



S1C31W74 Peripheral Library Manual

Rev. 2017-02

Generated by Doxygen 1.8.12

February 2017

Contents

1	S1C31W74 Peripheral Library Firmware Manual	1
1.1	Introduction	1
1.2	Installation	1
2	Module Documentation	3
2.1	sePeriphLibrary	3
2.2	Common	5
2.3	Common_Constants	6
2.3.1	Enumeration Type Documentation	6
2.3.1.1	seInterruptStatus	6
2.3.1.2	seState	6
2.3.1.3	seStatus	6
2.3.1.4	seTimeoutMs	7
2.3.1.5	seWriteProtect	7
2.4	Common_Macros	8
2.4.1	Macro Definition Documentation	8
2.4.1.1	WAIT	8
2.5	Common_Functions	9
2.5.1	Function Documentation	9
2.5.1.1	seAssert()	9
2.5.1.2	seClamp16()	9
2.5.1.3	seClamp32()	10
2.5.1.4	seProtectSys()	10
2.6	CLG	11
2.7	CLG_Constants	12
2.7.1	Enumeration Type Documentation	13
2.7.1.1	seCLG_CLG_ClkDiv	13
2.7.1.2	seCLG_ClkSrc	13
2.7.1.3	seCLG_EXOSC_ClkDiv	13

2.7.1.4	seCLG_Interrupt	13
2.7.1.5	seCLG_IntFlag	14
2.7.1.6	seCLG_IOSC_ClkDiv	14
2.7.1.7	seCLG_IOSC_loscFq	14
2.7.1.8	seCLG_OSC1_ClkDiv	15
2.7.1.9	seCLG_OSC3_ClkDiv	15
2.8	CLG_Types	16
2.9	CLG_Functions	17
2.9.1	Function Documentation	18
2.9.1.1	CLG_IRQHandler()	18
2.9.1.2	ConfigurePortsForOSC3()	18
2.9.1.3	seCLG_ClearIntFlag()	18
2.9.1.4	seCLG_DisableInt()	18
2.9.1.5	seCLG_EnableInt()	19
2.9.1.6	seCLG_GetIntFlag()	19
2.9.1.7	seCLG_GetloscFreqSel()	19
2.9.1.8	seCLG_GetOperInSlp()	19
2.9.1.9	seCLG_GetSysClk()	20
2.9.1.10	seCLG_GetSysClkDiv()	20
2.9.1.11	seCLG_GetSysClkSrc()	20
2.9.1.12	seCLG_Init()	20
2.9.1.13	seCLG_RunAutoTrimming()	21
2.9.1.14	seCLG_SetloscFreqSel()	21
2.9.1.15	seCLG_SetOperInSlp()	21
2.9.1.16	seCLG_SetStopDetection()	22
2.9.1.17	seCLG_SetWkUpSysClk()	22
2.9.1.18	seCLG_Start()	23
2.9.1.19	seCLG_Stop()	23
2.9.1.20	seCLG_SwitchSysClkSrc()	23
2.10	DMAC	24
2.11	DMAC_Constants	25
2.11.1	Enumeration Type Documentation	25
2.11.1.1	seDMAC_CHANNEL	26
2.11.1.2	seDMAC_Inc	26
2.11.1.3	seDMAC_InterruptSrc	26
2.11.1.4	seDMAC_Mode	26
2.11.1.5	seDMAC_Size	27

2.12	DMAC_Types	28
2.13	DMAC_Functions	29
2.13.1	Function Documentation	30
2.13.1.1	DMAC_IRQHandler()	30
2.13.1.2	seDMAC_AlternateDisable()	30
2.13.1.3	seDMAC_AlternateEnable()	30
2.13.1.4	seDMAC_ClearIntFlag()	31
2.13.1.5	seDMAC_ConfigMemToPeriph()	31
2.13.1.6	seDMAC_ConfigPeriphToMem()	32
2.13.1.7	seDMAC_Disable()	32
2.13.1.8	seDMAC_DisableInt()	32
2.13.1.9	seDMAC_DisableRequestMask()	33
2.13.1.10	seDMAC_Enable()	33
2.13.1.11	seDMAC_EnableInt()	33
2.13.1.12	seDMAC_EnableRequestMask()	33
2.13.1.13	seDMAC_GetAltDataStrucPtr()	34
2.13.1.14	seDMAC_GetDataStrucPtr()	34
2.13.1.15	seDMAC_GetIntFlag()	34
2.13.1.16	seDMAC_GetMode()	35
2.13.1.17	seDMAC_GetNMinus1()	35
2.13.1.18	seDMAC_Init()	35
2.13.1.19	seDMAC_NonBlockTransfMemToPeriph()	35
2.13.1.20	seDMAC_NonBlockTransfPeriphToMem()	36
2.13.1.21	seDMAC_PriorityDecrease()	36
2.13.1.22	seDMAC_PriorityIncrease()	37
2.13.1.23	seDMAC_SetChannel()	37
2.13.1.24	seDMAC_SetDataStrucPtr()	37
2.13.1.25	seDMAC_Start()	38
2.14	I2C	39
2.15	I2C_Constants	40
2.15.1	Enumeration Type Documentation	41
2.15.1.1	sel2C_AddrMode	41
2.15.1.2	sel2C_EXOSC_ClkDiv	41
2.15.1.3	sel2C_Interrupt	41
2.15.1.4	sel2C_IntFlag	42
2.15.1.5	sel2C_IOSC_ClkDiv	42
2.15.1.6	sel2C_mode	42

2.15.1.7	sel2C_OSC1_ClkDiv	43
2.15.1.8	sel2C_OSC3_ClkDiv	43
2.16	I2C_Types	44
2.17	I2C_Functions	45
2.17.1	Function Documentation	45
2.17.1.1	ConfigurePortsForI2C()	45
2.17.1.2	I2C_0_IRQHandler()	46
2.17.1.3	I2C_1_IRQHandler()	46
2.17.1.4	sel2C_ClearIntFlag()	46
2.17.1.5	sel2C_Disable()	46
2.17.1.6	sel2C_DisableInt()	47
2.17.1.7	sel2C_Enable()	47
2.17.1.8	sel2C_EnableInt()	47
2.17.1.9	sel2C_GetIntFlag()	48
2.17.1.10	sel2C_Init()	48
2.17.1.11	sel2C_InitStructForMaster()	48
2.17.1.12	sel2C_InitStructForSlave()	49
2.17.1.13	sel2C_MstReceiveData()	49
2.17.1.14	sel2C_MstSendData()	50
2.17.1.15	sel2C_Reset()	50
2.17.1.16	sel2C_SlvReceiveData()	50
2.17.1.17	sel2C_SlvSendData()	51
2.18	LCD32B	52
2.19	LCD32B_Constants	53
2.20	LCD32B_Clock	54
2.20.1	Enumeration Type Documentation	54
2.20.1.1	seLCD32B_BoostClk	54
2.20.1.2	seLCD32B_EXOSC_ClkDiv	55
2.20.1.3	seLCD32B_IOSC_ClkDiv	55
2.20.1.4	seLCD32B_OSC1_ClkDiv	55
2.20.1.5	seLCD32B_OSC3_ClkDiv	55
2.21	LCD32B_DisplayState	57
2.21.1	Enumeration Type Documentation	57
2.21.1.1	seLCD32B_DisplnvState	57
2.21.1.2	seLCD32B_DispNetState	57
2.22	LCD32B_Frame_Frequency_Control	58
2.22.1	Enumeration Type Documentation	58

2.22.1.1	seLCD32B_Duty	58
2.23	LCD32B_Power_Settings	61
2.23.1	Enumeration Type Documentation	61
2.23.1.1	seLCD32B_PwrBias	61
2.24	LCD32B_Contrast	62
2.24.1	Enumeration Type Documentation	62
2.24.1.1	seLCD32B_Contrast	62
2.25	LCD32B_RAM_Area	63
2.25.1	Enumeration Type Documentation	63
2.25.1.1	seLCD32B_Area	63
2.25.1.2	seLCD32B_SegRamAddress	63
2.26	LCD32B_Types	64
2.27	LCD32B_Functions	65
2.27.1	Function Documentation	66
2.27.1.1	ConfigurePortsForLcd32B()	66
2.27.1.2	LCD32B_IRQHandler()	66
2.27.1.3	seLCD32B_ClearDisplayMemory()	66
2.27.1.4	seLCD32B_ClearIntFlag()	66
2.27.1.5	seLCD32B_ConfigureClock()	67
2.27.1.6	seLCD32B_ConfigureDisplay()	67
2.27.1.7	seLCD32B_ConfigurePower()	67
2.27.1.8	seLCD32B_CopyToDisplayArea()	68
2.27.1.9	seLCD32B_Disable()	68
2.27.1.10	seLCD32B_DisableInt()	68
2.27.1.11	seLCD32B_Enable()	68
2.27.1.12	seLCD32B_EnableInt()	69
2.27.1.13	seLCD32B_GetDisplayArea()	69
2.27.1.14	seLCD32B_GetDisplayInvState()	69
2.27.1.15	seLCD32B_GetDisplayState()	69
2.27.1.16	seLCD32B_GetIntFlag()	70
2.27.1.17	seLCD32B_GetPanelContrast()	70
2.27.1.18	seLCD32B_Init()	70
2.27.1.19	seLCD32B_SetBoostClk()	70
2.27.1.20	seLCD32B_SetDisplayArea()	71
2.27.1.21	seLCD32B_SetDisplayInvState()	71
2.27.1.22	seLCD32B_SetDisplayMemory()	71
2.27.1.23	seLCD32B_SetDisplayState()	72

2.27.1.24	seLCD32B_SetFrameFreq()	72
2.27.1.25	seLCD32B_SetPanelContrast()	72
2.28	PPOINT	73
2.29	PPOINT_Exported_constants	74
2.29.1	Enumeration Type Documentation	77
2.29.1.1	sePPOINT_AltFunc	77
2.29.1.2	sePPOINT_Data	78
2.29.1.3	sePPOINT_Edge	78
2.29.1.4	sePPOINT_EXOSC_ClkDiv	78
2.29.1.5	sePPOINT_Id	78
2.29.1.6	sePPOINT_IOSC_ClkDiv	80
2.29.1.7	sePPOINT_OSC1_ClkDiv	80
2.29.1.8	sePPOINT_OSC3_ClkDiv	81
2.29.1.9	sePPOINT_PeriphPortInit	81
2.29.1.10	sePPOINT_PortGroup	82
2.29.1.11	sePPOINT_PortNumber	82
2.29.1.12	seUPMUX_Channel_Sel	82
2.29.1.13	seUPMUX_I2C_Fnc	83
2.29.1.14	seUPMUX_Peripheral_Sel	83
2.29.1.15	seUPMUX_SPIA_Fnc	83
2.29.1.16	seUPMUX_T16B_Fnc	83
2.29.1.17	seUPMUX_UART_Fnc	84
2.30	PPOINT_Exported_Types	85
2.31	PPOINT_Public_Functions	86
2.31.1	Function Documentation	87
2.31.1.1	PPOINT_IRQHandler()	87
2.31.1.2	sePPOINT_ClearIntFlag()	87
2.31.1.3	sePPOINT_ConfigureClock()	87
2.31.1.4	sePPOINT_DisableBuiltinResistor()	88
2.31.1.5	sePPOINT_DisableChatteringFilter()	88
2.31.1.6	sePPOINT_DisableInt()	88
2.31.1.7	sePPOINT_EnableChatteringFilter()	89
2.31.1.8	sePPOINT_EnableInt()	89
2.31.1.9	sePPOINT_EnablePullDownResistor()	89
2.31.1.10	sePPOINT_EnablePullUpResistor()	90
2.31.1.11	sePPOINT_GetChatteringFilter()	90
2.31.1.12	sePPOINT_GetInput()	90

2.31.1.13 sePPORT_GetIntFlag()	90
2.31.1.14 sePPORT_GetOutput()	91
2.31.1.15 sePPORT_Init()	91
2.31.1.16 sePPORT_InitAsAltFunction()	91
2.31.1.17 sePPORT_InitAsHiZ()	92
2.31.1.18 sePPORT_InitAsInput()	92
2.31.1.19 sePPORT_InitAsOutput()	92
2.31.1.20 sePPORT_SetOutput()	93
2.31.1.21 sePPORT_UpMuxFunction()	93
2.32 QSPI	94
2.33 QSPI_Constants	95
2.33.1 Enumeration Type Documentation	96
2.33.1.1 seQSPI_AddrMode	96
2.33.1.2 seQSPI_Format	96
2.33.1.3 seQSPI_Interrupt	96
2.33.1.4 seQSPI_IntFlag	97
2.33.1.5 seQSPI_IO	97
2.33.1.6 seQSPI_OperMode	97
2.33.1.7 seQSPI_Phase	97
2.33.1.8 seQSPI_Polarity	97
2.33.1.9 seQSPI_TransferMode	98
2.34 QSPI_Types	99
2.35 QSPI_Functions	100
2.35.1 Function Documentation	101
2.35.1.1 ConfigurePortsForQSPI()	101
2.35.1.2 QSPI_IRQHandler()	102
2.35.1.3 seQSPI_ASSERT_MST_CS0()	102
2.35.1.4 seQSPI_ASSERT_MST_CS1()	102
2.35.1.5 seQSPI_ASSERT_MST_CS2()	102
2.35.1.6 seQSPI_ASSERT_MST_CS3()	103
2.35.1.7 seQSPI_ClearIntFlag()	103
2.35.1.8 seQSPI_ClearMasterRxMMA()	103
2.35.1.9 seQSPI_DisableInt()	103
2.35.1.10 seQSPI_DmaRxBytes()	104
2.35.1.11 seQSPI_DmaRxHWords()	104
2.35.1.12 seQSPI_DmaRxMmaWords()	105
2.35.1.13 seQSPI_DmaTxBytes()	105

2.35.1.14	seQSPI_DmaTxHWords()	106
2.35.1.15	seQSPI_EnableInt()	106
2.35.1.16	seQSPI_GetBusSpeed()	107
2.35.1.17	seQSPI_GetIntFlag()	107
2.35.1.18	seQSPI_Init()	107
2.35.1.19	seQSPI_InitStructForMaster()	108
2.35.1.20	seQSPI_InitStructForSlave()	108
2.35.1.21	seQSPI_NEGATE_MST_CS0()	108
2.35.1.22	seQSPI_NEGATE_MST_CS1()	108
2.35.1.23	seQSPI_NEGATE_MST_CS2()	109
2.35.1.24	seQSPI_NEGATE_MST_CS3()	109
2.35.1.25	seQSPI_Reset()	109
2.35.1.26	seQSPI_RxBytes()	109
2.35.1.27	seQSPI_RxHWords()	110
2.35.1.28	seQSPI_SetBusSpeed()	110
2.35.1.29	seQSPI_SetIO()	111
2.35.1.30	seQSPI_SetMasterRxMMA()	111
2.35.1.31	seQSPI_SetMode()	111
2.35.1.32	seQSPI_Start()	112
2.35.1.33	seQSPI_Stop()	112
2.35.1.34	seQSPI_TermMasterTx()	112
2.35.1.35	seQSPI_TxBytes()	113
2.35.1.36	seQSPI_TxHWords()	113
2.35.1.37	seQSPI_TxValue()	113
2.36	REMC2	115
2.37	REMC2_Constants	116
2.37.1	Enumeration Type Documentation	117
2.37.1.1	seREMC2_EXOSC_ClkDiv	117
2.37.1.2	seREMC2_Interrupt	117
2.37.1.3	seREMC2_IOSC_ClkDiv	117
2.37.1.4	seREMC2_OSC1_ClkDiv	118
2.37.1.5	seREMC2_OSC3_ClkDiv	118
2.38	REMC2_Types	119
2.39	REMC2_Functions	120
2.39.1	Function Documentation	120
2.39.1.1	seREMC2_ConfigureClock()	120
2.39.1.2	seREMC2_Init()	120

2.39.1.3	seREMC2_Start()	121
2.39.1.4	seREMC2_Stop()	121
2.40	RFC	122
2.41	RFC_Constants	123
2.41.1	Enumeration Type Documentation	123
2.41.1.1	seRFC__IOSC_ClkDiv	123
2.41.1.2	seRFC_EXOSC_ClkDiv	124
2.41.1.3	seRFC_Interrupt	124
2.41.1.4	seRFC_OSC1_ClkDiv	124
2.41.1.5	seRFC_OSC3_ClkDiv	125
2.41.1.6	seRFC_OscMode	125
2.42	RFC_Types	126
2.43	RFC_Functions	127
2.43.1	Function Documentation	127
2.43.1.1	ConfigurePortsForRFC()	127
2.43.1.2	seRFC_ClearIntFlag()	128
2.43.1.3	seRFC_ConfigureClock()	128
2.43.1.4	seRFC_DisableInt()	128
2.43.1.5	seRFC_EnableInt()	129
2.43.1.6	seRFC_GetMeasurementCounter()	129
2.43.1.7	seRFC_GetTimeBaseCounter()	129
2.43.1.8	seRFC_Init()	129
2.43.1.9	seRFC_RunConvertingOperation()	130
2.43.1.10	seRFC_SetMeasurementCounter()	130
2.43.1.11	seRFC_SetTimeBaseCounter()	130
2.43.1.12	seRFC_Start()	131
2.43.1.13	seRFC_Stop()	131
2.44	RTCA	132
2.45	RTCA_Constants	133
2.45.1	Enumeration Type Documentation	133
2.45.1.1	seRTCA_AM_PM	133
2.45.1.2	seRTCA_DayOfTheWeek	134
2.45.1.3	seRTCA_Hours12_24	134
2.45.1.4	seRTCA_Interrupt	134
2.46	RTCA_Types	136
2.47	RTCA_Functions	137
2.47.1	Function Documentation	138

2.47.1.1	RTCA_IRQHandler()	138
2.47.1.2	seRTCA_CalculateTrm()	138
2.47.1.3	seRTCA_CalcWeekDay()	139
2.47.1.4	seRTCA_ClearIntFlag()	139
2.47.1.5	seRTCA_Disable1SecTimer()	140
2.47.1.6	seRTCA_DisableInt()	140
2.47.1.7	seRTCA_Enable1SecTimer()	140
2.47.1.8	seRTCA_EnableInt()	140
2.47.1.9	seRTCA_Get12_24Mode()	141
2.47.1.10	seRTCA_GetAlarm()	141
2.47.1.11	seRTCA_GetAM_PM()	141
2.47.1.12	seRTCA_GetHourMinuteSecond()	141
2.47.1.13	seRTCA_GetIntFlag()	142
2.47.1.14	seRTCA_GetYearMonthDayWeek()	142
2.47.1.15	seRTCA_Init()	142
2.47.1.16	seRTCA_InitTheoreticalRegulation()	143
2.47.1.17	seRTCA_ReadStopWatchCount()	143
2.47.1.18	seRTCA_ResetStopWatchCount()	143
2.47.1.19	seRTCA_Set12_24Mode()	144
2.47.1.20	seRTCA_Set30secCorrection()	144
2.47.1.21	seRTCA_SetAlarm()	144
2.47.1.22	seRTCA_SetAM_PM()	145
2.47.1.23	seRTCA_SetHourMinuteSecond()	145
2.47.1.24	seRTCA_SetSecondsAlarm()	145
2.47.1.25	seRTCA_SetYearMonthDayWeek()	146
2.47.1.26	seRTCA_Start()	146
2.47.1.27	seRTCA_StartStopWatchCount()	146
2.47.1.28	seRTCA_Stop()	147
2.47.1.29	seRTCA_StopStopWatchCount()	147
2.47.1.30	seRTCA_TheoreticalRegulationTrim()	147
2.48	SNDA	148
2.49	SNDA_Constants	149
2.49.1	Enumeration Type Documentation	149
2.49.1.1	seSNDA_DriveMode	149
2.49.1.2	seSNDA_EXOSC_ClkDiv	150
2.49.1.3	seSNDA_InterruptSrc	150
2.49.1.4	seSNDA_IOSC_ClkDiv	150

2.49.1.5	seSNDA_ModeSel	150
2.49.1.6	seSNDA_OSC1_ClkDiv	150
2.49.1.7	seSNDA_OSC3_ClkDiv	151
2.50	SNDA_Types	152
2.51	SNDA_Functions	153
2.51.1	Function Documentation	153
2.51.1.1	seSNDA_ClearIntFlag()	153
2.51.1.2	seSNDA_ConfigureClock()	154
2.51.1.3	seSNDA_Disable()	154
2.51.1.4	seSNDA_DisableInt()	154
2.51.1.5	seSNDA_Enable()	155
2.51.1.6	seSNDA_EnableInt()	155
2.51.1.7	seSNDA_GetClk()	155
2.51.1.8	seSNDA_GetIntFlag()	156
2.51.1.9	seSNDA_Init()	156
2.51.1.10	seSNDA_InitStruct()	156
2.51.1.11	seSNDA_Start()	157
2.51.1.12	seSNDA_StartMelody()	157
2.51.1.13	seSNDA_StartOneShot()	157
2.51.1.14	seSNDA_Stop()	158
2.51.1.15	SNDA_IRQHandler()	158
2.52	SPIA	159
2.53	SPIA_Constants	160
2.53.1	Enumeration Type Documentation	161
2.53.1.1	seSPIA_Format	161
2.53.1.2	seSPIA_Interrupt	161
2.53.1.3	seSPIA_IntFlag	161
2.53.1.4	seSPIA_OperMode	161
2.53.1.5	seSPIA_Phase	162
2.53.1.6	seSPIA_Polarity	162
2.54	SPIA_Types	163
2.55	SPIA_Functions	164
2.55.1	Function Documentation	165
2.55.1.1	ConfigurePortsForSPI()	165
2.55.1.2	seSPIA_ASSERT_MST_CS0()	165
2.55.1.3	seSPIA_ASSERT_MST_CS1()	165
2.55.1.4	seSPIA_ClearIntFlag()	166

