#define PORT_E_UNINIT ((uint8)0x0F)
```

Det Error value, returned by a function if API service called prior to module initialization.

Definition at line 106 of file Port_GeneralTypes.h.

5.34.2.16 PORT_GETVERSIONINFO_ID

```
#define PORT_GETVERSIONINFO_ID ((uint8)0x03)
```

API service ID for PORT get version info function.

Definition at line 79 of file Port_GeneralTypes.h.

5.34.2.17 PORT_INIT_ID

```
#define PORT_INIT_ID ((uint8)0x00)
```

API service ID for PORT Init function.

Definition at line 70 of file Port_GeneralTypes.h.

5.34.2.18 PORT_INSTANCE_ID

```
#define PORT_INSTANCE_ID ((uint8)0x0)
```

Instance ID of port driver.

Definition at line 67 of file Port_GeneralTypes.h.

5.34.2.19 PORT_REFRESHPINIRECTION_ID

```
#define PORT_REFRESHPINIRECTION_ID ((uint8)0x02)
```

API service ID for PORT refresh pin direction function.

Definition at line 76 of file Port_GeneralTypes.h.

5.34.2.20 PORT_SETPINDIRECTION_ID

```
#define PORT_SETPINDIRECTION_ID ((uint8)0x01)
```

API service ID for PORT set pin direction function.

Definition at line 73 of file Port_GeneralTypes.h.

5.34.2.21 PORT_SETPINMODE_ID

```
#define PORT_SETPINMODE_ID ((uint8)0x04)
```

API service ID for PORT set pin mode.

Definition at line 82 of file Port_GeneralTypes.h.

5.34.3 Typedef Documentation

5.34.3.1 Port_PinModeType

```
typedef uint8 Port_PinModeType
```

Port Pin mode type.

Definition at line 124 of file Port_GeneralTypes.h.

5.34.3.2 Port_PinType

```
typedef uint32 Port_PinType
```

Port Pin ID number.

Definition at line 121 of file Port_GeneralTypes.h.

5.34.4 Enumeration Type Documentation

5.34.4.1 Port_PinDirectionType

```
enum Port_PinDirectionType
```

pin direction define

Enumerator

PORT_PIN_IN	Sets port pin as input.
PORT_PIN_OUT	Sets port pin as output.

Definition at line 113 of file Port_GeneralTypes.h.

5.35 Pwm.h File Reference

This file provides all mcal Pwm api.

```
#include "Pwm_Cfg.h"
```

Classes

- struct [Pwm_ChannelConfigType](#)
- struct [Pwm_ModuleConfigType](#)
- struct [Pwm_IpConfigType](#)
- struct [Pwm_ConfigType](#)

Macros

- #define [PWM_DUTY_CYCLE_MAX](#) (0x8000U)
- #define [PWM_PERIOD_MAX](#) (0xFFFFU)

Typedefs

- typedef uint8 [Pwm_ChannelType](#)
- typedef uint32 [Pwm_PeriodType](#)

Enumerations

- enum [Pwm_OutputStateType](#) { [PWM_LOW](#) = 0U, [PWM_HIGH](#) }
- enum [Pwm_EdgeNotificationType](#) { [PWM_NO_EDGES](#) = 0U, [PWM_RISING_EDGE](#), [PWM_FALLING_EDGE](#), [PWM_BOTH_EDGES](#) }
- enum [Pwm_ChannelClassType](#) { [PWM_VARIABLE_PERIOD](#) = 0U, [PWM_FIXED_PERIOD](#), [PWM_FIXED_PERIOD_SHIFTED](#) }
- enum [Pwm_ServiceIdType](#) { [PWM_INIT_ID](#) = 0x00U, [PWM_DEINIT_ID](#), [PWM_SETDUTYCYCLE_ID](#), [PWM_SETPERIODANDDUTY_ID](#), [PWM_SETOUTPUTIDLE_ID](#), [PWM_GETOUTPUTSTATE_ID](#), [PWM_DISABLENOTIFICATION_ID](#), [PWM_ENABLENOTIFICATION_ID](#), [PWM_GETVERSIONINFO_ID](#) }

Functions

- void [Pwm_Init](#) (const [Pwm_ConfigType](#) *ConfigPtr)
: *Pwm_Init: service for PWM initialization*
- void [Pwm_DeInit](#) (void)
: *Pwm_DeInit: service for PWM De-Initialization*
- void [Pwm_SetDutyCycle](#) ([Pwm_ChannelType](#) ChannelNumber, uint16 DutyCycle)
: *Pwm_SetDutyCycle: set the duty cycle of the PWM channel*
- void [Pwm_SetPeriodAndDuty](#) ([Pwm_ChannelType](#) ChannelNumber, [Pwm_PeriodType](#) Period, uint16 DutyCycle)
: *Pwm_SetPeriodAndDuty: set the duty cycle and period of the PWM channel*
- void [Pwm_SetOutputToldle](#) ([Pwm_ChannelType](#) ChannelNumber)
: *Pwm_SetOutputToldle: set the PWM output to the configured Idle state*
- [Pwm_OutputStateType](#) [Pwm_GetOutputState](#) ([Pwm_ChannelType](#) ChannelNumber)
: *Pwm_GetOutputState: read the internal state of the PWM output signal*
- void [Pwm_DisableNotification](#) ([Pwm_ChannelType](#) ChannelNumber)
: *Pwm_DisableNotification: disable the PWM signal edge notification*
- void [Pwm_EnableNotification](#) ([Pwm_ChannelType](#) ChannelNumber, [Pwm_EdgeNotificationType](#) NotifyEdge)
: *Pwm_EnableNotification: enable the PWM signal edge notification*
- void [Pwm_GetVersionInfo](#) (Std_VersionInfoType *versioninfo)
: *Pwm_GetVersionInfo: check init config pointer is null or not*

Variables

- const [Pwm_ConfigType](#) [Pwm_GenerateConfigPC](#)

5.35.1 Detailed Description

This file provides all mcal Pwm api.

5.35.2 Macro Definition Documentation

5.35.2.1 PWM_DUTY_CYCLE_MAX

```
#define PWM_DUTY_CYCLE_MAX (0x8000U)
```

Definition at line 58 of file Pwm.h.

5.35.2.2 PWM_PERIOD_MAX

```
#define PWM_PERIOD_MAX (0xFFFFU)
```

Definition at line 61 of file Pwm.h.

5.35.3 Typedef Documentation

5.35.3.1 Pwm_ChannelType

```
typedef uint8 Pwm\_ChannelType
```

Definition at line 104 of file Pwm.h.

5.35.3.2 Pwm_PeriodType

```
typedef uint32 Pwm\_PeriodType
```

Definition at line 107 of file Pwm.h.

5.35.4 Enumeration Type Documentation

5.35.4.1 Pwm_ChannelClassType

```
enum Pwm\_ChannelClassType
```

Enumerator

PWM_VARIABLE_PERIOD	
PWM_FIXED_PERIOD	
PWM_FIXED_PERIOD_SHIFTED	

Definition at line 81 of file Pwm.h.

5.35.4.2 Pwm_EdgeNotificationType

```
enum Pwm_EdgeNotificationType
```

Enumerator

PWM_NO_EDGES	
PWM_RISING_EDGE	
PWM_FALLING_EDGE	
PWM_BOTH_EDGES	

Definition at line 72 of file Pwm.h.

5.35.4.3 Pwm_OutputStateType

```
enum Pwm_OutputStateType
```

Enumerator

PWM_LOW	
PWM_HIGH	

Definition at line 65 of file Pwm.h.

5.35.4.4 Pwm_ServiceIdType

```
enum Pwm_ServiceIdType
```

Enumerator

PWM_INIT_ID	
PWM_DEINIT_ID	
PWM_SETDUTYCYCLE_ID	
PWM_SETPERIODANDDUTY_ID	
PWM_SETOUTPUTTOIDLE_ID	
PWM_GETOUTPUTSTATE_ID	

Enumerator

PWM_DISABLENOTIFICATION_ID	
PWM_ENABLENOTIFICATION_ID	
PWM_GETVERSIONINFO_ID	

Definition at line 89 of file Pwm.h.

5.35.5 Function Documentation

5.35.5.1 Pwm_DeInit()

```
void Pwm_DeInit (
    void )
```

: Pwm_DeInit: service for PWM De-Initialization

Note

Function ID: DES_PWM_API_002
Service ID: 01

Parameters

in		
----	--	--

5.35.5.2 Pwm_DisableNotification()

```
void Pwm_DisableNotification (
    Pwm_ChannelType ChannelNumber )
```

: Pwm_DisableNotification: disable the PWM signal edge notification

Note

Function ID: DES_PWM_API_008
Service ID: 06

Parameters

in		
----	--	--

5.35.5.3 Pwm_EnableNotification()

```
void Pwm_EnableNotification (
    Pwm_ChannelType ChannelNumber,
    Pwm_EdgeNotificationType NotifyEdge )
```

: Pwm_EnableNotification: enable the PWM signal edge notification

Note

Function ID: DES_PWM_API_009
Service ID: 07

Parameters

in		
----	--	--

5.35.5.4 Pwm_GetOutputState()

```
Pwm_OutputStateType Pwm_GetOutputState (
    Pwm_ChannelType ChannelNumber )
```

: Pwm_GetOutputState: read the internal state of the PWM output signal

Note

Function ID: DES_PWM_API_007
Service ID: 05

Parameters

in		
----	--	--

5.35.5.5 Pwm_GetVersionInfo()

```
void Pwm_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

: Pwm_GetVersionInfo: check init config pointer is null or not

Note

Function ID: DES_PWM_API_005
Service ID: 08

Parameters

in		
----	--	--

5.35.5.6 Pwm_Init()

```
void Pwm_Init (
    const Pwm_ConfigType * ConfigPtr )
```

: Pwm_Init: service for PWM initialization

Note

Function ID: DES_PWM_API_001
Service ID: 00

Parameters

in		
----	--	--

5.35.5.7 Pwm_SetDutyCycle()

```
void Pwm_SetDutyCycle (
    Pwm_ChannelType ChannelNumber,
    uint16 DutyCycle )
```

: Pwm_SetDutyCycle: set the duty cycle of the PWM channel

Note

Function ID: DES_PWM_API_003
Service ID: 02

Parameters

in		
----	--	--

5.35.5.8 Pwm_SetOutputToIdle()

```
void Pwm_SetOutputToIdle (
    Pwm_ChannelType ChannelNumber )
```

: Pwm_SetOutputToIdle: set the PWM output to the configured Idle state

Note

Function ID: DES_PWM_API_006
Service ID: 04

Parameters

in		
----	--	--

5.35.5.9 Pwm_SetPeriodAndDuty()

```
void Pwm_SetPeriodAndDuty (
    Pwm_ChannelType ChannelNumber,
    Pwm_PeriodType Period,
    uint16 DutyCycle )
```

: Pwm_SetPeriodAndDuty: set the duty cycle and period of the PWM channel

Note

Function ID: DES_PWM_API_004
Service ID: 03

Parameters

in		
----	--	--

5.35.6 Variable Documentation**5.35.6.1 Pwm_GenerateConfigPC**

```
const Pwm_ConfigType Pwm_GenerateConfigPC
```

5.36 Spi.h File Reference

Spi driver mcal api header file.

```
#include "Spi_Ipw.h"
```

Macros

- #define `SPI_INSTANCE` ((uint8)0U)
SPI instance ID.
- #define `SPI_INIT_ID` ((uint8)0x00U)
API service ID.
- #define `SPI_DEINIT_ID` ((uint8)0x01U)
- #define `SPI_WRITEIB_ID` ((uint8)0x02U)
- #define `SPI_ASYNCTRANSMIT_ID` ((uint8)0x03U)
- #define `SPI_READIB_ID` ((uint8)0x04U)
- #define `SPI_SETUPEB_ID` ((uint8)0x05U)
- #define `SPI_GETSTATUS_ID` ((uint8)0x06U)
- #define `SPI_GETJOBRESULT_ID` ((uint8)0x07U)
- #define `SPI_GETSEQUENCERESULT_ID` ((uint8)0x08U)
- #define `SPI_GETVERSIONINFO_ID` ((uint8)0x09U)
- #define `SPI_SYNCTRANSMIT_ID` ((uint8)0x0AU)
- #define `SPI_GETHWUNITSTATUS_ID` ((uint8)0x0BU)
- #define `SPI_CANCEL_ID` ((uint8)0x0CU)
- #define `SPI_SETASYNCMODE_ID` ((uint8)0x0DU)
- #define `SPI_MAINFUNCTIONHANDLING_ID` ((uint8)0x10U)
- #define `SPI_E_PARAM_CHANNEL` ((uint8)0x0AU)
Development Errors. API service called with wrong channel.
- #define `SPI_E_PARAM_JOB` ((uint8)0x0BU)
Development Errors. API service called with wrong job.
- #define `SPI_E_PARAM_SEQ` ((uint8)0x0CU)
Development Errors. API service called with wrong sequence.
- #define `SPI_E_PARAM_LENGTH` ((uint8)0x0DU)
Development Errors. API service called with wrong length for EB.
- #define `SPI_E_PARAM_UNIT` ((uint8)0x0EU)
Development Errors. API service called with wrong hardware unit.
- #define `SPI_E_PARAM_POINTER` ((uint8)0x10U)
Development Errors. APIs called with a Null Pointer.
- #define `SPI_E_UNINIT` ((uint8)0x1AU)
Development Errors. API service used without module initialization.
- #define `SPI_E_ALREADY_INITIALIZED` ((uint8)0x4AU)
Development Errors. API SPI_Init service called while the SPI driver has already been initialized.
- #define `SPI_E_SEQ_PENDING` ((uint8)0x2AU)
Runtime Errors. Services called in a wrong sequence.
- #define `SPI_E_SEQ_IN_PROCESS` ((uint8)0x3AU)
Runtime Errors. Synchronous transmission service called at wrong time.
- #define `SPI_ALL_HW_MAP` ((uint32)0xFFFFFFFFU)
Map of all HWIDs.

Functions

- void `Spi_Init` (const Spi_ConfigType *ConfigPtr)
Service for SPI initialization.
- Std_ReturnType `Spi_DeInit` (void)
Service for SPI de-initialization.
- Std_ReturnType `Spi_WriteIB` (Spi_ChannelType Channel, const Spi_DataBufferType *DataBufferPtr)
Service for writing one or more data to an IB SPI Handler/Driver Channel specified by parameter.
- Std_ReturnType `Spi_AsyncTransmit` (Spi_SequenceType Sequence)
Service to transmit data on the SPI bus.

- Std_ReturnType [Spi_ReadIB](#) (Spi_ChannelType Channel, Spi_DataBufferType *DataBufferPointer)
Service for reading synchronously one or more data from an IB SPI Handler/Driver Channel specified by parameter.
- Std_ReturnType [Spi_SetupEB](#) (Spi_ChannelType Channel, const Spi_DataBufferType *SrcDataBufferPtr, Spi_DataBufferType *DesDataBufferPtr, Spi_NumberOfDataType Length)
Service to setup the buffers and the length of data for the EB SPI Handler/Driver Channel specified.
- Spi_StatusType [Spi_GetStatus](#) (void)
Service returns the SPI Handler/Driver software module status.
- Spi_JobResultType [Spi_GetJobResult](#) (Spi_JobType Job)
This service returns the last transmission result of the specified Job.
- Spi_SeqResultType [Spi_GetSequenceResult](#) (Spi_SequenceType Sequence)
This service returns the last transmission result of the specified Sequence.
- void [Spi_GetVersionInfo](#) (Std_VersionInfoType *versioninfo)
Service returns the version information of this module.
- Std_ReturnType [Spi_SyncTransmit](#) (Spi_SequenceType Sequence)
Service to transmit data on the SPI bus.
- Spi_StatusType [Spi_GetHWUnitStatus](#) (Spi_HWUnitType HWUnit)
This service returns the status of the specified SPI Hardware microcontroller peripheral.
- void [Spi_Cancel](#) (Spi_SequenceType Sequence)
Service cancels the specified on-going sequence transmission.
- Std_ReturnType [Spi_SetAsyncMode](#) (Spi_AsyncModeType Mode)
Service to set the asynchronous mechanism mode for SPI busses handled asynchronously.
- void [Spi_MainFunction_Handling](#) (void)
This function shall polls the SPI interrupts linked to HW Units allocated to the transmission of SPI sequences to enable the evolution of transmission state machine.

5.36.1 Detailed Description

Spi driver mcal api header file.

5.36.2 Macro Definition Documentation

5.36.2.1 SPI_ALL_HW_MAP

```
#define SPI_ALL_HW_MAP ((uint32)0xFFFFFFFFU)
```

Map of all HWIds.

Definition at line 101 of file Spi.h.

5.36.2.2 SPI_ASYNCTRANSMIT_ID

```
#define SPI_ASYNCTRANSMIT_ID ((uint8)0x03U)
```

Definition at line 65 of file Spi.h.

5.36.2.3 SPI_CANCEL_ID

```
#define SPI_CANCEL_ID ((uint8)0x0CU)
```

Definition at line 74 of file Spi.h.

5.36.2.4 SPI_DEINIT_ID

```
#define SPI_DEINIT_ID ((uint8)0x01U)
```

Definition at line 63 of file Spi.h.

5.36.2.5 SPI_E_ALREADY_INITIALIZED

```
#define SPI_E_ALREADY_INITIALIZED ((uint8)0x4AU)
```

Development Errors. API SPI_Init service called while the SPI driver has already been initialized.

Definition at line 93 of file Spi.h.

5.36.2.6 SPI_E_PARAM_CHANNEL

```
#define SPI_E_PARAM_CHANNEL ((uint8)0x0AU)
```

Development Errors. API service called with wrong channel.

Definition at line 79 of file Spi.h.

5.36.2.7 SPI_E_PARAM_JOB

```
#define SPI_E_PARAM_JOB ((uint8)0x0BU)
```

Development Errors. API service called with wrong job.

Definition at line 81 of file Spi.h.

5.36.2.8 SPI_E_PARAM_LENGTH

```
#define SPI_E_PARAM_LENGTH ((uint8)0x0DU)
```

Development Errors. API service called with wrong length for EB.

Definition at line 85 of file Spi.h.

5.36.2.9 SPI_E_PARAM_POINTER

```
#define SPI_E_PARAM_POINTER ((uint8)0x10U)
```

Development Errors. APIs called with a Null Pointer.

Definition at line 89 of file Spi.h.

5.36.2.10 SPI_E_PARAM_SEQ

```
#define SPI_E_PARAM_SEQ ((uint8)0x0CU)
```

Development Errors. API service called with wrong sequence.

Definition at line 83 of file Spi.h.

5.36.2.11 SPI_E_PARAM_UNIT

```
#define SPI_E_PARAM_UNIT ((uint8)0x0EU)
```

Development Errors. API service called with wrong hardware unit.

Definition at line 87 of file Spi.h.

5.36.2.12 SPI_E_SEQ_IN_PROCESS

```
#define SPI_E_SEQ_IN_PROCESS ((uint8)0x3AU)
```

Runtime Errors. Synchronous transmission service called at wrong time.

Definition at line 98 of file Spi.h.

5.36.2.13 SPI_E_SEQ_PENDING

```
#define SPI_E_SEQ_PENDING ((uint8)0x2AU)
```

Runtime Errors. Services called in a wrong sequence.

Definition at line 96 of file Spi.h.

5.36.2.14 SPI_E_UNINIT

```
#define SPI_E_UNINIT ((uint8)0x1AU)
```

Development Errors. API service used without module initialization.

Definition at line 91 of file Spi.h.

5.36.2.15 SPI_GETHWUNITSTATUS_ID

```
#define SPI_GETHWUNITSTATUS_ID ((uint8)0x0BU)
```

Definition at line 73 of file Spi.h.

5.36.2.16 SPI_GETJOBRESULT_ID

```
#define SPI_GETJOBRESULT_ID ((uint8)0x07U)
```

Definition at line 69 of file Spi.h.

5.36.2.17 SPI_GETSEQUENCERESULT_ID

```
#define SPI_GETSEQUENCERESULT_ID ((uint8)0x08U)
```

Definition at line 70 of file Spi.h.

5.36.2.18 SPI_GETSTATUS_ID

```
#define SPI_GETSTATUS_ID ((uint8)0x06U)
```

Definition at line 68 of file Spi.h.

5.36.2.19 SPI_GETVERSIONINFO_ID

```
#define SPI_GETVERSIONINFO_ID ((uint8)0x09U)
```

Definition at line 71 of file Spi.h.

5.36.2.20 SPI_INIT_ID

```
#define SPI_INIT_ID ((uint8)0x00U)
```

API service ID.

Definition at line 62 of file Spi.h.

5.36.2.21 SPI_INSTANCE

```
#define SPI_INSTANCE ((uint8)0U)
```

SPI instance ID.

Definition at line 59 of file Spi.h.

5.36.2.22 SPI_MAINFUNCTIONHANDLING_ID

```
#define SPI_MAINFUNCTIONHANDLING_ID ((uint8)0x10U)
```

Definition at line 76 of file Spi.h.

5.36.2.23 SPI_READIB_ID

```
#define SPI_READIB_ID ((uint8)0x04U)
```

Definition at line 66 of file Spi.h.

5.36.2.24 SPI_SETASYNCMODE_ID

```
#define SPI_SETASYNCMODE_ID ((uint8)0x0DU)
```

Definition at line 75 of file Spi.h.

5.36.2.25 SPI_SETUPEB_ID

```
#define SPI_SETUPEB_ID ((uint8)0x05U)
```

Definition at line 67 of file Spi.h.

5.36.2.26 SPI_SYNCTRANSMIT_ID

```
#define SPI_SYNCTRANSMIT_ID ((uint8)0x0AU)
```

Definition at line 72 of file Spi.h.

5.36.2.27 SPI_WRITEIB_ID

```
#define SPI_WRITEIB_ID ((uint8)0x02U)
```

Definition at line 64 of file Spi.h.

5.36.3 Function Documentation

5.36.3.1 Spi_AsyncTransmit()

```
Std_ReturnType Spi_AsyncTransmit (  
    Spi_SequenceType Sequence )
```

Service to transmit data on the SPI bus.

Note

Function ID: DES_SPI_API_003
Service ID: 0x03

Parameters

in	<i>Sequence</i>	Sequence ID.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

- E_OK: Transmission command has been accepted.
- E_NOT_OK: Transmission command has not been accepted.

5.36.3.2 Spi_Cancel()

```
void Spi_Cancel (  
    Spi_SequenceType Sequence )
```

Service cancels the specified on-going sequence transmission.

Note

Function ID: DES_SPI_API_012
Service ID: 0x0c

Parameters

in	Sequence	Sequence ID.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.36.3.3 Spi_DeInit()

```
Std_ReturnType Spi_DeInit (  
    void )
```

Service for SPI de-initialization.

Note

Function ID: DES_SPI_API_001
Service ID: 0x01

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

- E_OK: de-initialisation command has been accepted.
- E_NOT_OK: de-initialisation command has not been accepted.

5.36.3.4 Spi_GetHWUnitStatus()

```
Spi_StatusType Spi_GetHWUnitStatus (  
    Spi_HWUnitType HWUnit )
```

This service returns the status of the specified SPI Hardware microcontroller peripheral.

Note

Function ID: DES_SPI_API_011
Service ID: 0x0b

Parameters

<i>in</i>	<i>HWUnit</i>	SPI Hardware microcontroller peripheral (unit) ID.
<i>in, out</i>	<i>None</i>	
<i>out</i>	<i>None</i>	

Returns**Spi_StatusCode**

- SPI_UNINIT: The SPI Handler/Driver is not initialized or not usable.
- SPI_IDLE: The SPI Handler/Driver is not currently transmitting any Job.
- SPI_BUSY: The SPI Handler/Driver is performing a SPI Job (transmit).

5.36.3.5 Spi_GetJobResult()

```
Spi_JobResultType Spi_GetJobResult (  
    Spi_JobType Job )
```

This service returns the last transmission result of the specified Job.

Note

Function ID: DES_SPI_API_007
Service ID: 0x07

Parameters

<i>in</i>	<i>Job</i>	Job ID. An invalid job ID will return an undefined result.
<i>in, out</i>	<i>None</i>	
<i>out</i>	<i>None</i>	

Returns**Spi_JobResultType**

- SPI_JOB_OK: The last transmission of the Job has been finished successfully.
- SPI_JOB_PENDING: The SPI Handler/Driver is performing a SPI Job. The meaning of this status is equal to SPI_BUSY.
- SPI_JOB_FAILED: The last transmission of the Job has failed.
- SPI_JOB_QUEUED: An asynchronous transmit Job has been accepted, while actual transmission for this Job has not started yet.