2.55.1.5	seSPIA_DisableInt()	166
2.55.1.6	seSPIA_DmaRxHWords()	166
2.55.1.7	seSPIA_DmaTxHWords()	167
2.55.1.8	seSPIA_ENABLE_MST_CS0()	167
2.55.1.9	seSPIA_ENABLE_MST_CS1()	167
2.55.1.10	seSPIA_EnableInt()	167
2.55.1.11	seSPIA_GetBusSpeed()	168
2.55.1.12	seSPIA_GetIntFlag()	168
2.55.1.13	seSPIA_Init()	168
2.55.1.14	seSPIA_InitStructForMaster()	169
2.55.1.15	seSPIA_InitStructForSlave()	169
2.55.1.16	seSPIA_NEGATE_MST_CS0()	169
2.55.1.17	seSPIA_NEGATE_MST_CS1()	170
2.55.1.18	seSPIA_Reset()	170
2.55.1.19	seSPIA_RxBytes()	170
2.55.1.20	seSPIA_RxHWords()	171
2.55.1.21	seSPIA_SetBusSpeed()	171
2.55.1.22	seSPIA_Start()	171
2.55.1.23	seSPIA_Stop()	172
2.55.1.24	seSPIA_TxBytes()	172
2.55.1.25	seSPIA_TxHWords()	172
2.55.1.26	SPIA_0_IRQHandler()	173
2.56	SVD2	174
2.57	SVD2_Constants	175
2.57.1	Enumeration Type Documentation	176
2.57.1.1	seSVD2_DetectMode	176
2.57.1.2	seSVD2_EXOSC_ClkDiv	176
2.57.1.3	seSVD2_IntermittentMode	176
2.57.1.4	seSVD2_Interrupt	176
2.57.1.5	seSVD2_IntFlag	176
2.57.1.6	seSVD2_IOSC_ClkDiv	177
2.57.1.7	seSVD2_OSC1_ClkDiv	177
2.57.1.8	seSVD2_OSC3_ClkDiv	177
2.57.1.9	seSVD2_PowerSupply	177
2.57.1.10	seSVD2_SamplingResCnt	178
2.57.1.11	seSVD2_VoltageSource	178
2.58	SVD_Types	179

2.59	SVD_Functions	180
2.59.1	Function Documentation	180
2.59.1.1	ConfigurePortsForSVD2()	180
2.59.1.2	seSVD2_ClearIntLowVoltage()	181
2.59.1.3	seSVD2_GetVoltageDetection()	181
2.59.1.4	seSVD2_Init()	181
2.59.1.5	seSVD2_InitStruct()	182
2.59.1.6	seSVD2_IsIntLowVoltage()	182
2.59.1.7	seSVD2_SetComparisonVoltage()	182
2.59.1.8	seSVD2_SetVoltageSource()	182
2.59.1.9	seSVD2_Start()	183
2.59.1.10	seSVD2_Stop()	183
2.59.1.11	SVD2_0_IRQHandler()	183
2.59.1.12	SVD2_1_IRQHandler()	184
2.60	T16	185
2.61	T16_Constants	186
2.61.1	Enumeration Type Documentation	187
2.61.1.1	seT16_CounterMode	187
2.61.1.2	seT16_EXOSC_ClkDiv	187
2.61.1.3	seT16_IOSC_ClkDiv	187
2.61.1.4	seT16_OSC1_ClkDiv	188
2.61.1.5	seT16_OSC3_ClkDiv	188
2.62	T16_Types	189
2.63	T16_Functions	190
2.63.1	Function Documentation	190
2.63.1.1	seT16_ClearIntFlag()	190
2.63.1.2	seT16_ConfigureClock()	191
2.63.1.3	seT16_ConfigureCounterMode()	191
2.63.1.4	seT16_Disable()	191
2.63.1.5	seT16_DisableInt()	192
2.63.1.6	seT16_Enable()	192
2.63.1.7	seT16_EnableInt()	192
2.63.1.8	seT16_GetClk()	193
2.63.1.9	seT16_GetCounter()	193
2.63.1.10	seT16_GetIntFlag()	193
2.63.1.11	seT16_Init()	194
2.63.1.12	seT16_InitStruct()	194

2.63.1.13	seT16_SetCounter()	194
2.63.1.14	seT16_Start()	195
2.63.1.15	seT16_Stop()	195
2.63.1.16	T16_0_IRQHandler()	195
2.63.1.17	T16_1_IRQHandler()	195
2.63.1.18	T16_2_IRQHandler()	196
2.63.1.19	T16_3_IRQHandler()	196
2.64	T16B	197
2.65	T16B_Constants	198
2.65.1	Enumeration Type Documentation	200
2.65.1.1	seT16B_CAPIS	200
2.65.1.2	seT16B_CAPTRG	201
2.65.1.3	seT16B_CBUFMD	201
2.65.1.4	seT16B_CCMD	201
2.65.1.5	seT16B_ClkSrc	201
2.65.1.6	seT16B_CNTMD	201
2.65.1.7	seT16B_EX_ClkDiv	202
2.65.1.8	seT16B_IOSC_ClkDiv	202
2.65.1.9	seT16B_ONEST	202
2.65.1.10	seT16B_OSC1_ClkDiv	203
2.65.1.11	seT16B_OSC3_ClkDiv	203
2.65.1.12	seT16B_SCS	203
2.65.1.13	seT16B_TOUTINV	204
2.65.1.14	seT16B_TOUTMD	204
2.65.1.15	seT16B_TOUTMT	204
2.65.1.16	seT16B_TOUTO	204
2.66	T16B_Types	205
2.67	T16B_Functions	206
2.67.1	Function Documentation	207
2.67.1.1	ConfigurePortsForT16B()	207
2.67.1.2	seT16B_ClearIntFlag()	207
2.67.1.3	seT16B_ConfigureClock()	207
2.67.1.4	seT16B_ConfigureCounterMode()	208
2.67.1.5	seT16B_Disable()	208
2.67.1.6	seT16B_DisableInt()	208
2.67.1.7	seT16B_Enable()	209
2.67.1.8	seT16B_EnableInt()	209

2.67.1.9	seT16B_GetClk()	209
2.67.1.10	seT16B_GetCmpCapCnt()	209
2.67.1.11	seT16B_GetIntFlag()	210
2.67.1.12	seT16B_GetMaxCounter()	210
2.67.1.13	seT16B_GetTimerCount()	210
2.67.1.14	seT16B_Init()	211
2.67.1.15	seT16B_InitStruct()	211
2.67.1.16	seT16B_SetCmpCapCnt()	212
2.67.1.17	seT16B_SetMaxCounter()	212
2.67.1.18	seT16B_SetTriggerSignal()	212
2.67.1.19	seT16B_Start()	213
2.67.1.20	seT16B_Stop()	213
2.67.1.21	T16B_0_IRQHandler()	213
2.67.1.22	T16B_1_IRQHandler()	213
2.68	UART2	215
2.69	UART2_Constants	216
2.69.1	Enumeration Type Documentation	217
2.69.1.1	seUART2_EXOSC_ClkDiv	217
2.69.1.2	seUART2_Interrupt	217
2.69.1.3	seUART2_IOSC_ClkDiv	217
2.69.1.4	seUART2_MOD_Chln	218
2.69.1.5	seUART2_MOD_Invirrx	218
2.69.1.6	seUART2_MOD_Invirtx	218
2.69.1.7	seUART2_MOD_Irmd	218
2.69.1.8	seUART2_MOD_Outmd	219
2.69.1.9	seUART2_MOD_Pren	219
2.69.1.10	seUART2_MOD_Prmd	219
2.69.1.11	seUART2_MOD_Puen	219
2.69.1.12	seUART2_MOD_Stpb	220
2.69.1.13	seUART2_OSC1_ClkDiv	220
2.69.1.14	seUART2_OSC3_ClkDiv	220
2.70	UART_Types	221
2.71	UART_Functions	222
2.71.1	Function Documentation	223
2.71.1.1	ConfigurePortsForUart()	223
2.71.1.2	seUART2_ClearIntFlag()	223
2.71.1.3	seUART2_ConfigureClock()	223

2.71.1.4	seUART2_ConfigureMode()	224
2.71.1.5	seUART2_Disable()	224
2.71.1.6	seUART2_DisableInt()	224
2.71.1.7	seUART2_Enable()	225
2.71.1.8	seUART2_EnableInt()	225
2.71.1.9	seUART2_EnableRxDMAReq()	225
2.71.1.10	seUART2_EnableTxDMAReq()	226
2.71.1.11	seUART2_GetData()	226
2.71.1.12	seUART2_GetIntFlag()	226
2.71.1.13	seUART2_GetUartClk()	227
2.71.1.14	seUART2_Init()	227
2.71.1.15	seUART2_InitStruct()	227
2.71.1.16	seUART2_Receive()	228
2.71.1.17	seUART2_Send()	228
2.71.1.18	seUART2_SetBaudRate()	229
2.71.1.19	seUART2_SetBaudRateReg()	229
2.71.1.20	seUART2_SetData()	230
2.71.1.21	UART2_0_IRQHandler()	230
2.71.1.22	UART2_1_IRQHandler()	230
2.72	USB_Constants_and_Macros	231
2.72.1	Enumeration Type Documentation	232
2.72.1.1	seUSB_ClkSrc	232
2.72.1.2	seUSB_Ep0Interrupt	233
2.72.1.3	seUSB_EPCFG_DIR	233
2.72.1.4	seUSB_EPCFG_MAXSIZE	233
2.72.1.5	seUSB_EPCFG_TGLMOD	233
2.72.1.6	seUSB_EpmInterrupt	234
2.72.1.7	seUSB_GpeInterrupt	234
2.72.1.8	seUSB_MainInterrupt	234
2.72.1.9	seUSB_PSEL	234
2.72.1.10	seUSB_SieInterrupt	235
2.72.1.11	seUSB_TRCTL_OPMODE	235
2.73	USB_Types	236
2.74	USB_Functions	237
2.74.1	Function Documentation	238
2.74.1.1	PORT_IRQHandler()	238
2.74.1.2	seUSB_ActivateUSBCLK()	238

2.74.1.3	seUSB_Attach()	238
2.74.1.4	seUSB_ClrStall()	239
2.74.1.5	seUSB_ConfigureDefaultEndpoints()	239
2.74.1.6	seUSB_ConfigureEPm()	239
2.74.1.7	seUSB_ConfigurePortsForUsb()	239
2.74.1.8	seUSB_ConfSvdDetectDisconnect()	240
2.74.1.9	seUSB_DeactivateUSBCLK()	240
2.74.1.10	seUSB_Detach()	240
2.74.1.11	seUSB_Disable()	240
2.74.1.12	seUSB_DisableEPm()	241
2.74.1.13	seUSB_DisableInt()	241
2.74.1.14	seUSB_Enable()	241
2.74.1.15	seUSB_EnableEPm()	241
2.74.1.16	seUSB_EnableInt()	242
2.74.1.17	seUSB_Init()	242
2.74.1.18	seUSB_InitUsbModule()	242
2.74.1.19	seUSB_IsVbusConnected()	243
2.74.1.20	seUSB_SetStall()	243
2.74.1.21	SVD2_1_IRQHandler()	243
2.75	USB_EP0_Functions	244
2.75.1	Function Documentation	244
2.75.1.1	seseUSB_StatusOutStage()	244
2.75.1.2	seUSB_ClearEP0Fifo()	244
2.75.1.3	seUSB_DataInStage()	245
2.75.1.4	seUSB_DataOutStage()	245
2.75.1.5	seUSB_GetEp0Dir()	245
2.75.1.6	seUSB_GetSetupPacket()	245
2.75.1.7	seUSB_InitEp0()	245
2.75.1.8	seUSB_SetEp0Dir()	246
2.75.1.9	seUSB_SetupStage()	246
2.75.1.10	seUSB_StatusInStage()	246
2.76	USB_EPm_Functions	247
2.76.1	Function Documentation	247
2.76.1.1	seUSB_ClearEPmFifo()	247
2.76.1.2	seUSB_GetEPmDir()	247
2.76.1.3	seUSB_ResetEPm()	248
2.76.1.4	seUSB_SetEPmDir()	248

2.77	USB_EPn_Functions	249
2.77.1	Function Documentation	249
2.77.1.1	seUSB_ClearEPnFifos()	249
2.78	USB_Bus_Management	250
2.78.1	Function Documentation	250
2.78.1.1	seUSB_Connect()	250
2.78.1.2	seUSB_Disconnect()	251
2.78.1.3	seUSB_GetAddress()	251
2.78.1.4	seUSB_IsSnoozing()	251
2.78.1.5	seUSB_Reset()	251
2.78.1.6	seUSB_Resume()	251
2.78.1.7	seUSB_SetAddress()	252
2.78.1.8	seUSB_Snooze()	252
2.78.1.9	seUSB_Suspend()	252
2.78.1.10	seUSB_WakeUp()	253
2.79	USB_FIFO_Management	254
2.79.1	Function Documentation	254
2.79.1.1	seUSB_DmaCopyFromFifo()	254
2.79.1.2	seUSB_DmaCopyToFifo()	254
2.79.1.3	seUSB_ReadFifo()	255
2.79.1.4	seUSB_WriteFifo()	255
2.80	USB_Interrupt_Management	256
2.80.1	Function Documentation	257
2.80.1.1	seUSB_ClearAllIntFlags()	257
2.80.1.2	seUSB_DisableAllInt()	257
2.80.1.3	seUSB_EnableAllInt()	257
2.81	WDT2	258
2.82	WDT2_Constants	259
2.82.1	Enumeration Type Documentation	259
2.82.1.1	seWDT2_EXOSC_ClkDiv	259
2.82.1.2	seWDT2_IOSC_ClkDiv	259
2.82.1.3	seWDT2_Mode	260
2.82.1.4	seWDT2_OSC1_ClkDiv	260
2.82.1.5	seWDT2_OSC3_ClkDiv	260
2.83	WDT2_Types	261
2.84	WDT2_Functions	262
2.84.1	Function Documentation	262

2.84.1.1	NMI_Handler()	262
2.84.1.2	seWDT2_ChipReset()	263
2.84.1.3	seWDT2_ConfigureClock()	263
2.84.1.4	seWDT2_Get_tWDTSecs()	263
2.84.1.5	seWDT2_GetClk()	263
2.84.1.6	seWDT2_GetCMP()	264
2.84.1.7	seWDT2_Init()	264
2.84.1.8	seWDT2_InitStruct()	264
2.84.1.9	seWDT2_ResetCounter()	264
2.84.1.10	seWDT2_Set_tWDTSecs()	265
2.84.1.11	seWDT2_SetCMP()	265
2.84.1.12	seWDT2_SetMode()	265
2.84.1.13	seWDT2_Start()	266
2.84.1.14	seWDT2_Stop()	266
2.85	USB	267
3	Data Structure Documentation	269
3.1	seCLG_ClkDiv Union Reference	269
3.2	seCLG_InitTypeDef Struct Reference	269
3.3	seDMAC_CtrlData Union Reference	270
3.4	seDMAC_DataStruct Struct Reference	270
3.4.1	Field Documentation	271
3.4.1.1	transfer_source_end_pointer	271
3.5	sel2C_ChannelDef Struct Reference	271
3.6	sel2C_ClkDiv Union Reference	271
3.7	sel2C_InitTypeDef Struct Reference	271
3.7.1	Field Documentation	272
3.7.1.1	BRT	272
3.7.1.2	SlaveAddr	272
3.8	seLCD32B_ClkDiv Union Reference	272
3.9	seLCD32B_InitTypeDef Struct Reference	272
3.9.1	Field Documentation	273
3.9.1.1	CtlLcdIoDischargeEn	273
3.9.1.2	DspSelComPinDir	273
3.9.1.3	DspSelSegPinDir	273
3.10	sePPORT_Group Struct Reference	274
3.10.1	Field Documentation	274

3.10.1.1 DAT	274
3.11 sePPORT_InitTypeDef Struct Reference	274
3.12 sePPORT_PeriphPortDef Struct Reference	275
3.13 seQSPI_ChannelDef Struct Reference	275
3.14 seQSPI_InitTypeDef Struct Reference	275
3.15 seREMC2_ChannelDef Struct Reference	277
3.16 seREMC2_InitTypeDef Struct Reference	277
3.17 seRFC_ChannelDef Struct Reference	277
3.18 seRFC_InitTypeDef Struct Reference	278
3.19 seRTCA_InitTypeDef Struct Reference	278
3.20 seSNDA_ChannelDef Struct Reference	278
3.21 seSNDA_InitTypeDef Struct Reference	279
3.22 seSPIA_ChannelDef Struct Reference	279
3.23 seSPIA_InitTypeDef Struct Reference	279
3.24 seSVD2_ChannelDef Struct Reference	280
3.25 seSVD2_ClkDiv Union Reference	280
3.26 seSVD2_InitTypeDef Struct Reference	280
3.27 seT16_InitTypeDef Struct Reference	281
3.28 seT16B_CCCTL Struct Reference	282
3.29 seT16B_CCRegsDef Struct Reference	282
3.29.1 Field Documentation	283
3.29.1.1 CAPIS	283
3.29.1.2 CAPTRG	283
3.29.1.3 CBUFMD	283
3.29.1.4 CC	283
3.29.1.5 CC0DMAEN	283
3.29.1.6 CCCTL	283
3.29.1.7 CCCTL_b	283
3.29.1.8 CCDMAEN	284
3.29.1.9 CCDMAEN_b	284
3.29.1.10 CCMD	284
3.29.1.11 CCR	284
3.29.1.12 CCR_b	284
3.29.1.13 SCS	284
3.29.1.14 TOUTINV	284
3.29.1.15 TOUTMD	284
3.29.1.16 TOUTMT	284

3.29.1.17 TOUTO	285
3.30 seT16B_ChannelDef Struct Reference	285
3.31 seT16B_InitTypeDef Struct Reference	285
3.32 seUART2_ChannelDef Struct Reference	286
3.33 seUART2_ClkDiv Union Reference	286
3.34 seUART2_InitTypeDef Struct Reference	286
3.35 seUART2_Mode Union Reference	287
3.36 seUSB_InitTypeDef Struct Reference	287
3.37 seWDT2_InitTypeDef Struct Reference	288
3.37.1 Field Documentation	288
3.37.1.1 ClkDivider	288
3.38 swCounter Struct Reference	288
Index	291

Chapter 1

S1C31W74 Peripheral Library Firmware Manual

1.1 Introduction

The S1C31W74 microcontroller features ARM(R) Cortex(R)-M0+ CPU core, Nested Vectored Interrupt Controller (NVIC), System Timer (Systick), Serial-Wire Debug Port (SW-DP), Micro Trace Buffer (MTB), four hardware break points, and two watch points.

This microcontroller supports up to 4G bytes of accessible memory space for both instructions and data.

The S1C31W74 microcontroller provides a number of peripheral modules to control power supply, reset circuitry, clock generator, and IO ports. There are several other modules, like communication peripherals and a LCD driver.

1.2 Installation

There are multiple demo programs that show examples of calling the Peripheral Library modules to interface hardware components on this chip. The sample demo software can be built and executed on the Epson SVT board. If you haven't already done so, download the latest S1C31W74 archive from the web, and follow the instructions found inside the README.txt file.

Chapter 2

Module Documentation

2.1 sePeriphLibrary

The S1C31W74 Peripheral library firmware is a set of drivers. The library provides customers with uniform access to chip peripherals. The driver functions responsible for modules initialization, features configuration, modules access. The driver functions take advantage of chip peripheral addresses and register layout description in the Device Definition file.

Modules

- [Common](#)

All common definitions, data structures, and functions prototypes for the peripheral library.

- [CLG](#)

CLG module is the clock generator that controls the clock sources and manages clock supply to the CPU and the peripheral circuits.

- [DMAC](#)

The Direct Memory Access (DMAC) module is designed to move data between peripherals and memory without core participation. It speeds up transfers and saves power. The channels have separate controls and flags. Each channel has one or more transfer descriptors stored in chip's memory. Descriptors contain details of the data transfers.

- [I2C](#)

The I2C module is a subset of the I2C bus interface.

- [LCD32B](#)

The LCD32B is a module to drive an LCD panel.

- [PPORT](#)

The PPORT (I/O Ports) circuit. This circuit allows the selection of I/O functions on processor pins between GPIO, and several alternate functions. Selected PPORT groups also can be routed to the UPMUX (Universal Port Multiplexer) where additional Peripheral Circuits can be selected to go a variety of pins.

- [QSPI](#)

The QSPI module is a subset of the QSPI bus interface.

- [REMC2](#)

The REMC2 circuit generates infrared remote control output signals. This circuit can also be applicable to an EL lamp drive circuit by adding a simple external circuit.

- [RFC](#)

RFC is a R/F Converter module.

- [RTCA](#)

RTCA is a real-time clock with a perpetual calendar function.

- [SNDA](#)

SNDA is a sound generator that generates melodies and buzzer signals.

- [SPIA](#)

The SPIA module is a subset of the SPI bus interface.

- [SVD2](#)

SVD2 is a supply voltage detector to monitor the power supply voltage on the VDD pin or the voltage applied to an external pin.

- [T16](#)

T16 is a 16 bit timer.

- [T16B](#)

T16B is a 16-bit PWM timer with comparator or capture functions.

- [UART2](#)

The UART is an asynchronous serial interface.

- [WDT2](#)

WDT2 restarts the system if a problem occurs, such as when the program cannot be executed normally.

- [USB](#)

The USB 2.0 FS Device Controller is a USB target device controller that supports FS mode based on the USB 2.0 standard.

2.2 Common

All common definitions, data structures, and functions prototypes for the peripheral library.

Modules

- [Common_Constants](#)
- [Common_Macros](#)
- [Common_Functions](#)

2.3 Common_Constants

Enumerations

- enum `seState` {
`seDISABLE` = 0,
`seENABLE` = !`seDISABLE` }
- enum `seStatus` {
`seSTATUS_NG` = 0,
`seSTATUS_OK` = !`seSTATUS_NG` }
- enum `seInterruptStatus` {
`seINTERRUPT_NOT_OCCURRED` = 0,
`seINTERRUPT_OCCURRED` = !`seINTERRUPT_NOT_OCCURRED` }
- enum `seWriteProtect` {
`seWRITE_PROTECT_ON` = 0x00,
`seWRITE_PROTECT_OFF` = 0x96 }
- enum `seTimeoutMs` {
`seSHORT_WAIT_TIMEOUT_MS` = 100,
`seLONG_WAIT_TIMEOUT_MS` = 500 }

2.3.1 Enumeration Type Documentation

2.3.1.1 `seInterruptStatus`

enum `seInterruptStatus`

Enumerator

<code>seINTERRUPT_NOT_OCCURRED</code>	Interrupt did not occur.
<code>seINTERRUPT_OCCURRED</code>	Interrupt did occur.

2.3.1.2 `seState`

enum `seState`

Enumerator

<code>seDISABLE</code>	Module Disable state.
<code>seENABLE</code>	Module Enable state.

2.3.1.3 `seStatus`

enum `seStatus`

Enumerator

<code>seSTATUS_NG</code>	Represents state of failure (No Good).
<code>seSTATUS_OK</code>	Represents state of success (Ok).

2.3.1.4 seTimeoutMs

```
enum seTimeoutMs
```

Enumerator

seSHORT_WAIT_TIMEOUT_MS	Timeout in time critical-situations.
seLONG_WAIT_TIMEOUT_MS	Dead bit timeout.

2.3.1.5 seWriteProtect

```
enum seWriteProtect
```

Enumerator

seWRITE_PROTECT_ON	Write-protect on.
seWRITE_PROTECT_OFF	Write-Protect off.

2.4 Common_Macros

Macros

- #define **seASSERT**(expr) ((expr) ? ((void)0) : [seAssert](#)((uint8_t *)__FILE__, __LINE__))
- #define [seDBRUN](#) [seENABLE](#)
Default behavior is DBRUN=1 for both Debug and Release targets.
- #define **WAITLOOPFACTOR** 2220
- #define **WHILE**(CONDITION, STATUS) while(CONDITION);
- #define **SANITY_CHECK**(STATUS)
- #define **WAIT**(CONDITION, MS)

2.4.1 Macro Definition Documentation

2.4.1.1 WAIT

```
#define WAIT(  
    CONDITION,  
    MS )
```

Value:

```
{ \
    uint32_t Timer = 500/*ms*/ * WAITLOOPFACTOR/*~2220 for 1ms at 20MHz optimized code*/; \
    while(CONDITION) \
    { \
        if (--Timer == 0) \
        { \
            break; \
        } \
    } \
}
```


2.5 Common_Functions

Functions

- void [seAssert](#) (uint8_t *file, uint32_t line)
Prints filename and code line of the failed statement. Do not call directly, use the [seAssert\(\)](#) macro instead.
- void [seProtectSys](#) ([seWriteProtect](#) protect)
System protection.
- int16_t [seClamp16](#) (int16_t value, int16_t low, int16_t high)
Clamps the given 16-bit value between a low and high value.
- int32_t [seClamp32](#) (int32_t value, int32_t low, int32_t high)
Clamps the given 32-bit value between a low and high value.

2.5.1 Function Documentation

2.5.1.1 seAssert()

```
void seAssert (
    uint8_t * file,
    uint32_t line )
```

Parameters

<i>file</i>	Name of the source file.
<i>line</i>	Line number this function is being called on.

Return values

<i>None</i>	
-------------	--

2.5.1.2 seClamp16()

```
int16_t seClamp16 (
    int16_t value,
    int16_t low,
    int16_t high )
```

Parameters

<i>value</i>	The given value to be clamped
<i>low</i>	Minimum value
<i>high</i>	Maximum value

Return values

<i>clamped</i>	value
----------------	-------

2.5.1.3 seClamp32()

```
int32_t seClamp32 (
    int32_t value,
    int32_t low,
    int32_t high )
```

Parameters

<i>value</i>	The given value to be clamped
<i>low</i>	Minimum value
<i>high</i>	Maximum value

Return values

<i>clamped</i>	value
----------------	-------

2.5.1.4 seProtectSys()

```
void seProtectSys (
    seWriteProtect protect )
```

Parameters

<i>protect</i>	On or off value, see seWRITE_PROTECT_OFF
----------------	--

Return values

<i>None</i>	
-------------	--

2.6 CLG

CLG module is the clock generator that controls the clock sources and manages clock supply to the CPU and the peripheral circuits.

Modules

- [CLG_Constants](#)
- [CLG_Types](#)
- [CLG_Functions](#)

2.7 CLG_Constants

Macros

- #define `seCLG_FLGS(a)` ((`seCLG_IntFlag`)((a)))
Combination any of the `seCLG_IntFlag` enumerations.
- #define `seCLG_INTS(a)` ((`seCLG_Interrupt`)((a)))
Combination of any of the `seCLG_Interrupt` enumerations.

Enumerations

- enum `seCLG_ClkSrc` {
 `seCLG_OSC1` = 1,
 `seCLG_IOSC` = 0,
 `seCLG_OSC3` = 2,
 `seCLG_EXOSC` = 3 }
- enum `seCLG_IOSC_ClkDiv` {
 `seCLG_IOSC_CLKDIV_1` = 0,
 `seCLG_IOSC_CLKDIV_2` = 1,
 `seCLG_IOSC_CLKDIV_4` = 2,
 `seCLG_IOSC_CLKDIV_8` = 3 }
- enum `seCLG_OSC1_ClkDiv` {
 `seCLG_OSC1_CLKDIV_1` = 0,
 `seCLG_OSC1_CLKDIV_2` = 1 }
- enum `seCLG_OSC3_ClkDiv` {
 `seCLG_OSC3_CLKDIV_1` = 0,
 `seCLG_OSC3_CLKDIV_2` = 1,
 `seCLG_OSC3_CLKDIV_8` = 2,
 `seCLG_OSC3_CLKDIV_16` = 3 }
- enum `seCLG_EXOSC_ClkDiv` { `seCLG_EXOSC_CLKDIV_1` = 0 }
- enum `seCLG_CLG_ClkDiv` { `seCLG_CLG_CLKDIV_1` = 0 }
- enum `seCLG_IntFlag` {
 `seCLG_IOSCTERIF` = 0x0100U,
 `seCLG_OSC1TRMBSY` = 0x0080U,
 `seCLG_OSC1TEDIF` = 0x0040U,
 `seCLG_OSC1STPIF` = 0x0020U,
 `seCLG_IOSCTEDIF` = 0x0010U,
 `seCLG_OSC3STAIF` = 0x0004U,
 `seCLG_OSC1STAIF` = 0x0002U,
 `seCLG_IOSCSTAIF` = 0x0001U,
 `seCLG_ALLIF` }
- enum `seCLG_Interrupt` {
 `seCLG_IOSCTERIE` = 0x0100U,
 `seCLG_OSC1TEDIE` = 0x0040U,
 `seCLG_OSC1STPIE` = 0x0020U,
 `seCLG_IOSCTEDIE` = 0x0010U,
 `seCLG_OSC3STAIE` = 0x0004U,
 `seCLG_OSC1STAIE` = 0x0002U,
 `seCLG_IOSCSTAIE` = 0x0001U,
 `seCLG_ALLIE` }

```

• enum seCLG_IOSC_loscFq {
    seCLG_IOSC_IOSCFQ_1 = 0,
    seCLG_IOSC_IOSCFQ_2 = 1,
    seCLG_IOSC_IOSCFQ_8 = 4,
    seCLG_IOSC_IOSCFQ_12 = 5,
    seCLG_IOSC_IOSCFQ_16 = 6,
    seCLG_IOSC_IOSCFQ_20 = 7 }

```

2.7.1 Enumeration Type Documentation

2.7.1.1 seCLG_CLG_ClkDiv

```
enum seCLG_CLG_ClkDiv
```

Enumerator

seCLG_CLG_CLKDIV↔ _1	General division ratio of 1/1.
-------------------------	--------------------------------

2.7.1.2 seCLG_ClkSrc

```
enum seCLG_ClkSrc
```

Enumerator

seCLG_OSC1	OSC1 as a SYSCLK source.
seCLG_IOSC	IOSC as a SYSCLK source.
seCLG_OSC3	OSC3 as a SYSCLK source.
seCLG_EXOSC	EXOSC as a SYSCLK source.

2.7.1.3 seCLG_EXOSC_ClkDiv

```
enum seCLG_EXOSC_ClkDiv
```

Enumerator

seCLG_EXOSC_CLKDIV↔ _1	EXOSC division ratio of 1/1.
---------------------------	------------------------------

2.7.1.4 seCLG_Interrupt

```
enum seCLG_Interrupt
```

Enumerator

seCLG_IOSCTERIE	IOSC trimming error interrupt enable.
seCLG_OSC1TEDIE	OSC1 trimming end interrupt enable.

Enumerator

seCLG_OSC1STPIE	OSC1osc stop interrupt enable.
seCLG_IOSCTEDIE	IOSC trimming end interrupt enable.
seCLG_OSC3STAIE	OSC3 osc stable interrupt enable.
seCLG_OSC1STAIE	OSC1 osc stable interrupt enable.
seCLG_IOSCSTAIE	IOSC osc stable interrupt enable.
seCLG_ALLIE	All interrupts enable.

2.7.1.5 seCLG_IntFlag

```
enum seCLG_IntFlag
```

Enumerator

seCLG_IOSCTERIF	IOSC trimming error interrupt flag.
seCLG_OSC1TRMBSY	OSC1 theoretical regulation busy status.
seCLG_OSC1TEDIF	OSC1 theoretical regulation end interrupt flag.
seCLG_OSC1STPIF	OSC1osc stop interrupt flag.
seCLG_IOSCTEDIF	IOSC trimming end interrupt flag.
seCLG_OSC3STAIF	OSC3 osc stable interrupt flag.
seCLG_OSC1STAIF	OSC1 osc stable interrupt flag.
seCLG_IOSCSTAIF	IOSC osc stable interrupt flag.
seCLG_ALLIF	All interrupt flags.

2.7.1.6 seCLG_IOSC_ClkDiv

```
enum seCLG_IOSC_ClkDiv
```

Enumerator

seCLG_IOSC_CLKDIV↔ _1	IOSC division ratio of 1/1.
seCLG_IOSC_CLKDIV↔ _2	IOSC division ratio of 1/2.
seCLG_IOSC_CLKDIV↔ _4	IOSC division ratio of 1/4.
seCLG_IOSC_CLKDIV↔ _8	IOSC division ratio of 1/8.

2.7.1.7 seCLG_IOSC_IoscFq

```
enum seCLG_IOSC_IoscFq
```

Enumerator

seCLG_IOSC_IOSCFQ_1	IOSC frequency is 1 MHz.
seCLG_IOSC_IOSCFQ_2	IOSC frequency is 2 MHz.
seCLG_IOSC_IOSCFQ_8	IOSC frequency is 8 MHz.
seCLG_IOSC_IOSCFQ_12	IOSC frequency is 12 MHz.
seCLG_IOSC_IOSCFQ_16	IOSC frequency is 16 MHz.
seCLG_IOSC_IOSCFQ_20	IOSC frequency is 20 MHz.

2.7.1.8 seCLG_OSC1_ClkDiv

```
enum seCLG_OSC1_ClkDiv
```

Enumerator

seCLG_OSC1_CLKDIV↔ _1	OSC1 division ratio of 1/1.
seCLG_OSC1_CLKDIV↔ _2	OSC1 division ratio of 1/2.

2.7.1.9 seCLG_OSC3_ClkDiv

```
enum seCLG_OSC3_ClkDiv
```

Enumerator

seCLG_OSC3_CLKDIV_1	OSC3 division ratio of 1/1.
seCLG_OSC3_CLKDIV_2	OSC3 division ratio of 1/2.
seCLG_OSC3_CLKDIV_8	OSC3 division ratio of 1/8.
seCLG_OSC3_CLKDIV_16	OSC3 division ratio of 1/16.

2.8 CLG_Types

Data Structures

- union [seCLG_ClkDiv](#)
- struct [seCLG_InitTypeDef](#)
CLG Init structure definition.

2.9 CLG_Functions

Functions

- **seStatus seCLG_Init** (seCLG_InitTypeDef *InitStruct)
Initializes the CLG peripheral according to the specified parameters in the CLG_InitStruct.
- **seStatus seCLG_SwitchSysClkSrc** (seCLG_ClkSrc clock, seCLG_ClkDiv ClkDiv)
Selects a new clock source for the System clock.
- **uint16_t seCLG_GetSysClkSrc** (void)
Gets System clock source.
- **uint16_t seCLG_GetSysClkDiv** (void)
Gets System clock divider.
- **void seCLG_SetIoscFreqSel** (seCLG_IOSC_IoscFq freq)
Sets the IOSC frequency.
- **uint32_t seCLG_GetIoscFreqSel** (void)
Gets the IOSC frequency.
- **uint32_t seCLG_GetSysClk** (void)
Gets the SysClock frequency.
- **seStatus seCLG_Start** (seCLG_ClkSrc clock)
Starts Clock.
- **seStatus seCLG_Stop** (seCLG_ClkSrc clock)
Stops Clock.
- **seStatus seCLG_SetStopDetection** (seCLG_ClkSrc clock, seState StopDetectionEn)
Enables/Disables oscillator stop detection function.
- **seStatus seCLG_SetWkUpSysClk** (seCLG_ClkSrc WkUpSysClkSrc, seCLG_ClkDiv WkUpClkDiv, seState Sys↔ ClkSwitchOnWkUpEn)
Set wake-up SYSCLK source and divide.
- **seStatus seCLG_RunAutoTrimming** (seCLG_ClkSrc clock, seCLG_ClkSrc temp_clock)
The auto-trimming function adjusts the IOSCCLK clock frequency by trimming the clock with reference to the high precision OSC1CLK clock generated by the OSC1 oscillator circuit. As a side effect of this function SYSCLK is switched to different OSC.
- **seStatus seCLG_SetOperInSlp** (seCLG_ClkSrc clock, seState SlpEnable)
Sets Stop Clock mode used while in sleep.
- **seStatus seCLG_GetOperInSlp** (seCLG_ClkSrc clock)
Gets Stop Clock mode used while in sleep.
- **void seCLG_EnableInt** (seCLG_Interrupt irq)
Enable CLG interrupt(s).
- **void seCLG_DisableInt** (seCLG_Interrupt irq)
Disable CLG interrupt(s).
- **seInterruptStatus seCLG_GetIntFlag** (seCLG_IntFlag flag)
Returns CLG interrupt flag(s).
- **void seCLG_ClearIntFlag** (seCLG_IntFlag flag)
Clears CLG interrupt(s).
- **void CLG_IRQHandler** (void)
CLG Interrupt Service Routine.
- **seStatus ConfigurePortsForOSC3** (void)
Configures ports for this module. Override this function to configure specific ports.

2.9.1 Function Documentation

2.9.1.1 CLG_IRQHandler()

```
void CLG_IRQHandler (
    void )
```

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

2.9.1.2 ConfigurePortsForOSC3()

```
seStatus ConfigurePortsForOSC3 (
    void )
```

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.9.1.3 seCLG_ClearIntFlag()

```
void seCLG_ClearIntFlag (
    seCLG_IntFlag flag )
```

Parameters

<i>flag</i>	This parameter can be a value of seCLG_IntFlag .
-------------	--

Return values

<i>None</i>	
-------------	--

2.9.1.4 seCLG_DisableInt()

```
void seCLG_DisableInt (
    seCLG_Interrupt irq )
```

Parameters

<i>irq</i>	This parameter can be a value of seCLG_Interrupt .
------------	--

Return values

<i>None</i>	
-------------	--

2.9.1.5 seCLG_EnableInt()

```
void seCLG_EnableInt (
    seCLG_Interrupt irq )
```

Parameters

<i>irq</i>	This parameter can be a value of seCLG_Interrupt .
------------	--

Return values

<i>None</i>	
-------------	--

2.9.1.6 seCLG_GetIntFlag()

```
seInterruptStatus seCLG_GetIntFlag (
    seCLG_IntFlag flag )
```

Parameters

<i>flag</i>	This parameter can be a value of seCLG_IntFlag .
-------------	--

Return values

<i>InterruptStatus</i>	can be a value of seInterruptStatus
------------------------	---

2.9.1.7 seCLG_GetIoscFreqSel()

```
uint32_t seCLG_GetIoscFreqSel (
    void )
```

Return values

<i>Hz</i>	Actual value of IOSC frequency in Hz.
-----------	---------------------------------------

2.9.1.8 seCLG_GetOperInSlp()

```
seStatus seCLG_GetOperInSlp (
    seCLG_ClkSrc clock )
```

Parameters

<i>clock</i>	This parameter can be a value of seCLG_ClkSrc .
--------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.9.1.9 seCLG_GetSysClk()

```
uint32_t seCLG_GetSysClk (
    void )
```

Return values

<i>Hz</i>	Actual value of SysClock frequency in Hz.
-----------	---

2.9.1.10 seCLG_GetSysClkDiv()

```
uint16_t seCLG_GetSysClkDiv (
    void )
```

Return values

<i>Divider</i>	a value of seCLG_ClkDiv
----------------	---

2.9.1.11 seCLG_GetSysClkSrc()

```
uint16_t seCLG_GetSysClkSrc (
    void )
```

Return values

<i>Clock</i>	a value of seCLG_ClkSrc
--------------	---

2.9.1.12 seCLG_Init()

```
seStatus seCLG_Init (
    seCLG_InitTypeDef * InitStruct )
```

Parameters

<i>InitStruct</i>	Pointer to a seCLG_InitTypeDef structure that contains the configuration information for the specified CLG peripheral.
-------------------	--

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.9.1.13 seCLG_RunAutoTrimming()

```
seStatus seCLG_RunAutoTrimming (
    seCLG_ClkSrc clock,
    seCLG_ClkSrc temp_clock )
```

Note

Although the trimming time depends on the temperature, you will sacrifice some time when you use this function, likely quite a few ms. When IOSCCCLK is being used as the system clock or a peripheral circuit clock, do not use the auto-trimming function. OSC1 is started as side effect of this function.

Parameters

<i>clock</i>	Clock to do the procedure on, this parameter can be a value of seCLG_ClkSrc .
<i>temp_clock</i>	Temporary clock to run CPU while doing the procedure, can be a value of seCLG_ClkSrc .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.9.1.14 seCLG_SetIoscFreqSel()

```
void seCLG_SetIoscFreqSel (
    seCLG_IOSC_IoscFq freq )
```

Parameters

<i>freq</i>	New frequency can be a value of seCLG_IOSC_loscFq .
-------------	---

Return values

<i>None</i>	
-------------	--

2.9.1.15 seCLG_SetOperInSlp()

```
seStatus seCLG_SetOperInSlp (
    seCLG_ClkSrc clock,
    seState SlpEnable )
```

Parameters

<i>clock</i>	This parameter can be a value of seCLG_ClkSrc .
<i>SlpEnable</i>	When this parameter is seENABLE the clock stops in SLEEP mode. When this parameter is seDISABLE the clock runs in SLEEP mode.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.9.1.16 [seCLG_SetStopDetection\(\)](#)

```
seStatus seCLG_SetStopDetection (
    seCLG_ClkSrc clock,
    seState StopDetectionEn )
```

Note

The oscillation stop detection function restarts the OSC1 oscillator circuit when it detects the oscillation was stopped (due to under adverse environments, etc.).

Parameters

<i>clock</i>	This parameter can be a value of seCLG_ClkSrc .
<i>StopDetectionEn</i>	seState . This enables/disables oscillator stop detection function.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.9.1.17 [seCLG_SetWkUpSysClk\(\)](#)

```
seStatus seCLG_SetWkUpSysClk (
    seCLG_ClkSrc WkUpSysClkSrc,
    seCLG_ClkDiv WkUpClkDiv,
    seState SysClkSwitchOnWkUpEn )
```

Parameters

<i>WkUpSysClkSrc</i>	This parameter can be a value of seCLG_ClkSrc .
<i>WkUpClkDiv</i>	This parameter can be a value of seCLG_ClkDiv .
<i>SysClkSwitchOnWkUpEn</i>	seState When set to seDISABLE CPU, starts up with the same clock as one that used before SLEEP mode was entered. When set to seENABLE CPU, switches to a different wake-up clock that used before SLEEP mode was entered. The wake-up clock does not have to be enabled at this time. It will be enabled on the wake-up.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.9.1.18 seCLG_Start()

```
seStatus seCLG_Start (
    seCLG_ClkSrc clock )
```

Parameters

<i>clock</i>	The clock to start.
--------------	---------------------

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.9.1.19 seCLG_Stop()

```
seStatus seCLG_Stop (
    seCLG_ClkSrc clock )
```

Parameters

<i>clock</i>	The clock seCLG_ClkSrc to stop.
--------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.9.1.20 seCLG_SwitchSysClkSrc()

```
seStatus seCLG_SwitchSysClkSrc (
    seCLG_ClkSrc clock,
    seCLG_ClkDiv ClkDiv )
```

Parameters

<i>clock</i>	Clock source see seCLG_ClkSrc .
<i>ClkDiv</i>	Clock divide see seCLG_ClkDiv .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.10 DMAC

The Direct Memory Access (DMAC) module is designed to move data between peripherals and memory without core participation. It speeds up transfers and saves power. The channels have separate controls and flags. Each channel has one or more transfer descriptors stored in chip's memory. Descriptors contain details of the data transfers.