5.36.3.6 Spi_GetSequenceResult()

```
Spi_SeqResultType Spi_GetSequenceResult (  
    Spi_SequenceType Sequence )
```

This service returns the last transmission result of the specified Sequence.

Note

Function ID: DES_SPI_API_008
Service ID: 0x08

Parameters

in	<i>Sequence</i>	Sequence ID. An invalid sequence ID will return an undefined result.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Spi_SeqResultType

- SPI_SEQ_OK: The last transmission of the Sequence has been finished successfully.
- SPI_SEQ_PENDING: The SPI Handler/Driver is performing a SPI Sequence. The meaning of this status is equal to SPI_BUSY.
- SPI_SEQ_FAILED: The last transmission of the Sequence has failed.
- SPI_SEQ_CANCELED: The last transmission of the Sequence has been canceled by user.

5.36.3.7 Spi_GetStatus()

```
Spi_StatusType Spi_GetStatus (  
    void )
```

Service returns the SPI Handler/Driver software module status.

Note

Function ID: DES_SPI_API_006
Service ID: 0x06

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Spi_StatusType

- SPI_UNINIT: The SPI Handler/Driver is not initialized or not usable.
- SPI_IDLE: The SPI Handler/Driver is not currently transmitting any Job.
- SPI_BUSY: The SPI Handler/Driver is performing a SPI Job (transmit).

5.36.3.8 Spi_GetVersionInfo()

```
void Spi_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

Service returns the version information of this module.

Note

Function ID: DES_SPI_API_009
Service ID: 0x09

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>versioninfo</i>	Pointer to where to store the version information of this module.

Returns

None

5.36.3.9 Spi_Init()

```
void Spi_Init (
    const Spi_ConfigType * ConfigPtr )
```

Service for SPI initialization.

Note

Function ID: DES_SPI_API_000
Service ID: 0x00

Parameters

in	<i>ConfigPtr</i>	Pointer to configuration set.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.36.3.10 Spi_MainFunction_Handling()

```
void Spi_MainFunction_Handling (
    void )
```

This function shall polls the SPI interrupts linked to HW Units allocated to the transmission of SPI sequences to enable the evolution of transmission state machine.

Note

Function ID: DES_SPI_API_014
Service ID: 0x10

Parameters

in	<i>None</i>	
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

5.36.3.11 Spi_ReadIB()

```
Std_ReturnType Spi_ReadIB (
    Spi_ChannelType Channel,
    Spi_DataBufferType * DataBufferPointer )
```

Service for reading synchronously one or more data from an IB SPI Handler/Driver Channel specified by parameter.

Note

Function ID: DES_SPI_API_004
Service ID: 0x04

Parameters

in	<i>Channel</i>	Channel ID.
in, out	<i>None</i>	
out	<i>DataBufferPointer</i>	Pointer to destination data buffer in RAM.

Returns

Std_ReturnType

- E_OK: read command has been accepted.
- E_NOT_OK: read command has not been accepted.

5.36.3.12 Spi_SetAsyncMode()

```
Std_ReturnType Spi_SetAsyncMode (
    Spi_AsyncModeType Mode )
```

Service to set the asynchronous mechanism mode for SPI busses handled asynchronously.

Note

Function ID: DES_SPI_API_013
Service ID: 0x0d

Parameters

<i>in</i>	<i>Mode</i>	New mode required.
<i>in, out</i>	<i>None</i>	
<i>out</i>	<i>None</i>	

Returns

Std_ReturnType

- E_OK: Setting command has been done.
- E_NOT_OK: setting command has not been accepted.

5.36.3.13 Spi_SetupEB()

```
Std_ReturnType Spi_SetupEB (  
    Spi_ChannelType Channel,  
    const Spi_DataBufferType * SrcDataBufferPtr,  
    Spi_DataBufferType * DesDataBufferPtr,  
    Spi_NumberOfDataType Length )
```

Service to setup the buffers and the length of data for the EB SPI Handler/Driver Channel specified.

Note

Function ID: DES_SPI_API_005
Service ID: 0x05

Parameters

<i>in</i>	<i>Channel</i>	Channel ID.
<i>in</i>	<i>SrcDataBufferPtr</i>	Pointer to source data buffer.
<i>in</i>	<i>Length</i>	Length (number of data elements) of the data to be transmitted from SrcDataBufferPtr and/or received from DesDataBufferPtr. -Min.: 1 -Max.: Max of data specified at configuration for this channel.
<i>in, out</i>	<i>DesDataBufferPtr</i>	Pointer to destination data buffer in RAM.
<i>out</i>	<i>None</i>	

Returns

Std_ReturnType

- E_OK: Setup command has been accepted.
- E_NOT_OK: Setup command has not been accepted.

5.36.3.14 Spi_SyncTransmit()

```
Std_ReturnType Spi_SyncTransmit (
    Spi_SequenceType Sequence )
```

Service to transmit data on the SPI bus.

Note

Function ID: DES_SPI_API_010
Service ID: 0x0a

Parameters

in	<i>Sequence</i>	Sequence ID.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

- E_OK: Transmission has been successful.
- E_NOT_OK: Transmission failed.

5.36.3.15 Spi_WriteIB()

```
Std_ReturnType Spi_WriteIB (
    Spi_ChannelType Channel,
    const Spi_DataBufferType * DataBufferPtr )
```

Service for writing one or more data to an IB SPI Handler/Driver Channel specified by parameter.

Note

Function ID: DES_SPI_API_002
Service ID: 0x02

Parameters

in	<i>Channel</i>	Channel ID.
in	<i>DataBufferPtr</i>	Pointer to source data buffer. If this pointer is null, it is assumed that the data to be transmitted is not relevant and the default transmit value of this channel will be used instead.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Std_ReturnType

- E_OK: write command has been accepted.

- E_NOT_OK: write command has not been accepted.

5.37 Wdg.h File Reference

Header file of WDG MCAL driver.

```
#include "Wdg_GeneralTypes.h"
#include "Wdg_Cfg.h"
#include "Wdg_Hal.h"
#include "Wdg_PBcfg.h"
```

Enumerations

- enum [Wdg_ServiceIdType](#) {
[WDG_INIT_ID](#) = (uint8)0x00U, [WDG_SETMODE_ID](#) = (uint8)0x01U, [WDG_TRIGGER_ID](#) = (uint8)0x02U, [WDG←
SETTRIGGERCONDITION_ID](#) = (uint8)0x03U,
[WDG_GETVERSION_ID](#) = (uint8)0x04U }

This enumerated type specify service ID of WDG MCAL APIs.

Functions

- void [Wdg_Init](#) (const [Wdg_ConfigType](#) *ConfigPtr)
Initializes the WDG MCAL driver.
- Std_ReturnType [Wdg_SetMode](#) ([WdgIf_ModeType](#) Mode)
Switches the watchdog into the mode Mode.
- void [Wdg_SetTriggerCondition](#) (uint16 timeout)
Sets the timeout value for the trigger counter.
- void [Wdg_GetVersionInfo](#) (Std_VersionInfoType *versioninfo)
Returns the version information of the module.

5.37.1 Detailed Description

Header file of WDG MCAL driver.

5.37.2 Enumeration Type Documentation

5.37.2.1 Wdg_ServiceIdType

```
enum Wdg\_ServiceIdType
```

This enumerated type specify service ID of WDG MCAL APIs.

Enumerator

WDG_INIT_ID	ID for Wdg_Init() .
WDG_SETMODE_ID	ID for Wdg_SetMode() .
WDG_TRIGGER_ID	ID for Set Trigger.
WDG_SETTRIGGERCONDITION_ID	ID for Wdg_SetTriggerCondition()
WDG_GETVERSION_ID	ID for Wdg_GetVersionInfo()

Definition at line 56 of file Wdg.h.

5.37.3 Function Documentation

5.37.3.1 Wdg_GetVersionInfo()

```
void Wdg_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

Returns the version information of the module.

Note

Function ID: DES_WDG_API_001

Parameters

<i>in</i>	<i>versioninfo</i>	: Pointer to VersionInfo.
-----------	--------------------	---------------------------

Returns

none

See also

struct Std_VersionInfoType

5.37.3.2 Wdg_Init()

```
void Wdg_Init (
    const Wdg_ConfigType * ConfigPtr )
```

Initializes the WDG MCAL driver.

Note

Function ID: DES_WDG_API_002

Parameters

in	<i>ConfigPtr</i>	: Pointer to configuration set.
----	------------------	---------------------------------

Returns

none

See also

struct [Wdg_ConfigType](#)

5.37.3.3 Wdg_SetMode()

```
Std_ReturnType Wdg_SetMode (
    WdgIf_ModeType Mode )
```

Switches the watchdog into the mode Mode.

Note

Function ID: DES_WDG_API_004

Parameters

in	<i>Mode</i>	: Specify the mode to set.
----	-------------	----------------------------

Returns

none

See also

enum [WdgIf_ModeType](#)

5.37.3.4 Wdg_SetTriggerCondition()

```
void Wdg_SetTriggerCondition (
    uint16 timeout )
```

Sets the timeout value for the trigger counter.

Note

Function ID: DES_WDG_API_003

Parameters

<code>in</code>	<code>timeout</code>	: timeout value (milliseconds) for setting the trigger counter.
-----------------	----------------------	-----------------------------------------------------------------

Returns

none

5.38 Wdg_GeneralTypes.h File Reference

This file provides extern Wdg module structure and macro.

```
#include "StandardTypes.h"
#include "Wdg_Hal.h"
#include "Mcal.h"
#include "WdgIf.h"
```

Classes

- struct [Wdg_ConfigType](#)
Defines the configuration structure for WDG Driver.

Macros

- #define [WDG_E_DRIVER_STATE](#) 0x10U
Error code for function is called in wrong status .
- #define [WDG_E_PARAM_MODE](#) 0x11U
Error code for invalid mode .
- #define [WDG_E_PARAM_CONFIG](#) 0x12U
Error code for invalid configuration .
- #define [WDG_E_PARAM_TIMEOUT](#) 0x13U
Error code for timeout value is out of range.
- #define [WDG_E_PARAM_POINTER](#) 0x14U
Error code for NULL pointer.
- #define [WDG_E_DISABLE_REJECTED](#) 0x15U
Error code for WDG can't be disabled.

Enumerations

- enum [WdgIf_ModeType](#) { [WDGIF_OFF_MODE](#) = 0x00U, [WDGIF_SLOW_MODE](#), [WDGIF_FAST_MODE](#) }
This enumerated type will contain the watchdog driver's possible modes.

5.38.1 Detailed Description

This file provides extern Wdg module structure and macro.

5.38.2 Macro Definition Documentation

5.38.2.1 WDG_E_DISABLE_REJECTED

```
#define WDG_E_DISABLE_REJECTED 0x15U
```

Error code for WDG can't be disabled.

Definition at line 82 of file Wdg_GeneralTypes.h.

5.38.2.2 WDG_E_DRIVER_STATE

```
#define WDG_E_DRIVER_STATE 0x10U
```

Error code for funciton is called in wrong status .

Define error codes for Wdg driver.

Definition at line 67 of file Wdg_GeneralTypes.h.

5.38.2.3 WDG_E_PARAM_CONFIG

```
#define WDG_E_PARAM_CONFIG 0x12U
```

Error code for invalid configuration .

Definition at line 73 of file Wdg_GeneralTypes.h.

5.38.2.4 WDG_E_PARAM_MODE

```
#define WDG_E_PARAM_MODE 0x11U
```

Error code for invalid mode .

Definition at line 70 of file Wdg_GeneralTypes.h.

5.38.2.5 WDG_E_PARAM_POINTER

```
#define WDG_E_PARAM_POINTER 0x14U
```

Error code for NULL pointer.

Definition at line 79 of file Wdg_GeneralTypes.h.

5.38.2.6 WDG_E_PARAM_TIMEOUT

```
#define WDG_E_PARAM_TIMEOUT 0x13U
```

Error code for timeout value is out of range.

Definition at line 76 of file Wdg_GeneralTypes.h.

5.38.3 Enumeration Type Documentation

5.38.3.1 WdgIf_ModeType

```
enum WdgIf_ModeType
```

This enumerated type will contain the watchdog driver's possible modes.

Enumerator

WDGIF_OFF_MODE	In this mode, the watchdog driver is disabled (switched off).
WDGIF_SLOW_MODE	=0x01 In this mode, the watchdog driver is set up for a long timeout period (slow triggering).
WDGIF_FAST_MODE	=0x02 In this mode, the watchdog driver is set up for a short timeout period (fast triggering).

Definition at line 91 of file Wdg_GeneralTypes.h.

Index

- [_Crypto_QueueType, 8](#)
 - [JobPtr, 8](#)
 - [Next, 8](#)
- [ACMP_DEINIT_ID](#)
 - [CDD_Acmp.h, 144](#)
- [ACMP_E_INVALID_CHANNEL](#)
 - [CDD_Acmp.h, 144](#)
- [ACMP_E_INVALID_POINTER](#)
 - [CDD_Acmp.h, 144](#)
- [ACMP_E_STATE_TRANSITION](#)
 - [CDD_Acmp.h, 144](#)
- [ACMP_E_UNINIT](#)
 - [CDD_Acmp.h, 144](#)
- [ACMP_GET_OUTPUTDATA_ID](#)
 - [CDD_Acmp.h, 144](#)
- [ACMP_GET_POLLINGDATA_ID](#)
 - [CDD_Acmp.h, 145](#)
- [ACMP_GET_VERSIONINFO_ID](#)
 - [CDD_Acmp.h, 145](#)
- [ACMP_INIT_ID](#)
 - [CDD_Acmp.h, 145](#)
- [ACMP_INSTANCE_ID](#)
 - [CDD_Acmp.h, 145](#)
- [ACMP_MODULE_INIT](#)
 - [CDD_Acmp.h, 145](#)
- [ACMP_MODULE_UNINIT](#)
 - [CDD_Acmp.h, 145](#)
- [ADC_DEINIT_ID](#)
 - [Adc.h, 109](#)
- [ADC_DISABLEGROUPNOTIFICATION_ID](#)
 - [Adc.h, 110](#)
- [ADC_DISABLEHARDWARETRIGGER_ID](#)
 - [Adc.h, 110](#)
- [ADC_E_ALREADY_INITIALIZED](#)
 - [Adc.h, 110](#)
- [ADC_E_BUFFER_UNINIT](#)
 - [Adc.h, 110](#)
- [ADC_E_BUSY](#)
 - [Adc.h, 110](#)
- [ADC_E_IDLE](#)
 - [Adc.h, 110](#)
- [ADC_E_NOT_DISENGAGED](#)
 - [Adc.h, 111](#)
- [ADC_E_NOTIF_CAPABILITY](#)
 - [Adc.h, 111](#)
- [ADC_E_PARAM_CONFIG](#)
 - [Adc.h, 111](#)
- [ADC_E_PARAM_GROUP](#)
 - [Adc.h, 111](#)
- [ADC_E_PARAM_POINTER](#)
 - [Adc.h, 111](#)
- [ADC_E_PERIPHERAL_NOT_PREPARED](#)
 - [Adc.h, 111](#)
- [ADC_E_POWER_STATE_NOT_SUPPORTED](#)
 - [Adc.h, 112](#)
- [ADC_E_TRANSITION_NOT_POSSIBLE](#)
 - [Adc.h, 112](#)
- [ADC_E_UNINIT](#)
 - [Adc.h, 112](#)
- [ADC_E_WRONG_CONV_MODE](#)
 - [Adc.h, 112](#)
- [ADC_E_WRONG_TRIGG_SRC](#)
 - [Adc.h, 112](#)
- [ADC_ENABLEGROUPNOTIFICATION_ID](#)
 - [Adc.h, 112](#)
- [ADC_ENABLEHARDWARETRIGGER_ID](#)
 - [Adc.h, 113](#)
- [ADC_GETCURRENTPOWERSTATE_ID](#)
 - [Adc.h, 113](#)
- [ADC_GETGROUPSTATUS_ID](#)
 - [Adc.h, 113](#)
- [ADC_GETSTREAMLASTPOINTER_ID](#)
 - [Adc.h, 113](#)
- [ADC_GETTARGETPOWERSTATE_ID](#)
 - [Adc.h, 113](#)
- [ADC_GETVERSIONINFO_ID](#)
 - [Adc.h, 113](#)
- [ADC_INIT_ID](#)
 - [Adc.h, 114](#)
- [ADC_INSTANCE](#)
 - [Adc.h, 114](#)
- [ADC_PREPAREPOWERSTATE_ID](#)
 - [Adc.h, 114](#)
- [ADC_READGROUP_ID](#)
 - [Adc.h, 114](#)
- [ADC_SETPOWERSTATE_ID](#)
 - [Adc.h, 114](#)
- [ADC_SETUPRESULTBUFFER_ID](#)
 - [Adc.h, 114](#)
- [ADC_STARTGROUPCONVERSION_ID](#)
 - [Adc.h, 115](#)
- [ADC_STOPGROUPCONVERSION_ID](#)
 - [Adc.h, 115](#)
- [Acmp_ConfigType, 9](#)
 - [HwConfigPtr, 9](#)
 - [MaxChannelNum, 9](#)
- [Acmp_Deinit](#)
 - [CDD_Acmp.h, 146](#)
- [Acmp_GetOutputData](#)
 - [CDD_Acmp.h, 146](#)
- [Acmp_GetPollingData](#)
 - [CDD_Acmp.h, 147](#)
- [Acmp_Init](#)

- CDD_Acmp.h, 147
- ActiveTime
 - Icu_DutyCycleType, 82
- ActualSize
 - CryptoKeyElement, 43
- Adc.h, 108
 - ADC_DEINIT_ID, 109
 - ADC_DISABLEGROUPNOTIFICATION_ID, 110
 - ADC_DISABLEHARDWARETRIGGER_ID, 110
 - ADC_E_ALREADY_INITIALIZED, 110
 - ADC_E_BUFFER_UNINIT, 110
 - ADC_E_BUSY, 110
 - ADC_E_IDLE, 110
 - ADC_E_NOT_DISENGAGED, 111
 - ADC_E_NOTIF_CAPABILITY, 111
 - ADC_E_PARAM_CONFIG, 111
 - ADC_E_PARAM_GROUP, 111
 - ADC_E_PARAM_POINTER, 111
 - ADC_E_PERIPHERAL_NOT_PREPARED, 111
 - ADC_E_POWER_STATE_NOT_SUPPORTED, 112
 - ADC_E_TRANSITION_NOT_POSSIBLE, 112
 - ADC_E_UNINIT, 112
 - ADC_E_WRONG_CONV_MODE, 112
 - ADC_E_WRONG_TRIGG_SRC, 112
 - ADC_ENABLEGROUPNOTIFICATION_ID, 112
 - ADC_ENABLEHARDWARETRIGGER_ID, 113
 - ADC_GETCURRENTPOWERSTATE_ID, 113
 - ADC_GETGROUPSTATUS_ID, 113
 - ADC_GETSTREAMLASTPOINTER_ID, 113
 - ADC_GETTARGETPOWERSTATE_ID, 113
 - ADC_GETVERSIONINFO_ID, 113
 - ADC_INIT_ID, 114
 - ADC_INSTANCE, 114
 - ADC_PREPAREPOWERSTATE_ID, 114
 - ADC_READGROUP_ID, 114
 - ADC_SETPOWERSTATE_ID, 114
 - ADC_SETUPRESULTBUFFER_ID, 114
 - ADC_STARTGROUPCONVERSION_ID, 115
 - ADC_STOPGROUPCONVERSION_ID, 115
 - Adc_DelInit, 115
 - Adc_DisableGroupNotification, 115
 - Adc_DisableHardwareTrigger, 116
 - Adc_EnableGroupNotification, 116
 - Adc_EnableHardwareTrigger, 116
 - Adc_GenerateConfigPC, 120
 - Adc_GetGroupStatus, 117
 - Adc_GetStreamLastPointer, 117
 - Adc_GetVersionInfo, 118
 - Adc_Init, 118
 - Adc_ReadGroup, 118
 - Adc_SetupResultBuffer, 119
 - Adc_StartGroupConversion, 119
 - Adc_StopGroupConversion, 119
- Adc_DelInit
 - Adc.h, 115
- Adc_DisableGroupNotification
 - Adc.h, 115
- Adc_DisableHardwareTrigger
 - Adc.h, 116
- Adc_EnableGroupNotification
 - Adc.h, 116
- Adc_EnableHardwareTrigger
 - Adc.h, 116
- Adc_EnableHardwareTrigger
 - Adc.h, 116
- Adc_GenerateConfigPC
 - Adc.h, 120
- Adc_GetGroupStatus
 - Adc.h, 117
- Adc_GetStreamLastPointer
 - Adc.h, 117
- Adc_GetVersionInfo
 - Adc.h, 118
- Adc_Init
 - Adc.h, 118
- Adc_ReadGroup
 - Adc.h, 118
- Adc_SetupResultBuffer
 - Adc.h, 119
- Adc_StartGroupConversion
 - Adc.h, 119
- Adc_StopGroupConversion
 - Adc.h, 119
- algorithm
 - Crypto_PrimitiveInfoType, 41
- AllowPartialAccess
 - CryptoKeyElement, 44
- Assigned
 - Fee_BlockHeaderType, 48
- AutoBusOffRecover
 - Can_ControllerDescriptorType, 14
- BgndBlockCellLength
 - FlsTst_CommonVariableType, 70
- BgndBlockCellNumber
 - FlsTst_CommonVariableType, 70
- BgndBlockCellSignatureValue
 - FlsTst_CommonVariableType, 70
- BgndBlockCellStartId
 - FlsTst_CommonVariableType, 70
- BgndBlockConfig
 - FlsTst_ConfigType, 72
- BgndBlockIndex
 - FlsTst_CommonVariableType, 70
- BgndBlockMaxNumber
 - FlsTst_ConfigType, 72
- BgndLastSignature
 - FlsTst_CommonVariableType, 70
- BgndOverallResult
 - FlsTst_CommonVariableType, 71
- BgndTestAbortOrSuspendFlag
 - FlsTst_CommonVariableType, 71
- BlockAssign
 - Fee_BlockType, 51
- BlockBaseAddress
 - FlsTst_BlockConfigType, 68
- BlockId
 - Fee_BlockHeaderType, 48
- BlockIdx
 - Fee_JobInfoType, 59
- BlockIndex
 - FlsTst_BlockConfigType, 68
- BlockNumber
 - Fee_BlockType, 51

- Fee_JobPendingInfoType, [62](#)
- BlockOffset
 - Fee_JobPendingInfoType, [62](#)
- BlockPtr
 - Fee_ClusterGroupType, [53](#)
- BlockSize
 - Fee_BlockType, [52](#)
 - FlsTst_BlockConfigType, [68](#)
- BufferValid
 - Fee_JobInfoType, [59](#)
- BusoffInterrupt
 - Can_ControllerDescriptorType, [14](#)
- CAN_BUSY
 - Can_GeneralTypes.h, [139](#)
- CAN_CANCTRL_MAX_PAYLOAD12_U8
 - Can_GeneralTypes.h, [139](#)
- CAN_CANCTRL_MAX_PAYLOAD16_U8
 - Can_GeneralTypes.h, [139](#)
- CAN_CANCTRL_MAX_PAYLOAD20_U8
 - Can_GeneralTypes.h, [139](#)
- CAN_CANCTRL_MAX_PAYLOAD24_U8
 - Can_GeneralTypes.h, [139](#)
- CAN_CANCTRL_MAX_PAYLOAD32_U8
 - Can_GeneralTypes.h, [140](#)
- CAN_CANCTRL_MAX_PAYLOAD48_U8
 - Can_GeneralTypes.h, [140](#)
- CAN_CANCTRL_MAX_PAYLOAD64_U8
 - Can_GeneralTypes.h, [140](#)
- CAN_CANCTRL_MAX_PAYLOAD8_U8
 - Can_GeneralTypes.h, [140](#)
- CAN_E_DATA_LOST
 - Can.h, [122](#)
- CAN_E_ECC_ERROR
 - Can.h, [123](#)
- CAN_E_ICOM_CONFIG_INVALID
 - Can.h, [123](#)
- CAN_E_INIT_FAILED
 - Can.h, [123](#)
- CAN_E_INVALID_CONTROLLER_MODE
 - Can.h, [123](#)
- CAN_E_INVALID_CONTROLLER
 - Can.h, [123](#)
- CAN_E_PARAM_BAUDRATE
 - Can.h, [123](#)
- CAN_E_PARAM_CONTROLLER
 - Can.h, [124](#)
- CAN_E_PARAM_DATA_LENGTH
 - Can.h, [124](#)
- CAN_E_PARAM_HANDLE
 - Can.h, [124](#)
- CAN_E_PARAM_POINTER
 - Can.h, [124](#)
- CAN_E_TRANSITION
 - Can.h, [124](#)
- CAN_E_UNINIT
 - Can.h, [124](#)
- CAN_EXTEND_FD_ID_DESCRIPTOR
 - Can_GeneralTypes.h, [140](#)
- CAN_EXTEND_ID_DESCRIPTOR
 - Can_GeneralTypes.h, [140](#)
- CAN_FD_ID_DESCRIPTOR
 - Can_GeneralTypes.h, [141](#)
- CAN_FD_ID_EXTEND_DEF_MASK
 - Can_GeneralTypes.h, [141](#)
- CAN_ID_EXTENDED_MASK_U32
 - Can_GeneralTypes.h, [141](#)
- CAN_ID_EXTENDEDDIFF_MASK_U32
 - Can_GeneralTypes.h, [141](#)
- CAN_ID_STANDARD_MASK_U32
 - Can_GeneralTypes.h, [141](#)
- CAN_IDE_ID_DESCRIPTOR
 - Can_GeneralTypes.h, [141](#)
- CAN_INSTANCE_ID
 - Can.h, [125](#)
- CAN_SID_CBK_CHECK_WAKEUP
 - Can.h, [125](#)
- CAN_SID_CHANGE_BAUDRATE
 - Can.h, [125](#)
- CAN_SID_CHECK_BAUDRATE
 - Can.h, [125](#)
- CAN_SID_DEINIT
 - Can.h, [125](#)
- CAN_SID_DISABLE_CONTROLLER_INTERRUPTS
 - Can.h, [125](#)
- CAN_SID_ENABLE_CONTROLLER_INTERRUPTS
 - Can.h, [126](#)
- CAN_SID_GET_VERSION_INFO
 - Can.h, [126](#)
- CAN_SID_GETCONTROLLERERRORSTATE
 - Can.h, [126](#)
- CAN_SID_GETCONTROLLERMODE
 - Can.h, [126](#)
- CAN_SID_GetControllerRxErrorCounter
 - Can.h, [126](#)
- CAN_SID_GetControllerTxErrorCounter
 - Can.h, [126](#)
- CAN_SID_INIT
 - Can.h, [127](#)
- CAN_SID_MAIN_FUNCTION_BUS_OFF
 - Can.h, [127](#)
- CAN_SID_MAIN_FUNCTION_MODE
 - Can.h, [127](#)
- CAN_SID_MAIN_FUNCTION_READ
 - Can.h, [127](#)
- CAN_SID_MAIN_FUNCTION_WAKEUP
 - Can.h, [127](#)
- CAN_SID_MAIN_FUNCTION_WRITE
 - Can.h, [127](#)
- CAN_SID_SET_BAUDRATE
 - Can.h, [128](#)
- CAN_SID_SET_CONTROLLER_MODE
 - Can.h, [128](#)
- CAN_SID_WRITE
 - Can.h, [128](#)
- CDD_Acmp.h, [143](#)
 - ACMP_DEINIT_ID, [144](#)
 - ACMP_E_INVALID_CHANNEL, [144](#)
 - ACMP_E_INVALID_POINTER, [144](#)
 - ACMP_E_STATE_TRANSITION, [144](#)
 - ACMP_E_UNINIT, [144](#)
 - ACMP_GET_OUTPUTDATA_ID, [144](#)

- ACMP_GET_POLLINGDATA_ID, [145](#)
- ACMP_GET_VERSIONINFO_ID, [145](#)
- ACMP_INIT_ID, [145](#)
- ACMP_INSTANCE_ID, [145](#)
- ACMP_MODULE_INIT, [145](#)
- ACMP_MODULE_UNINIT, [145](#)
- Acmp_Deinit, [146](#)
- Acmp_GetOutputData, [146](#)
- Acmp_GetPollingData, [147](#)
- Acmp_Init, [147](#)
- CDD_Crc.h, [148](#)
 - CRC_CALCULATECRC_ID, [149](#)
 - CRC_E_PARAM_POINTER, [149](#)
 - CRC_E_UNINIT, [149](#)
 - CRC_GETCONFIG_ID, [149](#)
 - CRC_GETVERSIONINFO_ID, [150](#)
 - CRC_GetCRCRESULT_ID, [150](#)
 - CRC_INIT_ID, [150](#)
 - Crc_CalculateCRC16, [151](#)
 - Crc_CalculateCRC16ARC, [151](#)
 - Crc_CalculateCRC32, [152](#)
 - Crc_CalculateCRC32P4, [152](#)
 - Crc_CalculateCRC64, [153](#)
 - Crc_CalculateCRC8, [153](#)
 - Crc_CalculateCRC8H2F, [154](#)
 - Crc_CalculateCRC, [150](#)
 - Crc_Deinit, [155](#)
 - Crc_GetConfig, [155](#)
 - Crc_GetCrcResult, [156](#)
 - Crc_GetVersionInfo, [156](#)
 - Crc_Init, [156](#)
- CDD_Crc_Types.h, [157](#)
 - CRC_CALCULATECRC_ID, [158](#)
 - CRC_CRC16_POLY, [158](#)
 - CRC_CRC16ARC_POLY, [158](#)
 - CRC_CRC32_POLY, [158](#)
 - CRC_CRC32P4_POLY, [158](#)
 - CRC_E_PARAM_POINTER, [158](#)
 - CRC_E_UNINIT, [159](#)
 - CRC_GETCONFIG_ID, [159](#)
 - CRC_GETVERSIONINFO_ID, [159](#)
 - CRC_GetCRCRESULT_ID, [159](#)
 - CRC_INIT_ID, [159](#)
- CDD_I2c.h, [160](#)
 - I2C_ASYNCTRANSMIT_ID, [161](#)
 - I2C_DEINIT_ID, [161](#)
 - I2C_E_ALREADY_INIT, [161](#)
 - I2C_E_BUSY, [162](#)
 - I2C_E_INVALID_CHANNEL, [162](#)
 - I2C_E_INVALID_POINTER, [162](#)
 - I2C_E_STATUES, [162](#)
 - I2C_E_UNINIT, [162](#)
 - I2C_GETSTATUS_ID, [162](#)
 - I2C_GETVERSION_INFO_ID, [163](#)
 - I2C_INIT_ID, [163](#)
 - I2C_INIT_STATE, [163](#)
 - I2C_UNINIT_STATE, [163](#)
 - I2c_AbortTransmit, [164](#)
 - I2c_AsyncTransmit, [164](#)
 - I2c_DeInit, [165](#)
 - I2c_GenerateConfigPC, [168](#)
 - I2c_GetBaudRate, [165](#)
 - I2c_GetStatus, [166](#)
 - I2c_GetVersionInfo, [166](#)
 - I2c_Init, [167](#)
 - I2c_SetBaudRate, [167](#)
 - I2c_StatusType, [163](#)
 - I2c_SyncTransmit, [167](#)
- CDD_Mcl.h, [168](#)
 - MCL_DEINIT_ID, [169](#)
 - MCL_E_PARAM_POINTER, [169](#)
 - MCL_GET_VERSION_INFO_ID, [169](#)
 - MCL_INIT_ID, [169](#)
 - MCL_INSTANCE_ID, [170](#)
 - Mcl_DeInit, [170](#)
 - Mcl_GetVersionInfo, [170](#)
 - Mcl_Init, [171](#)
- CDD_Sent.h, [171](#)
 - SENT_DEINIT_ID, [172](#)
 - SENT_E_INVALID_CHANNEL, [172](#)
 - SENT_E_INVALID_POINTER, [172](#)
 - SENT_E_STATE_TRANSITION, [173](#)
 - SENT_E_UNINIT, [173](#)
 - SENT_ENABLE_CHANNEL_ID, [173](#)
 - SENT_GET_CHANNELSTAUTS_ID, [173](#)
 - SENT_GET_VERSIONINFO_ID, [173](#)
 - SENT_INIT_ID, [173](#)
 - SENT_INIT_TX_CTRL_ID, [174](#)
 - SENT_INSTANCE_ID, [174](#)
 - Sent_Deinit, [174](#)
 - Sent_EnableChannel, [174](#)
 - Sent_GetChannelStatus, [175](#)
 - Sent_Init, [176](#)
 - Sent_InitChannelTxCtrl, [176](#)
- CDD_Uart.h, [177](#)
 - UART_ABORTTRANSMIT_ID, [178](#)
 - UART_ASYNCTRANSMIT_ID, [178](#)
 - UART_DEINIT_ID, [178](#)
 - UART_E_BUSY_TRANSMIT, [178](#)
 - UART_E_INVALID_CHANNEL, [178](#)
 - UART_E_INVALID_POINTER, [178](#)
 - UART_E_STATE_TRANSITION, [179](#)
 - UART_E_TRANSMIT_LENGTH, [179](#)
 - UART_E_UNINIT, [179](#)
 - UART_GET_STATUS_ID, [179](#)
 - UART_GET_VERSIONINFO_ID, [179](#)
 - UART_INIT_ID, [179](#)
 - UART_INSTANCE_ID, [180](#)
 - Uart_AbortTransmit, [180](#)
 - Uart_AsyncTransmit, [181](#)
 - Uart_DeInit, [181](#)
 - Uart_DirType, [180](#)
 - Uart_GenerateConfigPC, [183](#)
 - Uart_GetStatus, [182](#)
 - Uart_GetVersionInfo, [182](#)
 - Uart_Init, [183](#)
- CMU_HAL_ID1
 - Oslf_Critical.h, [341](#)
- CRC_CALCULATECRC_ID
 - CDD_Crc.h, [149](#)
 - CDD_Crc_Types.h, [158](#)
- CRC_CRC16_POLY

- CDD_Crc_Types.h, 158
- CRC_CRC16ARC_POLY
 - CDD_Crc_Types.h, 158
- CRC_CRC32_POLY
 - CDD_Crc_Types.h, 158
- CRC_CRC32P4_POLY
 - CDD_Crc_Types.h, 158
- CRC_E_PARAM_POINTER
 - CDD_Crc.h, 149
 - CDD_Crc_Types.h, 158
- CRC_E_UNINIT
 - CDD_Crc.h, 149
 - CDD_Crc_Types.h, 159
- CRC_GETCONFIG_ID
 - CDD_Crc.h, 149
 - CDD_Crc_Types.h, 159
- CRC_GETVERSIONINFO_ID
 - CDD_Crc.h, 150
 - CDD_Crc_Types.h, 159
- CRC_GetCRCRESULT_ID
 - CDD_Crc.h, 150
 - CDD_Crc_Types.h, 159
- CRC_INIT_ID
 - CDD_Crc.h, 150
 - CDD_Crc_Types.h, 159
- CRYPTO_CSE_KE_SHE_M1_OFFSET
 - Csm_Types.h, 202
- CRYPTO_CSE_KE_SHE_M2_OFFSET
 - Csm_Types.h, 202
- CRYPTO_CSE_KE_SHE_M3_OFFSET
 - Csm_Types.h, 202
- CRYPTO_CSE_KE_SHE_M4_OFFSET
 - Csm_Types.h, 203
- CRYPTO_CSE_KE_SHE_M5_OFFSET
 - Csm_Types.h, 203
- CRYPTO_E_INIT_FAILED
 - Crypto.h, 185
- CRYPTO_E_PARAM_HANDLE
 - Crypto.h, 185
- CRYPTO_E_PARAM_POINTER
 - Crypto.h, 185
- CRYPTO_E_PARAM_VALUE
 - Crypto.h, 185
- CRYPTO_E_UNINIT
 - Crypto.h, 185
- CRYPTO_FUNCTION_BOOTDEFINE
 - Crypto.h, 185
- CRYPTO_FUNCTION_BOOTFAILURE
 - Crypto.h, 186
- CRYPTO_FUNCTION_BOOTOK
 - Crypto.h, 186
- CRYPTO_FUNCTION_CANCELJOB
 - Crypto.h, 186
- CRYPTO_FUNCTION_CERTIFICATEPARSE
 - Crypto.h, 186
- CRYPTO_FUNCTION_CERTIFICATEVERIFY
 - Crypto.h, 186
- CRYPTO_FUNCTION_DEBUGAUTH
 - Crypto.h, 186
- CRYPTO_FUNCTION_DEBUGCHAL
 - Crypto.h, 187
- CRYPTO_FUNCTION_GETID
 - Crypto.h, 187
- CRYPTO_FUNCTION_GETSTATUS
 - Crypto.h, 187
- CRYPTO_FUNCTION_GETVERSIONINFO
 - Crypto.h, 187
- CRYPTO_FUNCTION_INIT
 - Crypto.h, 187
- CRYPTO_FUNCTION_KEYCOPY
 - Crypto.h, 187
- CRYPTO_FUNCTION_KEYDERIVE
 - Crypto.h, 188
- CRYPTO_FUNCTION_KEYELEMENTCOPY
 - Crypto.h, 188
- CRYPTO_FUNCTION_KEYELEMENTGET
 - Crypto.h, 188
- CRYPTO_FUNCTION_KEYELEMENTIDSGET
 - Crypto.h, 188
- CRYPTO_FUNCTION_KEYELEMENTSET
 - Crypto.h, 188
- CRYPTO_FUNCTION_KEYEXCHANGEALCPUBVAL
 - Crypto.h, 188
- CRYPTO_FUNCTION_KEYEXCHANGEALCSECRET
 - Crypto.h, 189
- CRYPTO_FUNCTION_KEYGENERATE
 - Crypto.h, 189
- CRYPTO_FUNCTION_KEYVALIDSET
 - Crypto.h, 189
- CRYPTO_FUNCTION_MPCCOMPRESSION
 - Crypto.h, 189
- CRYPTO_FUNCTION_PROCESSJOB
 - Crypto.h, 189
- CRYPTO_FUNCTION_RANDOMSEED
 - Crypto.h, 189
- CRYPTO_HW_NOT_SUPPORTED
 - Crypto_Types.h, 195
- CRYPTO_KE_CERTIFICATE_CURRENT_TIME
 - Csm_Types.h, 203
- CRYPTO_KE_CERTIFICATE_DATA
 - Csm_Types.h, 203
- CRYPTO_KE_CERTIFICATE_EXTENSIONS
 - Csm_Types.h, 203
- CRYPTO_KE_CERTIFICATE_ISSUER
 - Csm_Types.h, 203
- CRYPTO_KE_CERTIFICATE_PARSING_FORMAT
 - Csm_Types.h, 204
- CRYPTO_KE_CERTIFICATE_SERIALNUMBER
 - Csm_Types.h, 204
- CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM
 - Csm_Types.h, 204
- CRYPTO_KE_CERTIFICATE_SIGNATURE
 - Csm_Types.h, 204
- CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY
 - Csm_Types.h, 204
- CRYPTO_KE_CERTIFICATE_SUBJECT
 - Csm_Types.h, 204
- CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER
 - Csm_Types.h, 205
- CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE
 - Csm_Types.h, 205
- CRYPTO_KE_CERTIFICATE_VERSION

- Csm_Types.h, [205](#)
- CRYPTO_KE_CIPHER_2NDKEY
 - Csm_Types.h, [205](#)
- CRYPTO_KE_CIPHER_IV
 - Csm_Types.h, [205](#)
- CRYPTO_KE_CIPHER_KEY
 - Csm_Types.h, [205](#)
- CRYPTO_KE_CIPHER_PROOF
 - Csm_Types.h, [206](#)
- CRYPTO_KE_KEYDERIVATION_ALGORITHM
 - Csm_Types.h, [206](#)
- CRYPTO_KE_KEYDERIVATION_ITERATIONS
 - Csm_Types.h, [206](#)
- CRYPTO_KE_KEYDERIVATION_PASSWORD
 - Csm_Types.h, [206](#)
- CRYPTO_KE_KEYDERIVATION_SALT
 - Csm_Types.h, [206](#)
- CRYPTO_KE_KEYEXCHANGE_ALGORITHM
 - Csm_Types.h, [206](#)
- CRYPTO_KE_KEYEXCHANGE_BASE
 - Csm_Types.h, [207](#)
- CRYPTO_KE_KEYEXCHANGE_OWNPUKEY
 - Csm_Types.h, [207](#)
- CRYPTO_KE_KEYEXCHANGE_PRIVKEY
 - Csm_Types.h, [207](#)
- CRYPTO_KE_KEYGENERATE_ALGORITHM
 - Csm_Types.h, [207](#)
- CRYPTO_KE_KEYGENERATE_KEY
 - Csm_Types.h, [207](#)
- CRYPTO_KE_KEYGENERATE_SEED
 - Csm_Types.h, [207](#)
- CRYPTO_KE_MAC_KEY
 - Csm_Types.h, [208](#)
- CRYPTO_KE_MAC_PROOF
 - Csm_Types.h, [208](#)
- CRYPTO_KE_RANDOM_ALGORITHM
 - Csm_Types.h, [208](#)
- CRYPTO_KE_RANDOM_SEED_STATE
 - Csm_Types.h, [208](#)
- CRYPTO_KE_SIGNATURE_KEY
 - Csm_Types.h, [208](#)
- CRYPTO_KEY_MATERIAL
 - Csm_Types.h, [208](#)
- CRYPTO_M1SIZE_U32
 - Crypto_Types.h, [196](#)
- CRYPTO_M2SIZE_U32
 - Crypto_Types.h, [196](#)
- CRYPTO_M3SIZE_U32
 - Crypto_Types.h, [196](#)
- CRYPTO_M4SIZE_U32
 - Crypto_Types.h, [196](#)
- CRYPTO_M5SIZE_U32
 - Crypto_Types.h, [196](#)
- CRYPTO_MCAL_ID
 - Oslf_Critical.h, [341](#)
- CRYPTO_PARAM_LEN_ALIGN
 - Crypto_Types.h, [196](#)
- CRYPTO_PARAM_LEN_IN
 - Crypto_Types.h, [197](#)
- CRYPTO_PARAM_LEN_LARGER
 - Crypto_Types.h, [197](#)
- CRYPTO_PARAM_LEN_MASK
 - Crypto_Types.h, [197](#)
- CRYPTO_SIZE_OUT_U32
 - Crypto_Types.h, [197](#)
- CRYPTO_SerViceNum
 - Crypto_Types.h, [197](#)
- CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE
 - Csm_Types.h, [209](#)
- callbackId
 - Crypto_JobPrimitiveInfoType, [27](#)
- callbackUpdateNotification
 - Crypto_JobPrimitiveInfoType, [27](#)
- Can.h, [120](#)
 - CAN_E_DATA_LOST, [122](#)
 - CAN_E_ECC_ERROR, [123](#)
 - CAN_E_ICOM_CONFIG_INVALID, [123](#)
 - CAN_E_INIT_FAILED, [123](#)
 - CAN_E_INVALID_CONTROLLER_MODE, [123](#)
 - CAN_E_INVALID_CONTROLLER, [123](#)
 - CAN_E_PARAM_BAUDRATE, [123](#)
 - CAN_E_PARAM_CONTROLLER, [124](#)
 - CAN_E_PARAM_DATA_LENGTH, [124](#)
 - CAN_E_PARAM_HANDLE, [124](#)
 - CAN_E_PARAM_POINTER, [124](#)
 - CAN_E_TRANSITION, [124](#)
 - CAN_E_UNINIT, [124](#)
 - CAN_INSTANCE_ID, [125](#)
 - CAN_SID_CBK_CHECK_WAKEUP, [125](#)
 - CAN_SID_CHANGE_BAUDRATE, [125](#)
 - CAN_SID_CHECK_BAUDRATE, [125](#)
 - CAN_SID_DEINIT, [125](#)
 - CAN_SID_DISABLE_CONTROLLER_INTERRUPTS, [125](#)
 - CAN_SID_ENABLE_CONTROLLER_INTERRUPTS, [126](#)
 - CAN_SID_GET_VERSION_INFO, [126](#)
 - CAN_SID_GETCONTROLLERERRORSTATE, [126](#)
 - CAN_SID_GETCONTROLLERMODE, [126](#)
 - CAN_SID_GetControllerRxErrorCounter, [126](#)
 - CAN_SID_GetControllerTxErrorCounter, [126](#)
 - CAN_SID_INIT, [127](#)
 - CAN_SID_MAIN_FUNCTION_BUS_OFF, [127](#)
 - CAN_SID_MAIN_FUNCTION_MODE, [127](#)
 - CAN_SID_MAIN_FUNCTION_READ, [127](#)
 - CAN_SID_MAIN_FUNCTION_WAKEUP, [127](#)
 - CAN_SID_MAIN_FUNCTION_WRITE, [127](#)
 - CAN_SID_SET_BAUDRATE, [128](#)
 - CAN_SID_SET_CONTROLLER_MODE, [128](#)
 - CAN_SID_WRITE, [128](#)
 - Can_CheckWakeup, [130](#)
 - Can_DeInit, [130](#)
 - Can_DisableControllerInterrupts, [131](#)
 - Can_EnableControllerInterrupts, [131](#)
 - Can_GenerateConfigPB, [138](#)
 - Can_GetControllerErrorState, [131](#)
 - Can_GetControllerMode, [132](#)
 - Can_GetControllerRxErrorCounter, [132](#)
 - Can_GetControllerTxErrorCounter, [133](#)
 - Can_GetVersionInfo, [133](#)
 - Can_HandleType, [128](#)
 - Can_Init, [134](#)

- Can_MainFunction_BusOff, 134
- Can_MainFunction_Mode, 135
- Can_MainFunction_Read, 135
- Can_MainFunction_Wakeup, 135
- Can_MainFunction_Write, 136
- Can_ModeType, 128
- Can_OverflowModeType, 129
- Can_PduldType, 129
- Can_SetBaudrate, 136
- Can_SetControllerMode, 137
- Can_StatusType, 129
- Can_Write, 137
- Can_BitrateConfigType, 10
 - PRESC, 10
 - SEG_1, 10
 - SEG_2, 10
 - SJW, 11
- Can_CheckWakeup
 - Can.h, 130
- Can_ConfigType, 11
 - CanHOHPtr, 11
 - ControlerNumber, 11
 - ControllerDescriptorsPtr, 12
 - HohNumber, 12
- Can_ControllerBaudrateConfigType, 12
 - ControllerBaudRateConfigID, 12
 - ControllerBaudRateKbps, 13
 - ControllerBitrate, 13
 - ControllerFDConfig, 13
- Can_ControllerDescriptorType, 13
 - AutoBusOffRecover, 14
 - BusoffInterrupt, 14
 - CanControllerActivation, 14
 - Channel, 14
 - ControlerID, 14
 - ControllerBaudrateConfigsPtr, 14
 - DefaultBaudRateIndex, 15
 - ECUMWakeupSourceId, 15
 - MaxBaudRateCount, 15
 - MemEccEn, 15
 - OverflowMode, 15
 - RxInterrupt, 15
 - TimeStampEn, 16
 - TxInterrupt, 16
 - WakeUpEn, 16
 - WakeUpInterrupt, 16
- Can_ControllerFdConfigType, 16
 - CanBrsEn, 17
 - CanContrTrcvDelayCompensation, 17
 - CanFdBitrate, 17
 - CanFdEnable, 17
 - CanFdIsoMode, 17
- Can_ControllerStateType
 - Can_GeneralTypes.h, 142
- Can_DeInit
 - Can.h, 130
- Can_DisableControllerInterrupts
 - Can.h, 131
- Can_EnableControllerInterrupts
 - Can.h, 131
- Can_ErrorStateType
 - Can_GeneralTypes.h, 142
- Can_FilterControlType, 18
 - Code, 18
 - IdType, 18
 - Mask, 18
- Can_GeneralTypes.h, 138
 - CAN_BUSY, 139
 - CAN_CANCTRL_MAX_PAYLOAD12_U8, 139
 - CAN_CANCTRL_MAX_PAYLOAD16_U8, 139
 - CAN_CANCTRL_MAX_PAYLOAD20_U8, 139
 - CAN_CANCTRL_MAX_PAYLOAD24_U8, 139
 - CAN_CANCTRL_MAX_PAYLOAD32_U8, 140
 - CAN_CANCTRL_MAX_PAYLOAD48_U8, 140
 - CAN_CANCTRL_MAX_PAYLOAD64_U8, 140
 - CAN_CANCTRL_MAX_PAYLOAD8_U8, 140
 - CAN_EXTEND_FD_ID_DESCRIPTOR, 140
 - CAN_EXTEND_ID_DESCRIPTOR, 140
 - CAN_FD_ID_DESCRIPTOR, 141
 - CAN_FD_ID_EXTEND_DEF_MASK, 141
 - CAN_ID_EXTENDED_MASK_U32, 141
 - CAN_ID_EXTENDEDDIFF_MASK_U32, 141
 - CAN_ID_STANDARD_MASK_U32, 141
 - CAN_IDE_ID_DESCRIPTOR, 141
 - Can_ControllerStateType, 142
 - Can_ErrorStateType, 142
 - Can_HwHandleType, 142
 - Can_IdType, 142
- Can_GenerateConfigPB
 - Can.h, 138
- Can_GetControllerErrorState
 - Can.h, 131
- Can_GetControllerMode
 - Can.h, 132
- Can_GetControllerRxErrorCounter
 - Can.h, 132
- Can_GetControllerTxErrorCounter
 - Can.h, 133
- Can_GetVersionInfo
 - Can.h, 133
- Can_HandleType
 - Can.h, 128
- Can_HardwareObjectType, 19
 - CanControllerRef, 19
 - CanFdPaddingValue, 19
 - CanFilterListPtr, 20
 - CanHWObjectCount, 20
 - CanHandle, 20
 - CanHwObjectUsesPolling, 20
 - CanIdType, 20
 - CanMainFunctionRWPeriodRef, 20
 - CanObjectIndex, 21
 - CanRwMode, 21
 - FilterNumber, 21
- Can_HwHandleType
 - Can_GeneralTypes.h, 142
- Can_HwType, 21
 - CanId, 22
 - ControllerId, 22
 - Hoh, 22
- Can_IdType
 - Can_GeneralTypes.h, 142