Modules

- [DMAC_Constants](#)
- [DMAC_Types](#)
- [DMAC_Functions](#)

2.11 DMAC_Constants

Macros

- #define **seDMAC_CNLS**(a) ((seDMAC_CHANNEL)((a)))
Combination of any of the above channels.
- #define **seDMAC_IDX**(chan) (((chan) == seDMAC_CH0)? 0 : ((chan) == seDMAC_CH1)? 1 : ((chan) == seDMAC_CH2)? 2 : 3)
Mapping index for a channel descriptor.
- #define **seDMAC_IntFlag** seDMAC_CHANNEL
- #define **seDMAC_NM_MAX** 0x3FF
- #define **seDMAC_RP_MAX** 0xF
- #define **seDMAC_cdata**(cc, nm, Rp, ss, ds, si, di) ((cc)&0x7) | ((nm)&seDMAC_NM_MAX)<<4 | ((Rp)&seDMAC_RP_MAX)<<14 | ((ss)&3)<<24 | ((si)&3)<<26 | ((ds)&3)<<28 | (uint32_t)((di)&3)<<30

Enumerations

- enum **seDMAC_CHANNEL** {
seDMAC_CH_NONE = 0U,
seDMAC_CH0 = 1U,
seDMAC_CH1 = 2U,
seDMAC_CH2 = 4U,
seDMAC_CH3 = 8U,
seDMAC_CH_ALL = 0xfU }
- enum **seDMAC_Inc** {
seDMAC_INC_1 = 0,
seDMAC_INC_2 = 1,
seDMAC_INC_4 = 2,
seDMAC_INC_NO = 3 }
- enum **seDMAC_Size** {
seDMAC_SIZE_BYTE = 0,
seDMAC_SIZE_HALF_WORD = 1,
seDMAC_SIZE_WORD = 2,
seDMAC_SIZE_RESERVED = 3 }
- enum **seDMAC_Mode** {
seDMAC_MODE_ALTERNT_PERIF_SCATTER = 7,
seDMAC_MODE_PRIMARY_PERIF_SCATTER = 6,
seDMAC_MODE_ALTERNT_MEM_SCATTER = 5,
seDMAC_MODE_PRIMARY_MEM_SCATTER = 4,
seDMAC_MODE_PING_PONG = 3,
seDMAC_MODE_AUTO_REQ = 2,
seDMAC_MODE_BASIC = 1,
seDMAC_MODE_STOP = 0 }
- enum **seDMAC_InterruptSrc** {
seDMAC_ERR_INT = 1,
seDMAC_TRANSF_COMPL = 2,
seDMAC_ALL_INT = 3 }

2.11.1 Enumeration Type Documentation

2.11.1.1 seDMAC_CHANNEL

```
enum seDMAC_CHANNEL
```

Enumerator

seDMAC_CH_NONE	Implemented channel 0.
seDMAC_CH0	Implemented channel 0.
seDMAC_CH1	Implemented channel 1.
seDMAC_CH2	Implemented channel 2.
seDMAC_CH3	Implemented channel 2.
seDMAC_CH_ALL	All implemented channels.

2.11.1.2 seDMAC_Inc

```
enum seDMAC_Inc
```

Enumerator

seDMAC_INC_1	Set the increment value to one.
seDMAC_INC_2	Set the increment value to two.
seDMAC_INC_4	Set the increment value to four.
seDMAC_INC_NO	Set the increment value to no increment.

2.11.1.3 seDMAC_InterruptSrc

```
enum seDMAC_InterruptSrc
```

Enumerator

seDMAC_ERR_INT	Interrupt cause is an error.
seDMAC_TRANSF_COMPL	Interrupt cause is transfer completion.
seDMAC_ALL_INT	All possible causes of interrupt.

2.11.1.4 seDMAC_Mode

```
enum seDMAC_Mode
```

Enumerator

seDMAC_MODE_ALTERNT_PERIF_SCATTER	Set transfer mode to Peripheral scatter-gather transfer (for alternate data structure)
seDMAC_MODE_PRIMARY_PERIF_SCATTER	Set transfer mode to Peripheral scatter-gather transfer (for primary data structure)
seDMAC_MODE_ALTERNT_MEM_SCATTER	Set transfer mode to Memory scatter-gather transfer (for alternate data structure)

Enumerator

seDMAC_MODE_PRIMARY_MEM_SCATTER	Set transfer mode to Memory scatter-gather transfer (for primary data structure)
seDMAC_MODE_PING_PONG	Set transfer mode to Ping-pong transfer.
seDMAC_MODE_AUTO_REQ	Set transfer mode to Auto-request transfer.
seDMAC_MODE_BASIC	Set transfer mode to Basic transfer.
seDMAC_MODE_STOP	Set transfer mode to Stop.

2.11.1.5 seDMAC_Size

enum `seDMAC_Size`

Enumerator

seDMAC_SIZE_BYTE	Set the size to one byte.
seDMAC_SIZE_HALF_WORD	Set the size to two bytes.
seDMAC_SIZE_WORD	Set the size to four bytes.
seDMAC_SIZE_RESERVED	Size value is reserved.

2.12 DMAC_Types

Data Structures

- union [seDMAC_CtrlData](#)
DMAC Control Data definition.
- struct [seDMAC_DataStruct](#)
DMAC Descriptors structure definition.

2.13 DMAC_Functions

Functions

- [seStatus seDMAC_Init](#) (uint32_t dma_data_struct_ptr, int chnls)
Initializes DMAC descriptors.
- [seStatus seDMAC_SetChannel](#) (uint32_t ctrl_data, uint32_t transf_src_end, uint32_t transf_dest_end, [seDMAC_CHANNEL](#) chan)
Initializes a DMAC channel.
- void [seDMAC_Start](#) ([seDMAC_CHANNEL](#) chan)
Starts DMAC channel.
- void [seDMAC_EnableInt](#) ([seDMAC_InterruptSrc](#) src, [seDMAC_IntFlag](#) flag)
Enables interrupt(s).
- void [seDMAC_DisableInt](#) ([seDMAC_InterruptSrc](#) src, [seDMAC_IntFlag](#) flag)
Disables interrupt(s).
- [seInterruptStatus seDMAC_GetIntFlag](#) ([seDMAC_InterruptSrc](#) src, [seDMAC_IntFlag](#) flag)
Returns the status of interrupt flag for selected interrupt sources. Use caution when using this function with a combination of interrupt sources.
- void [seDMAC_ClearIntFlag](#) ([seDMAC_InterruptSrc](#) src, [seDMAC_IntFlag](#) flag)
Clears DMAC interrupt(s).
- void [seDMAC_EnableRequestMask](#) ([seDMAC_CHANNEL](#) chnls)
Enables masking of the Peripheral devices.
- void [seDMAC_DisableRequestMask](#) ([seDMAC_CHANNEL](#) chnls)
Disables masking of the Peripheral devices.
- void [seDMAC_Enable](#) ([seDMAC_CHANNEL](#) chnls)
Enables DMAC channel.
- void [seDMAC_Disable](#) ([seDMAC_CHANNEL](#) chnls)
Disables DMAC channel.
- void [seDMAC_AlternateEnable](#) ([seDMAC_CHANNEL](#) chnls)
Enables DMAC's alternate descriptors.
- void [seDMAC_AlternateDisable](#) ([seDMAC_CHANNEL](#) chnls)
Enables DMAC's primary descriptors.
- void [seDMAC_PriorityIncrease](#) ([seDMAC_CHANNEL](#) chnls)
Causes DMAC channel priority increase.
- void [seDMAC_PriorityDecrease](#) ([seDMAC_CHANNEL](#) chnls)
Causes DMAC channel priority decrease.
- void [seDMAC_SetDataStrucPtr](#) (uint32_t dma_data_struct_ptr)
Sets DMAC channel's data structure pointer.
- uint32_t [seDMAC_GetDataStrucPtr](#) (void)
Returns DMAC channel's data structure pointer.
- uint32_t [seDMAC_GetAltDataStrucPtr](#) (void)
Returns DMAC channel's alternate data structure pointer.
- uint32_t [seDMAC_GetMode](#) ([seDMAC_CHANNEL](#) chan)
Gets DMAC mode from the valid data structure. Returns mode [seDMAC_MODE_STOP](#) if DMA structures are not initialized.
- uint32_t [seDMAC_GetNMinus1](#) ([seDMAC_CHANNEL](#) chan)
Gets DMAC Total number of units to transmit minus 1 from the data structure.

- volatile uint16_t * [seDMAC_ConfigPeriphToMem](#) (uint32_t periph, uint32_t daddress, uint32_t size_m1, [seDMAC_CHANNEL](#) chan)
For Peripheral-to-Memory DMAC transfer this function generates DMAC control data and sets it in the DMAC channel descriptor. Returns a register pointer further used to trigger DMAC transfer.
- volatile uint16_t * [seDMAC_ConfigMemToPeriph](#) (uint32_t saddress, uint32_t periph, uint32_t size_m1, [seDMAC_CHANNEL](#) chan)
For Memory-to-Peripheral DMAC transfer this function generates DMAC control data and sets it in the DMAC channel descriptor. Returns a register pointer further used to enable DMAC request.
- volatile uint16_t * [seDMAC_NonBlockTransfPeriphToMem](#) (uint32_t periph, uint32_t daddress, uint32_t transf-count, [seDMAC_CHANNEL](#) chan)
For Peripheral-to-Memory DMAC transfer this function configures a DMAC channel and generates a DMAC request. Returns a register pointer further used to disable DMAC request.
- volatile uint16_t * [seDMAC_NonBlockTransfMemToPeriph](#) (uint32_t saddress, uint32_t periph, uint32_t transf-count, [seDMAC_CHANNEL](#) chan)
For Memory-to-Peripheral DMAC transfer this function configures a DMAC channel and generates a DMAC request. Returns a register pointer further used to disable DMAC request.
- void [DMAC_IRQHandler](#) (void)
DMAC Interrupt Service Routine.

2.13.1 Function Documentation

2.13.1.1 DMAC_IRQHandler()

```
void DMAC_IRQHandler (
    void )
```

Return values

None	
------	--

2.13.1.2 seDMAC_AlternateDisable()

```
void seDMAC_AlternateDisable (
    seDMAC\_CHANNEL chnls )
```

Parameters

<i>chnls</i>	This parameter is a value of seDMAC_CHANNEL .
--------------	---

Return values

None	
------	--

2.13.1.3 seDMAC_AlternateEnable()

```
void seDMAC_AlternateEnable (
    seDMAC\_CHANNEL chnls )
```

Parameters

<i>chnls</i>	This parameter is a value of seDMAC_CHANNEL .
--------------	---

Return values

<i>None</i>	
-------------	--

2.13.1.4 seDMAC_ClearIntFlag()

```
void seDMAC_ClearIntFlag (
    seDMAC_InterruptSrc src,
    seDMAC_IntFlag flag )
```

Parameters

<i>src</i>	This parameter is a value of seDMAC_InterruptSrc .
<i>flag</i>	This parameter is a value of seDMAC_CHANNEL .

Return values

<i>None</i>	
-------------	--

2.13.1.5 seDMAC_ConfigMemToPeriph()

```
volatile uint16_t* seDMAC_ConfigMemToPeriph (
    uint32_t saddress,
    uint32_t periph,
    uint32_t size_m1,
    seDMAC_CHANNEL chan )
```

Parameters

<i>saddress</i>	Source address. The size of the transfer defined in the function based on the peripheral type.
<i>periph</i>	Pointer to a peripheral.
<i>size_m1</i>	Transfer size minus one.
<i>chan</i>	DMAC channel can be seDMAC_CHANNEL

Return values

<i>Ptr</i>	The register pointer further used to trigger DMAC transfer.
------------	---

2.13.1.6 seDMAC_ConfigPeriphToMem()

```
volatile uint16_t* seDMAC_ConfigPeriphToMem (
    uint32_t periph,
    uint32_t daddress,
    uint32_t size_m1,
    seDMAC_CHANNEL chan )
```

Parameters

<i>periph</i>	Pointer to a peripheral.
<i>daddress</i>	Destination address. The size of the transfer defined in the function based on the peripheral type.
<i>size_m1</i>	Transfer size minus one.
<i>chan</i>	DMAC channel can be seDMAC_CHANNEL

Return values

<i>Ptr</i>	The register pointer further used to trigger DMAC transfer.
------------	---

2.13.1.7 seDMAC_Disable()

```
void seDMAC_Disable (
    seDMAC_CHANNEL chnls )
```

Parameters

<i>chnls</i>	This parameter is a value of seDMAC_CHANNEL .
--------------	---

Return values

<i>None</i>	
-------------	--

2.13.1.8 seDMAC_DisableInt()

```
void seDMAC_DisableInt (
    seDMAC_InterruptSrc src,
    seDMAC_IntFlag flag )
```

Parameters

<i>src</i>	This parameter is a value of seDMAC_InterruptSrc .
<i>flag</i>	This parameter is a value of seDMAC_CHANNEL .

Return values

<i>None</i>	
-------------	--

2.13.1.9 seDMAC_DisableRequestMask()

```
void seDMAC_DisableRequestMask (
    seDMAC_CHANNEL chnls )
```

Parameters

<i>chnls</i>	This parameter is a value of seDMAC_CHANNEL .
--------------	---

Return values

<i>None</i>	
-------------	--

2.13.1.10 seDMAC_Enable()

```
void seDMAC_Enable (
    seDMAC_CHANNEL chnls )
```

Parameters

<i>chnls</i>	This parameter is a value of seDMAC_CHANNEL .
--------------	---

Return values

<i>None</i>	
-------------	--

2.13.1.11 seDMAC_EnableInt()

```
void seDMAC_EnableInt (
    seDMAC_InterruptSrc src,
    seDMAC_IntFlag flag )
```

Parameters

<i>src</i>	This parameter is a value of seDMAC_InterruptSrc .
<i>flag</i>	This parameter is a value of seDMAC_CHANNEL .

Return values

<i>None</i>	
-------------	--

2.13.1.12 seDMAC_EnableRequestMask()

```
void seDMAC_EnableRequestMask (
    seDMAC_CHANNEL chnls )
```

Parameters

<i>chnls</i>	This parameter is a value of seDMAC_CHANNEL .
--------------	---

Return values

<i>None</i>	
-------------	--

2.13.1.13 `seDMAC_GetAltDataStrucPtr()`

```
uint32_t seDMAC_GetAltDataStrucPtr (
    void )
```

Return values

<i>Ptr</i>	A valid aligned memory pointer.
------------	---------------------------------

2.13.1.14 `seDMAC_GetDataStrucPtr()`

```
uint32_t seDMAC_GetDataStrucPtr (
    void )
```

Return values

<i>Ptr</i>	A valid aligned memory pointer.
------------	---------------------------------

2.13.1.15 `seDMAC_GetIntFlag()`

```
seInterruptStatus seDMAC_GetIntFlag (
    seDMAC_InterruptSrc src,
    seDMAC_IntFlag flag )
```

Parameters

<i>src</i>	This parameter is a value of seDMAC_InterruptSrc .
<i>flag</i>	This parameter is a value of seDMAC_CHANNEL .

Return values

<i>InterruptStatus</i>	can be a value of seInterruptStatus .
------------------------	---

2.13.1.16 seDMAC_GetMode()

```
uint32_t seDMAC_GetMode (
    seDMAC_CHANNEL chan )
```

Parameters

<i>chan</i>	is a value of seDMAC_CHANNEL .
-------------	--

Return values

<i>Mode</i>	Value matching seDMAC_Mode
-------------	--

2.13.1.17 seDMAC_GetNMinus1()

```
uint32_t seDMAC_GetNMinus1 (
    seDMAC_CHANNEL chan )
```

Parameters

<i>chan</i>	is a value of seDMAC_CHANNEL .
-------------	--

Return values

<i>units</i>	Total number of units to transmit minus 1.
--------------	--

2.13.1.18 seDMAC_Init()

```
seStatus seDMAC_Init (
    uint32_t dma_data_struct_ptr,
    int chnls )
```

Parameters

<i>dma_data_struct_ptr</i>	Address of an allocated buffer for use for DMAC descriptors. Must be aligned.
<i>chnls</i>	This parameter declares the maximum number of requested channels.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.13.1.19 seDMAC_NonBlockTransfMemToPeriph()

```
volatile uint16_t* seDMAC_NonBlockTransfMemToPeriph (
    uint32_t saddress,
```

```
uint32_t periph,
uint32_t transfcoun,
seDMAC_CHANNEL chan )
```

Parameters

<i>saddress</i>	Source address. The size of the transfer defined in the function based on the peripheral type.
<i>periph</i>	Pointer to a peripheral.
<i>transfcoun</i>	Transfer size minus one.
<i>chan</i>	DMAC channel can be seDMAC_CHANNEL

Return values

<i>Ptr</i>	The register pointer further used to trigger DMAC transfer.
------------	---

2.13.1.20 seDMAC_NonBlockTransfPeriphToMem()

```
volatile uint16_t* seDMAC_NonBlockTransfPeriphToMem (
    uint32_t periph,
    uint32_t daddress,
    uint32_t transfcoun,
    seDMAC_CHANNEL chan )
```

Parameters

<i>periph</i>	Pointer to a peripheral.
<i>daddress</i>	Destination address. The size of the transfer defined in the function based on the peripheral type.
<i>transfcoun</i>	Transfer size minus one.
<i>chan</i>	DMAC channel can be seDMAC_CHANNEL

Return values

<i>Ptr</i>	The register pointer further used to trigger DMAC transfer.
------------	---

2.13.1.21 seDMAC_PriorityDecrease()

```
void seDMAC_PriorityDecrease (
    seDMAC_CHANNEL chnls )
```

Parameters

<i>chnls</i>	This parameter is a value of seDMAC_CHANNEL .
--------------	---

Return values

<i>None</i>	
-------------	--

2.13.1.22 seDMAC_PriorityIncrease()

```
void seDMAC_PriorityIncrease (
    seDMAC_CHANNEL chnls )
```

Parameters

<i>chnls</i>	This parameter is a value of seDMAC_CHANNEL .
--------------	---

Return values

<i>None</i>	
-------------	--

2.13.1.23 seDMAC_SetChannel()

```
seStatus seDMAC_SetChannel (
    uint32_t ctrl_data,
    uint32_t transf_src_end,
    uint32_t transf_dest_end,
    seDMAC_CHANNEL chan )
```

Note

It does not check if the channel was already active.

Parameters

<i>ctrl_data</i>	This parameter is a DMA channel control data 32-bit value.
<i>transf_src_end</i>	This parameter is a source end address.
<i>transf_dest_end</i>	This parameter is a destination end address.
<i>chan</i>	This parameter is a value of seDMAC_CHANNEL .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.13.1.24 seDMAC_SetDataStrucPtr()

```
void seDMAC_SetDataStrucPtr (
    uint32_t dma_data_struct_ptr )
```

Parameters

<i>dma_data_struct_ptr</i>	This parameter shall be a valid aligned memory pointer.
----------------------------	---

Return values

<i>None</i>	
-------------	--

2.13.1.25 seDMAC_Start()

```
void seDMAC_Start (
    seDMAC_CHANNEL chan )
```

Parameters

<i>chan</i>	This parameter is a value of seDMAC_CHANNEL .
-------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.14 I2C

The I2C module is a subset of the I2C bus interface.

Modules

- [I2C_Constants](#)
- [I2C_Types](#)
- [I2C_Functions](#)

2.15 I2C_Constants

Data Structures

- union [sel2C_ClkDiv](#)

Macros

- #define [sel2C_FLGS\(a\)](#) (([sel2C_IntFlag](#))((a)))
Combination of any of the [sel2C_IntFlag](#) enumerations.
- #define [sel2C_INTS\(a\)](#) (([sel2C_Interrupt](#))((a)))
Combination of any of the [sel2C_Interrupt](#) enumerations.

Typedefs

- typedef [seCLG_ClkSrc](#) **sel2C_ClkSrc**

Enumerations

- enum [sel2C_IOSC_ClkDiv](#) {
 [sel2C_IOSC_CLKDIV_1](#) = 0,
 [sel2C_IOSC_CLKDIV_2](#) = 1,
 [sel2C_IOSC_CLKDIV_4](#) = 2,
 [sel2C_IOSC_CLKDIV_8](#) = 3 }
- enum [sel2C_OSC1_ClkDiv](#) { [sel2C_OSC1_CLKDIV_1](#) = 0 }
- enum [sel2C_OSC3_ClkDiv](#) {
 [sel2C_OSC3_CLKDIV_1](#) = 0,
 [sel2C_OSC3_CLKDIV_2](#) = 1,
 [sel2C_OSC3_CLKDIV_4](#) = 2,
 [sel2C_OSC3_CLKDIV_8](#) = 3 }
- enum [sel2C_EXOSC_ClkDiv](#) { [sel2C_EXOSC_CLKDIV_1](#) = 0 }
- enum [sel2C_mode](#) {
 [sel2C_MODE_SLAVE](#) = 0,
 [sel2C_MODE_MASTER](#) = 1 }
- enum [sel2C_AddrMode](#) {
 [sel2C_7BIT_SLV_ADDR](#) = 0,
 [sel2C_10BIT_SLV_ADDR](#) = 1 }
- enum [sel2C_IntFlag](#) {
 [sel2C_SDALOW](#) = 0x1000U,
 [sel2C_SCLLOW](#) = 0x0800U,
 [sel2C_BSY](#) = 0x0400U,
 [sel2C_TR](#) = 0x0200U,
 [sel2C_BYTEENDIF](#) = 0x0080U,
 [sel2C_GCIF](#) = 0x0040U,
 [sel2C_NACKIF](#) = 0x0020U,
 [sel2C_STOPIF](#) = 0x0010U,
 [sel2C_STARTIF](#) = 0x0008U,
 [sel2C_ERRIF](#) = 0x0004U,
 [sel2C_RBFIF](#) = 0x0002U,
 [sel2C_TBEIF](#) = 0x0001U,
 [sel2C_ALL_IF](#) }


```

• enum sel2C_Interrupt {
    sel2C_BYTEENDIE = 0x0080U,
    sel2C_GCIE = 0x0040U,
    sel2C_NACKIE = 0x0020U,
    sel2C_STOPIE = 0x0010U,
    sel2C_STARTIE = 0x0008U,
    sel2C_ERRIE = 0x0004U,
    sel2C_RBFIE = 0x0002U,
    sel2C_TBEIE = 0x0001U,
    sel2C_ALL_IE }

```

2.15.1 Enumeration Type Documentation

2.15.1.1 sel2C_AddrMode

```
enum sel2C_AddrMode
```

Enumerator

sel2C_7BIT_SLV_ADDR	Specifies the I2C mode slave.
sel2C_10BIT_SLV_ADDR	Specifies the I2C mode master.

2.15.1.2 sel2C_EXOSC_ClkDiv

```
enum sel2C_EXOSC_ClkDiv
```

Enumerator

sel2C_EXOSC_CLKDIV↔ _1	EXOSC division ratio is 1/1.
---------------------------	------------------------------

2.15.1.3 sel2C_Interrupt

```
enum sel2C_Interrupt
```

Enumerator

sel2C_BYTEENDIE	End of transfer interrupt.
sel2C_GCIE	General call address reception interrupt.
sel2C_NACKIE	NACK reception interrupt.
sel2C_STOPIE	STOP condition interrupt.
sel2C_STARTIE	START condition interrupt.
sel2C_ERRIE	Error detection interrupt.
sel2C_RBFIE	Receive buffer full interrupt.
sel2C_TBEIE	Transmit buffer empty interrupt.

2.15.1.4 sel2C_IntFlag

```
enum sel2C_IntFlag
```

Enumerator

sel2C_SDALOW	SDA low flag.
sel2C_SCLLOW	SCL low flag.
sel2C_BSY	Busy flag.
sel2C_TR	Transmitter/Receiver flag.
sel2C_BYTEENDIF	End of transfer flag.
sel2C_GCIF	General call address reception flag.
sel2C_NACKIF	NACK reception flag.
sel2C_STOPIF	STOP condition flag.
sel2C_STARTIF	START condition flag.
sel2C_ERRIF	Error detection flag.
sel2C_RBFIF	Receive buffer full flag.
sel2C_TBEIF	Transmit buffer empty flag.

2.15.1.5 sel2C_IOSC_ClkDiv

```
enum sel2C_IOSC_ClkDiv
```

Enumerator

sel2C_IOSC_CLKDIV↔ _1	IOSC division ratio is 1/1.
sel2C_IOSC_CLKDIV↔ _2	IOSC division ratio is 1/2.
sel2C_IOSC_CLKDIV↔ _4	IOSC division ratio is 1/4.
sel2C_IOSC_CLKDIV↔ _8	IOSC division ratio is 1/8.

2.15.1.6 sel2C_mode

```
enum sel2C_mode
```

Enumerator

sel2C_MODE_SLAVE	Specifies the I2C mode slave.
sel2C_MODE_MASTER	Specifies the I2C mode master.

2.15.1.7 sel2C_OSC1_ClkDiv

```
enum sel2C_OSC1_ClkDiv
```

Enumerator

sel2C_OSC1_CLKDIV↔ _1	OSC1 division ratio is 1/1.
--------------------------	-----------------------------

2.15.1.8 sel2C_OSC3_ClkDiv

```
enum sel2C_OSC3_ClkDiv
```

Enumerator

sel2C_OSC3_CLKDIV↔ _1	OSC3 division ratio is 1/1.
sel2C_OSC3_CLKDIV↔ _2	OSC3 division ratio is 1/2.
sel2C_OSC3_CLKDIV↔ _4	OSC3 division ratio is 1/4.
sel2C_OSC3_CLKDIV↔ _8	OSC3 division ratio is 1/8.

2.16 I2C_Types

Data Structures

- struct [sel2C_InitTypeDef](#)
I2C Init structure definition.
- struct [sel2C_ChannelDef](#)
I2C Channel definition.

Variables

- [sel2C_ChannelDef](#) **I2C_CH0**
- [sel2C_ChannelDef](#) **I2C_CH1**

2.17 I2C_Functions

Functions

- void `selI2C_InitStructForMaster` (`selI2C_InitTypeDef` *`I2C_InitStruct`)
Fills each I2C_InitStruct member with its default value for Master mode.
- void `selI2C_InitStructForSlave` (`selI2C_InitTypeDef` *`I2C_InitStruct`)
Fills each I2C_InitStruct member with its default value for Slave mode.
- `seStatus` `selI2C_Init` (`selI2C_ChannelDef` *`I2CCHx`, `selI2C_InitTypeDef` *`I2C_InitStruct`)
Initializes the I2Cx peripheral according to the specified parameters in the I2C_InitStruct.
- `seStatus` `selI2C_Enable` (`I2C_0_Type` *`I2Cx`)
Enables the specified I2C channel.
- void `selI2C_Disable` (`I2C_0_Type` *`I2Cx`)
Disables the specified I2C channel.
- `seStatus` `selI2C_MstSendData` (`I2C_0_Type` *`I2Cx`, `uint16_t` address, `uint8_t` data[], `uint32_t` size, `uint32_t` stop←_pending)
Sends a data byte through the I2Cx peripheral if Master mode.
- `seStatus` `selI2C_MstReceiveData` (`I2C_0_Type` *`I2Cx`, `uint16_t` address, `uint8_t` data[], `uint32_t` size, `uint32_t` stop_pending)
Returns the most recent received data by the I2Cx peripheral if Master mode.
- `seStatus` `selI2C_SlvSendData` (`I2C_0_Type` *`I2Cx`, `uint8_t` data[], `uint32_t` size)
Sends a data byte through the I2Cx peripheral if Slave mode.
- `seStatus` `selI2C_SlvReceiveData` (`I2C_0_Type` *`I2Cx`, `uint8_t` data[], `uint32_t` size)
Returns the most recent received data by the I2Cx peripheral if Slave mode.
- `seStatus` `selI2C_Reset` (`I2C_0_Type` *`I2Cx`)
Resets the I2C channel.
- void `selI2C_EnableInt` (`I2C_0_Type` *`I2Cx`, `selI2C_Interrupt` irq)
Enables I2C channel interrupt.
- void `selI2C_DisableInt` (`I2C_0_Type` *`I2Cx`, `selI2C_Interrupt` irq)
Disables I2C channel interrupt.
- `selInterruptStatus` `selI2C_GetIntFlag` (`I2C_0_Type` *`I2Cx`, `selI2C_IntFlag` flag)
Gets I2C channel interrupt flag.
- void `selI2C_ClearIntFlag` (`I2C_0_Type` *`I2Cx`, `selI2C_IntFlag` flag)
Clears I2C channel interrupt flag.
- void `I2C_0_IRQHandler` (void)
I2C_0 Interrupt Service Routine.
- void `I2C_1_IRQHandler` (void)
I2C_CH1 Interrupt Service Routine.
- `seStatus` `ConfigurePortsForI2C` (`selI2C_ChannelDef` *`I2CCHx`)
Configures ports for the I2C module.

2.17.1 Function Documentation

2.17.1.1 ConfigurePortsForI2C()

```
seStatus ConfigurePortsForI2C (
    selI2C_ChannelDef * I2CCHx )
```

Parameters

<i>I2CCHx</i>	I2C channel definition of type seI2C_ChannelDef
---------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.17.1.2 I2C_0_IRQHandler()

```
void I2C_0_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.17.1.3 I2C_1_IRQHandler()

```
void I2C_1_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.17.1.4 seI2C_ClearIntFlag()

```
void seI2C_ClearIntFlag (
    I2C_0_Type * I2Cx,
    seI2C_IntFlag flag )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
<i>flag</i>	The interrupt flag, can be value of seI2C_IntFlag .

Return values

<i>None</i>	
-------------	--

2.17.1.5 seI2C_Disable()

```
void seI2C_Disable (
```

```
I2C_0_Type * I2Cx )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
-------------	-----------------

Return values

<i>None</i>	
-------------	--

2.17.1.6 selI2C_DisableInt()

```
void selI2C_DisableInt (
    I2C_0_Type * I2Cx,
    selI2C_Interrupt irq )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
<i>irq</i>	The interrupt to disable, can be value of selI2C_Interrupt .

Return values

<i>None</i>	
-------------	--

2.17.1.7 selI2C_Enable()

```
seStatus selI2C_Enable (
    I2C_0_Type * I2Cx )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
-------------	-----------------

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.17.1.8 selI2C_EnableInt()

```
void selI2C_EnableInt (
    I2C_0_Type * I2Cx,
    selI2C_Interrupt irq )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
<i>irq</i>	The interrupt to enable, can be value of selI2C_Interrupt .

Return values

<i>None</i>	
-------------	--

2.17.1.9 selI2C_GetIntFlag()

```
seInterruptStatus selI2C_GetIntFlag (
    I2C_0_Type * I2Cx,
    seI2C_IntFlag flag )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
<i>flag</i>	The interrupt flag, can be value of selI2C_IntFlag .

Return values

<i>InterruptStatus</i>	can be a value of seInterruptStatus
------------------------	---

2.17.1.10 selI2C_Init()

```
seStatus selI2C_Init (
    seI2C_ChannelDef * I2CCHx,
    seI2C_InitTypeDef * I2C_InitStruct )
```

Parameters

<i>I2CCHx</i>	I2C channel definition of type selI2C_ChannelDef
<i>I2C_InitStruct</i>	Pointer to a selI2C_InitTypeDef structure that contains the configuration information for the specified I2C peripheral.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.17.1.11 selI2C_InitStructForMaster()

```
void selI2C_InitStructForMaster (
    seI2C_InitTypeDef * I2C_InitStruct )
```


Parameters

<i>I2C_InitStruct</i>	Pointer to an seI2C_InitTypeDef structure which will be initialized.
-----------------------	--

Return values

<i>None</i>	
-------------	--

2.17.1.12 `seI2C_InitStructForSlave()`

```
void seI2C_InitStructForSlave (
    seI2C\_InitTypeDef * I2C_InitStruct )
```

Parameters

<i>I2C_InitStruct</i>	Pointer to an seI2C_InitTypeDef structure which will be initialized.
-----------------------	--

Return values

<i>None</i>	
-------------	--

2.17.1.13 `seI2C_MstReceiveData()`

```
seStatus seI2C_MstReceiveData (
    I2C_0_Type * I2Cx,
    uint16_t address,
    uint8_t data[],
    uint32_t size,
    uint32_t stop_pending )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
<i>address</i>	I2C addres, can be 10-bit address.
<i>data</i>	ptr to place recieved data.
<i>size</i>	Number of bytes in data to receive.
<i>stop_pending</i>	if zero - send stop condition, if non zero - stop is pending

Return values

<i>Status</i>	can be a value of seStatus received data is store in <i>data</i> [].
---------------	--

2.17.1.14 selI2C_MstSendData()

```
seStatus selI2C_MstSendData (
    I2C_0_Type * I2Cx,
    uint16_t address,
    uint8_t data[],
    uint32_t size,
    uint32_t stop_pending )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
<i>address</i>	I2C addres, can be 10-bit address.
<i>data</i>	Data to be transmitted.
<i>size</i>	Number of bytes in data to send.
<i>stop_pending</i>	if zero - send stop condition, if non zero - stop pending.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.17.1.15 selI2C_Reset()

```
seStatus selI2C_Reset (
    I2C_0_Type * I2Cx )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
-------------	-----------------

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.17.1.16 selI2C_SlvReceiveData()

```
seStatus selI2C_SlvReceiveData (
    I2C_0_Type * I2Cx,
    uint8_t data[],
    uint32_t size )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
<i>data</i>	ptr to be place received data.
<i>size</i>	Number of bytes in data to receive.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.17.1.17 selI2C_SlvSendData()

```
seStatus selI2C_SlvSendData (
    I2C_0_Type * I2Cx,
    uint8_t data[],
    uint32_t size )
```

Parameters

<i>I2Cx</i>	I2C peripheral.
<i>data</i>	Data to be transmitted.
<i>size</i>	Number of bytes in data to send.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.18 LCD32B

The LCD32B is a module to drive an LCD panel.

Modules

- [LCD32B_Constants](#)
- [LCD32B_Types](#)
- [LCD32B_Functions](#)

2.19 LCD32B_Constants

Modules

- [LCD32B_Clock](#)
- [LCD32B_DisplayState](#)
- [LCD32B_Frame_Frequency_Control](#)
- [LCD32B_Power_Settings](#)
- [LCD32B_Contrast](#)
- [LCD32B_RAM_Area](#)

2.20 LCD32B_Clock

Data Structures

- union [seLCD32B_ClkDiv](#)

Typedefs

- typedef [seCLG_ClkSrc](#) [seLCD32B_ClkSrc](#)
LCD32B Clock Source selection.

Enumerations

- enum [seLCD32B_IOSC_ClkDiv](#) {
[seLCD32B_IOSC_CLKDIV_16](#) = 0,
[seLCD32B_IOSC_CLKDIV_32](#) = 1,
[seLCD32B_IOSC_CLKDIV_64](#) = 2,
[seLCD32B_IOSC_CLKDIV_128](#) = 3,
[seLCD32B_IOSC_CLKDIV_256](#) = 4,
[seLCD32B_IOSC_CLKDIV_512](#) = 5 }
LCD32B Clock source divider options.
- enum [seLCD32B_OSC1_ClkDiv](#) { [seLCD32B_OSC1_CLKDIV_1](#) = 0 }
- enum [seLCD32B_OSC3_ClkDiv](#) {
[seLCD32B_OSC3_CLKDIV_16](#) = 0,
[seLCD32B_OSC3_CLKDIV_32](#) = 1,
[seLCD32B_OSC3_CLKDIV_64](#) = 2,
[seLCD32B_OSC3_CLKDIV_128](#) = 3,
[seLCD32B_OSC3_CLKDIV_256](#) = 4,
[seLCD32B_OSC3_CLKDIV_512](#) = 5 }
- enum [seLCD32B_EXOSC_ClkDiv](#) { [seLCD32B_EXOSC_CLKDIV_1](#) = 0 }
- enum [seLCD32B_BoostClk](#) {
[seLCD32B_FLCLK_DIV_64](#) = 0,
[seLCD32B_FLCLK_DIV_32](#) = 1,
[seLCD32B_FLCLK_DIV_16](#) = 2,
[seLCD32B_FLCLK_DIV_4](#) = 3 }
LCD32B Booster clock control options.

2.20.1 Enumeration Type Documentation

2.20.1.1 [seLCD32B_BoostClk](#)

enum [seLCD32B_BoostClk](#)

Note

fCLK_LCD32B: LCD32B operating clock frequency [Hz]

Enumerator

seLCD32B_FLCLK_DIV_64	fCLK 1/64.
---------------------------------------	------------

Enumerator

seLCD32B_FLCLK_DIV_32	fLCLK 1/32.
seLCD32B_FLCLK_DIV_16	fLCLK 1/16.
seLCD32B_FLCLK_DIV_4	fLCLK 1/4.

2.20.1.2 seLCD32B_EXOSC_ClkDiv

```
enum seLCD32B_EXOSC_ClkDiv
```

Enumerator

seLCD32B_EXOSC_CLKDIV↔ _1	EXOSC division ratio is 1/1.
------------------------------	------------------------------

2.20.1.3 seLCD32B_IOSC_ClkDiv

```
enum seLCD32B_IOSC_ClkDiv
```

Enumerator

seLCD32B_IOSC_CLKDIV_16	IOSC division ratio is 1/16.
seLCD32B_IOSC_CLKDIV_32	IOSC division ratio is 1/32.
seLCD32B_IOSC_CLKDIV_64	IOSC division ratio is 1/64.
seLCD32B_IOSC_CLKDIV_128	IOSC division ratio is 1/128.
seLCD32B_IOSC_CLKDIV_256	IOSC division ratio is 1/256.
seLCD32B_IOSC_CLKDIV_512	IOSC division ratio is 1/512.

2.20.1.4 seLCD32B_OSC1_ClkDiv

```
enum seLCD32B_OSC1_ClkDiv
```

Enumerator

seLCD32B_OSC1_CLKDIV↔ _1	OSC1 division ratio is 1/1.
-----------------------------	-----------------------------

2.20.1.5 seLCD32B_OSC3_ClkDiv

```
enum seLCD32B_OSC3_ClkDiv
```

Enumerator

seLCD32B_OSC3_CLKDIV_16	OSC3 division ratio is 1/4.
seLCD32B_OSC3_CLKDIV_32	OSC3 division ratio is 1/8.

Enumerator

seLCD32B_OSC3_CLKDIV_64	OSC3 division ratio is 1/16.
seLCD32B_OSC3_CLKDIV_128	OSC3 division ratio is 1/32.
seLCD32B_OSC3_CLKDIV_256	OSC3 division ratio is 1/64.
seLCD32B_OSC3_CLKDIV_512	OSC3 division ratio is 1/128.

2.21 LCD32B_DisplayState

Enumerations

- enum `seLCD32B_DisplInvState` {
`seLCD32B_DISP_INVERSE` = 0,
`seLCD32B_DISP_NORMAL` = 1 }
LCD32B Display inversion.
- enum `seLCD32B_DisplState` {
`seLCD32B_DISP_STATE_ALL_OFF` = 3,
`seLCD32B_DISP_STATE_ALL_ON` = 2,
`seLCD32B_DISP_STATE_NORMAL` = 1,
`seLCD32B_DISP_STATE_DISP_OFF` = 0 }
LCD32B Display state settings.

The LCD panel display can be inverted without re-writing data RAM.

2.21.1 Enumeration Type Documentation

2.21.1.1 `seLCD32B_DisplInvState`

```
enum seLCD32B_DisplInvState
```

Enumerator

<code>seLCD32B_DISP_INVERSE</code>	Display State is Inverted.
<code>seLCD32B_DISP_NORMAL</code>	Display State is Normal.

2.21.1.2 `seLCD32B_DisplState`

```
enum seLCD32B_DisplState
```

Enumerator

<code>seLCD32B_DISP_STATE_ALL_OFF</code>	Display State is All off (Static Drive).
<code>seLCD32B_DISP_STATE_ALL_ON</code>	Display State is All on.
<code>seLCD32B_DISP_STATE_NORMAL</code>	Display State is Normal.
<code>seLCD32B_DISP_STATE_DISP_OFF</code>	Display State is Display off.

2.22 LCD32B_Frame_Frequency_Control

Macros

- #define `IS_LCD32B_LDUTY`(LDUTY) (!((LDUTY) & ~LCD32B_TIM1_LDUTY_Msk))
Lcd32b Duty parameter is within limits.
- #define `IS_LCD32B_FRMCNT`(FRMCNT) (!((FRMCNT) & ~LCD32B_TIM1_FRMCNT_Msk))
Lcd32b Frame Count is within limits.

Enumerations

- enum `seLCD32B_Duty` {
`seLCD32B_DUTY_STATIC` = 0,
`seLCD32B_DUTY_1_2` = 1,
`seLCD32B_DUTY_1_3` = 2,
`seLCD32B_DUTY_1_4` = 3,
`seLCD32B_DUTY_1_5` = 4,
`seLCD32B_DUTY_1_6` = 5,
`seLCD32B_DUTY_1_7` = 6,
`seLCD32B_DUTY_1_8` = 7,
`seLCD32B_DUTY_1_9` = 8,
`seLCD32B_DUTY_1_10` = 9,
`seLCD32B_DUTY_1_11` = 10,
`seLCD32B_DUTY_1_12` = 11,
`seLCD32B_DUTY_1_13` = 12,
`seLCD32B_DUTY_1_14` = 13,
`seLCD32B_DUTY_1_15` = 14,
`seLCD32B_DUTY_1_16` = 15,
`seLCD32B_DUTY_1_17` = 16,
`seLCD32B_DUTY_1_18` = 17,
`seLCD32B_DUTY_1_19` = 18,
`seLCD32B_DUTY_1_20` = 19,
`seLCD32B_DUTY_1_21` = 20,
`seLCD32B_DUTY_1_22` = 21,
`seLCD32B_DUTY_1_23` = 22,
`seLCD32B_DUTY_1_24` = 23,
`seLCD32B_DUTY_1_25` = 24,
`seLCD32B_DUTY_1_26` = 25,
`seLCD32B_DUTY_1_27` = 26,
`seLCD32B_DUTY_1_28` = 27,
`seLCD32B_DUTY_1_29` = 28,
`seLCD32B_DUTY_1_30` = 29,
`seLCD32B_DUTY_1_31` = 30,
`seLCD32B_DUTY_1_32` = 31 }

The frame frequency is determined by selecting a division ratio from 32 variations depending on the drive duty.

2.22.1 Enumeration Type Documentation

2.22.1.1 seLCD32B_Duty

enum `seLCD32B_Duty`

The following equation is to calculate the frame frequency.

$$fFR = 1/8 * fLCLK * 1/(FRMCNT + 1) * 1/(LDUTY + 1)$$

Where

fFR: Frame frequency [Hz]

fLCLK: LCD32B operating clock frequency [Hz]

FRMCNT: value (0 to 31)

LDUTY: value [seLCD32B_Duty](#)

Note

To achieve a target frame rate the following formula can be used to calculate a value of FRMCNT based on selected fLCLK and LDUTY.

$$FRMCNT = (1/8) * fLCLK * 1/(LDUTY + 1) * (1/FR) - 1.$$

Enumerator

seLCD32B_DUTY_STATIC	Static duty.
seLCD32B_DUTY_1_2	1/2 duty
seLCD32B_DUTY_1_3	1/3 duty
seLCD32B_DUTY_1_4	1/4 duty
seLCD32B_DUTY_1_5	1/5 duty
seLCD32B_DUTY_1_6	1/6 duty
seLCD32B_DUTY_1_7	1/7 duty
seLCD32B_DUTY_1_8	1/8 duty
seLCD32B_DUTY_1_9	1/9 duty
seLCD32B_DUTY_1_10	1/10 duty
seLCD32B_DUTY_1_11	1/11 duty
seLCD32B_DUTY_1_12	1/12 duty
seLCD32B_DUTY_1_13	1/13 duty
seLCD32B_DUTY_1_14	1/14 duty
seLCD32B_DUTY_1_15	1/15 duty
seLCD32B_DUTY_1_16	1/16 duty
seLCD32B_DUTY_1_17	1/17 duty
seLCD32B_DUTY_1_18	1/18 duty
seLCD32B_DUTY_1_19	1/19 duty
seLCD32B_DUTY_1_20	1/20 duty
seLCD32B_DUTY_1_21	1/21 duty
seLCD32B_DUTY_1_22	1/22 duty
seLCD32B_DUTY_1_23	1/23 duty
seLCD32B_DUTY_1_24	1/24 duty
seLCD32B_DUTY_1_25	1/25 duty
seLCD32B_DUTY_1_26	1/26 duty
seLCD32B_DUTY_1_27	1/27 duty
seLCD32B_DUTY_1_28	1/28 duty
seLCD32B_DUTY_1_29	1/29 duty

Enumerator

seLCD32B_DUTY_1_30	1/30 duty
seLCD32B_DUTY_1_31	1/31 duty
seLCD32B_DUTY_1_32	1/32 duty

2.23 LCD32B_Power_Settings

Enumerations

- enum `seLCD32B_PwrBias` {
`seLCD32B_PWR_BIAS_1_4` = 1,
`seLCD32B_PWR_BIAS_1_5` = 0 }

LCD32B drive bias.