- Can_Init
 - Can.h, [134](#)
- Can_MainFunction_BusOff
 - Can.h, [134](#)
- Can_MainFunction_Mode
 - Can.h, [135](#)
- Can_MainFunction_Read
 - Can.h, [135](#)
- Can_MainFunction_Wakeup
 - Can.h, [135](#)
- Can_MainFunction_Write
 - Can.h, [136](#)
- Can_ModeType
 - Can.h, [128](#)
- Can_OverflowModeType
 - Can.h, [129](#)
- Can_PduType
 - Can.h, [129](#)
- Can_PduType, [22](#)
 - id, [23](#)
 - length, [23](#)
 - sdu, [23](#)
 - swPduHandle, [23](#)
- Can_SetBaudrate
 - Can.h, [136](#)
- Can_SetControllerMode
 - Can.h, [137](#)
- Can_StatusType
 - Can.h, [129](#)
- Can_Write
 - Can.h, [137](#)
- CanBrsEn
 - Can_ControllerFdConfigType, [17](#)
- CanContrTrcvDelayCompensation
 - Can_ControllerFdConfigType, [17](#)
- CanControllerActivation
 - Can_ControllerDescriptorType, [14](#)
- CanControllerRef
 - Can_HardwareObjectType, [19](#)
- CanFdBitrate
 - Can_ControllerFdConfigType, [17](#)
- CanFdEnable
 - Can_ControllerFdConfigType, [17](#)
- CanFdIsoMode
 - Can_ControllerFdConfigType, [17](#)
- CanFdPaddingValue
 - Can_HardwareObjectType, [19](#)
- CanFilterListPtr
 - Can_HardwareObjectType, [20](#)
- CanHOHPtr
 - Can_ConfigType, [11](#)
- CanHWObjectCount
 - Can_HardwareObjectType, [20](#)
- CanHandle
 - Can_HardwareObjectType, [20](#)
- CanHwObjectUsesPolling
 - Can_HardwareObjectType, [20](#)
- CanId
 - Can_HwType, [22](#)
- CanIdType
 - Can_HardwareObjectType, [20](#)
- CanMainFunctionRWPeriodRef
 - Can_HardwareObjectType, [20](#)
- CanObjectIndex
 - Can_HardwareObjectType, [21](#)
- CanRwMode
 - Can_HardwareObjectType, [21](#)
- ChConfigPtr
 - I2c_CHConfigType, [77](#)
- Channel
 - Can_ControllerDescriptorType, [14](#)
 - Icu_ChannelConfigType, [78](#)
 - Ocu_ChannelConfigType, [87](#)
 - Pwm_ChannelConfigType, [98](#)
- ChannelCfg
 - Pwm_ConfigType, [101](#)
- ChannelClass
 - Pwm_ChannelConfigType, [98](#)
- ChannelConfigPtr
 - I2c_ConfigType, [77](#)
 - Icu_ConfigType, [81](#)
- ChannelDefaultThreshold
 - Ocu_ChannelConfigType, [87](#)
- ChannelGroupList
 - Dio_ConfigType, [47](#)
- ChannelInitLevel
 - Ocu_ChannelConfigType, [87](#)
- ChannelMode
 - Pwm_ChannelConfigType, [98](#)
- ChannelNotification
 - Icu_ChannelConfigType, [78](#)
 - Ocu_ChannelConfigType, [88](#)
 - Pwm_ChannelConfigType, [99](#)
- ChannelOutputPinUsed
 - Ocu_ChannelConfigType, [88](#)
- ChannelPinAction
 - Ocu_ChannelConfigType, [88](#)
- CheckSum
 - Fee_BlockHeaderType, [48](#)
 - Fee_ClusterHeaderType, [54](#)
- Clock
 - Port_DigitalFilterConfigType, [94](#)
- ClockSource
 - Icu_PwmModuleConfigType, [84](#)
 - Ocu_ModuleConfigType, [91](#)
 - Pwm_ModuleConfigType, [103](#)
- ClusterCount
 - Fee_ClusterGroupType, [53](#)
- ClusterGrp
 - Fee_BlockType, [52](#)
- ClusterGrpPtr
 - Fee_ConfigType, [58](#)
- ClusterId
 - Fee_ClusterHeaderType, [54](#)
- ClusterIdx
 - Fee_JobInfoType, [59](#)
- ClusterPtr
 - Fee_ClusterGroupType, [53](#)
- Code
 - Can_FilterControlType, [18](#)
- ControlerID
 - Can_ControllerDescriptorType, [14](#)

- ControlerNumber
 - Can_ConfigType, 11
- ControllerBaudRateConfigID
 - Can_ControllerBaudrateConfigType, 12
- ControllerBaudRateKbps
 - Can_ControllerBaudrateConfigType, 13
- ControllerBaudrateConfigsPtr
 - Can_ControllerDescriptorType, 14
- ControllerBitrate
 - Can_ControllerBaudrateConfigType, 13
- ControllerDescriptorsPtr
 - Can_ConfigType, 12
- ControllerFDConfig
 - Can_ControllerBaudrateConfigType, 13
- ControllerId
 - Can_HwType, 22
- Count
 - Crypto_Key, 38
- CountMode
 - Pwm_ModuleConfigType, 103
- Crc_CalculateCRC16
 - CDD_Crc.h, 151
- Crc_CalculateCRC16ARC
 - CDD_Crc.h, 151
- Crc_CalculateCRC32
 - CDD_Crc.h, 152
- Crc_CalculateCRC32P4
 - CDD_Crc.h, 152
- Crc_CalculateCRC64
 - CDD_Crc.h, 153
- Crc_CalculateCRC8
 - CDD_Crc.h, 153
- Crc_CalculateCRC8H2F
 - CDD_Crc.h, 154
- Crc_CalculateCRC
 - CDD_Crc.h, 150
- Crc_Deinit
 - CDD_Crc.h, 155
- Crc_DmaConfig, 23
 - DmaChannel, 24
 - DmaUsed, 24
 - Params, 24
- Crc_GetConfig
 - CDD_Crc.h, 155
- Crc_GetCrcResult
 - CDD_Crc.h, 156
- Crc_GetVersionInfo
 - CDD_Crc.h, 156
- Crc_Init
 - CDD_Crc.h, 156
- crylfKeyId
 - Crypto_JobPrimitiveInfoType, 28
 - Crypto_JobPrimitiveInputOutputType, 29
- Crypto.h, 183
 - CRYPTO_E_INIT_FAILED, 185
 - CRYPTO_E_PARAM_HANDLE, 185
 - CRYPTO_E_PARAM_POINTER, 185
 - CRYPTO_E_PARAM_VALUE, 185
 - CRYPTO_E_UNINIT, 185
 - CRYPTO_FUNCTION_BOOTDEFINE, 185
 - CRYPTO_FUNCTION_BOOTFAILURE, 186
 - CRYPTO_FUNCTION_BOOTOK, 186
 - CRYPTO_FUNCTION_CANCELJOB, 186
 - CRYPTO_FUNCTION_CERTIFICATEPARSE, 186
 - CRYPTO_FUNCTION_CERTIFICATEVERIFY, 186
 - CRYPTO_FUNCTION_DEBUGAUTH, 186
 - CRYPTO_FUNCTION_DEBUGCHAL, 187
 - CRYPTO_FUNCTION_GETID, 187
 - CRYPTO_FUNCTION_GETSTATUS, 187
 - CRYPTO_FUNCTION_GETVERSIONINFO, 187
 - CRYPTO_FUNCTION_INIT, 187
 - CRYPTO_FUNCTION_KEYCOPY, 187
 - CRYPTO_FUNCTION_KEYDERIVE, 188
 - CRYPTO_FUNCTION_KEYELEMENTCOPY, 188
 - CRYPTO_FUNCTION_KEYELEMENTGET, 188
 - CRYPTO_FUNCTION_KEYELEMENTIDSGET, 188
 - CRYPTO_FUNCTION_KEYELEMENTSET, 188
 - CRYPTO_FUNCTION_KEYEXCHANGEALCPUB←
VAL, 188
 - CRYPTO_FUNCTION_KEYEXCHANGEALCSEC←
RET, 189
 - CRYPTO_FUNCTION_KEYGENERATE, 189
 - CRYPTO_FUNCTION_KEYVALIDSET, 189
 - CRYPTO_FUNCTION_MPCOMPRESSION, 189
 - CRYPTO_FUNCTION_PROCESSJOB, 189
 - CRYPTO_FUNCTION_RANDOMSEED, 189
 - Crypto_CancelJob, 190
 - Crypto_GetVersionInfo, 190
 - Crypto_Init, 191
 - Crypto_KeyElementGet, 191
 - Crypto_KeyElementIdsGet, 192
 - Crypto_KeyElementSet, 192
 - Crypto_KeySetValid, 193
 - Crypto_MainFunction, 193
 - Crypto_ProcessJob, 194
- Crypto_AlgorithmFamilyType
 - Csm_Types.h, 209
- Crypto_AlgorithmInfoType, 24
 - family, 25
 - keyLength, 25
 - mode, 25
 - secondaryFamily, 25
- Crypto_AlgorithmModeType
 - Csm_Types.h, 210
- Crypto_CancelJob
 - Crypto.h, 190
- Crypto_DriverStateType
 - Crypto_Types.h, 198
- Crypto_GetVersionInfo
 - Crypto.h, 190
- Crypto_Init
 - Crypto.h, 191
- Crypto_InputOutputRedirectionConfigType
 - Csm_Types.h, 210
- Crypto_JobInfoType, 26
 - JobId, 26
 - JobPriority, 26
- Crypto_JobPrimitiveInfoType, 27
 - callbackId, 27
 - callbackUpdateNotification, 27
 - crylfKeyId, 28
 - primitiveInfo, 28

- processingType, 28
- Crypto_JobPrimitiveInputOutputType, 28
 - crylfKeyId, 29
 - input64, 29
 - inputLength, 29
 - inputPtr, 30
 - mode, 30
 - output64Ptr, 30
 - outputLengthPtr, 30
 - outputPtr, 30
 - secondaryInputLength, 31
 - secondaryInputPtr, 31
 - secondaryOutputLengthPtr, 31
 - secondaryOutputPtr, 31
 - targetCrylfKeyId, 31
 - tertiaryInputLength, 32
 - tertiaryInputPtr, 32
 - verifyPtr, 32
- Crypto_JobRedirectionInfoType, 32
 - inputKeyElementId, 33
 - inputKeyId, 33
 - outputKeyElementId, 33
 - outputKeyId, 33
 - redirectionConfig, 34
 - secondaryInputKeyElementId, 34
 - secondaryInputKeyId, 34
 - secondaryOutputKeyElementId, 34
 - secondaryOutputKeyId, 34
 - tertiaryInputKeyElementId, 35
 - tertiaryInputKeyId, 35
- Crypto_JobStateType
 - Csm_Types.h, 211
- Crypto_JobType, 35
 - cryptoKeyId, 36
 - jobId, 36
 - jobInfo, 36
 - jobPrimitiveInfo, 36
 - jobPrimitiveInputOutput, 36
 - jobRedirectionInfoRef, 37
 - jobState, 37
- Crypto_Key, 37
 - Count, 38
 - KeyId, 38
 - KeyTypePtr, 38
 - KeyValid, 38
- Crypto_KeyElementGet
 - Crypto.h, 191
- Crypto_KeyElementIdsGet
 - Crypto.h, 192
- Crypto_KeyElementSet
 - Crypto.h, 192
- Crypto_KeySetValid
 - Crypto.h, 193
- Crypto_MainFunction
 - Crypto.h, 193
- Crypto_ObjectType, 38
 - DriverObjectId, 39
 - HeadOfFree, 39
 - HeadOfJobs, 39
 - PrimitivesCount, 39
 - PrimitivesPtr, 40
- QueueSize, 40
- QueuedJobs, 40
- Crypto_OperationModeType
 - Csm_Types.h, 211
- Crypto_PrimitiveInfoType, 40
 - algorithm, 41
 - resultLength, 41
 - service, 41
- Crypto_PrimitiveType, 42
 - Family, 42
 - Mode, 42
 - SecondaryFamily, 42
 - Service, 43
- Crypto_ProcessJob
 - Crypto.h, 194
- Crypto_ProcessingType
 - Csm_Types.h, 211
- Crypto_QueueType
 - Crypto_Types.h, 198
- Crypto_ServiceInfoType
 - Csm_Types.h, 212
- Crypto_Types.h, 194
 - CRYPTO_HW_NOT_SUPPORTED, 195
 - CRYPTO_M1SIZE_U32, 196
 - CRYPTO_M2SIZE_U32, 196
 - CRYPTO_M3SIZE_U32, 196
 - CRYPTO_M4SIZE_U32, 196
 - CRYPTO_M5SIZE_U32, 196
 - CRYPTO_PARAM_LEN_ALIGN, 196
 - CRYPTO_PARAM_LEN_IN, 197
 - CRYPTO_PARAM_LEN_LARGER, 197
 - CRYPTO_PARAM_LEN_MASK, 197
 - CRYPTO_SIZE_OUT_U32, 197
 - CRYPTO_SerViceNum, 197
 - Crypto_DriverStateType, 198
 - Crypto_QueueType, 198
 - CryptoKeyElementFormat, 198
 - CryptoKeyElementReadAccess, 199
 - CryptoKeyElementWriteAccess, 199
- Crypto_VerifyResultType
 - Csm_Types.h, 212
- CryptoKeyElement, 43
 - ActualSize, 43
 - AllowPartialAccess, 44
 - Id, 44
 - KeyFormatType, 44
 - MaxSize, 44
 - Persist, 44
 - ReadAccess, 44
 - Uniqueld, 45
 - ValuePtr, 45
 - WriteAccess, 45
- CryptoKeyElementFormat
 - Crypto_Types.h, 198
- CryptoKeyElementReadAccess
 - Crypto_Types.h, 199
- CryptoKeyElementWriteAccess
 - Crypto_Types.h, 199
- cryptoKeyId
 - Crypto_JobType, 36
- Csm_Types.h, 200

- CRYPTO_CSE_KE_SHE_M1_OFFSET, [202](#)
- CRYPTO_CSE_KE_SHE_M2_OFFSET, [202](#)
- CRYPTO_CSE_KE_SHE_M3_OFFSET, [202](#)
- CRYPTO_CSE_KE_SHE_M4_OFFSET, [203](#)
- CRYPTO_CSE_KE_SHE_M5_OFFSET, [203](#)
- CRYPTO_KE_CERTIFICATE_CURRENT_TIME, [203](#)
- CRYPTO_KE_CERTIFICATE_DATA, [203](#)
- CRYPTO_KE_CERTIFICATE_EXTENSIONS, [203](#)
- CRYPTO_KE_CERTIFICATE_ISSUER, [203](#)
- CRYPTO_KE_CERTIFICATE_PARSING_FORMAT, [204](#)
- CRYPTO_KE_CERTIFICATE_SERIALNUMBER, [204](#)
- CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM, [204](#)
- CRYPTO_KE_CERTIFICATE_SIGNATURE, [204](#)
- CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY, [204](#)
- CRYPTO_KE_CERTIFICATE_SUBJECT, [204](#)
- CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER, [205](#)
- CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE, [205](#)
- CRYPTO_KE_CERTIFICATE_VERSION, [205](#)
- CRYPTO_KE_CIPHER_2NDKEY, [205](#)
- CRYPTO_KE_CIPHER_IV, [205](#)
- CRYPTO_KE_CIPHER_KEY, [205](#)
- CRYPTO_KE_CIPHER_PROOF, [206](#)
- CRYPTO_KE_KEYDERIVATION_ALGORITHM, [206](#)
- CRYPTO_KE_KEYDERIVATION_ITERATIONS, [206](#)
- CRYPTO_KE_KEYDERIVATION_PASSWORD, [206](#)
- CRYPTO_KE_KEYDERIVATION_SALT, [206](#)
- CRYPTO_KE_KEYEXCHANGE_ALGORITHM, [206](#)
- CRYPTO_KE_KEYEXCHANGE_BASE, [207](#)
- CRYPTO_KE_KEYEXCHANGE_OWN_PUBKEY, [207](#)
- CRYPTO_KE_KEYEXCHANGE_PRIVKEY, [207](#)
- CRYPTO_KE_KEYGENERATE_ALGORITHM, [207](#)
- CRYPTO_KE_KEYGENERATE_KEY, [207](#)
- CRYPTO_KE_KEYGENERATE_SEED, [207](#)
- CRYPTO_KE_MAC_KEY, [208](#)
- CRYPTO_KE_MAC_PROOF, [208](#)
- CRYPTO_KE_RANDOM_ALGORITHM, [208](#)
- CRYPTO_KE_RANDOM_SEED_STATE, [208](#)
- CRYPTO_KE_SIGNATURE_KEY, [208](#)
- CRYPTO_KEY_MATERIAL, [208](#)
- CRYPTO_KE_KEYEXCHANGE_SHAREDVALUE, [209](#)
- Crypto_AlgorithmFamilyType, [209](#)
- Crypto_AlgorithmModeType, [210](#)
- Crypto_InputOutputRedirectionConfigType, [210](#)
- Crypto_JobStateType, [211](#)
- Crypto_OperationModeType, [211](#)
- Crypto_ProcessingType, [211](#)
- Crypto_ServiceInfoType, [212](#)
- Crypto_VerifyResultType, [212](#)
- DIO_E_PARAM_CONFIG
 - Dio_GeneralTypes.h, [219](#)
- DIO_E_PARAM_INVALID_CHANNEL_ID
 - Dio_GeneralTypes.h, [219](#)
- DIO_E_PARAM_INVALID_GROUP
 - Dio_GeneralTypes.h, [219](#)
- DIO_E_PARAM_INVALID_PORT_ID
 - Dio_GeneralTypes.h, [219](#)
- DIO_E_PARAM_LEVEL
 - Dio_GeneralTypes.h, [219](#)
- DIO_E_PARAM_POINTER
 - Dio_GeneralTypes.h, [220](#)
- DIO_FLIPCHANNEL_ID
 - Dio_GeneralTypes.h, [220](#)
- DIO_GETVERSIONINFO_ID
 - Dio_GeneralTypes.h, [220](#)
- DIO_INSTANCE_ID
 - Dio_GeneralTypes.h, [220](#)
- DIO_MASKEDWRITEPORT_ID
 - Dio_GeneralTypes.h, [220](#)
- DIO_READCHANNEL_ID
 - Dio_GeneralTypes.h, [221](#)
- DIO_READCHANNELGROUP_ID
 - Dio_GeneralTypes.h, [221](#)
- DIO_READPORT_ID
 - Dio_GeneralTypes.h, [221](#)
- DIO_WRITECHANNEL_ID
 - Dio_GeneralTypes.h, [221](#)
- DIO_WRITECHANNELGROUP_ID
 - Dio_GeneralTypes.h, [221](#)
- DIO_WRITEPORT_ID
 - Dio_GeneralTypes.h, [222](#)
- DMA_HAL_ID1
 - Oslf_Critical.h, [341](#)
- DMA_HAL_ID2
 - Oslf_Critical.h, [342](#)
- DataLength
 - Fee_JobInfoType, [59](#)
- DataOffset
 - Fee_JobInfoType, [60](#)
- DataOut
 - Port_PinConfigType, [95](#)
 - Port_UnUsedPinConfigType, [97](#)
- DataPtr
 - Fee_JobInfoType, [60](#)
- DeadTimeValue
 - Pwm_ChannelConfigType, [99](#)
- DefaultBaudRateIndex
 - Can_ControllerDescriptorType, [15](#)
- DefaultMode
 - Fls_ConfigType, [64](#)
 - Wdg_ConfigType, [107](#)
- DefaultStartEdge
 - Icu_ChannelConfigType, [79](#)
- DigitalFilterConfig
 - Port_ConfigType, [92](#)
- Dio.h, [213](#)
 - Dio_FlipChannel, [213](#)
 - Dio_GetVersionInfo, [214](#)
 - Dio_MaskedWritePort, [214](#)
 - Dio_ReadChannel, [215](#)
 - Dio_ReadChannelGroup, [215](#)
 - Dio_ReadPort, [215](#)
 - Dio_WriteChannel, [216](#)
 - Dio_WriteChannelGroup, [216](#)
 - Dio_WritePort, [217](#)
- Dio_ChannelGroupType, [45](#)

- Mask, [46](#)
- Offset, [46](#)
- Port, [46](#)
- Dio_ChannelType
 - Dio_GeneralTypes.h, [222](#)
- Dio_ConfigType, [46](#)
 - ChannelGroupList, [47](#)
 - NumChannelGroups, [47](#)
- Dio_FlipChannel
 - Dio.h, [213](#)
- Dio_GeneralTypes.h, [217](#)
 - DIO_E_PARAM_CONFIG, [219](#)
 - DIO_E_PARAM_INVALID_CHANNEL_ID, [219](#)
 - DIO_E_PARAM_INVALID_GROUP, [219](#)
 - DIO_E_PARAM_INVALID_PORT_ID, [219](#)
 - DIO_E_PARAM_LEVEL, [219](#)
 - DIO_E_PARAM_POINTER, [220](#)
 - DIO_FLIPCHANNEL_ID, [220](#)
 - DIO_GETVERSIONINFO_ID, [220](#)
 - DIO_INSTANCE_ID, [220](#)
 - DIO_MASKEDWRITEPORT_ID, [220](#)
 - DIO_READCHANNEL_ID, [221](#)
 - DIO_READCHANNELGROUP_ID, [221](#)
 - DIO_READPORT_ID, [221](#)
 - DIO_WRITECHANNEL_ID, [221](#)
 - DIO_WRITECHANNELGROUP_ID, [221](#)
 - DIO_WRITEPORT_ID, [222](#)
 - Dio_ChannelType, [222](#)
 - Dio_LevelType, [222](#)
 - Dio_PortLevelType, [222](#)
 - Dio_PortType, [222](#)
- Dio_GetVersionInfo
 - Dio.h, [214](#)
- Dio_LevelType
 - Dio_GeneralTypes.h, [222](#)
- Dio_MaskedWritePort
 - Dio.h, [214](#)
- Dio_PortLevelType
 - Dio_GeneralTypes.h, [222](#)
- Dio_PortType
 - Dio_GeneralTypes.h, [222](#)
- Dio_ReadChannel
 - Dio.h, [215](#)
- Dio_ReadChannelGroup
 - Dio.h, [215](#)
- Dio_ReadPort
 - Dio.h, [215](#)
- Dio_WriteChannel
 - Dio.h, [216](#)
- Dio_WriteChannelGroup
 - Dio.h, [216](#)
- Dio_WritePort
 - Dio.h, [217](#)
- DirChangeable
 - Port_PinConfigType, [95](#)
- Direction
 - Port_PinConfigType, [95](#)
 - Port_UnusedPinConfigType, [97](#)
- DmaChannel
 - Crc_DmaConfig, [24](#)
- DmaUsed
 - Crc_DmaConfig, [24](#)
 - DriverObjectId
 - Crypto_ObjectType, [39](#)
 - ECUMWakeupSourceId
 - Can_ControllerDescriptorType, [15](#)
 - EccFaultAddress
 - FlsTst_ErrorDetailsType, [73](#)
 - EccStatus
 - FlsTst_ErrorDetailsType, [74](#)
 - Eep.h, [223](#)
 - Eep_Cancel, [223](#)
 - Eep_Compare, [223](#)
 - Eep_Erase, [223](#)
 - Eep_GetJobResult, [224](#)
 - Eep_GetStatus, [224](#)
 - Eep_GetVersionInfo, [224](#)
 - Eep_Init, [224](#)
 - Eep_Read, [224](#)
 - Eep_SetMode, [224](#)
 - Eep_Write, [224](#)
 - Eep_Cancel
 - Eep.h, [223](#)
 - Eep_Compare
 - Eep.h, [223](#)
 - Eep_Erase
 - Eep.h, [223](#)
 - Eep_GetJobResult
 - Eep.h, [224](#)
 - Eep_GetStatus
 - Eep.h, [224](#)
 - Eep_GetVersionInfo
 - Eep.h, [224](#)
 - Eep_Init
 - Eep.h, [224](#)
 - Eep_Read
 - Eep.h, [224](#)
 - Eep_SetMode
 - Eep.h, [224](#)
 - Eep_Write
 - Eep.h, [224](#)
 - EnableCombaineComple
 - Pwm_ChannelConfigType, [99](#)
 - EnableDeadTime
 - Pwm_ChannelConfigType, [99](#)
 - EnableInitTrigger
 - Pwm_ModuleConfigType, [103](#)
 - EnableInterrupt
 - Pwm_ChannelConfigType, [99](#)
 - Pwm_ModuleConfigType, [103](#)
 - EnableMatchTrigger
 - Pwm_ChannelConfigType, [99](#)
 - EnableMaxTrigger
 - Pwm_ModuleConfigType, [104](#)
 - EnableOverflowEvent
 - Pwm_ModuleConfigType, [104](#)
 - EraseTimeout
 - Fls_ConfigType, [65](#)
 - FEE_BLOCK_INVALID_VALUE
 - Fee_InternalTypes.h, [239](#)

FEE_BLOCKHEAD_OVERHEAD
 Fee_GeneralTypes.h, 233

FEE_CANCEL_ID
 Fee_GeneralTypes.h, 233

FEE_CLUSTERHEAD_OVERHEAD
 Fee_GeneralTypes.h, 233

FEE_E_BUSY_INTERNAL
 Fee_GeneralTypes.h, 233

FEE_E_BUSY
 Fee_GeneralTypes.h, 233

FEE_E_CANCEL_API
 Fee_GeneralTypes.h, 234

FEE_E_CLUSTER_GROUP_IDX
 Fee_GeneralTypes.h, 234

FEE_E_FOREIGN_BLOCKS_OVF
 Fee_GeneralTypes.h, 234

FEE_E_INIT_FAILED
 Fee_GeneralTypes.h, 234

FEE_E_INVALID_BLOCK_LEN
 Fee_GeneralTypes.h, 234

FEE_E_INVALID_BLOCK_NO
 Fee_GeneralTypes.h, 234

FEE_E_INVALID_BLOCK_OFS
 Fee_GeneralTypes.h, 235

FEE_E_INVALID_CANCEL
 Fee_GeneralTypes.h, 235

FEE_E_PARAM_POINTER
 Fee_GeneralTypes.h, 235

FEE_E_UNINIT
 Fee_GeneralTypes.h, 235

FEE_ERASEIMMEDIATEBLOCK_ID
 Fee_GeneralTypes.h, 235

FEE_FLAG_OVERHEAD
 Fee_GeneralTypes.h, 235

FEE_GETJOBRESULT_ID
 Fee_GeneralTypes.h, 236

FEE_GETSTATUS_ID
 Fee_GeneralTypes.h, 236

FEE_GETVERSIONINFO_ID
 Fee_GeneralTypes.h, 236

FEE_INIT_ID
 Fee_GeneralTypes.h, 236

FEE_INSTANCE_ID
 Fee_GeneralTypes.h, 236

FEE_INVALIDATEBLOCK_ID
 Fee_GeneralTypes.h, 236

FEE_JOBENDNOTIFICATION_ID
 Fee_GeneralTypes.h, 237

FEE_JOBERRORNOTIFICATION_ID
 Fee_GeneralTypes.h, 237

FEE_MAINFUNCTION_ID
 Fee_GeneralTypes.h, 237

FEE_READ_ID
 Fee_GeneralTypes.h, 237

FEE_SETMODE_ID
 Fee_GeneralTypes.h, 237

FEE_WRITE_ID
 Fee_GeneralTypes.h, 237

FLS_BLANK_CHECK_ID
 Fls_GeneralTypes.h, 247

FLS_CANCEL_ID
 Fls_GeneralTypes.h, 248

FLS_COMPARE_ID
 Fls_GeneralTypes.h, 248

FLS_E_BUSY
 Fls_GeneralTypes.h, 248

FLS_E_COMPARE_FAILED
 Fls_GeneralTypes.h, 248

FLS_E_ERASE_FAILED
 Fls_GeneralTypes.h, 248

FLS_E_PARAM_ADDRESS
 Fls_GeneralTypes.h, 248

FLS_E_PARAM_CONFIG
 Fls_GeneralTypes.h, 249

FLS_E_PARAM_DATA
 Fls_GeneralTypes.h, 249

FLS_E_PARAM_LENGTH
 Fls_GeneralTypes.h, 249

FLS_E_PARAM_POINTER
 Fls_GeneralTypes.h, 249

FLS_E_READ_FAILED
 Fls_GeneralTypes.h, 249

FLS_E_TIMEOUT
 Fls_GeneralTypes.h, 249

FLS_E_UNEXPECTED_FLASH_ID
 Fls_GeneralTypes.h, 250

FLS_E_UNINIT
 Fls_GeneralTypes.h, 250

FLS_E_VERIFY_ERASE_FAILED
 Fls_GeneralTypes.h, 250

FLS_E_VERIFY_WRITE_FAILED
 Fls_GeneralTypes.h, 250

FLS_E_WRITE_FAILED
 Fls_GeneralTypes.h, 250

FLS_ERASE_ID
 Fls_GeneralTypes.h, 250

FLS_GETJOBRESULT_ID
 Fls_GeneralTypes.h, 251

FLS_GETSTATUS_ID
 Fls_GeneralTypes.h, 251

FLS_GETVERSIONINFO_ID
 Fls_GeneralTypes.h, 251

FLS_HAL_ID1
 Oslf_Critical.h, 342

FLS_INIT_ID
 Fls_GeneralTypes.h, 251

FLS_INSTANCE_ID
 Fls_GeneralTypes.h, 251

FLS_MAINFUNCTION_ID
 Fls_GeneralTypes.h, 251

FLS_PAGE_WRITE_ASYNC
 Fls_GeneralTypes.h, 252

FLS_READ_ID
 Fls_GeneralTypes.h, 252

FLS_SECTOR_ERASE_ASYNC
 Fls_GeneralTypes.h, 252

FLS_SETMODE_ID
 Fls_GeneralTypes.h, 252

FLS_WRITE_ID
 Fls_GeneralTypes.h, 252

FLSTST_ABORT_SVCID
 FlsTst.h, 257

- FLSTST_DEINIT_SVCID
 - FlsTst.h, [257](#)
- FLSTST_E_ALREADY_INITIALIZED
 - FlsTst.h, [257](#)
- FLSTST_E_FLSTST_FAILURE
 - FlsTst.h, [257](#)
- FLSTST_E_INIT_FAILED
 - FlsTst.h, [258](#)
- FLSTST_E_PARAM_INVALID
 - FlsTst.h, [258](#)
- FLSTST_E_PARAM_POINTER
 - FlsTst.h, [258](#)
- FLSTST_E_STATE_FAILURE
 - FlsTst.h, [258](#)
- FLSTST_E_UNINIT
 - FlsTst.h, [258](#)
- FLSTST_GET_CURRENT_STATE_SVCID
 - FlsTst.h, [259](#)
- FLSTST_GET_ERROR_DETAILS_SVCID
 - FlsTst.h, [259](#)
- FLSTST_GET_TEST_RESULT_BGND_SVCID
 - FlsTst.h, [259](#)
- FLSTST_GET_TEST_RESULT_FGND_SVCID
 - FlsTst.h, [259](#)
- FLSTST_GET_TEST_SIGNATURE_BGND_SVCID
 - FlsTst.h, [259](#)
- FLSTST_GET_TEST_SIGNATURE_FGND_SVCID
 - FlsTst.h, [260](#)
- FLSTST_GET_VERSION_INFO_SVCID
 - FlsTst.h, [260](#)
- FLSTST_HAL_ID1
 - Oslf_Critical.h, [342](#)
- FLSTST_INIT_SVCID
 - FlsTst.h, [260](#)
- FLSTST_MAIN_FUNCTION_SVCID
 - FlsTst.h, [260](#)
- FLSTST_RESUME_SVCID
 - FlsTst.h, [260](#)
- FLSTST_START_FGND_SVCID
 - FlsTst.h, [261](#)
- FLSTST_SUSPEND_SVCID
 - FlsTst.h, [261](#)
- FLSTST_TEST_COMPLETED_NOTIFICATION_SVCID
 - FlsTst.h, [261](#)
- FLSTST_TEST_ECC_SVCID
 - FlsTst.h, [261](#)
- Family
 - Crypto_PrimitiveType, [42](#)
- family
 - Crypto_AlgorithmInfoType, [25](#)
- Fee.h, [225](#)
 - Fee_Cancel, [225](#)
 - Fee_EraseImmediateBlock, [226](#)
 - Fee_GetJobResult, [226](#)
 - Fee_GetStatus, [227](#)
 - Fee_GetVersionInfo, [228](#)
 - Fee_Init, [228](#)
 - Fee_InvalidateBlock, [228](#)
 - Fee_JobEndNotification, [229](#)
 - Fee_JobErrorNotification, [229](#)
 - Fee_Read, [230](#)
 - Fee_SetMode, [230](#)
 - Fee_Write, [231](#)
- Fee_BlockAssignmentType
 - Fee_GeneralTypes.h, [237](#)
- Fee_BlockHeaderType, [47](#)
 - Assigned, [48](#)
 - BlockId, [48](#)
 - Checksum, [48](#)
 - ImmediateBlock, [48](#)
 - Invalid, [49](#)
 - Length, [49](#)
 - TargetAddr, [49](#)
 - Valid, [49](#)
- Fee_BlockInfoType, [50](#)
 - HeadAddr, [50](#)
 - Header, [50](#)
 - Status, [50](#)
- Fee_BlockStatusType
 - Fee_InternalTypes.h, [239](#)
- Fee_BlockType, [51](#)
 - BlockAssign, [51](#)
 - BlockNumber, [51](#)
 - BlockSize, [52](#)
 - ClusterGrp, [52](#)
 - ImmediateData, [52](#)
- Fee_Cancel
 - Fee.h, [225](#)
- Fee_ClusterGroupType, [52](#)
 - BlockPtr, [53](#)
 - ClusterCount, [53](#)
 - ClusterPtr, [53](#)
 - ReservedSize, [53](#)
- Fee_ClusterHeaderType, [54](#)
 - Checksum, [54](#)
 - ClusterId, [54](#)
 - Invalid, [55](#)
 - Length, [55](#)
 - StartAddr, [55](#)
 - Valid, [55](#)
- Fee_ClusterInfoType, [56](#)
 - Header, [56](#)
 - Status, [56](#)
- Fee_ClusterStatusType
 - Fee_InternalTypes.h, [239](#)
- Fee_ClusterType, [56](#)
 - Length, [57](#)
 - StartAddr, [57](#)
- Fee_ConfigType, [57](#)
 - ClusterGrpPtr, [58](#)
 - GrpCount, [58](#)
- Fee_EraseImmediateBlock
 - Fee.h, [226](#)
- Fee_GeneralTypes.h, [232](#)
 - FEE_BLOCKHEAD_OVERHEAD, [233](#)
 - FEE_CANCEL_ID, [233](#)
 - FEE_CLUSTERHEAD_OVERHEAD, [233](#)
 - FEE_E_BUSY_INTERNAL, [233](#)
 - FEE_E_BUSY, [233](#)
 - FEE_E_CANCEL_API, [234](#)
 - FEE_E_CLUSTER_GROUP_IDX, [234](#)
 - FEE_E_FOREIGN_BLOCKS_OVF, [234](#)

- FEE_E_INIT_FAILED, [234](#)
- FEE_E_INVALID_BLOCK_LEN, [234](#)
- FEE_E_INVALID_BLOCK_NO, [234](#)
- FEE_E_INVALID_BLOCK_OFS, [235](#)
- FEE_E_INVALID_CANCEL, [235](#)
- FEE_E_PARAM_POINTER, [235](#)
- FEE_E_UNINIT, [235](#)
- FEE_ERASEIMMEDIATEBLOCK_ID, [235](#)
- FEE_FLAG_OVERHEAD, [235](#)
- FEE_GETJOBRESULT_ID, [236](#)
- FEE_GETSTATUS_ID, [236](#)
- FEE_GETVERSIONINFO_ID, [236](#)
- FEE_INIT_ID, [236](#)
- FEE_INSTANCE_ID, [236](#)
- FEE_INVALIDATEBLOCK_ID, [236](#)
- FEE_JOBENDNOTIFICATION_ID, [237](#)
- FEE_JOBERRORNOTIFICATION_ID, [237](#)
- FEE_MAINFUNCTION_ID, [237](#)
- FEE_READ_ID, [237](#)
- FEE_SETMODE_ID, [237](#)
- FEE_WRITE_ID, [237](#)
- Fee_BlockAssignmentType, [237](#)
- Fee_GetJobResult
 - Fee.h, [226](#)
- Fee_GetStatus
 - Fee.h, [227](#)
- Fee_GetVersionInfo
 - Fee.h, [228](#)
- Fee_Init
 - Fee.h, [228](#)
- Fee_InternalTypes.h, [238](#)
 - FEE_BLOCK_INVALID_VALUE, [239](#)
 - Fee_BlockStatusType, [239](#)
 - Fee_ClusterStatusType, [239](#)
 - Fee_JobType, [240](#)
- Fee_InvalidateBlock
 - Fee.h, [228](#)
- Fee_JobEndNotification
 - Fee.h, [229](#)
- Fee_JobErrorNotification
 - Fee.h, [229](#)
- Fee_JobInfoType, [58](#)
 - BlockIdx, [59](#)
 - BufferValid, [59](#)
 - ClusterIdx, [59](#)
 - DataLength, [59](#)
 - DataOffset, [60](#)
 - DataPtr, [60](#)
 - GroupIdx, [60](#)
 - Mode, [60](#)
 - ModuleStatus, [60](#)
 - OldState, [61](#)
 - Result, [61](#)
 - SrcDataPtr, [61](#)
 - State, [61](#)
- Fee_JobPendingInfoType, [62](#)
 - BlockNumber, [62](#)
 - BlockOffset, [62](#)
 - Length, [62](#)
 - Mode, [63](#)
 - PendingJob, [63](#)
 - PendingValid, [63](#)
 - RdDataPtr, [63](#)
 - WrDataPtr, [63](#)
- Fee_JobType
 - Fee_InternalTypes.h, [240](#)
- Fee_Read
 - Fee.h, [230](#)
- Fee_SetMode
 - Fee.h, [230](#)
- Fee_Write
 - Fee.h, [231](#)
- FgndBlockConfig
 - FlsTst_ConfigType, [72](#)
- FgndBlockMaxNumber
 - FlsTst_ConfigType, [73](#)
- FgndLastResult
 - FlsTst_CommonVariableType, [71](#)
- FgndLastSignature
 - FlsTst_CommonVariableType, [71](#)
- FilterNumber
 - Can_HardwareObjectType, [21](#)
- Fls.h, [240](#)
 - Fls_BlankCheck, [241](#)
 - Fls_Cancel, [242](#)
 - Fls_Compare, [242](#)
 - Fls_Erase, [243](#)
 - Fls_GetJobResult, [243](#)
 - Fls_GetStatus, [244](#)
 - Fls_GetVersionInfo, [244](#)
 - Fls_Init, [244](#)
 - Fls_Read, [245](#)
 - Fls_SetMode, [245](#)
 - Fls_Write, [246](#)
- Fls_AcCallbackPtrType
 - Fls_GeneralTypes.h, [252](#)
- Fls_AcErasePtrType
 - Fls_GeneralTypes.h, [253](#)
- Fls_AcWritePtrType
 - Fls_GeneralTypes.h, [253](#)
- Fls_AddressType
 - Fls_GeneralTypes.h, [253](#)
- Fls_BlankCheck
 - Fls.h, [241](#)
- Fls_Cancel
 - Fls.h, [242](#)
- Fls_Compare
 - Fls.h, [242](#)
- Fls_ConfigType, [64](#)
 - DefaultMode, [64](#)
 - EraseTimeout, [65](#)
 - JobEndNotificationPtr, [65](#)
 - JobErrorNotificationPtr, [65](#)
 - MaxReadFastMode, [65](#)
 - MaxReadNormalMode, [65](#)
 - MaxWriteFastMode, [65](#)
 - MaxWriteNormalMode, [66](#)
 - PCallBackPtr, [66](#)
 - PDType, [66](#)
 - PErasePtr, [66](#)
 - PWritePtr, [66](#)
 - SectorCount, [66](#)

- SectorEndAddr, [67](#)
- SectorFlags, [67](#)
- SectorPageSize, [67](#)
- SectorSize, [67](#)
- SectorStartAddr, [67](#)
- WriteTimeout, [67](#)
- Fls_Erase
 - Fls.h, [243](#)
- Fls_GeneralTypes.h, [246](#)
 - FLS_BLANK_CHECK_ID, [247](#)
 - FLS_CANCEL_ID, [248](#)
 - FLS_COMPARE_ID, [248](#)
 - FLS_E_BUSY, [248](#)
 - FLS_E_COMPARE_FAILED, [248](#)
 - FLS_E_ERASE_FAILED, [248](#)
 - FLS_E_PARAM_ADDRESS, [248](#)
 - FLS_E_PARAM_CONFIG, [249](#)
 - FLS_E_PARAM_DATA, [249](#)
 - FLS_E_PARAM_LENGTH, [249](#)
 - FLS_E_PARAM_POINTER, [249](#)
 - FLS_E_READ_FAILED, [249](#)
 - FLS_E_TIMEOUT, [249](#)
 - FLS_E_UNEXPECTED_FLASH_ID, [250](#)
 - FLS_E_UNINIT, [250](#)
 - FLS_E_VERIFY_ERASE_FAILED, [250](#)
 - FLS_E_VERIFY_WRITE_FAILED, [250](#)
 - FLS_E_WRITE_FAILED, [250](#)
 - FLS_ERASE_ID, [250](#)
 - FLS_GETJOBRESULT_ID, [251](#)
 - FLS_GETSTATUS_ID, [251](#)
 - FLS_GETVERSIONINFO_ID, [251](#)
 - FLS_INIT_ID, [251](#)
 - FLS_INSTANCE_ID, [251](#)
 - FLS_MAINFUNCTION_ID, [251](#)
 - FLS_PAGE_WRITE_ASYNC, [252](#)
 - FLS_READ_ID, [252](#)
 - FLS_SECTOR_ERASE_ASYNC, [252](#)
 - FLS_SETMODE_ID, [252](#)
 - FLS_WRITE_ID, [252](#)
 - Fls_AcCallbackPtrType, [252](#)
 - Fls_AcErasePtrType, [253](#)
 - Fls_AcWritePtrType, [253](#)
 - Fls_AddressType, [253](#)
 - Fls_JobEndNotificationPtrType, [253](#)
 - Fls_JobErrorNotificationPtrType, [253](#)
 - Fls_JobType, [253](#)
 - Fls_LengthType, [253](#)
 - Fls_PDType, [254](#)
- Fls_GetJobResult
 - Fls.h, [243](#)
- Fls_GetStatus
 - Fls.h, [244](#)
- Fls_GetVersionInfo
 - Fls.h, [244](#)
- Fls_Init
 - Fls.h, [244](#)
- Fls_JobEndNotificationPtrType
 - Fls_GeneralTypes.h, [253](#)
- Fls_JobErrorNotificationPtrType
 - Fls_GeneralTypes.h, [253](#)
- Fls_JobType
- Fls_GeneralTypes.h, [253](#)
- Fls_LengthType
 - Fls_GeneralTypes.h, [253](#)
- Fls_PDType
 - Fls_GeneralTypes.h, [254](#)
- Fls_Read
 - Fls.h, [245](#)
- Fls_SetMode
 - Fls.h, [245](#)
- Fls_Write
 - Fls.h, [246](#)
- FlsTst.h, [254](#)
 - FLSTST_ABORT_SVCID, [257](#)
 - FLSTST_DEINIT_SVCID, [257](#)
 - FLSTST_E_ALREADY_INITIALIZED, [257](#)
 - FLSTST_E_FLSTST_FAILURE, [257](#)
 - FLSTST_E_INIT_FAILED, [258](#)
 - FLSTST_E_PARAM_INVALID, [258](#)
 - FLSTST_E_PARAM_POINTER, [258](#)
 - FLSTST_E_STATE_FAILURE, [258](#)
 - FLSTST_E_UNINIT, [258](#)
 - FLSTST_GET_CURRENT_STATE_SVCID, [259](#)
 - FLSTST_GET_ERROR_DETAILS_SVCID, [259](#)
 - FLSTST_GET_TEST_RESULT_BGND_SVCID, [259](#)
 - FLSTST_GET_TEST_RESULT_FGND_SVCID, [259](#)
 - FLSTST_GET_TEST_SIGNATURE_BGND_SVCID, [259](#)
 - FLSTST_GET_TEST_SIGNATURE_FGND_SVCID, [260](#)
 - FLSTST_GET_VERSION_INFO_SVCID, [260](#)
 - FLSTST_INIT_SVCID, [260](#)
 - FLSTST_MAIN_FUNCTION_SVCID, [260](#)
 - FLSTST_RESUME_SVCID, [260](#)
 - FLSTST_START_FGND_SVCID, [261](#)
 - FLSTST_SUSPEND_SVCID, [261](#)
 - FLSTST_TEST_COMPLETED_NOTIFICATION_SVCID, [261](#)
 - FLSTST_TEST_ECC_SVCID, [261](#)
 - FlsTst_Abort, [263](#)
 - FlsTst_BlockIdFgndType, [261](#)
 - FlsTst_DeInit, [264](#)
 - FlsTst_GenerateConfigPC, [271](#)
 - FlsTst_GetCurrentState, [264](#)
 - FlsTst_GetErrorDetails, [265](#)
 - FlsTst_GetTestResultBgnd, [265](#)
 - FlsTst_GetTestResultFgnd, [266](#)
 - FlsTst_GetTestSignatureBgnd, [266](#)
 - FlsTst_GetTestSignatureFgnd, [267](#)
 - FlsTst_GetVersionInfo, [267](#)
 - FlsTst_GlobalCommonVar, [271](#)
 - FlsTst_Init, [268](#)
 - FlsTst_MainFunction, [268](#)
 - FlsTst_NotificationType, [262](#)
 - FlsTst_Resume, [269](#)
 - FlsTst_StartFgnd, [269](#)
 - FlsTst_StateType, [262](#)
 - FlsTst_Suspend, [269](#)
 - FlsTst_TestAlgorithmType, [262](#)
 - FlsTst_TestCompletedNotification, [270](#)
 - FlsTst_TestEcc, [270](#)
 - FlsTst_TestResultFgndType, [263](#)