When using internal generation mode, select the reference voltage for boosting voltage generated by the LCD voltage regulator according to the power supply voltage.

Heavy load protection mode should be set when the display has inconsistencies in density. Current consumption increases in heavy load protection mode, therefore do not set heavy load protection mode if unnecessary.

2.23.1 Enumeration Type Documentation

2.23.1.1 seLCD32B_PwrBias

enum `seLCD32B_PwrBias`

Enumerator

<code>seLCD32B_PWR_BIAS_1↔ _4</code>	Bias 1/4.
<code>seLCD32B_PWR_BIAS_1↔ _5</code>	Bias 1/5.

2.24 LCD32B_Contrast

Enumerations

- enum `seLCD32B_Contrast` {
`seLCD32B_CONTRAST_LEVEL0` = 0,
`seLCD32B_CONTRAST_LEVEL1` = 1,
`seLCD32B_CONTRAST_LEVEL2` = 2,
`seLCD32B_CONTRAST_LEVEL3` = 3,
`seLCD32B_CONTRAST_LEVEL4` = 4,
`seLCD32B_CONTRAST_LEVEL5` = 5,
`seLCD32B_CONTRAST_LEVEL6` = 6,
`seLCD32B_CONTRAST_LEVEL7` = 7,
`seLCD32B_CONTRAST_LEVEL8` = 8,
`seLCD32B_CONTRAST_LEVEL9` = 9,
`seLCD32B_CONTRAST_LEVEL10` = 10,
`seLCD32B_CONTRAST_LEVEL11` = 11,
`seLCD32B_CONTRAST_LEVEL12` = 12,
`seLCD32B_CONTRAST_LEVEL13` = 13,
`seLCD32B_CONTRAST_LEVEL14` = 14,
`seLCD32B_CONTRAST_LEVEL15` = 15 }

Sets LCD panel contrast.

2.24.1 Enumeration Type Documentation

2.24.1.1 seLCD32B_Contrast

enum `seLCD32B_Contrast`

Enumerator

<code>seLCD32B_CONTRAST_LEVEL0</code>	Contrast Level 0 (Light)
<code>seLCD32B_CONTRAST_LEVEL1</code>	Contrast Level 1.
<code>seLCD32B_CONTRAST_LEVEL2</code>	Contrast Level 2.
<code>seLCD32B_CONTRAST_LEVEL3</code>	Contrast Level 3.
<code>seLCD32B_CONTRAST_LEVEL4</code>	Contrast Level 4.
<code>seLCD32B_CONTRAST_LEVEL5</code>	Contrast Level 5.
<code>seLCD32B_CONTRAST_LEVEL6</code>	Contrast Level 6.
<code>seLCD32B_CONTRAST_LEVEL7</code>	Contrast Level 7.
<code>seLCD32B_CONTRAST_LEVEL8</code>	Contrast Level 8 (Normal)
<code>seLCD32B_CONTRAST_LEVEL9</code>	Contrast Level 9.
<code>seLCD32B_CONTRAST_LEVEL10</code>	Contrast Level 10.
<code>seLCD32B_CONTRAST_LEVEL11</code>	Contrast Level 11.
<code>seLCD32B_CONTRAST_LEVEL12</code>	Contrast Level 12.
<code>seLCD32B_CONTRAST_LEVEL13</code>	Contrast Level 13.
<code>seLCD32B_CONTRAST_LEVEL14</code>	Contrast Level 14.
<code>seLCD32B_CONTRAST_LEVEL15</code>	Contrast Level 15 (Dark)

2.25 LCD32B_RAM_Area

Enumerations

- enum `seLCD32B_Area` {
`seLCD32B_DISPLAY_AREA0` = 0,
`seLCD32B_DISPLAY_AREA1` = 1 }
- enum `seLCD32B_SegRamAddress` {
`seLCD32B_SEGRAM_TOP_ADDR_AREA0` = 0x20200000,
`seLCD32B_SEGRAM_TOP_ADDR_AREA1` = 0x20200200,
`seLCD32B_SEGRAM_SIZE` = 0x160 }

The correspondence between the memory bits of the display data RAM and the common/segment pins varies depending on the selected conditions below.

Drive duty (1/32 to 1/2 or static drive)
 Segment pin assignment (normal or inverse)
 Common pin assignment (normal or inverse)

The Hardware specification shows the correspondence between display data RAM and the common/segment pins in some drive duties.

Writing 1 to the display data RAM bit corresponding to a segment on the LCD panel turns the segment on, while writing 0 turns the segment off. Since the display memory is RAM, allowing reading and writing, bits can be controlled individually using logic operation instructions (read-modify-write instructions).

2.25.1 Enumeration Type Documentation

2.25.1.1 `seLCD32B_Area`

```
enum seLCD32B_Area
```

Enumerator

<code>seLCD32B_DISPLAY_AREA0</code>	Area : Display Area 0.
<code>seLCD32B_DISPLAY_AREA1</code>	Area : Display Area 1.

2.25.1.2 `seLCD32B_SegRamAddress`

```
enum seLCD32B_SegRamAddress
```

Enumerator

<code>seLCD32B_SEGRAM_TOP_ADDR_AREA0</code>	The display data RAM address(Area0).
<code>seLCD32B_SEGRAM_TOP_ADDR_AREA1</code>	The display data RAM address(Area1).
<code>seLCD32B_SEGRAM_SIZE</code>	The display data RAM size.

2.26 LCD32B_Types

Data Structures

- struct [seLCD32B_InitTypeDef](#)
LCD32B Init structure definition.

2.27 LCD32B_Functions

Functions

- `seStatus seLCD32B_Init (seLCD32B_InitTypeDef *InitStruct)`
Initializes the LCD32B peripheral according to the specified parameters in the LCD32B_InitStruct.
- `void seLCD32B_Enable (seState IoDischarge)`
Enables LCD32B by start supplying operating clock.
- `void seLCD32B_Disable (void)`
Disables LCD32B by stop supplying operating clock.
- `void seLCD32B_ConfigureClock (seLCD32B_ClkSrc ClkSrc, uint16_t ClkDivider)`
Configures LCD32B clock source and clock divider.
- `void seLCD32B_SetFrameFreq (seLCD32B_InitTypeDef *InitStruct)`
Sets frame rate by selecting the appropriate FRMCNT and LDUTY values. Refer to [LCD32B_Frame_Frequency_Control](#) for details.
- `void seLCD32B_ConfigurePower (seLCD32B_InitTypeDef *InitStruct)`
Sets Voltages.
- `void seLCD32B_ConfigureDisplay (seLCD32B_InitTypeDef *InitStruct)`
Configures Display features.
- `void seLCD32B_SetBoostClk (seLCD32B_BoostClk clk)`
Configures Display features.
- `void seLCD32B_SetPanelContrast (seLCD32B_Contrast contrast)`
Sets Panel contrast.
- `uint16_t seLCD32B_GetPanelContrast (void)`
Gets Panel contrast.
- `void seLCD32B_SetDisplayState (seLCD32B_DispState state)`
Sets Display state.
- `uint16_t seLCD32B_GetDisplayState (void)`
Gets Display state.
- `void seLCD32B_SetDisplayInvState (seLCD32B_DisplInvState state)`
Sets Display Inverted state.
- `uint16_t seLCD32B_GetDisplayInvState (void)`
Gets Display Inverted state.
- `void seLCD32B_SetDisplayArea (seLCD32B_Area area)`
Sets Display Area.
- `uint16_t seLCD32B_GetDisplayArea (void)`
Gets Display Area.
- `void seLCD32B_ClearDisplayMemory (seLCD32B_Area area)`
Clears Display Area.
- `void seLCD32B_SetDisplayMemory (seLCD32B_Area area, const uint8_t byte)`
Sets Display Memory to a value.
- `void seLCD32B_CopyToDisplayArea (seLCD32B_Area area, const uint8_t *buf, uint16_t size)`
Copies user data to Display Area.
- `void LCD32B_IRQHandler (void)`
LCD32B Interrupt Service Routine.
- `void seLCD32B_EnableInt (void)`
Enables Lcd interrupt.

- void [seLCD32B_DisableInt](#) (void)
Disables Lcd interrupt.
- [seInterruptStatus seLCD32B_GetIntFlag](#) (void)
Returns Lcd interrupt flag.
- void [seLCD32B_ClearIntFlag](#) (void)
Clears Lcd interrupt.
- [seStatus ConfigurePortsForLcd32B](#) (void)
Configures ports for this module. Override this function to configure specific ports.

2.27.1 Function Documentation

2.27.1.1 ConfigurePortsForLcd32B()

```
seStatus ConfigurePortsForLcd32B (
    void )
```

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.27.1.2 LCD32B_IRQHandler()

```
void LCD32B_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.27.1.3 seLCD32B_ClearDisplayMemory()

```
void seLCD32B_ClearDisplayMemory (
    seLCD32B_Area area )
```

Parameters

<i>area</i>	This parameter can be a value of seLCD32B_Area .
-------------	--

Return values

<i>None</i>	
-------------	--

2.27.1.4 seLCD32B_ClearIntFlag()

```
void seLCD32B_ClearIntFlag (
```

```
void )
```

Return values

<i>None</i>	
-------------	--

2.27.1.5 seLCD32B_ConfigureClock()

```
void seLCD32B_ConfigureClock (
    seLCD32B_ClkSrc ClkSrc,
    uint16_t ClkDivider )
```

Parameters

<i>ClkSrc</i>	Clock source can be a value of seLCD32B_ClkSrc .
<i>ClkDivider</i>	Clock divider a value of seLCD32B_EXOSC_ClkDiv or seLCD32B_IOSC_ClkDiv or ref seLCD32B_OSC3_ClkDiv or seLCD32B_OSC1_ClkDiv depending on the selected clock source.

Return values

<i>None</i>	
-------------	--

2.27.1.6 seLCD32B_ConfigureDisplay()

```
void seLCD32B_ConfigureDisplay (
    seLCD32B_InitTypeDef * InitStruct )
```

Parameters

<i>InitStruct</i>	This parameter can be a value of seLCD32B_InitTypeDef .
-------------------	---

Return values

<i>None</i>	
-------------	--

2.27.1.7 seLCD32B_ConfigurePower()

```
void seLCD32B_ConfigurePower (
    seLCD32B_InitTypeDef * InitStruct )
```

Parameters

<i>InitStruct</i>	This parameter can be a value of seLCD32B_InitTypeDef .
-------------------	---

Return values

<i>None</i>	
-------------	--

2.27.1.8 seLCD32B_CopyToDisplayArea()

```
void seLCD32B_CopyToDisplayArea (
    seLCD32B_Area area,
    const uint8_t * buf,
    uint16_t size )
```

Parameters

<i>area</i>	This parameter can be a value of seLCD32B_Area .
<i>buf</i>	This parameter is the user data pointer.
<i>size</i>	This parameter is the user data size.

Return values

<i>None</i>	
-------------	--

2.27.1.9 seLCD32B_Disable()

```
void seLCD32B_Disable (
    void )
```

Return values

<i>None</i>	
-------------	--

2.27.1.10 seLCD32B_DisableInt()

```
void seLCD32B_DisableInt (
    void )
```

Return values

<i>None</i>	
-------------	--

2.27.1.11 seLCD32B_Enable()

```
void seLCD32B_Enable (
    seState IoDischarge )
```

Parameters

<i>IoDischarge</i>	Controls if LCDIO discharge seState .
--------------------	---

Return values

<i>None</i>	
-------------	--

2.27.1.12 seLCD32B_EnableInt()

```
void seLCD32B_EnableInt (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.27.1.13 seLCD32B_GetDisplayArea()

```
uint16_t seLCD32B_GetDisplayArea (  
    void )
```

Return values

<i>area</i>	A value of seLCD32B_DispSArea.
-------------	--------------------------------

2.27.1.14 seLCD32B_GetDisplayInvState()

```
uint16_t seLCD32B_GetDisplayInvState (  
    void )
```

Return values

<i>InvState</i>	A value of seLCD32B_DisplnvState .
-----------------	--

2.27.1.15 seLCD32B_GetDisplayState()

```
uint16_t seLCD32B_GetDisplayState (  
    void )
```

Return values

<i>A</i>	value of seLCD32B_DispState .
----------	---

2.27.1.16 seLCD32B_GetIntFlag()

```
seInterruptStatus seLCD32B_GetIntFlag (
    void )
```

Return values

<i>InterruptStatus</i>	can be a value of seInterruptStatus
------------------------	---

2.27.1.17 seLCD32B_GetPanelContrast()

```
uint16_t seLCD32B_GetPanelContrast (
    void )
```

Return values

<i>Contrast</i>	A value of seLCD32B_Contrast .
-----------------	--

2.27.1.18 seLCD32B_Init()

```
seStatus seLCD32B_Init (
    seLCD32B_InitTypeDef * InitStruct )
```

Note

This function configures the module, and module's interrupts. It clears module's interrupts but does not enable interrupt from the module to the CPU. This function enables module by start supplying operating clock. This module starts does not start display. It is user code responsibility to set appropriate display state.

Parameters

<i>InitStruct</i>	Pointer to a seLCD32B_InitTypeDef structure that contains the configuration information for the LCD32B peripheral.
-------------------	--

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.27.1.19 seLCD32B_SetBoostClk()

```
void seLCD32B_SetBoostClk (
    seLCD32B_BoostClk clk )
```

Parameters

<i>clk</i>	This parameter can be a value of seLCD32B_BoostClk .
------------	--

Return values

<i>None</i>	
-------------	--

2.27.1.20 seLCD32B_SetDisplayArea()

```
void seLCD32B_SetDisplayArea (
    seLCD32B_Area area )
```

Parameters

<i>area</i>	This parameter can be a value of seLCD32B_Area .
-------------	--

Return values

<i>None</i>	
-------------	--

2.27.1.21 seLCD32B_SetDisplayInvState()

```
void seLCD32B_SetDisplayInvState (
    seLCD32B_DisplnInvState state )
```

Parameters

<i>state</i>	This parameter can be a value of seLCD32B_DisplnInvState .
--------------	--

Return values

<i>None</i>	
-------------	--

2.27.1.22 seLCD32B_SetDisplayMemory()

```
void seLCD32B_SetDisplayMemory (
    seLCD32B_Area area,
    const uint8_t byte )
```

Parameters

<i>area</i>	This parameter can be a value of seLCD32B_Area .
<i>byte</i>	Value to set.

Return values

<i>None</i>	
-------------	--

2.27.1.23 seLCD32B_SetDisplayState()

```
void seLCD32B_SetDisplayState (
    seLCD32B_DispState state )
```

Parameters

<i>state</i>	This parameter can be a value of seLCD32B_DisplayState.
--------------	---

Return values

<i>None</i>	
-------------	--

2.27.1.24 seLCD32B_SetFrameFreq()

```
void seLCD32B_SetFrameFreq (
    seLCD32B_InitTypeDef * InitStruct )
```

Parameters

<i>InitStruct</i>	pointer to a @ ref seLCD32B_InitTypeDef structure that contains the configuration information for the LCD32B peripheral.
-------------------	--

Return values

<i>None</i>	
-------------	--

2.27.1.25 seLCD32B_SetPanelContrast()

```
void seLCD32B_SetPanelContrast (
    seLCD32B_Contrast contrast )
```

Parameters

<i>contrast</i>	This parameter can be a value of seLCD32B_Contrast .
-----------------	--

Return values

<i>None</i>	
-------------	--

2.28 PPORT

The PPORT (I/O Ports) circuit. This circuit allows the selection of I/O functions on processor pins between GPIO, and several alternate functions. Selected PPORT groups also can be routed to the UPMUX (Universal Port Multiplexer) where additional Peripheral Circuits can be selected to go a variety of pins.

Modules

- [PPORT_Exported_constants](#)
- [PPORT_Exported_Types](#)
- [PPORT_Public_Functions](#)

2.29 PPORT_Exported_constants

Typedefs

- typedef [seCLG_ClkSrc](#) **sePPORT_ClkSrc**

Enumerations

- enum [sePPORT_IOSC_ClkDiv](#) {
[sePPORT_IOSC_CLKDIV_1](#) = 0,
[sePPORT_IOSC_CLKDIV_2](#) = 1,
[sePPORT_IOSC_CLKDIV_4](#) = 2,
[sePPORT_IOSC_CLKDIV_8](#) = 3,
[sePPORT_IOSC_CLKDIV_16](#) = 4,
[sePPORT_IOSC_CLKDIV_32](#) = 5,
[sePPORT_IOSC_CLKDIV_64](#) = 6,
[sePPORT_IOSC_CLKDIV_128](#) = 7,
[sePPORT_IOSC_CLKDIV_256](#) = 8,
[sePPORT_IOSC_CLKDIV_512](#) = 9,
[sePPORT_IOSC_CLKDIV_1024](#) = 10,
[sePPORT_IOSC_CLKDIV_2048](#) = 11,
[sePPORT_IOSC_CLKDIV_4096](#) = 12,
[sePPORT_IOSC_CLKDIV_8192](#) = 13,
[sePPORT_IOSC_CLKDIV_16384](#) = 14,
[sePPORT_IOSC_CLKDIV_32768](#) = 15 }
- enum [sePPORT_OSC1_ClkDiv](#) {
[sePPORT_OSC1_CLKDIV_1](#) = 0,
[sePPORT_OSC1_CLKDIV_2](#) = 1,
[sePPORT_OSC1_CLKDIV_4](#) = 2,
[sePPORT_OSC1_CLKDIV_8](#) = 3,
[sePPORT_OSC1_CLKDIV_16](#) = 4,
[sePPORT_OSC1_CLKDIV_32](#) = 5,
[sePPORT_OSC1_CLKDIV_64](#) = 6,
[sePPORT_OSC1_CLKDIV_128](#) = 7,
[sePPORT_OSC1_CLKDIV_256](#) = 8,
[sePPORT_OSC1_CLKDIV_512](#) = 9,
[sePPORT_OSC1_CLKDIV_1024](#) = 10,
[sePPORT_OSC1_CLKDIV_2048](#) = 11,
[sePPORT_OSC1_CLKDIV_4096](#) = 12,
[sePPORT_OSC1_CLKDIV_8192](#) = 13,
[sePPORT_OSC1_CLKDIV_16384](#) = 14,
[sePPORT_OSC1_CLKDIV_32768](#) = 15 }
- enum [sePPORT_OSC3_ClkDiv](#) {
[sePPORT_OSC3_CLKDIV_1](#) = 0,
[sePPORT_OSC3_CLKDIV_2](#) = 1,
[sePPORT_OSC3_CLKDIV_4](#) = 2,
[sePPORT_OSC3_CLKDIV_8](#) = 3,
[sePPORT_OSC3_CLKDIV_16](#) = 4,
[sePPORT_OSC3_CLKDIV_32](#) = 5,
[sePPORT_OSC3_CLKDIV_64](#) = 6,
[sePPORT_OSC3_CLKDIV_128](#) = 7,
[sePPORT_OSC3_CLKDIV_256](#) = 8,
[sePPORT_OSC3_CLKDIV_512](#) = 9,

```
sePPORT_OSC3_CLKDIV_1024 = 10,  
sePPORT_OSC3_CLKDIV_2048 = 11,  
sePPORT_OSC3_CLKDIV_4096 = 12,  
sePPORT_OSC3_CLKDIV_8192 = 13,  
sePPORT_OSC3_CLKDIV_16384 = 14,  
sePPORT_OSC3_CLKDIV_32768 = 15 }  
  
• enum sePPORT_EXOSC_ClkDiv { sePPORT_EXOSC_CLKDIV_1 = 0 }  
  
• enum sePPORT_Id {  
sePPORT_P00 = 0,  
sePPORT_P01 = 1,  
sePPORT_P02 = 2,  
sePPORT_P03 = 3,  
sePPORT_P04 = 4,  
sePPORT_P05 = 5,  
sePPORT_P06 = 6,  
sePPORT_P07 = 7,  
sePPORT_P10 = 8,  
sePPORT_P11 = 9,  
sePPORT_P12 = 10,  
sePPORT_P13 = 11,  
sePPORT_P14 = 12,  
sePPORT_P15 = 13,  
sePPORT_P16 = 14,  
sePPORT_P17 = 15,  
sePPORT_P20 = 16,  
sePPORT_P21 = 17,  
sePPORT_P22 = 18,  
sePPORT_P23 = 19,  
sePPORT_P24 = 20,  
sePPORT_P25 = 21,  
sePPORT_P26 = 22,  
sePPORT_P27 = 23,  
sePPORT_P30 = 24,  
sePPORT_P31 = 25,  
sePPORT_P32 = 26,  
sePPORT_P33 = 27,  
sePPORT_P34 = 28,  
sePPORT_P35 = 29,  
sePPORT_P36 = 30,  
sePPORT_P37 = 31,  
sePPORT_P40 = 32,  
sePPORT_P41 = 33,  
sePPORT_P42 = 34,  
sePPORT_P43 = 35,  
sePPORT_P44 = 36,  
sePPORT_P45 = 37,  
sePPORT_P46 = 38,  
sePPORT_P47 = 39,  
sePPORT_P50 = 40,  
sePPORT_P51 = 41,  
sePPORT_P52 = 42,  
sePPORT_P53 = 43,  
sePPORT_P54 = 44,  
sePPORT_P55 = 45,
```

```

sePPORT_P56 = 46,
sePPORT_P57 = 47,
sePPORT_P60 = 48,
sePPORT_P61 = 49,
sePPORT_P62 = 50,
sePPORT_P63 = 51,
sePPORT_P64 = 52,
sePPORT_P65 = 53,
sePPORT_P66 = 54,
sePPORT_P67 = 55,
sePPORT_P70 = 56,
sePPORT_P71 = 57,
sePPORT_P72 = 58,
sePPORT_P73 = 59,
sePPORT_P74 = 60,
sePPORT_P75 = 61,
sePPORT_P76 = 62,
sePPORT_P77 = 63,
sePPORT_P80 = 64,
sePPORT_P81 = 65,
sePPORT_P90 = 72,
sePPORT_Pd0 = 80,
sePPORT_Pd1 = 81,
sePPORT_Pd2 = 82,
sePPORT_Pd3 = 83 }
• enum sePPORT_PortNumber {
sePPORT_P0 = ( 0 ),
sePPORT_P1 = ( 1 ),
sePPORT_P2 = ( 2 ),
sePPORT_P3 = ( 3 ),
sePPORT_P4 = ( 4 ),
sePPORT_P5 = ( 5 ),
sePPORT_P6 = ( 6 ),
sePPORT_P7 = ( 7 ) }
• enum sePPORT_PortGroup {
sePPORT_G0 = ( 0 ),
sePPORT_G1 = ( 1 ),
sePPORT_G2 = ( 2 ),
sePPORT_G3 = ( 3 ),
sePPORT_G4 = ( 4 ),
sePPORT_G5 = ( 5 ),
sePPORT_G6 = ( 6 ),
sePPORT_G7 = ( 7 ),
sePPORT_G8 = ( 8 ),
sePPORT_G9 = ( 9 ),
sePPORT_Gd = ( 10 ) }
• enum sePPORT_AltFunc {
sePPORT_ALT_0 = ( 0 ),
sePPORT_ALT_1 = ( 1 ),
sePPORT_ALT_2 = ( 2 ),
sePPORT_ALT_3 = ( 3 ) }
• enum sePPORT_PeriphPortInit {
sePPORT_PERIPHPORT_NOINIT = ( 0 ),
sePPORT_PERIPHPORT_NOTUPMUX = ( 1 ),