- FlsTst_TestResultType, [263](#)
- FlsTst_Abort
 - FlsTst.h, [263](#)
- FlsTst_BlockConfigType, [68](#)
 - BlockBaseAddress, [68](#)
 - BlockIndex, [68](#)
 - BlockSize, [68](#)
 - NumberOfTestedCells, [69](#)
 - SignatureAddress, [69](#)
 - TestAlgorithm, [69](#)
- FlsTst_BlockIdFgndType
 - FlsTst.h, [261](#)
- FlsTst_CommonVariableType, [69](#)
 - BgndBlockCellLength, [70](#)
 - BgndBlockCellNumber, [70](#)
 - BgndBlockCellSignatureValue, [70](#)
 - BgndBlockCellStartId, [70](#)
 - BgndBlockIndex, [70](#)
 - BgndLastSignature, [70](#)
 - BgndOverallResult, [71](#)
 - BgndTestAbortOrSuspendFlag, [71](#)
 - FgndLastResult, [71](#)
 - FgndLastSignature, [71](#)
 - TestCompletion, [71](#)
 - TestIntervallId, [71](#)
- FlsTst_ConfigType, [72](#)
 - BgndBlockConfig, [72](#)
 - BgndBlockMaxNumber, [72](#)
 - FgndBlockConfig, [72](#)
 - FgndBlockMaxNumber, [73](#)
 - TestCompletedNotification, [73](#)
- FlsTst_Delnit
 - FlsTst.h, [264](#)
- FlsTst_ErrorDetailsType, [73](#)
 - EccFaultAddress, [73](#)
 - EccStatus, [74](#)
- FlsTst_GenerateConfigPC
 - FlsTst.h, [271](#)
- FlsTst_GetCurrentState
 - FlsTst.h, [264](#)
- FlsTst_GetErrorDetails
 - FlsTst.h, [265](#)
- FlsTst_GetTestResultBgnd
 - FlsTst.h, [265](#)
- FlsTst_GetTestResultFgnd
 - FlsTst.h, [266](#)
- FlsTst_GetTestSignatureBgnd
 - FlsTst.h, [266](#)
- FlsTst_GetTestSignatureFgnd
 - FlsTst.h, [267](#)
- FlsTst_GetVersionInfo
 - FlsTst.h, [267](#)
- FlsTst_GlobalCommonVar
 - FlsTst.h, [271](#)
- FlsTst_Init
 - FlsTst.h, [268](#)
- FlsTst_MainFunction
 - FlsTst.h, [268](#)
- FlsTst_NotificationType
 - FlsTst.h, [262](#)
- FlsTst_Resume
 - FlsTst.h, [269](#)
- FlsTst_StartFgnd
 - FlsTst.h, [269](#)
- FlsTst_StateType
 - FlsTst.h, [262](#)
- FlsTst_Suspend
 - FlsTst.h, [269](#)
- FlsTst_TestAlgorithmType
 - FlsTst.h, [262](#)
- FlsTst_TestCompletedNotification
 - FlsTst.h, [270](#)
- FlsTst_TestEcc
 - FlsTst.h, [270](#)
- FlsTst_TestResultBgndType, [74](#)
 - TestIntervallId, [74](#)
 - TestResultBgnd, [74](#)
- FlsTst_TestResultFgndType
 - FlsTst.h, [263](#)
- FlsTst_TestResultType
 - FlsTst.h, [263](#)
- FlsTst_TestSignatureBgndType, [75](#)
 - TestIntervallId, [75](#)
 - TestSignatureValue, [75](#)
- FlsTst_TestSignatureFgndType, [76](#)
 - TestSignatureValue, [76](#)
- GPIO_FUN0
 - Port_GeneralTypes.h, [357](#)
- GPIO_FUN1
 - Port_GeneralTypes.h, [357](#)
- GPIO_FUN2
 - Port_GeneralTypes.h, [357](#)
- GPIO_FUN3
 - Port_GeneralTypes.h, [357](#)
- GPIO_FUN4
 - Port_GeneralTypes.h, [357](#)
- GPIO_FUN5
 - Port_GeneralTypes.h, [358](#)
- GPIO_FUN6
 - Port_GeneralTypes.h, [358](#)
- GPIO_FUN7
 - Port_GeneralTypes.h, [358](#)
- GPIO_Mode
 - Port_PinConfigType, [96](#)
- GPT_CHECK_WAKEUP_ID
 - Gpt.h, [273](#)
- GPT_DEINIT_ID
 - Gpt.h, [273](#)
- GPT_DISABLE_NOTIFICATION_ID
 - Gpt.h, [273](#)
- GPT_DISABLE_WAKEUP_ID
 - Gpt.h, [273](#)
- GPT_E_ALREADY_INITIALIZED
 - Gpt.h, [273](#)
- GPT_E_BUSY
 - Gpt.h, [273](#)
- GPT_E_INIT_FAILED
 - Gpt.h, [273](#)
- GPT_E_MODE
 - Gpt.h, [274](#)
- GPT_E_PARAM_CHANNEL

- Gpt.h, [274](#)
- GPT_E_PARAM_MODE
 - Gpt.h, [274](#)
- GPT_E_PARAM_POINTER
 - Gpt.h, [274](#)
- GPT_E_PARAM_PREDEF_TIMER
 - Gpt.h, [274](#)
- GPT_E_PARAM_VALUE
 - Gpt.h, [274](#)
- GPT_E_UNINIT
 - Gpt.h, [275](#)
- GPT_ENABLE_NOTIFICATION_ID
 - Gpt.h, [275](#)
- GPT_ENABLE_WAKEUP_ID
 - Gpt.h, [275](#)
- GPT_GET_PREDEF_TIMER_VALUE_ID
 - Gpt.h, [275](#)
- GPT_GET_TIME_ELAPSED_ID
 - Gpt.h, [275](#)
- GPT_GET_TIME_REMAINING_ID
 - Gpt.h, [275](#)
- GPT_GET_VERSION_INFO_ID
 - Gpt.h, [276](#)
- GPT_INIT_ID
 - Gpt.h, [276](#)
- GPT_INSTANCE_ID
 - Gpt.h, [276](#)
- GPT_MODULE_ID
 - Gpt.h, [276](#)
- GPT_SET_MODE_ID
 - Gpt.h, [276](#)
- GPT_START_TIMER_ID
 - Gpt.h, [276](#)
- GPT_STOP_TIMER_ID
 - Gpt.h, [277](#)
- Gpt.h, [271](#)
 - GPT_CHECK_WAKEUP_ID, [273](#)
 - GPT_DEINIT_ID, [273](#)
 - GPT_DISABLE_NOTIFICATION_ID, [273](#)
 - GPT_DISABLE_WAKEUP_ID, [273](#)
 - GPT_E_ALREADY_INITIALIZED, [273](#)
 - GPT_E_BUSY, [273](#)
 - GPT_E_INIT_FAILED, [273](#)
 - GPT_E_MODE, [274](#)
 - GPT_E_PARAM_CHANNEL, [274](#)
 - GPT_E_PARAM_MODE, [274](#)
 - GPT_E_PARAM_POINTER, [274](#)
 - GPT_E_PARAM_PREDEF_TIMER, [274](#)
 - GPT_E_PARAM_VALUE, [274](#)
 - GPT_E_UNINIT, [275](#)
 - GPT_ENABLE_NOTIFICATION_ID, [275](#)
 - GPT_ENABLE_WAKEUP_ID, [275](#)
 - GPT_GET_PREDEF_TIMER_VALUE_ID, [275](#)
 - GPT_GET_TIME_ELAPSED_ID, [275](#)
 - GPT_GET_TIME_REMAINING_ID, [275](#)
 - GPT_GET_VERSION_INFO_ID, [276](#)
 - GPT_INIT_ID, [276](#)
 - GPT_INSTANCE_ID, [276](#)
 - GPT_MODULE_ID, [276](#)
 - GPT_SET_MODE_ID, [276](#)
 - GPT_START_TIMER_ID, [276](#)
 - GPT_STOP_TIMER_ID, [277](#)
 - Gpt_CheckWakeup, [277](#)
 - Gpt_DeInit, [277](#)
 - Gpt_DisableNotification, [278](#)
 - Gpt_DisableWakeup, [278](#)
 - Gpt_EnableNotification, [278](#)
 - Gpt_EnableWakeup, [279](#)
 - Gpt_GenerateConfigPC, [282](#)
 - Gpt_GetPredefTimerValue, [279](#)
 - Gpt_GetTimeElapsed, [279](#)
 - Gpt_GetTimeRemaining, [280](#)
 - Gpt_GetVersionInfo, [280](#)
 - Gpt_Init, [280](#)
 - Gpt_SetMode, [281](#)
 - Gpt_StartTimer, [281](#)
 - Gpt_StopTimer, [282](#)
 - Gpt_CheckWakeup
 - Gpt.h, [277](#)
 - Gpt_DeInit
 - Gpt.h, [277](#)
 - Gpt_DisableNotification
 - Gpt.h, [278](#)
 - Gpt_DisableWakeup
 - Gpt.h, [278](#)
 - Gpt_EnableNotification
 - Gpt.h, [278](#)
 - Gpt_EnableWakeup
 - Gpt.h, [279](#)
 - Gpt_GenerateConfigPC
 - Gpt.h, [282](#)
 - Gpt_GetPredefTimerValue
 - Gpt.h, [279](#)
 - Gpt_GetTimeElapsed
 - Gpt.h, [279](#)
 - Gpt_GetTimeRemaining
 - Gpt.h, [280](#)
 - Gpt_GetVersionInfo
 - Gpt.h, [280](#)
 - Gpt_Init
 - Gpt.h, [280](#)
 - Gpt_SetMode
 - Gpt.h, [281](#)
 - Gpt_StartTimer
 - Gpt.h, [281](#)
 - Gpt_StopTimer
 - Gpt.h, [282](#)
 - GroupIdx
 - Fee_JobInfoType, [60](#)
 - GrpCount
 - Fee_ConfigType, [58](#)
 - HeadAddr
 - Fee_BlockInfoType, [50](#)
 - HeadOfFree
 - Crypto_ObjectType, [39](#)
 - HeadOfJobs
 - Crypto_ObjectType, [39](#)
 - Header
 - Fee_BlockInfoType, [50](#)
 - Fee_ClusterInfoType, [56](#)
 - Hoh

- Can_HwType, [22](#)
- HohNumber
 - Can_ConfigType, [12](#)
- HwChannel
 - I2c_CHConfigType, [77](#)
- HwConfigPtr
 - Acmp_ConfigType, [9](#)
 - Sent_ConfigType, [105](#)
 - Uart_ConfigType, [106](#)
- HwSelect
 - Icu_ChannelConfigType, [79](#)
- I2C_ASYNCTRANSMIT_ID
 - CDD_I2c.h, [161](#)
- I2C_DEINIT_ID
 - CDD_I2c.h, [161](#)
- I2C_E_ALREADY_INIT
 - CDD_I2c.h, [161](#)
- I2C_E_BUSY
 - CDD_I2c.h, [162](#)
- I2C_E_INVALID_CHANNEL
 - CDD_I2c.h, [162](#)
- I2C_E_INVALID_POINTER
 - CDD_I2c.h, [162](#)
- I2C_E_STATUES
 - CDD_I2c.h, [162](#)
- I2C_E_UNINIT
 - CDD_I2c.h, [162](#)
- I2C_GETSTATUS_ID
 - CDD_I2c.h, [162](#)
- I2C_GETVERSION_INFO_ID
 - CDD_I2c.h, [163](#)
- I2C_INIT_ID
 - CDD_I2c.h, [163](#)
- I2C_INIT_STATE
 - CDD_I2c.h, [163](#)
- I2C_UNINIT_STATE
 - CDD_I2c.h, [163](#)
- I2c_AbortTransmit
 - CDD_I2c.h, [164](#)
- I2c_AsyncTransmit
 - CDD_I2c.h, [164](#)
- I2c_CHConfigType, [76](#)
 - ChConfigPtr, [77](#)
 - HwChannel, [77](#)
- I2c_ConfigType, [77](#)
 - ChannelConfigPtr, [77](#)
 - MaxChannelNum, [77](#)
- I2c_DelNit
 - CDD_I2c.h, [165](#)
- I2c_GenerateConfigPC
 - CDD_I2c.h, [168](#)
- I2c_GetBaudRate
 - CDD_I2c.h, [165](#)
- I2c_GetStatus
 - CDD_I2c.h, [166](#)
- I2c_GetVersionInfo
 - CDD_I2c.h, [166](#)
- I2c_Init
 - CDD_I2c.h, [167](#)
- I2c_SetBaudRate
 - CDD_I2c.h, [167](#)
- I2c_StatusType
 - CDD_I2c.h, [163](#)
- I2c_SyncTransmit
 - CDD_I2c.h, [167](#)
- ICU_CHECKWAKEUP_ID
 - Icu.h, [286](#)
- ICU_DEINIT_ID
 - Icu.h, [286](#)
- ICU_DISABLEEDGECOUNT_ID
 - Icu.h, [286](#)
- ICU_DISABLEEDGEDETECTION_ID
 - Icu.h, [286](#)
- ICU_DISABLENOTIFICATION_ID
 - Icu.h, [286](#)
- ICU_DISABLEWAKEUP_ID
 - Icu.h, [287](#)
- ICU_E_ALREADY_INITIALIZED
 - Icu.h, [287](#)
- ICU_E_INIT_FAILED
 - Icu.h, [287](#)
- ICU_E_NOT_STARTED
 - Icu.h, [287](#)
- ICU_E_PARAM_ACTIVATION
 - Icu.h, [287](#)
- ICU_E_PARAM_BUFFER_SIZE
 - Icu.h, [288](#)
- ICU_E_PARAM_CHANNEL
 - Icu.h, [288](#)
- ICU_E_PARAM_MODE
 - Icu.h, [288](#)
- ICU_E_PARAM_NOTIFY_INTERVAL
 - Icu.h, [288](#)
- ICU_E_PARAM_POINTER
 - Icu.h, [288](#)
- ICU_E_PARAM_VINFO
 - Icu.h, [289](#)
- ICU_E_UNINIT
 - Icu.h, [289](#)
- ICU_ENABLEEDGECOUNT_ID
 - Icu.h, [289](#)
- ICU_ENABLEEDGEDETECTION_ID
 - Icu.h, [289](#)
- ICU_ENABLENOTIFICATION_ID
 - Icu.h, [289](#)
- ICU_ENABLEWAKEUP_ID
 - Icu.h, [290](#)
- ICU_GETDUTYCYCLEVALUES_ID
 - Icu.h, [290](#)
- ICU_GETEDGENUMBERS_ID
 - Icu.h, [290](#)
- ICU_GETINPUTSTATE_ID
 - Icu.h, [290](#)
- ICU_GETTIMEELAPSED_ID
 - Icu.h, [290](#)
- ICU_GETTIMESTAMPINDEX_ID
 - Icu.h, [291](#)
- ICU_GETVERSIONINFO_ID
 - Icu.h, [291](#)
- ICU_INIT_ID
 - Icu.h, [291](#)

ICU_RESETEDEGECOUNT_ID
 Icu.h, 291
 ICU_SETACTIVATIONCONDITION_ID
 Icu.h, 291
 ICU_SETMODE_ID
 Icu.h, 292
 ICU_STARTSIGNALMEASUREMENT_ID
 Icu.h, 292
 ICU_STARTTIMESTAMP_ID
 Icu.h, 292
 ICU_STOPSIGNALMEASUREMENT_ID
 Icu.h, 292
 ICU_STOPTIMESTAMP_ID
 Icu.h, 292
 ISR
 Oslf_Irq.h, 348
 Icu.h, 282
 ICU_CHECKWAKEUP_ID, 286
 ICU_DEINIT_ID, 286
 ICU_DISABLEEDGECOUNT_ID, 286
 ICU_DISABLEEDGEDETECTION_ID, 286
 ICU_DISABLENOTIFICATION_ID, 286
 ICU_DISABLEWAKEUP_ID, 287
 ICU_E_ALREADY_INITIALIZED, 287
 ICU_E_INIT_FAILED, 287
 ICU_E_NOT_STARTED, 287
 ICU_E_PARAM_ACTIVATION, 287
 ICU_E_PARAM_BUFFER_SIZE, 288
 ICU_E_PARAM_CHANNEL, 288
 ICU_E_PARAM_MODE, 288
 ICU_E_PARAM_NOTIFY_INTERVAL, 288
 ICU_E_PARAM_POINTER, 288
 ICU_E_PARAM_VINFO, 289
 ICU_E_UNINIT, 289
 ICU_ENABLEEDGECOUNT_ID, 289
 ICU_ENABLEEDGEDETECTION_ID, 289
 ICU_ENABLENOTIFICATION_ID, 289
 ICU_ENABLEWAKEUP_ID, 290
 ICU_GETDUTYCYCLEVALUES_ID, 290
 ICU_GETEDGENUMBERS_ID, 290
 ICU_GETINPUTSTATE_ID, 290
 ICU_GETTIMEELAPSED_ID, 290
 ICU_GETTIMESTAMPINDEX_ID, 291
 ICU_GETVERSIONINFO_ID, 291
 ICU_INIT_ID, 291
 ICU_RESETEDEGECOUNT_ID, 291
 ICU_SETACTIVATIONCONDITION_ID, 291
 ICU_SETMODE_ID, 292
 ICU_STARTSIGNALMEASUREMENT_ID, 292
 ICU_STARTTIMESTAMP_ID, 292
 ICU_STOPSIGNALMEASUREMENT_ID, 292
 ICU_STOPTIMESTAMP_ID, 292
 Icu_ActivationType, 293
 Icu_ChannelType, 293
 Icu_CheckWakeup, 296
 Icu_DeInit, 296
 Icu_DisableEdgeCount, 296
 Icu_DisableEdgeDetection, 297
 Icu_DisableNotification, 297
 Icu_DisableWakeup, 297
 Icu_EdgeNumberType, 293
 Icu_EnableEdgeCount, 298
 Icu_EnableEdgeDetection, 298
 Icu_EnableNotification, 298
 Icu_EnableWakeup, 299
 Icu_GenerateConfigPC, 304
 Icu_GetDutyCycleValues, 299
 Icu_GetEdgeNumbers, 299
 Icu_GetInputState, 300
 Icu_GetTimeElapsed, 300
 Icu_GetTimestampIndex, 301
 Icu_GetVersionInfo, 301
 Icu_HwSelect, 294
 Icu_IndexType, 293
 Icu_Init, 301
 Icu_InputStateType, 294
 Icu_MeasurementModeType, 294
 Icu_ModeType, 295
 Icu_ResetEdgeCount, 302
 Icu_SetActivationCondition, 302
 Icu_SetMode, 302
 Icu_SignalMeasurementPropertyType, 295
 Icu_StartSignalMeasurement, 303
 Icu_StartTimestamp, 303
 Icu_StopSignalMeasurement, 303
 Icu_StopTimestamp, 304
 Icu_TimestampBufferType, 295
 Icu_ValueType, 293
 Icu_ActivationType
 Icu.h, 293
 Icu_ChannelConfigType, 78
 Channel, 78
 ChannelNotification, 78
 DefaultStartEdge, 79
 HwSelect, 79
 LogicId, 79
 MeasurementMode, 79
 ModuleId, 79
 SignalMeasurementPropertyType, 80
 TimestampBufferType, 80
 WakeupCapable, 80
 WakeupSource, 80
 Icu_ChannelType
 Icu.h, 293
 Icu_CheckWakeup
 Icu.h, 296
 Icu_ConfigType, 81
 ChannelConfigPtr, 81
 IpConfigPtr, 81
 Icu_DeInit
 Icu.h, 296
 Icu_DisableEdgeCount
 Icu.h, 296
 Icu_DisableEdgeDetection
 Icu.h, 297
 Icu_DisableNotification
 Icu.h, 297
 Icu_DisableWakeup
 Icu.h, 297
 Icu_DutyCycleType, 81
 ActiveTime, 82
 PeriodTime, 82

- Icu_EdgeNumberType
 - Icu.h, [293](#)
- Icu_EnableEdgeCount
 - Icu.h, [298](#)
- Icu_EnableEdgeDetection
 - Icu.h, [298](#)
- Icu_EnableNotification
 - Icu.h, [298](#)
- Icu_EnableWakeUp
 - Icu.h, [299](#)
- Icu_GenerateConfigPC
 - Icu.h, [304](#)
- Icu_GetDutyCycleValues
 - Icu.h, [299](#)
- Icu_GetEdgeNumbers
 - Icu.h, [299](#)
- Icu_GetInputState
 - Icu.h, [300](#)
- Icu_GetTimeElapsed
 - Icu.h, [300](#)
- Icu_GetTimestampIndex
 - Icu.h, [301](#)
- Icu_GetVersionInfo
 - Icu.h, [301](#)
- Icu_HwSelect
 - Icu.h, [294](#)
- Icu_IndexType
 - Icu.h, [293](#)
- Icu_Init
 - Icu.h, [301](#)
- Icu_InputStateType
 - Icu.h, [294](#)
- Icu_IpConfigType, [82](#)
 - PwmModuleConfigPtr, [83](#)
 - PwmModuleCount, [83](#)
- Icu_MeasurementModeType
 - Icu.h, [294](#)
- Icu_ModeType
 - Icu.h, [295](#)
- Icu_PwmModuleConfigType, [83](#)
 - ClockSource, [84](#)
 - Instance, [84](#)
 - Prescaler, [84](#)
- Icu_ResetEdgeCount
 - Icu.h, [302](#)
- Icu_SetActivationCondition
 - Icu.h, [302](#)
- Icu_SetMode
 - Icu.h, [302](#)
- Icu_SignalMeasurementPropertyType
 - Icu.h, [295](#)
- Icu_StartSignalMeasurement
 - Icu.h, [303](#)
- Icu_StartTimestamp
 - Icu.h, [303](#)
- Icu_StopSignalMeasurement
 - Icu.h, [303](#)
- Icu_StopTimestamp
 - Icu.h, [304](#)
- Icu_TimestampBufferType
 - Icu.h, [295](#)
- Icu_ValueType
 - Icu.h, [293](#)
- Id
 - CryptoKeyElement, [44](#)
- id
 - Can_PduType, [23](#)
- IdType
 - Can_FilterControlType, [18](#)
- IdleState
 - Pwm_ChannelConfigType, [100](#)
- ImmediateBlock
 - Fee_BlockHeaderType, [48](#)
- ImmediateData
 - Fee_BlockType, [52](#)
- input64
 - Crypto_JobPrimitiveInputOutputType, [29](#)
- inputKeyElementId
 - Crypto_JobRedirectionInfoType, [33](#)
- inputKeyId
 - Crypto_JobRedirectionInfoType, [33](#)
- inputLength
 - Crypto_JobPrimitiveInputOutputType, [29](#)
- inputPtr
 - Crypto_JobPrimitiveInputOutputType, [30](#)
- Instance
 - Icu_PwmModuleConfigType, [84](#)
 - Pwm_ModuleConfigType, [104](#)
- Invalid
 - Fee_BlockHeaderType, [49](#)
 - Fee_ClusterHeaderType, [55](#)
- IpConfig
 - Ocu_ConfigType, [89](#)
- IpConfigPtr
 - Icu_ConfigType, [81](#)
 - Pwm_ConfigType, [101](#)
- JobEndNotificationPtr
 - Fls_ConfigType, [65](#)
- JobErrorNotificationPtr
 - Fls_ConfigType, [65](#)
- JobId
 - Crypto_JobInfoType, [26](#)
- jobId
 - Crypto_JobType, [36](#)
- jobInfo
 - Crypto_JobType, [36](#)
- jobPrimitiveInfo
 - Crypto_JobType, [36](#)
- jobPrimitiveInputOutput
 - Crypto_JobType, [36](#)
- JobPriority
 - Crypto_JobInfoType, [26](#)
- JobPtr
 - _Crypto_QueueType, [8](#)
- jobRedirectionInfoRef
 - Crypto_JobType, [37](#)
- jobState
 - Crypto_JobType, [37](#)
- KeyFormatType
 - CryptoKeyElement, [44](#)

- KeyId
 - Crypto_Key, [38](#)
- keyLength
 - Crypto_AlgorithmInfoType, [25](#)
- KeyTypePtr
 - Crypto_Key, [38](#)
- KeyValid
 - Crypto_Key, [38](#)
- LIN_CH_OPERATIONAL
 - Lin.h, [306](#)
- LIN_CH_SLEEP_PENDING
 - Lin.h, [306](#)
- LIN_CH_SLEEP_STATE
 - Lin.h, [306](#)
- LIN_CHECK_WAKEUP_ID
 - Lin.h, [306](#)
- LIN_E_INVALID_CHANNEL
 - Lin.h, [306](#)
- LIN_E_INVALID_POINTER
 - Lin.h, [306](#)
- LIN_E_PARAM_POINTER
 - Lin.h, [307](#)
- LIN_E_STATE_TRANSITION
 - Lin.h, [307](#)
- LIN_E_TIMEOUT
 - Lin.h, [307](#)
- LIN_E_UNINIT
 - Lin.h, [307](#)
- LIN_GETSTATUS_ID
 - Lin.h, [307](#)
- LIN_GETVERSIONINFO_ID
 - Lin.h, [307](#)
- LIN_GOTOSLEEP_ID
 - Lin.h, [308](#)
- LIN_GOTOSLEEPINTERNAL_ID
 - Lin.h, [308](#)
- LIN_INIT_ID
 - Lin.h, [308](#)
- LIN_INSTANCE
 - Lin.h, [308](#)
- LIN_SENDFRAME_ID
 - Lin.h, [308](#)
- LIN_STATE_INIT
 - Lin.h, [308](#)
- LIN_STATE_UNINIT
 - Lin.h, [309](#)
- LIN_WAKEUP_ID
 - Lin.h, [309](#)
- LIN_WAKEUPINTERNAL_ID
 - Lin.h, [309](#)
- Length
 - Fee_BlockHeaderType, [49](#)
 - Fee_ClusterHeaderType, [55](#)
 - Fee_ClusterType, [57](#)
 - Fee_JobPendingInfoType, [62](#)
- length
 - Can_PduType, [23](#)
- Lin.h, [304](#)
 - LIN_CH_OPERATIONAL, [306](#)
 - LIN_CH_SLEEP_PENDING, [306](#)
 - LIN_CH_SLEEP_STATE, [306](#)
 - LIN_CHECK_WAKEUP_ID, [306](#)
 - LIN_E_INVALID_CHANNEL, [306](#)
 - LIN_E_INVALID_POINTER, [306](#)
 - LIN_E_PARAM_POINTER, [307](#)
 - LIN_E_STATE_TRANSITION, [307](#)
 - LIN_E_TIMEOUT, [307](#)
 - LIN_E_UNINIT, [307](#)
 - LIN_GETSTATUS_ID, [307](#)
 - LIN_GETVERSIONINFO_ID, [307](#)
 - LIN_GOTOSLEEP_ID, [308](#)
 - LIN_GOTOSLEEPINTERNAL_ID, [308](#)
 - LIN_INIT_ID, [308](#)
 - LIN_INSTANCE, [308](#)
 - LIN_SENDFRAME_ID, [308](#)
 - LIN_STATE_INIT, [308](#)
 - LIN_STATE_UNINIT, [309](#)
 - LIN_WAKEUP_ID, [309](#)
 - LIN_WAKEUPINTERNAL_ID, [309](#)
 - Lin_CheckWakeup, [309](#)
 - Lin_GenerateConfigPC, [314](#)
 - Lin_GetStatus, [310](#)
 - Lin_GetVersionInfo, [310](#)
 - Lin_GoToSleep, [311](#)
 - Lin_GoToSleepInternal, [311](#)
 - Lin_Init, [312](#)
 - Lin_SendFrame, [312](#)
 - Lin_Wakeup, [313](#)
 - Lin_WakeupInternal, [313](#)
 - Lin_CheckWakeup
 - Lin.h, [309](#)
 - Lin_ConfigType, [84](#)
 - Lin_HwConfigPtr, [85](#)
 - Lin_FrameDType
 - Lin_GeneralTypes.h, [314](#)
 - Lin_FramePidType
 - Lin_GeneralTypes.h, [314](#)
 - Lin_GeneralTypes.h, [314](#)
 - Lin_FrameDType, [314](#)
 - Lin_FramePidType, [314](#)
 - Lin_SlaveErrorType, [314](#)
 - Lin_GenerateConfigPC
 - Lin.h, [314](#)
 - Lin_GetStatus
 - Lin.h, [310](#)
 - Lin_GetVersionInfo
 - Lin.h, [310](#)
 - Lin_GoToSleep
 - Lin.h, [311](#)
 - Lin_GoToSleepInternal
 - Lin.h, [311](#)
 - Lin_HwConfigPtr
 - Lin_ConfigType, [85](#)
 - Lin_Init
 - Lin.h, [312](#)
 - Lin_SendFrame
 - Lin.h, [312](#)
 - Lin_SlaveErrorType
 - Lin_GeneralTypes.h, [314](#)
 - Lin_Wakeup
 - Lin.h, [313](#)

- Lin_WakeupInternal
 - Lin.h, [313](#)
- LogicId
 - Icu_ChannelConfigType, [79](#)
 - Ocu_ChannelConfigType, [88](#)
 - Pwm_ChannelConfigType, [100](#)
- MCL_DEINIT_ID
 - CDD_Mcl.h, [169](#)
- MCL_E_PARAM_POINTER
 - CDD_Mcl.h, [169](#)
- MCL_GET_VERSION_INFO_ID
 - CDD_Mcl.h, [169](#)
- MCL_INIT_ID
 - CDD_Mcl.h, [169](#)
- MCL_INSTANCE_ID
 - CDD_Mcl.h, [170](#)
- MCU_DISTRIBUTEPLLCLOCK_ID
 - Mcu.h, [316](#)
- MCU_E_ALREADY_INITIALIZED
 - Mcu.h, [316](#)
- MCU_E_CMU_INDEX_OUT_OF_RANGE
 - Mcu.h, [317](#)
- MCU_E_INIT_FAILED
 - Mcu.h, [317](#)
- MCU_E_PARAM_CLOCK
 - Mcu.h, [317](#)
- MCU_E_PARAM_CONFIG
 - Mcu.h, [317](#)
- MCU_E_PARAM_MODE
 - Mcu.h, [317](#)
- MCU_E_PARAM_POINTER
 - Mcu.h, [317](#)
- MCU_E_PARAM_RAMSECTION
 - Mcu.h, [318](#)
- MCU_E_PLL_NOT_LOCKED
 - Mcu.h, [318](#)
- MCU_E_UNINIT
 - Mcu.h, [318](#)
- MCU_GETPLLSTATUS_ID
 - Mcu.h, [318](#)
- MCU_GETRAMSTATE_ID
 - Mcu.h, [318](#)
- MCU_GETRESETRAWVALUE_ID
 - Mcu.h, [318](#)
- MCU_GETRESETREASON_ID
 - Mcu.h, [319](#)
- MCU_GETVERSIONINFO_ID
 - Mcu.h, [319](#)
- MCU_INIT_ID
 - Mcu.h, [319](#)
- MCU_INITCLOCK_ID
 - Mcu.h, [319](#)
- MCU_INITRAMSECTION_ID
 - Mcu.h, [319](#)
- MCU_INSTANCE_ID
 - Mcu.h, [319](#)
- MCU_PERFORMRESET_ID
 - Mcu.h, [320](#)
- MCU_SETMODE_ID
 - Mcu.h, [320](#)
- Mask
 - Can_FilterControlType, [18](#)
 - Dio_ChannelGroupType, [46](#)
- MaxBaudRateCount
 - Can_ControllerDescriptorType, [15](#)
- MaxChannelNum
 - Acmp_ConfigType, [9](#)
 - I2c_ConfigType, [77](#)
 - Sent_ConfigType, [105](#)
 - Uart_ConfigType, [106](#)
- MaxReadFastMode
 - FIs_ConfigType, [65](#)
- MaxReadNormalMode
 - FIs_ConfigType, [65](#)
- MaxSize
 - CryptoKeyElement, [44](#)
- MaxWriteFastMode
 - FIs_ConfigType, [65](#)
- MaxWriteNormalMode
 - FIs_ConfigType, [66](#)
- Mcl_DeInit
 - CDD_Mcl.h, [170](#)
- Mcl_GetVersionInfo
 - CDD_Mcl.h, [170](#)
- Mcl_Init
 - CDD_Mcl.h, [171](#)
- Mcu.h, [315](#)
 - MCU_DISTRIBUTEPLLCLOCK_ID, [316](#)
 - MCU_E_ALREADY_INITIALIZED, [316](#)
 - MCU_E_CMU_INDEX_OUT_OF_RANGE, [317](#)
 - MCU_E_INIT_FAILED, [317](#)
 - MCU_E_PARAM_CLOCK, [317](#)
 - MCU_E_PARAM_CONFIG, [317](#)
 - MCU_E_PARAM_MODE, [317](#)
 - MCU_E_PARAM_POINTER, [317](#)
 - MCU_E_PARAM_RAMSECTION, [318](#)
 - MCU_E_PLL_NOT_LOCKED, [318](#)
 - MCU_E_UNINIT, [318](#)
 - MCU_GETPLLSTATUS_ID, [318](#)
 - MCU_GETRAMSTATE_ID, [318](#)
 - MCU_GETRESETRAWVALUE_ID, [318](#)
 - MCU_GETRESETREASON_ID, [319](#)
 - MCU_GETVERSIONINFO_ID, [319](#)
 - MCU_INIT_ID, [319](#)
 - MCU_INITCLOCK_ID, [319](#)
 - MCU_INITRAMSECTION_ID, [319](#)
 - MCU_INSTANCE_ID, [319](#)
 - MCU_PERFORMRESET_ID, [320](#)
 - MCU_SETMODE_ID, [320](#)
 - Mcu_DistributePllClock, [320](#)
 - Mcu_GenerateConfigPC, [325](#)
 - Mcu_GetPllStatus, [320](#)
 - Mcu_GetRamState, [321](#)
 - Mcu_GetResetRawValue, [321](#)
 - Mcu_GetResetReason, [322](#)
 - Mcu_GetVersionInfo, [322](#)
 - Mcu_Init, [323](#)
 - Mcu_InitClock, [323](#)
 - Mcu_InitRamSection, [324](#)
 - Mcu_PerformReset, [324](#)
 - Mcu_SetMode, [325](#)

- Mcu_ClkConfigNum
 - Mcu_ConfigType, 85
- Mcu_ClockConfigPtr
 - Mcu_ConfigType, 85
- Mcu_CommonConfigPtr
 - Mcu_ConfigType, 86
- Mcu_ConfigType, 85
 - Mcu_ClkConfigNum, 85
 - Mcu_ClockConfigPtr, 85
 - Mcu_CommonConfigPtr, 86
 - Mcu_DemConfigPtr, 86
 - Mcu_ModeConfigNum, 86
 - Mcu_ModeConfigPtr, 86
 - Mcu_RamConfigNum, 86
 - Mcu_RamConfigPtr, 86
- Mcu_DemConfigPtr
 - Mcu_ConfigType, 86
- Mcu_DistributePllClock
 - Mcu.h, 320
- Mcu_GenerateConfigPC
 - Mcu.h, 325
- Mcu_GetPllStatus
 - Mcu.h, 320
- Mcu_GetRamState
 - Mcu.h, 321
- Mcu_GetResetRawValue
 - Mcu.h, 321
- Mcu_GetResetReason
 - Mcu.h, 322
- Mcu_GetVersionInfo
 - Mcu.h, 322
- Mcu_Init
 - Mcu.h, 323
- Mcu_InitClock
 - Mcu.h, 323
- Mcu_InitRamSection
 - Mcu.h, 324
- Mcu_ModeConfigNum
 - Mcu_ConfigType, 86
- Mcu_ModeConfigPtr
 - Mcu_ConfigType, 86
- Mcu_PerformReset
 - Mcu.h, 324
- Mcu_RamConfigNum
 - Mcu_ConfigType, 86
- Mcu_RamConfigPtr
 - Mcu_ConfigType, 86
- Mcu_SetMode
 - Mcu.h, 325
- MeasurementMode
 - Icu_ChannelConfigType, 79
- MemEccEn
 - Can_ControllerDescriptorType, 15
- Mode
 - Crypto_PrimitiveType, 42
 - Fee_JobInfoType, 60
 - Fee_JobPendingInfoType, 63
- mode
 - Crypto_AlgorithmInfoType, 25
 - Crypto_JobPrimitiveInputOutputType, 30
- ModeChangeable
 - Port_PinConfigType, 96
- ModeSetting
 - Wdg_ConfigType, 107
- ModuleCfg
 - Pwm_IpConfigType, 102
- ModuleConfig
 - Sent_ConfigType, 106
- ModuleId
 - Icu_ChannelConfigType, 79
 - Ocu_ModuleConfigType, 91
- ModuleMaxValue
 - Ocu_ModuleConfigType, 91
- ModuleStatus
 - Fee_JobInfoType, 60
- Next
 - _Crypto_QueueType, 8
- NumChannelGroups
 - Dio_ConfigType, 47
- NumDigitalFilterPorts
 - Port_ConfigType, 92
- NumPins
 - Port_ConfigType, 92
- NumUnusedPins
 - Port_ConfigType, 92
- NumberOfTestedCells
 - FlsTst_BlockConfigType, 69
- OCU_DEINIT_ID
 - Ocu.h, 327
- OCU_DISABLENOTIFICATION_ID
 - Ocu.h, 327
- OCU_E_ALREADY_INITIALIZED
 - Ocu.h, 328
- OCU_E_BUSY
 - Ocu.h, 328
- OCU_E_INIT_FAILED
 - Ocu.h, 328
- OCU_E_NO_VALID_NOTIF
 - Ocu.h, 328
- OCU_E_PARAM_INVALID_ACTION
 - Ocu.h, 328
- OCU_E_PARAM_INVALID_CHANNEL
 - Ocu.h, 328
- OCU_E_PARAM_INVALID_STATE
 - Ocu.h, 329
- OCU_E_PARAM_INVALID_VALUE
 - Ocu.h, 329
- OCU_E_PARAM_NO_PIN
 - Ocu.h, 329
- OCU_E_PARAM_POINTER
 - Ocu.h, 329
- OCU_E_UNINIT
 - Ocu.h, 329
- OCU_ENABLENOTIFICATION_ID
 - Ocu.h, 329
- OCU_GETCOUNTER_ID
 - Ocu.h, 330
- OCU_GETVERSIONINFO_ID
 - Ocu.h, 330
- OCU_INIT_ID

Ocu.h, 330
 OCU_SETABSOLUTETHRESHOLD_ID
 Ocu.h, 330
 OCU_SETPINACTION_ID
 Ocu.h, 330
 OCU_SETPINSTATE_ID
 Ocu.h, 330
 OCU_SETRELATIVETHRESHOLD_ID
 Ocu.h, 331
 OCU_STARTCHANNEL_ID
 Ocu.h, 331
 OCU_STOPCHANNEL_ID
 Ocu.h, 331
 OS_PLATFORM
 Oslf.h, 338
 OSIF_BAREMETAL
 Oslf.h, 338
 OSIF_CRITICAL_SW_MAJOR_VERSION
 Oslf_Critical.h, 342
 OSIF_CRITICAL_SW_MINOR_VERSION
 Oslf_Critical.h, 342
 OSIF_CRITICAL_SW_PATCH_VERSION
 Oslf_Critical.h, 343
 OSIF_ENTER_CRITICAL_PROTOTYPES
 Oslf_Critical.h, 343–346
 OSIF_ENTER_CRITICAL
 Oslf_Critical.h, 343
 OSIF_EXIT_CRITICAL_PROTOTYPES
 Oslf_Critical.h, 343, 346, 347
 OSIF_EXIT_CRITICAL
 Oslf_Critical.h, 343
 OSIF_IRQ_SW_MAJOR_VERSION
 Oslf_Irq.h, 348
 OSIF_IRQ_SW_MINOR_VERSION
 Oslf_Irq.h, 348
 OSIF_IRQ_SW_PATCH_VERSION
 Oslf_Irq.h, 348
 OSIF_OS
 Oslf.h, 338
 OSIF_SW_MAJOR_VERSION
 Oslf.h, 338
 OSIF_SW_MINOR_VERSION
 Oslf.h, 338
 OSIF_SW_PATCH_VERSION
 Oslf.h, 338
 OSIF_TIME_SW_MAJOR_VERSION
 Oslf_Time.h, 350
 OSIF_TIME_SW_MINOR_VERSION
 Oslf_Time.h, 350
 OSIF_TIME_SW_PATCH_VERSION
 Oslf_Time.h, 351
 Ocu.h, 325
 OCU_DEINIT_ID, 327
 OCU_DISABLENOTIFICATION_ID, 327
 OCU_E_ALREADY_INITIALIZED, 328
 OCU_E_BUSY, 328
 OCU_E_INIT_FAILED, 328
 OCU_E_NO_VALID_NOTIF, 328
 OCU_E_PARAM_INVALID_ACTION, 328
 OCU_E_PARAM_INVALID_CHANNEL, 328
 OCU_E_PARAM_INVALID_STATE, 329
 OCU_E_PARAM_INVALID_VALUE, 329
 OCU_E_PARAM_NO_PIN, 329
 OCU_E_PARAM_POINTER, 329
 OCU_E_UNINIT, 329
 OCU_ENABLENOTIFICATION_ID, 329
 OCU_GETCOUNTER_ID, 330
 OCU_GETVERSIONINFO_ID, 330
 OCU_INIT_ID, 330
 OCU_SETABSOLUTETHRESHOLD_ID, 330
 OCU_SETPINACTION_ID, 330
 OCU_SETPINSTATE_ID, 330
 OCU_SETRELATIVETHRESHOLD_ID, 331
 OCU_STARTCHANNEL_ID, 331
 OCU_STOPCHANNEL_ID, 331
 Ocu_ChannelType, 331
 Ocu_Delnit, 332
 Ocu_DisableNotification, 333
 Ocu_EnableNotification, 333
 Ocu_GenerateConfigPC, 337
 Ocu_GetCounter, 333
 Ocu_GetVersionInfo, 334
 Ocu_Init, 334
 Ocu_PinActionType, 331
 Ocu_PinStateType, 332
 Ocu_ReturnType, 332
 Ocu_SetAbsoluteThreshold, 334
 Ocu_SetPinAction, 335
 Ocu_SetPinState, 335
 Ocu_SetRelativeThreshold, 336
 Ocu_StartChannel, 336
 Ocu_StopChannel, 336
 Ocu_ValueType, 331
 Ocu_ChannelConfigType, 87
 Channel, 87
 ChannelDefaultThreshold, 87
 ChannelInitLevel, 87
 ChannelNotification, 88
 ChannelOutputPinUsed, 88
 ChannelPinAction, 88
 LogicId, 88
 PwmModulesId, 88
 Ocu_ChannelType
 Ocu.h, 331
 Ocu_ConfigType, 89
 IpConfig, 89
 OcuChannelConfig, 89
 OcuChannelCount, 89
 Ocu_Delnit
 Ocu.h, 332
 Ocu_DisableNotification
 Ocu.h, 333
 Ocu_EnableNotification
 Ocu.h, 333
 Ocu_GenerateConfigPC
 Ocu.h, 337
 Ocu_GetCounter
 Ocu.h, 333
 Ocu_GetVersionInfo
 Ocu.h, 334
 Ocu_Init
 Ocu.h, 334

- Ocu_IpConfigType, 89
 - OcuModuleConfig, 90
 - OcuModuleCount, 90
- Ocu_ModuleConfigType, 90
 - ClockSource, 91
 - ModuleId, 91
 - ModuleMaxValue, 91
 - Prescaler, 91
- Ocu_PinActionType
 - Ocu.h, 331
- Ocu_PinStateType
 - Ocu.h, 332
- Ocu_ReturnType
 - Ocu.h, 332
- Ocu_SetAbsoluteThreshold
 - Ocu.h, 334
- Ocu_SetPinAction
 - Ocu.h, 335
- Ocu_SetPinState
 - Ocu.h, 335
- Ocu_SetRelativeThreshold
 - Ocu.h, 336
- Ocu_StartChannel
 - Ocu.h, 336
- Ocu_StopChannel
 - Ocu.h, 336
- Ocu_ValueType
 - Ocu.h, 331
- OcuChannelConfig
 - Ocu_ConfigType, 89
- OcuChannelCount
 - Ocu_ConfigType, 89
- OcuModuleConfig
 - Ocu_IpConfigType, 90
- OcuModuleCount
 - Ocu_IpConfigType, 90
- Offset
 - Dio_ChannelGroupType, 46
- OldState
 - Fee_JobInfoType, 61
- Oslf.h, 337
 - OS_PLATFORM, 338
 - OSIF_BAREMETAL, 338
 - OSIF_OS, 338
 - OSIF_SW_MAJOR_VERSION, 338
 - OSIF_SW_MINOR_VERSION, 338
 - OSIF_SW_PATCH_VERSION, 338
 - Oslf_Deinit, 339
 - Oslf_GetCoreId, 339
 - Oslf_Init, 339
- Oslf_Critical.h, 340
 - CMU_HAL_ID1, 341
 - CRYPTO_MCAL_ID, 341
 - DMA_HAL_ID1, 341
 - DMA_HAL_ID2, 342
 - FLS_HAL_ID1, 342
 - FLSTST_HAL_ID1, 342
 - OSIF_CRITICAL_SW_MAJOR_VERSION, 342
 - OSIF_CRITICAL_SW_MINOR_VERSION, 342
 - OSIF_CRITICAL_SW_PATCH_VERSION, 343
 - OSIF_ENTER_CRITICAL_PROTOTYPES, 343–346
 - OSIF_ENTER_CRITICAL, 343
 - OSIF_EXIT_CRITICAL_PROTOTYPES, 343, 346, 347
 - OSIF_EXIT_CRITICAL, 343
 - UART_HAL_ID1, 343
 - WDG_HAL_CS_ID1, 344
 - WDG_HAL_CS_ID2, 344
 - WDG_HAL_CS_ID3, 344
- Oslf_CriticalNesting
 - Oslf_Irq.h, 349
- Oslf_Deinit
 - Oslf.h, 339
- Oslf_GetCoreId
 - Oslf.h, 339
- Oslf_GetCounter
 - Oslf_Time.h, 351
- Oslf_GetElapsed
 - Oslf_Time.h, 351
- Oslf_Init
 - Oslf.h, 339
- Oslf_Irq.h, 347
 - ISR, 348
 - OSIF_IRQ_SW_MAJOR_VERSION, 348
 - OSIF_IRQ_SW_MINOR_VERSION, 348
 - OSIF_IRQ_SW_PATCH_VERSION, 348
 - Oslf_CriticalNesting, 349
 - Oslf_PriMaskValue, 350
 - Oslf_ResumeAllInterrupts, 349
 - Oslf_SuspendAllInterrupts, 349
- Oslf_MicrosToTicks
 - Oslf_Time.h, 352
- Oslf_PriMaskValue
 - Oslf_Irq.h, 350
- Oslf_ResumeAllInterrupts
 - Oslf_Irq.h, 349
- Oslf_SuspendAllInterrupts
 - Oslf_Irq.h, 349
- Oslf_Time.h, 350
 - OSIF_TIME_SW_MAJOR_VERSION, 350
 - OSIF_TIME_SW_MINOR_VERSION, 350
 - OSIF_TIME_SW_PATCH_VERSION, 351
 - Oslf_GetCounter, 351
 - Oslf_GetElapsed, 351
 - Oslf_MicrosToTicks, 352
 - Oslf_UDelay, 352
- Oslf_UDelay
 - Oslf_Time.h, 352
- output64Ptr
 - Crypto_JobPrimitiveInputOutputType, 30
- outputKeyElementId
 - Crypto_JobRedirectionInfoType, 33
- outputKeyId
 - Crypto_JobRedirectionInfoType, 33
- outputLengthPtr
 - Crypto_JobPrimitiveInputOutputType, 30
- outputPtr
 - Crypto_JobPrimitiveInputOutputType, 30
- OverflowMode
 - Can_ControllerDescriptorType, 15
- OverflowNotification
 - Pwm_ModuleConfigType, 104