```

```

sePPORT_PERIPHPORT_UPMUX = ( 2 ) }
• enum sePPORT_Data {
    sePPORT_DATA_LOW = ( 0 ),
    sePPORT_DATA_HIGH = ( 1 ) }
• enum sePPORT_Edge {
    sePPORT_INT_EDGE_RISING = ( 0 ),
    sePPORT_INT_EDGE_FALLING = ( 1 ) }
• enum seUPMUX_Peripheral_Sel {
    seUPMUX_PER_DISABLE = ( 0 ),
    seUPMUX_PER_I2C = ( 1 ),
    seUPMUX_PER_SPIA = ( 2 ),
    seUPMUX_PER_UART = ( 3 ),
    seUPMUX_PER_T16B = ( 4 ) }
• enum seUPMUX_Channel_Sel {
    seUPMUX_CH_0 = ( 0 ),
    seUPMUX_CH_1 = ( 1 ),
    seUPMUX_CH_2 = ( 2 ) }
• enum seUPMUX_SPIA_Fnc {
    seUPMUX_SPIA_SDI = ( 1 ),
    seUPMUX_SPIA_SDO = ( 2 ),
    seUPMUX_SPIA_SPICLK = ( 3 ),
    seUPMUX_SPIA_SPISS = ( 4 ) }
• enum seUPMUX_I2C_Fnc {
    seUPMUX_I2C_SCL = ( 1 ),
    seUPMUX_I2C_SDA = ( 2 ) }
• enum seUPMUX_UART_Fnc {
    seUPMUX_UART_USIN = ( 1 ),
    seUPMUX_UART_USOUT = ( 2 ) }
• enum seUPMUX_T16B_Fnc {
    seUPMUX_T16B_TOUT_CAP0 = ( 1 ),
    seUPMUX_T16B_TOUT_CAP1 = ( 2 ),
    seUPMUX_T16B_TOUT_CAP2 = ( 3 ),
    seUPMUX_T16B_TOUT_CAP3 = ( 4 ),
    seUPMUX_T16B_TOUT_CAP4 = ( 5 ),
    seUPMUX_T16B_TOUT_CAP5 = ( 6 ) }

```

2.29.1 Enumeration Type Documentation

2.29.1.1 sePPORT_AltFunc

```
enum sePPORT_AltFunc
```

Enumerator

sePPORT_ALT↔ _0	Port Alt Function 0.
sePPORT_ALT↔ _1	Port Alt Function 1.
sePPORT_ALT↔ _2	Port Alt Function 2.
sePPORT_ALT↔ _3	Port Alt Function 3.

2.29.1.2 sePPORT_Data

```
enum sePPORT_Data
```

Enumerator

sePPORT_DATA_LOW	Port Data:LOW.
sePPORT_DATA_HIGH	Port Data:HIG.

2.29.1.3 sePPORT_Edge

```
enum sePPORT_Edge
```

Enumerator

sePPORT_INT_EDGE_RISING	Input signal:Rising edge.
sePPORT_INT_EDGE_FALLING	Input signal:Falling edge.

2.29.1.4 sePPORT_EXOSC_ClkDiv

```
enum sePPORT_EXOSC_ClkDiv
```

Enumerator

sePPORT_EXOSC_CLKDIV↔ _1	EXOSC division ratio is 1/1.
-----------------------------	------------------------------

2.29.1.5 sePPORT_Id

```
enum sePPORT_Id
```

Enumerator

sePPORT_P00	P00.
sePPORT_P01	P01.
sePPORT_P02	P02.
sePPORT_P03	P03.
sePPORT_P04	P04.
sePPORT_P05	P05.
sePPORT_P06	P06.
sePPORT_P07	P07.
sePPORT_P10	P10.
sePPORT_P11	P11.
sePPORT_P12	P12.
sePPORT_P13	P13.
sePPORT_P14	P14.

Enumerator

sePPORT_P15	P15.
sePPORT_P16	P16.
sePPORT_P17	P17.
sePPORT_P20	P20.
sePPORT_P21	P21.
sePPORT_P22	P22.
sePPORT_P23	P23.
sePPORT_P24	P24.
sePPORT_P25	P25.
sePPORT_P26	P26.
sePPORT_P27	P27.
sePPORT_P30	P30.
sePPORT_P31	P31.
sePPORT_P32	P32.
sePPORT_P33	P33.
sePPORT_P34	P34.
sePPORT_P35	P35.
sePPORT_P36	P36.
sePPORT_P37	P37.
sePPORT_P40	P40.
sePPORT_P41	P41.
sePPORT_P42	P42.
sePPORT_P43	P43.
sePPORT_P44	P44.
sePPORT_P45	P45.
sePPORT_P46	P46.
sePPORT_P47	P47.
sePPORT_P50	P50.
sePPORT_P51	P51.
sePPORT_P52	P52.
sePPORT_P53	P53.
sePPORT_P54	P54.
sePPORT_P55	P55.
sePPORT_P56	P56.
sePPORT_P57	P57.
sePPORT_P60	P60.
sePPORT_P61	P61.
sePPORT_P62	P62.
sePPORT_P63	P63.
sePPORT_P64	P64.
sePPORT_P65	P65.
sePPORT_P66	P66.
sePPORT_P67	P67.
sePPORT_P70	P70.
sePPORT_P71	P71.
sePPORT_P72	P72.

Enumerator

sePPORT_P73	P73.
sePPORT_P74	P74.
sePPORT_P75	P75.
sePPORT_P76	P76.
sePPORT_P77	P77.
sePPORT_P80	P80.
sePPORT_P81	P81.
sePPORT_P90	P90.
sePPORT_Pd0	Pd0.
sePPORT_Pd1	Pd1.
sePPORT_Pd2	Pd2.
sePPORT_Pd3	Pd3.

2.29.1.6 sePPORT_IOSC_ClkDiv

```
enum sePPORT_IOSC_ClkDiv
```

Enumerator

sePPORT_IOSC_CLKDIV_1	IOSC division ratio is 1/1.
sePPORT_IOSC_CLKDIV_2	IOSC division ratio is 1/2.
sePPORT_IOSC_CLKDIV_4	IOSC division ratio is 1/4.
sePPORT_IOSC_CLKDIV_8	IOSC division ratio is 1/8.
sePPORT_IOSC_CLKDIV_16	IOSC division ratio is 1/16.
sePPORT_IOSC_CLKDIV_32	IOSC division ratio is 1/32.
sePPORT_IOSC_CLKDIV_64	IOSC division ratio is 1/64.
sePPORT_IOSC_CLKDIV_128	IOSC division ratio is 1/128.
sePPORT_IOSC_CLKDIV_256	IOSC division ratio is 1/256.
sePPORT_IOSC_CLKDIV_512	IOSC division ratio is 1/512.
sePPORT_IOSC_CLKDIV_1024	IOSC division ratio is 1/1024.
sePPORT_IOSC_CLKDIV_2048	IOSC division ratio is 1/2048.
sePPORT_IOSC_CLKDIV_4096	IOSC division ratio is 1/4096.
sePPORT_IOSC_CLKDIV_8192	IOSC division ratio is 1/8192.
sePPORT_IOSC_CLKDIV_16384	IOSC division ratio is 1/16384.
sePPORT_IOSC_CLKDIV_32768	IOSC division ratio is 1/32768.

2.29.1.7 sePPORT_OSC1_ClkDiv

```
enum sePPORT_OSC1_ClkDiv
```

Enumerator

sePPORT_OSC1_CLKDIV_1	OSC1 division ratio is 1/1.
sePPORT_OSC1_CLKDIV_2	OSC1 division ratio is 1/2.

Enumerator

sePPORT_OSC1_CLKDIV_4	OSC1 division ratio is 1/4.
sePPORT_OSC1_CLKDIV_8	OSC1 division ratio is 1/8.
sePPORT_OSC1_CLKDIV_16	OSC1 division ratio is 1/16.
sePPORT_OSC1_CLKDIV_32	OSC1 division ratio is 1/32.
sePPORT_OSC1_CLKDIV_64	OSC1 division ratio is 1/64.
sePPORT_OSC1_CLKDIV_128	OSC1 division ratio is 1/128.
sePPORT_OSC1_CLKDIV_256	OSC1 division ratio is 1/256.
sePPORT_OSC1_CLKDIV_512	OSC1 division ratio is 1/512.
sePPORT_OSC1_CLKDIV_1024	OSC1 division ratio is 1/1024.
sePPORT_OSC1_CLKDIV_2048	OSC1 division ratio is 1/2048.
sePPORT_OSC1_CLKDIV_4096	OSC1 division ratio is 1/4096.
sePPORT_OSC1_CLKDIV_8192	OSC3 division ratio is 1/8192.
sePPORT_OSC1_CLKDIV_16384	OSC3 division ratio is 1/16384.
sePPORT_OSC1_CLKDIV_32768	OSC3 division ratio is 1/32768.

2.29.1.8 sePPORT_OSC3_ClkDiv

```
enum sePPORT_OSC3_ClkDiv
```

Enumerator

sePPORT_OSC3_CLKDIV_1	OSC3 division ratio is 1/1.
sePPORT_OSC3_CLKDIV_2	OSC3 division ratio is 1/2.
sePPORT_OSC3_CLKDIV_4	OSC3 division ratio is 1/4.
sePPORT_OSC3_CLKDIV_8	OSC3 division ratio is 1/8.
sePPORT_OSC3_CLKDIV_16	OSC1 division ratio is 1/16.
sePPORT_OSC3_CLKDIV_32	OSC1 division ratio is 1/32.
sePPORT_OSC3_CLKDIV_64	OSC1 division ratio is 1/64.
sePPORT_OSC3_CLKDIV_128	OSC1 division ratio is 1/128.
sePPORT_OSC3_CLKDIV_256	OSC1 division ratio is 1/256.
sePPORT_OSC3_CLKDIV_512	OSC1 division ratio is 1/512.
sePPORT_OSC3_CLKDIV_1024	OSC1 division ratio is 1/1024.
sePPORT_OSC3_CLKDIV_2048	OSC1 division ratio is 1/2048.
sePPORT_OSC3_CLKDIV_4096	OSC1 division ratio is 1/4096.
sePPORT_OSC3_CLKDIV_8192	OSC3 division ratio is 1/8192.
sePPORT_OSC3_CLKDIV_16384	OSC3 division ratio is 1/16384.
sePPORT_OSC3_CLKDIV_32768	OSC3 division ratio is 1/32768.

2.29.1.9 sePPORT_PeriphPortInit

```
enum sePPORT_PeriphPortInit
```

Enumerator

sePPORT_PERIPHPORT_NOINIT	Do not initialize pin.
---------------------------	------------------------

Enumerator

sePPORT_PERIPHPORT_NOTUPMUX	Initialize pin as not UPMUX.
sePPORT_PERIPHPORT_UPMUX	Initialize pin as UPMUX.

2.29.1.10 sePPORT_PortGroup

enum `sePPORT_PortGroup`

Enumerator

sePPORT_G0	Port group number P0x.
sePPORT_G1	Port group number P1x.
sePPORT_G2	Port group number P2x.
sePPORT_G3	Port group number P3x.
sePPORT_G4	Port group number P4x.
sePPORT_G5	Port group number P5x.
sePPORT_G6	Port group number P6x.
sePPORT_G7	Port group number P7x.
sePPORT_G8	Port group number P8x.
sePPORT_G9	Port group number P9x.
sePPORT_Gd	Port group number Pdx.

2.29.1.11 sePPORT_PortNumber

enum `sePPORT_PortNumber`

Enumerator

sePPORT_P0	Port number Px0.
sePPORT_P1	Port number Px1.
sePPORT_P2	Port number Px2.
sePPORT_P3	Port number Px3.
sePPORT_P4	Port number Px4.
sePPORT_P5	Port number Px5.
sePPORT_P6	Port number Px6.
sePPORT_P7	Port number Px7.

2.29.1.12 seUPMUX_Channel_Sel

enum `seUPMUX_Channel_Sel`

Enumerator

seUPMUX_CH↔ _0	UpMux Ch 0.
seUPMUX_CH↔ _1	UpMux Ch 1.
seUPMUX_CH↔ _2	UpMux Ch 2.

2.29.1.13 seUPMUX_I2C_Fnc

```
enum seUPMUX_I2C_Fnc
```

Enumerator

seUPMUX_I2C_SCL	UpMux Fnc I2C SCL.
seUPMUX_I2C_SDA	UpMux Fnc I2C SDA.

2.29.1.14 seUPMUX_Peripheral_Sel

```
enum seUPMUX_Peripheral_Sel
```

Enumerator

seUPMUX_PER_DISABLE	UpMux Disabled.
seUPMUX_PER_I2C	UpMux I2C Peripheral.
seUPMUX_PER_SPIA	UpMux SPIA Peripheral.
seUPMUX_PER_UART	UpMux UART Peripheral.
seUPMUX_PER_T16B	UpMux T16B Peripheral.

2.29.1.15 seUPMUX_SPIA_Fnc

```
enum seUPMUX_SPIA_Fnc
```

Enumerator

seUPMUX_SPIA_SDI	UpMux Fnc SPIA SDI.
seUPMUX_SPIA_SDO	UpMux Fnc SPIA SDO.
seUPMUX_SPIA_SPICLK	UpMux Fnc SPIA SPICLK.
seUPMUX_SPIA_SPISS	UpMux Fnc SPIA SPISS.

2.29.1.16 seUPMUX_T16B_Fnc

```
enum seUPMUX_T16B_Fnc
```

Enumerator

seUPMUX_T16B_TOUT_CAP0	UpMux Fnc T16B TOUT_CAP0.
seUPMUX_T16B_TOUT_CAP1	UpMux Fnc T16B TOUT_CAP1.
seUPMUX_T16B_TOUT_CAP2	UpMux Fnc T16B TOUT_CAP2.
seUPMUX_T16B_TOUT_CAP3	UpMux Fnc T16B TOUT_CAP3.
seUPMUX_T16B_TOUT_CAP4	UpMux Fnc T16B TOUT_CAP4.
seUPMUX_T16B_TOUT_CAP5	UpMux Fnc T16B TOUT_CAP5.

2.29.1.17 seUPMUX_UART_Fnc

enum [seUPMUX_UART_Fnc](#)

Enumerator

seUPMUX_UART_USIN	UpMux Fnc UART USIN.
seUPMUX_UART_USOUT	UpMux Fnc UART USOUT.

2.30 PPORT_Exported_Types

Data Structures

- struct [sePPORT_InitTypeDef](#)
PPORTInit structure definition.
- struct [sePPORT_Group](#)
- struct [sePPORT_PeriphPortDef](#)

Macros

- #define [PERISEL_Pos](#) 0
UPMUX Setting Register bit positions.
- #define **PERICH_Pos** 3
- #define **PFNC_Pos** 5

2.31 PPORT_Public_Functions

Functions

- `seStatus sePPORT_Init (sePPORT_InitTypeDef *InitStruct)`
Initializes the PPORT peripheral according to the specified parameters in the sePPORT_InitStruct. Stop PPORT before initializing.
- `void sePPORT_ConfigureClock (sePPORT_ClkSrc clock, uint16_t divider)`
Configures PPORT timer clock source and clock divider.
- `void sePPORT_InitAsInput (sePPORT_Id portId)`
Initialize for using a port as a general-purpose input port.
- `void sePPORT_EnablePullUpResistor (sePPORT_Id portId)`
Enables the pull-up resistor.
- `void sePPORT_EnablePullDownResistor (sePPORT_Id portId)`
Enables the pull-down resistor.
- `void sePPORT_DisableBuiltInResistor (sePPORT_Id portId)`
Disables built-in resistor.
- `void sePPORT_InitAsOutput (sePPORT_Id portId)`
Initialize for using a port as a general-purpose output port.
- `void sePPORT_EnableInt (sePPORT_Id portId, sePPORT_Edge edge)`
Enable port interrupt.
- `void sePPORT_DisableInt (sePPORT_Id portId)`
Disable port interrupt.
- `seInterruptStatus sePPORT_GetIntFlag (sePPORT_Id portId)`
Get port interrupt.
- `void sePPORT_ClearIntFlag (sePPORT_Id portId)`
Clear port interrupt.
- `sePPORT_Data sePPORT_GetOutput (sePPORT_Id portId)`
Get port output value.
- `void sePPORT_SetOutput (sePPORT_Id portId, sePPORT_Data data)`
Set port output data.
- `sePPORT_Data sePPORT_GetInput (sePPORT_Id portId)`
Get port input value.
- `void sePPORT_EnableChatteringFilter (sePPORT_Id portId)`
Enable port Chattering Filter.
- `void sePPORT_DisableChatteringFilter (sePPORT_Id portId)`
Disable port Chattering Filter.
- `seState sePPORT_GetChatteringFilter (sePPORT_Id portId)`
Gets the Chattering Filter.
- `void PORT_IRQHandler (void)`
PPORT Interrupt Service Routine (defined by user).
- `void sePPORT_InitAsHiZ (sePPORT_Id portId)`
Initialize for using a port as HiZ that is GPIO with input and output disabled, also disables the Interrupt.
- `seStatus sePPORT_UpMuxFunction (sePPORT_Id portId, seUPMUX_Peripheral_Sel peripheralNo, seUPMUX_X_Channel_Sel channelNo, uint8_t ioFuncNo)`
Initialize for using UpMux use for the requested function.
- `seStatus sePPORT_InitAsAltFunction (sePPORT_Id portId, sePPORT_AltFunc funcNo)`
Initialize for using a port as an alternate function.

2.31.1 Function Documentation

2.31.1.1 PORT_IRQHandler()

```
void PORT_IRQHandler (
    void )
```

Return values

None	
------	--

2.31.1.2 sePPORT_ClearIntFlag()

```
void sePPORT_ClearIntFlag (
    sePPORT_Id portId )
```

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
---------------	---

Return values

None	
------	--

2.31.1.3 sePPORT_ConfigureClock()

```
void sePPORT_ConfigureClock (
    sePPORT_ClkSrc clock,
    uint16_t divider )
```

Parameters

<i>clock</i>	This parameter can be a value of sePPORT_ClkSrc .
<i>divider</i>	This parameter can be a value of sePPORT_ClkDiv .

Note

If Chattering filter is enabled application shall clear interrupts when changing the clock settings.

Return values

None	
------	--

2.31.1.4 sePPORT_DisableBuiltInResistor()

```
void sePPORT_DisableBuiltInResistor (
    sePPORT_Id portId )
```

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
---------------	---

Return values

<i>None</i>	
-------------	--

2.31.1.5 sePPORT_DisableChatteringFilter()

```
void sePPORT_DisableChatteringFilter (
    sePPORT_Id portId )
```

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
---------------	---

Note

Application shall clear interrupts when changing Chattering Filter.

Return values

<i>None</i>	
-------------	--

2.31.1.6 sePPORT_DisableInt()

```
void sePPORT_DisableInt (
    sePPORT_Id portId )
```

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
---------------	---

Return values

<i>None</i>	
-------------	--

2.31.1.7 sePPORT_EnableChatteringFilter()

```
void sePPORT_EnableChatteringFilter (
    sePPORT_Id portId )
```

Parameters

<i>port↔ Id</i>	This parameter can be a value of sePPORT_Id .
---------------------	---

Note

Application shall clear interrupts when changing Chattering Filter.