- PCallBackPtr
 - Fls_ConfigType, 66
- PCR
 - Port_PinConfigType, 96
 - Port_UnUsedPinConfigType, 97
- PDType
 - Fls_ConfigType, 66
- PErasePtr
 - Fls_ConfigType, 66
- PORT_E_DIRECTION_UNCHANGEABLE
 - Port_GeneralTypes.h, 358
- PORT_E_INIT_FAILED
 - Port_GeneralTypes.h, 358
- PORT_E_MODE_UNCHANGEABLE
 - Port_GeneralTypes.h, 358
- PORT_E_PARAM_INVALID_MODE
 - Port_GeneralTypes.h, 359
- PORT_E_PARAM_PIN
 - Port_GeneralTypes.h, 359
- PORT_E_PARAM_POINTER
 - Port_GeneralTypes.h, 359
- PORT_E_UNINIT
 - Port_GeneralTypes.h, 359
- PORT_GETVERSIONINFO_ID
 - Port_GeneralTypes.h, 359
- PORT_INIT_ID
 - Port_GeneralTypes.h, 360
- PORT_INSTANCE_ID
 - Port_GeneralTypes.h, 360
- PORT_REFRESHPINDIRECTION_ID
 - Port_GeneralTypes.h, 360
- PORT_SETPINDIRECTION_ID
 - Port_GeneralTypes.h, 360
- PORT_SETPINMODE_ID
 - Port_GeneralTypes.h, 360
- PRESC
 - Can_BitrateConfigType, 10
- PWM_DUTY_CYCLE_MAX
 - Pwm.h, 363
- PWM_PERIOD_MAX
 - Pwm.h, 363
- PWritePtr
 - Fls_ConfigType, 66
- Params
 - Crc_DmaConfig, 24
- PendingJob
 - Fee_JobPendingInfoType, 63
- PendingValid
 - Fee_JobPendingInfoType, 63
- Period
 - Pwm_ModuleConfigType, 104
- PeriodTime
 - Icu_DutyCycleType, 82
- Persist
 - CryptoKeyElement, 44
- Pin
 - Port_PinConfigType, 96
- PinMask
 - Port_DigitalFilterConfigType, 94
- Polarity
 - Pwm_ChannelConfigType, 100
- Port
 - Dio_ChannelGroupType, 46
- Port.h, 352
 - Port_GetVersionInfo, 353
 - Port_Init, 353
 - Port_RefreshPortDirection, 354
 - Port_SetPinDirection, 354
 - Port_SetPinMode, 355
- Port_ConfigType, 91
 - DigitalFilterConfig, 92
 - NumDigitalFilterPorts, 92
 - NumPins, 92
 - NumUnusedPins, 92
 - UnusedPadConfig, 93
 - UnusedPads, 93
 - UsedPadConfig, 93
- Port_DigitalFilterConfigType, 93
 - Clock, 94
 - PinMask, 94
 - PortID, 94
 - Width, 94
- Port_GeneralTypes.h, 355
 - GPIO_FUN0, 357
 - GPIO_FUN1, 357
 - GPIO_FUN2, 357
 - GPIO_FUN3, 357
 - GPIO_FUN4, 357
 - GPIO_FUN5, 358
 - GPIO_FUN6, 358
 - GPIO_FUN7, 358
 - PORT_E_DIRECTION_UNCHANGEABLE, 358
 - PORT_E_INIT_FAILED, 358
 - PORT_E_MODE_UNCHANGEABLE, 358
 - PORT_E_PARAM_INVALID_MODE, 359
 - PORT_E_PARAM_PIN, 359
 - PORT_E_PARAM_POINTER, 359
 - PORT_E_UNINIT, 359
 - PORT_GETVERSIONINFO_ID, 359
 - PORT_INIT_ID, 360
 - PORT_INSTANCE_ID, 360
 - PORT_REFRESHPINDIRECTION_ID, 360
 - PORT_SETPINDIRECTION_ID, 360
 - PORT_SETPINMODE_ID, 360
 - Port_PinDirectionType, 361
 - Port_PinModeType, 361
 - Port_PinType, 361
- Port_GetVersionInfo
 - Port.h, 353
- Port_Init
 - Port.h, 353
- Port_PinConfigType, 95
 - DataOut, 95
 - DirChangeable, 95
 - Direction, 95
 - GIOMode, 96
 - ModeChangeable, 96
 - PCR, 96
 - Pin, 96
- Port_PinDirectionType
 - Port_GeneralTypes.h, 361
- Port_PinModeType

- Port_GeneralTypes.h, [361](#)
- Port_PinType
 - Port_GeneralTypes.h, [361](#)
- Port_RefreshPortDirection
 - Port.h, [354](#)
- Port_SetPinDirection
 - Port.h, [354](#)
- Port_SetPinMode
 - Port.h, [355](#)
- Port_UnUsedPinConfigType, [97](#)
 - DataOut, [97](#)
 - Direction, [97](#)
 - PCR, [97](#)
- PortID
 - Port_DigitalFilterConfigType, [94](#)
- Prescaler
 - Icu_PwmModuleConfigType, [84](#)
 - Ocu_ModuleConfigType, [91](#)
 - Pwm_ModuleConfigType, [104](#)
- primitiveInfo
 - Crypto_JobPrimitiveInfoType, [28](#)
- PrimitivesCount
 - Crypto_ObjectType, [39](#)
- PrimitivesPtr
 - Crypto_ObjectType, [40](#)
- processingType
 - Crypto_JobPrimitiveInfoType, [28](#)
- Pwm.h, [362](#)
 - PWM_DUTY_CYCLE_MAX, [363](#)
 - PWM_PERIOD_MAX, [363](#)
 - Pwm_ChannelClassType, [363](#)
 - Pwm_ChannelType, [363](#)
 - Pwm_DeInit, [365](#)
 - Pwm_DisableNotification, [365](#)
 - Pwm_EdgeNotificationType, [364](#)
 - Pwm_EnableNotification, [365](#)
 - Pwm_GenerateConfigPC, [368](#)
 - Pwm_GetOutputState, [366](#)
 - Pwm_GetVersionInfo, [366](#)
 - Pwm_Init, [367](#)
 - Pwm_OutputStateType, [364](#)
 - Pwm_PeriodType, [363](#)
 - Pwm_ServiceIdType, [364](#)
 - Pwm_SetDutyCycle, [367](#)
 - Pwm_SetOutputTogle, [367](#)
 - Pwm_SetPeriodAndDuty, [368](#)
- Pwm_ChannelClassType
 - Pwm.h, [363](#)
- Pwm_ChannelConfigType, [98](#)
 - Channel, [98](#)
 - ChannelClass, [98](#)
 - ChannelMode, [98](#)
 - ChannelNotification, [99](#)
 - DeadTimeValue, [99](#)
 - EnableCombaineComple, [99](#)
 - EnableDeadTime, [99](#)
 - EnableInterrupt, [99](#)
 - EnableMatchTrigger, [99](#)
 - IdleState, [100](#)
 - LogicId, [100](#)
 - Polarity, [100](#)
- PwmDefaultDutycycle, [100](#)
- PwmModulesId, [100](#)
- TimePsc, [100](#)
- Pwm_ChannelType
 - Pwm.h, [363](#)
- Pwm_ConfigType, [101](#)
 - ChannelCfg, [101](#)
 - IpConfigPtr, [101](#)
 - PwmChannelCount, [101](#)
- Pwm_DeInit
 - Pwm.h, [365](#)
- Pwm_DisableNotification
 - Pwm.h, [365](#)
- Pwm_EdgeNotificationType
 - Pwm.h, [364](#)
- Pwm_EnableNotification
 - Pwm.h, [365](#)
- Pwm_GenerateConfigPC
 - Pwm.h, [368](#)
- Pwm_GetOutputState
 - Pwm.h, [366](#)
- Pwm_GetVersionInfo
 - Pwm.h, [366](#)
- Pwm_Init
 - Pwm.h, [367](#)
- Pwm_IpConfigType, [102](#)
 - ModuleCfg, [102](#)
 - PwmModuleCount, [102](#)
- Pwm_ModuleConfigType, [103](#)
 - ClockSource, [103](#)
 - CountMode, [103](#)
 - EnableInitTrigger, [103](#)
 - EnableInterrupt, [103](#)
 - EnableMaxTrigger, [104](#)
 - EnableOverflowEvent, [104](#)
 - Instance, [104](#)
 - OverflowNotification, [104](#)
 - Period, [104](#)
 - Prescaler, [104](#)
 - TriggerRatio, [105](#)
- Pwm_OutputStateType
 - Pwm.h, [364](#)
- Pwm_PeriodType
 - Pwm.h, [363](#)
- Pwm_ServiceIdType
 - Pwm.h, [364](#)
- Pwm_SetDutyCycle
 - Pwm.h, [367](#)
- Pwm_SetOutputTogle
 - Pwm.h, [367](#)
- Pwm_SetPeriodAndDuty
 - Pwm.h, [368](#)
- PwmChannelCount
 - Pwm_ConfigType, [101](#)
- PwmDefaultDutycycle
 - Pwm_ChannelConfigType, [100](#)
- PwmModuleConfigPtr
 - Icu_IpConfigType, [83](#)
- PwmModuleCount
 - Icu_IpConfigType, [83](#)
 - Pwm_IpConfigType, [102](#)

- PwmModulesId
 - Ocu_ChannelConfigType, [88](#)
 - Pwm_ChannelConfigType, [100](#)
- QueueSize
 - Crypto_ObjectType, [40](#)
- QueuedJobs
 - Crypto_ObjectType, [40](#)
- RdDataPtr
 - Fee_JobPendingInfoType, [63](#)
- ReadAccess
 - CryptoKeyElement, [44](#)
- redirectionConfig
 - Crypto_JobRedirectionInfoType, [34](#)
- ReservedSize
 - Fee_ClusterGroupType, [53](#)
- Result
 - Fee_JobInfoType, [61](#)
- resultLength
 - Crypto_PrimitiveInfoType, [41](#)
- RxInterrupt
 - Can_ControllerDescriptorType, [15](#)
- SEG_1
 - Can_BitrateConfigType, [10](#)
- SEG_2
 - Can_BitrateConfigType, [10](#)
- SENT_DEINIT_ID
 - CDD_Sent.h, [172](#)
- SENT_E_INVALID_CHANNEL
 - CDD_Sent.h, [172](#)
- SENT_E_INVALID_POINTER
 - CDD_Sent.h, [172](#)
- SENT_E_STATE_TRANSITION
 - CDD_Sent.h, [173](#)
- SENT_E_UNINIT
 - CDD_Sent.h, [173](#)
- SENT_ENABLE_CHANNEL_ID
 - CDD_Sent.h, [173](#)
- SENT_GET_CHANNELSTAUTS_ID
 - CDD_Sent.h, [173](#)
- SENT_GET_VERSIONINFO_ID
 - CDD_Sent.h, [173](#)
- SENT_INIT_ID
 - CDD_Sent.h, [173](#)
- SENT_INIT_TX_CTRL_ID
 - CDD_Sent.h, [174](#)
- SENT_INSTANCE_ID
 - CDD_Sent.h, [174](#)
- SJW
 - Can_BitrateConfigType, [11](#)
- SPI_ALL_HW_MAP
 - Spi.h, [370](#)
- SPI_ASYNCTRANSMIT_ID
 - Spi.h, [370](#)
- SPI_CANCEL_ID
 - Spi.h, [370](#)
- SPI_DEINIT_ID
 - Spi.h, [371](#)
- SPI_E_ALREADY_INITIALIZED
 - Spi.h, [371](#)
- SPI_E_PARAM_CHANNEL
 - Spi.h, [371](#)
- SPI_E_PARAM_JOB
 - Spi.h, [371](#)
- SPI_E_PARAM_LENGTH
 - Spi.h, [371](#)
- SPI_E_PARAM_POINTER
 - Spi.h, [371](#)
- SPI_E_PARAM_SEQ
 - Spi.h, [372](#)
- SPI_E_PARAM_UNIT
 - Spi.h, [372](#)
- SPI_E_SEQ_IN_PROCESS
 - Spi.h, [372](#)
- SPI_E_SEQ_PENDING
 - Spi.h, [372](#)
- SPI_E_UNINIT
 - Spi.h, [372](#)
- SPI_GETHWUNITSTATUS_ID
 - Spi.h, [373](#)
- SPI_GETJOBRESULT_ID
 - Spi.h, [373](#)
- SPI_GETSEQUENCERESULT_ID
 - Spi.h, [373](#)
- SPI_GETSTATUS_ID
 - Spi.h, [373](#)
- SPI_GETVERSIONINFO_ID
 - Spi.h, [373](#)
- SPI_INIT_ID
 - Spi.h, [373](#)
- SPI_INSTANCE
 - Spi.h, [374](#)
- SPI_MAINFUNCTIONHANDLING_ID
 - Spi.h, [374](#)
- SPI_READIB_ID
 - Spi.h, [374](#)
- SPI_SETASYNCMODE_ID
 - Spi.h, [374](#)
- SPI_SETUPEB_ID
 - Spi.h, [374](#)
- SPI_SYNCTRANSMIT_ID
 - Spi.h, [374](#)
- SPI_WRITEIB_ID
 - Spi.h, [375](#)
- sdu
 - Can_PduType, [23](#)
- SecondaryFamily
 - Crypto_PrimitiveType, [42](#)
- secondaryFamily
 - Crypto_AlgorithmInfoType, [25](#)
- secondaryInputKeyElementId
 - Crypto_JobRedirectionInfoType, [34](#)
- secondaryInputKeyId
 - Crypto_JobRedirectionInfoType, [34](#)
- secondaryInputLength
 - Crypto_JobPrimitiveInputOutputType, [31](#)
- secondaryInputPtr
 - Crypto_JobPrimitiveInputOutputType, [31](#)
- secondaryOutputKeyElementId
 - Crypto_JobRedirectionInfoType, [34](#)

- secondaryOutputKeyId
 - Crypto_JobRedirectionInfoType, 34
- secondaryOutputLengthPtr
 - Crypto_JobPrimitiveInputOutputType, 31
- secondaryOutputPtr
 - Crypto_JobPrimitiveInputOutputType, 31
- SectorCount
 - Fls_ConfigType, 66
- SectorEndAddr
 - Fls_ConfigType, 67
- SectorFlags
 - Fls_ConfigType, 67
- SectorPageSize
 - Fls_ConfigType, 67
- SectorSize
 - Fls_ConfigType, 67
- SectorStartAddr
 - Fls_ConfigType, 67
- Sent_ConfigType, 105
 - HwConfigPtr, 105
 - MaxChannelNum, 105
 - ModuleConfig, 106
- Sent_Deinit
 - CDD_Sent.h, 174
- Sent_EnableChannel
 - CDD_Sent.h, 174
- Sent_GetChannelStatus
 - CDD_Sent.h, 175
- Sent_Init
 - CDD_Sent.h, 176
- Sent_InitChannelTxCtrl
 - CDD_Sent.h, 176
- Service
 - Crypto_PrimitiveType, 43
- service
 - Crypto_PrimitiveInfoType, 41
- SignalMeasurementPropertyType
 - Icu_ChannelConfigType, 80
- SignatureAddress
 - FlsTst_BlockConfigType, 69
- Spi.h, 368
 - SPI_ALL_HW_MAP, 370
 - SPI_ASYNCTRANSMIT_ID, 370
 - SPI_CANCEL_ID, 370
 - SPI_DEINIT_ID, 371
 - SPI_E_ALREADY_INITIALIZED, 371
 - SPI_E_PARAM_CHANNEL, 371
 - SPI_E_PARAM_JOB, 371
 - SPI_E_PARAM_LENGTH, 371
 - SPI_E_PARAM_POINTER, 371
 - SPI_E_PARAM_SEQ, 372
 - SPI_E_PARAM_UNIT, 372
 - SPI_E_SEQ_IN_PROCESS, 372
 - SPI_E_SEQ_PENDING, 372
 - SPI_E_UNINIT, 372
 - SPI_GETHWUNITSTATUS_ID, 373
 - SPI_GETJOBRESULT_ID, 373
 - SPI_GETSEQUENCERESULT_ID, 373
 - SPI_GETSTATUS_ID, 373
 - SPI_GETVERSIONINFO_ID, 373
 - SPI_INIT_ID, 373
 - SPI_INSTANCE, 374
 - SPI_MAINFUNCTIONHANDLING_ID, 374
 - SPI_READIB_ID, 374
 - SPI_SETASYNCMODE_ID, 374
 - SPI_SETUPEB_ID, 374
 - SPI_SYNCTRANSMIT_ID, 374
 - SPI_WRITEIB_ID, 375
 - Spi_AsyncTransmit, 375
 - Spi_Cancel, 375
 - Spi_DeInit, 376
 - Spi_GetHWUnitStatus, 376
 - Spi_GetJobResult, 377
 - Spi_GetSequenceResult, 377
 - Spi_GetStatus, 378
 - Spi_GetVersionInfo, 378
 - Spi_Init, 379
 - Spi_MainFunction_Handling, 379
 - Spi_ReadIB, 380
 - Spi_SetAsyncMode, 380
 - Spi_SetupEB, 381
 - Spi_SyncTransmit, 381
 - Spi_WriteIB, 382
 - Spi_AsyncTransmit
 - Spi.h, 375
 - Spi_Cancel
 - Spi.h, 375
 - Spi_DeInit
 - Spi.h, 376
 - Spi_GetHWUnitStatus
 - Spi.h, 376
 - Spi_GetJobResult
 - Spi.h, 377
 - Spi_GetSequenceResult
 - Spi.h, 377
 - Spi_GetStatus
 - Spi.h, 378
 - Spi_GetVersionInfo
 - Spi.h, 378
 - Spi_Init
 - Spi.h, 379
 - Spi_MainFunction_Handling
 - Spi.h, 379
 - Spi_ReadIB
 - Spi.h, 380
 - Spi_SetAsyncMode
 - Spi.h, 380
 - Spi_SetupEB
 - Spi.h, 381
 - Spi_SyncTransmit
 - Spi.h, 381
 - Spi_WriteIB
 - Spi.h, 382
 - SrcDataPtr
 - Fee_JobInfoType, 61
 - StartAddr
 - Fee_ClusterHeaderType, 55
 - Fee_ClusterType, 57
 - State
 - Fee_JobInfoType, 61
 - Status
 - Fee_BlockInfoType, 50

- Fee_ClusterInfoType, 56
- swPduHandle
 - Can_PduType, 23
- TargetAddr
 - Fee_BlockHeaderType, 49
- targetCryIfKeyId
 - Crypto_JobPrimitiveInputOutputType, 31
- tertiaryInputKeyElementId
 - Crypto_JobRedirectionInfoType, 35
- tertiaryInputKeyId
 - Crypto_JobRedirectionInfoType, 35
- tertiaryInputLength
 - Crypto_JobPrimitiveInputOutputType, 32
- tertiaryInputPtr
 - Crypto_JobPrimitiveInputOutputType, 32
- TestAlgorithm
 - FlsTst_BlockConfigType, 69
- TestCompletedNotification
 - FlsTst_ConfigType, 73
- TestCompletion
 - FlsTst_CommonVariableType, 71
- TestIntervalId
 - FlsTst_CommonVariableType, 71
 - FlsTst_TestResultBgndType, 74
 - FlsTst_TestSignatureBgndType, 75
- TestResultBgnd
 - FlsTst_TestResultBgndType, 74
- TestSignatureValue
 - FlsTst_TestSignatureBgndType, 75
 - FlsTst_TestSignatureFgndType, 76
- TimePsc
 - Pwm_ChannelConfigType, 100
- TimeStampEn
 - Can_ControllerDescriptorType, 16
- TimestampBufferType
 - Icu_ChannelConfigType, 80
- TriggerRatio
 - Pwm_ModuleConfigType, 105
- TxInterrupt
 - Can_ControllerDescriptorType, 16
- UART_ABORTTRANSMIT_ID
 - CDD_Uart.h, 178
- UART_ASYNCTRANSMIT_ID
 - CDD_Uart.h, 178
- UART_DEINIT_ID
 - CDD_Uart.h, 178
- UART_E_BUSY_TRANSMIT
 - CDD_Uart.h, 178
- UART_E_INVALID_CHANNEL
 - CDD_Uart.h, 178
- UART_E_INVALID_POINTER
 - CDD_Uart.h, 178
- UART_E_STATE_TRANSITION
 - CDD_Uart.h, 179
- UART_E_TRANSMIT_LENGTH
 - CDD_Uart.h, 179
- UART_E_UNINIT
 - CDD_Uart.h, 179
- UART_GET_STATUS_ID
 - CDD_Uart.h, 179
- UART_GET_VERSIONINFO_ID
 - CDD_Uart.h, 179
- UART_HAL_ID1
 - Oslf_Critical.h, 343
- UART_INIT_ID
 - CDD_Uart.h, 179
- UART_INSTANCE_ID
 - CDD_Uart.h, 180
- Uart_AbortTransmit
 - CDD_Uart.h, 180
- Uart_AsyncTransmit
 - CDD_Uart.h, 181
- Uart_ConfigType, 106
 - HwConfigPtr, 106
 - MaxChannelNum, 106
- Uart_DeInit
 - CDD_Uart.h, 181
- Uart_DirType
 - CDD_Uart.h, 180
- Uart_GenerateConfigPC
 - CDD_Uart.h, 183
- Uart_GetStatus
 - CDD_Uart.h, 182
- Uart_GetVersionInfo
 - CDD_Uart.h, 182
- Uart_Init
 - CDD_Uart.h, 183
- Uniqueld
 - CryptoKeyElement, 45
- UnusedPadConfig
 - Port_ConfigType, 93
- UnusedPads
 - Port_ConfigType, 93
- UsedPadConfig
 - Port_ConfigType, 93
- Valid
 - Fee_BlockHeaderType, 49
 - Fee_ClusterHeaderType, 55
- ValuePtr
 - CryptoKeyElement, 45
- verifyPtr
 - Crypto_JobPrimitiveInputOutputType, 32
- WDG_E_DISABLE_REJECTED
 - Wdg_GeneralTypes.h, 387
- WDG_E_DRIVER_STATE
 - Wdg_GeneralTypes.h, 387
- WDG_E_PARAM_CONFIG
 - Wdg_GeneralTypes.h, 387
- WDG_E_PARAM_MODE
 - Wdg_GeneralTypes.h, 387
- WDG_E_PARAM_POINTER
 - Wdg_GeneralTypes.h, 387
- WDG_E_PARAM_TIMEOUT
 - Wdg_GeneralTypes.h, 387
- WDG_HAL_CS_ID1
 - Oslf_Critical.h, 344
- WDG_HAL_CS_ID2
 - Oslf_Critical.h, 344

- WDG_HAL_CS_ID3
 - Oslf_Critical.h, [344](#)
- WakeUpEn
 - Can_ControllerDescriptorType, [16](#)
- WakeUpInterrupt
 - Can_ControllerDescriptorType, [16](#)
- WakeupCapable
 - Icu_ChannelConfigType, [80](#)
- WakeupSource
 - Icu_ChannelConfigType, [80](#)
- Wdg.h, [383](#)
 - Wdg_GetVersionInfo, [384](#)
 - Wdg_Init, [384](#)
 - Wdg_ServiceIdType, [383](#)
 - Wdg_SetMode, [385](#)
 - Wdg_SetTriggerCondition, [385](#)
- Wdg_ConfigType, [107](#)
 - DefaultMode, [107](#)
 - ModeSetting, [107](#)
- Wdg_GeneralTypes.h, [386](#)
 - WDG_E_DISABLE_REJECTED, [387](#)
 - WDG_E_DRIVER_STATE, [387](#)
 - WDG_E_PARAM_CONFIG, [387](#)
 - WDG_E_PARAM_MODE, [387](#)
 - WDG_E_PARAM_POINTER, [387](#)
 - WDG_E_PARAM_TIMEOUT, [387](#)
 - WdgIf_ModeType, [388](#)
- Wdg_GetVersionInfo
 - Wdg.h, [384](#)
- Wdg_Init
 - Wdg.h, [384](#)
- Wdg_ServiceIdType
 - Wdg.h, [383](#)
- Wdg_SetMode
 - Wdg.h, [385](#)
- Wdg_SetTriggerCondition
 - Wdg.h, [385](#)
- WdgIf_ModeType
 - Wdg_GeneralTypes.h, [388](#)
- Width
 - Port_DigitalFilterConfigType, [94](#)
- WrDataPtr
 - Fee_JobPendingInfoType, [63](#)
- WriteAccess
 - CryptoKeyElement, [45](#)
- WriteTimeout
 - Fls_ConfigType, [67](#)