Return values

<i>None</i>	
-------------	--

2.31.1.8 sePPORT_EnableInt()

```
void sePPORT_EnableInt (
    sePPORT_Id portId,
    sePPORT_Edge edge )
```

Parameters

<i>port↔ Id</i>	This parameter can be a value of sePPORT_Id .
<i>edge</i>	This parameter can be a value of sePPORT_Edge .

Return values

<i>None</i>	
-------------	--

2.31.1.9 sePPORT_EnablePullDownResistor()

```
void sePPORT_EnablePullDownResistor (
    sePPORT_Id portId )
```

Parameters

<i>port↔ Id</i>	This parameter can be a value of sePPORT_Id .
---------------------	---

Return values

<i>None</i>	
-------------	--

2.31.1.10 sePPORT_EnablePullUpResistor()

```
void sePPORT_EnablePullUpResistor (
    sePPORT_Id portId )
```

Parameters

<i>port↔ Id</i>	This parameter can be a value of sePPORT_Id .
---------------------	---

Return values

<i>None</i>	
-------------	--

2.31.1.11 sePPORT_GetChatteringFilter()

```
seState sePPORT_GetChatteringFilter (
    sePPORT_Id portId )
```

Parameters

<i>port↔ Id</i>	This parameter can be a value of sePPORT_Id .
---------------------	---

Return values

<i>State</i>	can be a value of seState
--------------	---

2.31.1.12 sePPORT_GetInput()

```
sePPORT_Data sePPORT_GetInput (
    sePPORT_Id portId )
```

Parameters

<i>port↔ Id</i>	This parameter can be a value of sePPORT_Id .
---------------------	---

Return values

<i>value</i>	can be a value of sePPORT_Data .
--------------	--

2.31.1.13 sePPORT_GetIntFlag()

```
seInterruptStatus sePPORT_GetIntFlag (
```

```
sePPORT_Id portId )
```

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
---------------	---

Return values

<i>InterruptStatus</i>	can be a value of seInterruptStatus .
------------------------	---

2.31.1.14 sePPORT_GetOutput()

```
sePPORT_Data sePPORT_GetOutput (
    sePPORT_Id portId )
```

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
---------------	---

Return values

<i>value</i>	can be a value of sePPORT_Data .
--------------	--

2.31.1.15 sePPORT_Init()

```
seStatus sePPORT_Init (
    sePPORT_InitTypeDef * InitStruct )
```

Parameters

<i>InitStruct</i>	pointer to a sePPORT_InitTypeDef structure that contains the configuration information for the specified PPORT peripheral. sePPORT_InitTypeDef .
-------------------	--

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.31.1.16 sePPORT_InitAsAltFunction()

```
seStatus sePPORT_InitAsAltFunction (
    sePPORT_Id portId,
    sePPORT_AltFunc funcNo )
```

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
<i>funcNo</i>	This parameter can be a value of sePPORT_AltFunc .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.31.1.17 sePPORT_InitAsHiZ()

```
void sePPORT_InitAsHiZ (
    sePPORT_Id portId )
```

Parameters

<i>port↔ Id</i>	This parameter can be a value of sePPORT_Id .
---------------------	---

Return values

<i>None</i>	
-------------	--

2.31.1.18 sePPORT_InitAsInput()

```
void sePPORT_InitAsInput (
    sePPORT_Id portId )
```

Parameters

<i>port↔ Id</i>	This parameter can be a value of sePPORT_Id .
---------------------	---

Return values

<i>None</i>	
-------------	--

2.31.1.19 sePPORT_InitAsOutput()

```
void sePPORT_InitAsOutput (
    sePPORT_Id portId )
```

Parameters

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
---------------	---

Return values

<i>None</i>	
-------------	--

2.31.1.20 sePPORT_SetOutput()

```
void sePPORT_SetOutput (
    sePPORT_Id portId,
    sePPORT_Data data )
```

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
<i>value</i>	can be a value of sePPORT_Data .

Return values

<i>None</i>	
-------------	--

2.31.1.21 sePPORT_UpMuxFunction()

```
seStatus sePPORT_UpMuxFunction (
    sePPORT_Id portId,
    seUPMUX_Peripheral_Sel peripheralNo,
    seUPMUX_Channel_Sel channelNo,
    uint8_t ioFuncNo )
```

Parameters

<i>portId</i>	This parameter can be a value of sePPORT_Id .
<i>peripheralNo</i>	This parameter can be a value of seUPMUX_Peripheral_Sel
<i>channelNo</i>	This parameter can be a value of seUPMUX_Channel_Sel
<i>ioFuncNo</i>	This parameter selects I/O function

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.32 QSPI

The QSPI module is a subset of the QSPI bus interface.

Modules

- [QSPI_Constants](#)

The QSPI module exported constants.

- [QSPI_Types](#)
- [QSPI_Functions](#)

2.33 QSPI_Constants

The QSPI module exported constants.

Macros

- #define **seQSPI_FLGS**(a) ((seQSPI_InterruptFlags)((a)))
Combination of any of the seQSPI_IntFlag enumerations.
- #define **seQSPI_INTS**(a) ((seQSPI_Interrupts)((a)))
Combination of any of the seQSPI_Interrupt enumerations.

Enumerations

- enum **seQSPI_Clocks** {
 seQSPI_16CLK = 0xf,
 seQSPI_15CLK = 0xe,
 seQSPI_14CLK = 0xd,
 seQSPI_13CLK = 0xc,
 seQSPI_12CLK = 0xb,
 seQSPI_11CLK = 0xa,
 seQSPI_10CLK = 0x9,
 seQSPI_09CLK = 0x8,
 seQSPI_08CLK = 0x7,
 seQSPI_07CLK = 0x6,
 seQSPI_06CLK = 0x5,
 seQSPI_05CLK = 0x4,
 seQSPI_04CLK = 0x3,
 seQSPI_03CLK = 0x2,
 seQSPI_02CLK = 0x1,
 seQSPI_01CLK = 0x0 }
- enum **seQSPI_Format** {
 seQSPI_MSB_FST = 0,
 seQSPI_LSB_FST = 1 }
- enum **seQSPI_Polarity** {
 seQSPI_POL_LOW = 0,
 seQSPI_POL_HIGH = 1 }
- enum **seQSPI_Phase** {
 seQSPI_PH_RISE = 0,
 seQSPI_PH_FALL = 1 }
- enum **seQSPI_OperMode** {
 seQSPI_MODE_SLAVE = 0,
 seQSPI_MODE_MASTER = 1 }
- enum **seQSPI_TransferMode** {
 seQSPI_MODE_SINGLE = 0,
 seQSPI_MODE_DUAL = 1,
 seQSPI_MODE_QUAD = 2 }
- enum **seQSPI_IntFlag** {
 seQSPI_BSY = 0x0080U,
 seQSPI_OEIF = 0x0008U,
 seQSPI_TENDIF = 0x0004U,
 seQSPI_RBFIF = 0x0002U,

```

seQSPI_TBEIF = 0x0001U,
seQSPI_ALL_IF = seQSPI_OEIF | seQSPI_TENDIF | seQSPI_RBFIF | seQSPI_TBEIF }
• enum seQSPI_Interrupt {
seQSPI_OEIE = 0x0008U,
seQSPI_TENDIE = 0x0004U,
seQSPI_RBFIE = 0x0002U,
seQSPI_TBEIE = 0x0001U,
seQSPI_ALL_IE = seQSPI_OEIE | seQSPI_TENDIE | seQSPI_RBFIE | seQSPI_TBEIE }
• enum seQSPI_IO {
seQSPI_Output = 0,
seQSPI_Input = 1 }
• enum seQSPI_AddrMode {
seQSPI_24BIT_ADDR = 0,
seQSPI_32BIT_ADDR = 1 }

```

2.33.1 Enumeration Type Documentation

2.33.1.1 seQSPI_AddrMode

```
enum seQSPI_AddrMode
```

Enumerator

seQSPI_24BIT_ADDR	Specifies IO direction as output.
seQSPI_32BIT_ADDR	Specifies IO direction as input.

2.33.1.2 seQSPI_Format

```
enum seQSPI_Format
```

Enumerator

seQSPI_MSB_FST	Specify the data format (input/output permutation) MSB first.
seQSPI_LSB_FST	Specify the data format (input/output permutation) LSB first.

2.33.1.3 seQSPI_Interrupt

```
enum seQSPI_Interrupt
```

Enumerator

seQSPI_OEIE	Overrun error interrupt.
seQSPI_TENDIE	End-of-transmission interrupt.
seQSPI_RBFIE	Receive buffer full interrupt.
seQSPI_TBEIE	Transmit buffer empty interrupt.

2.33.1.4 seQSPI_IntFlag

```
enum seQSPI_IntFlag
```

Enumerator

seQSPI_BSY	Transfer busy/slave selected.
seQSPI_OEIF	Overrun error interrupt.
seQSPI_TENDIF	End-of-transmission interrupt.
seQSPI_RBFIF	Receive buffer full interrupt.
seQSPI_TBEIF	Transmit buffer empty interrupt.

2.33.1.5 seQSPI_IO

```
enum seQSPI_IO
```

Enumerator

seQSPI_Output	Specifies IO direction as output.
seQSPI_Input	Specifies IO direction as input.

2.33.1.6 seQSPI_OperMode

```
enum seQSPI_OperMode
```

Enumerator

seQSPI_MODE_SLAVE	Specifies the QSPI mode slave.
seQSPI_MODE_MASTER	Specifies the QSPI mode master.

2.33.1.7 seQSPI_Phase

```
enum seQSPI_Phase
```

Enumerator

seQSPI_PH_RISE	Triggers on positive edge of clock.
seQSPI_PH_FALL	Triggers on negative edge of clock.

2.33.1.8 seQSPI_Polarity

```
enum seQSPI_Polarity
```

Enumerator

seQSPI_POL_LOW	Output is low when clock is off.
seQSPI_POL_HIGH	Output is high when clock is off.

2.33.1.9 seQSPI_TransferMode

enum [seQSPI_TransferMode](#)

Enumerator

seQSPI_MODE_SINGLE	Single transfer mode.
seQSPI_MODE_DUAL	Dual transfer mode.
seQSPI_MODE_QUAD	Quad transfer mode.

2.34 QSPI_Types

Data Structures

- struct [seQSPI_InitTypeDef](#)
QSPI Init structure definition.
- struct [seQSPI_ChannelDef](#)
QSPI Channel definition.

Variables

- [seQSPI_ChannelDef](#) **QSPI_CH0**
- const uint32_t **QSPI_MMA_START_ADDR**

2.35 QSPI_Funcions

Functions

- void [seQSPI_InitStructForMaster](#) ([seQSPI_InitTypeDef](#) *QSPI_InitStruct)
This function initializes the QSPIx peripheral as Master according to the specified parameters in the QSPI_InitStruct.
- void [seQSPI_InitStructForSlave](#) ([seQSPI_InitTypeDef](#) *QSPI_InitStruct)
This function initializes the QSPIx peripheral as Slave according to the specified parameters in the QSPI_InitStruct.
- [seStatus seQSPI_Init](#) ([seQSPI_ChannelDef](#) *QSPICHx, [seQSPI_InitTypeDef](#) *QSPI_InitStruct)
This function initializes the QSPIx peripheral according to the specified parameters in the QSPI_InitStruct.
- [seStatus seQSPI_Start](#) ([seQSPI_ChannelDef](#) *QSPICHx)
This function starts or enables the specified QSPI channel.
- void [seQSPI_Stop](#) ([seQSPI_ChannelDef](#) *QSPICHx)
This function stops or disables the specified QSPI channel.
- [seStatus seQSPI_Reset](#) (QSPI_0_Type *QSPIx)
This function is software reset of the specified QSPI channel.
- [seStatus seQSPI_TxHWords](#) (QSPI_0_Type *QSPIx, uint16_t data[], uint32_t size)
This function sends data through the QSPIx peripheral.
- [seStatus seQSPI_DmaTxHWords](#) (QSPI_0_Type *QSPIx, uint16_t data[], uint32_t size)
This function sends data by DMA through the QSPIx peripheral.
- [seStatus seQSPI_TxBytes](#) (QSPI_0_Type *QSPIx, uint8_t data[], uint32_t size)
This function sends data through the QSPIx peripheral.
- [seStatus seQSPI_DmaTxBytes](#) (QSPI_0_Type *QSPIx, uint8_t data[], uint32_t size)
This function sends data by through the QSPIx peripheral.
- [seStatus seQSPI_TxValue](#) (QSPI_0_Type *QSPIx, uint8_t value, uint32_t count)
This function sends single data value through the QSPIx peripheral.
- [seStatus seQSPI_RxHWords](#) (QSPI_0_Type *QSPIx, uint16_t data[], uint32_t size)
This function returns the received data by the QSPIx peripheral.
- [seStatus seQSPI_DmaRxHWords](#) (QSPI_0_Type *QSPIx, uint16_t data[], uint32_t size)
This function returns the received data from the QSPIx peripheral by DMA.
- [seStatus seQSPI_RxBytes](#) (QSPI_0_Type *QSPIx, uint8_t data[], uint32_t size)
This function returns the received data from the QSPIx peripheral.
- [seStatus seQSPI_DmaRxBytes](#) (QSPI_0_Type *QSPIx, uint8_t data[], uint32_t size)
This function returns the received data from the QSPIx peripheral by DMA.
- [seStatus seQSPI_DmaRxMmaWords](#) (QSPI_0_Type *QSPIx, uint32_t offset, uint32_t data[], uint32_t size)
This function returns the received data by DMA from the QSPIx peripheral in memory mapped mode.
- [seStatus seQSPI_SetMode](#) (QSPI_0_Type *QSPIx, [seQSPI_TransferMode](#) mode, [seQSPI_Clocks](#) chIn, [seQSPI_Clocks](#) chDl)
This function sets QSPI mode (single, dual, quad).
- void [seQSPI_EnableInt](#) (QSPI_0_Type *QSPIx, [seQSPI_Interrupt](#) irq)
This function enables QSPI channel interrupt.
- void [seQSPI_DisableInt](#) (QSPI_0_Type *QSPIx, [seQSPI_Interrupt](#) irq)
This function disables QSPI channel interrupt.
- [seInterruptStatus seQSPI_GetIntFlag](#) (QSPI_0_Type *QSPIx, [seQSPI_IntFlag](#) flag)
This function gets QSPI channel interrupt flag.
- void [seQSPI_ClearIntFlag](#) (QSPI_0_Type *QSPIx, [seQSPI_IntFlag](#) flag)
This function clears QSPI channel interrupt flag.

- `seStatus seQSPI_SetBusSpeed` (`seQSPI_ChannelDef *QSPICHx`, `uint32_t speed`)
This function configures QSPICLK frequency [Hz] (= baud rate [bps]).
- `uint32_t seQSPI_GetBusSpeed` (`seQSPI_ChannelDef *QSPICHx`)
This function discovers QSPICLK frequency [Hz] (= baud rate [bps]).
- `seStatus seQSPI_SetMasterRxMMA` (`QSPI_0_Type *QSPIx`, `uint32_t raddr`, `uint16_t flash_rcmd`)
This function sets the mapping mode for reading from External flash.
- `seStatus seQSPI_ClearMasterRxMMA` (`QSPI_0_Type *QSPIx`)
This function ends mapping mode for reading from External flash.
- `seStatus seQSPI_TermMasterTx` (`seQSPI_ChannelDef *QSPICHx`)
This function terminates QSPI module.
- `void seQSPI_SetIO` (`QSPI_0_Type *QSPIx`, `seQSPI_IO direction`)
This function configures I/O for Rx or Tx direction (Master mode only).
- `void seQSPI_ASSERT_MST_CS0` (`void`)
This function asserts Chip Slave Select 0 (Master mode only) It controls a native QSPI slave select line that can be used with MMA.
- `void seQSPI_NEGATE_MST_CS0` (`void`)
This function negates Chip Slave Select 0 (Master mode only) It controls a native QSPI slave select line that can be used with MMA.
- `void seQSPI_ASSERT_MST_CS1` (`void`)
Function asserts Chip Slave Select 1 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.
- `void seQSPI_ASSERT_MST_CS2` (`void`)
Function asserts Chip Slave Select 2 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.
- `void seQSPI_ASSERT_MST_CS3` (`void`)
Function asserts Chip Slave Select 3 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.
- `void seQSPI_NEGATE_MST_CS1` (`void`)
Function negates Chip Slave Select 1 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.
- `void seQSPI_NEGATE_MST_CS2` (`void`)
This function negates Chip Slave Select 2 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.
- `void seQSPI_NEGATE_MST_CS3` (`void`)
This function negates Chip Slave Select 3 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.
- `void QSPI_IRQHandler` (`void`)
QSPI Interrupt Service Routine (defined by user).
- `seStatus ConfigurePortsForQSPI` (`seQSPI_ChannelDef *QSPICHx`)
Configures ports for this module. Override this function to configure specific ports.

2.35.1 Function Documentation

2.35.1.1 ConfigurePortsForQSPI()

```
seStatus ConfigurePortsForQSPI (
    seQSPI_ChannelDef * QSPICHx )
```

Parameters

<i>QSPICHx</i>	QSPI channel definition of type seQSPI_ChannelDef
----------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.2 QSPI_IRQHandler()

```
void QSPI_IRQHandler (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.35.1.3 seQSPI_ASSERT_MST_CS0()

```
void seQSPI_ASSERT_MST_CS0 (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.35.1.4 seQSPI_ASSERT_MST_CS1()

```
void seQSPI_ASSERT_MST_CS1 (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.35.1.5 seQSPI_ASSERT_MST_CS2()

```
void seQSPI_ASSERT_MST_CS2 (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.35.1.6 seQSPI_ASSERT_MST_CS3()

```
void seQSPI_ASSERT_MST_CS3 (
    void )
```

Return values

None	
------	--

2.35.1.7 seQSPI_ClearIntFlag()

```
void seQSPI_ClearIntFlag (
    QSPI_0_Type * QSPIx,
    seQSPI_IntFlag flag )
```

Parameters

QSPIx	QSPI to select the QSPI peripheral.
flag	Interrupt to clear, see seQSPI_IntFlag .

Return values

None	
------	--

2.35.1.8 seQSPI_ClearMasterRxMMA()

```
seStatus seQSPI_ClearMasterRxMMA (
    QSPI_0_Type * QSPIx )
```

Parameters

QSPIx	QSPIx_CH0 to select the QSPI peripheral.
-------	--

Return values

Status	can be a value of seStatus
--------	--

2.35.1.9 seQSPI_DisableInt()

```
void seQSPI_DisableInt (
    QSPI_0_Type * QSPIx,
    seQSPI_Interrupt irq )
```

Parameters

<i>QSPiPlx</i>	QSPI to select the QSPI peripheral.
<i>irq</i>	Interrupt to be disabled, see seQSPI_Interrupt .

Return values

<i>None</i>	
-------------	--

2.35.1.10 seQSPI_DmaRxBytes()

```
seStatus seQSPI_DmaRxBytes (
    QSPI_0_Type * QSPiPlx,
    uint8_t data[],
    uint32_t size )
```

Note

: DMAC must be initialized prior calling of this function. DMA Channels 1,3 are used by this function. They must be available for duration of the function call.

Parameters

<i>QSPiPlx</i>	QSPI to select the QSPI peripheral.
<i>data</i>	The received data pointer.
<i>size</i>	Data size in number of bytes.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.11 seQSPI_DmaRxHWords()

```
seStatus seQSPI_DmaRxHWords (
    QSPI_0_Type * QSPiPlx,
    uint16_t data[],
    uint32_t size )
```

Note

: DMAC must be initialized prior calling of this function. DMA Channels 1,3 are used by this function. They must be available for duration of the function call.

Parameters

<i>QSPiPlx</i>	QSPI to select the QSPI peripheral.
<i>data</i>	The received data pointer.
<i>size</i>	Data size in number of half words.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.12 seQSPI_DmaRxMmaWords()

```
seStatus seQSPI_DmaRxMmaWords (
    QSPI_0_Type * QSPiPlx,
    uint32_t offset,
    uint32_t data[],
    uint32_t size )
```

Note

: DMAC must be initialized prior calling of this function. DMA Channel 2 is used by this function. They must be available for duration of the function call.

Parameters

<i>QSPiPlx</i>	QSPI to select the QSPI peripheral.
<i>data</i>	The received data pointer.
<i>size</i>	Data size in number of long words.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.13 seQSPI_DmaTxBytes()

```
seStatus seQSPI_DmaTxBytes (
    QSPI_0_Type * QSPiPlx,
    uint8_t data[],
    uint32_t size )
```

Note

: DMAC must be initialized prior calling of this function. DMA Channel 0 is used by this function. It must be available for duration of the function call.

Parameters

<i>QSPiPlx</i>	QSPI to select the QSPI peripheral.
<i>data</i>	Pointer to data to be transmitted.
<i>size</i>	Data size in number of bytes.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.14 seQSPI_DmaTxHWords()

```
seStatus seQSPI_DmaTxHWords (
    QSPI_0_Type * QSPiPlx,
    uint16_t data[],
    uint32_t size )
```

Note

: DMAC must be initialized prior calling of this function. DMA Channel 0 is used by this function. It must be available for the duration of the function call.

Parameters

<i>QSPiPlx</i>	QSPI to select the QSPI peripheral.
<i>data</i>	Pointer to data to be transmitted.
<i>size</i>	Data size in number of half words.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.15 seQSPI_EnableInt()

```
void seQSPI_EnableInt (
    QSPI_0_Type * QSPiPlx,
    seQSPI_Interrupt irq )
```

Parameters

<i>QSPiPlx</i>	QSPI to select the QSPI peripheral.
<i>irq</i>	Interrupt to be enabled, see seQSPI_Interrupt .

Return values

<i>None</i>	
-------------	--

2.35.1.16 seQSPI_GetBusSpeed()

```
uint32_t seQSPI_GetBusSpeed (
    seQSPI_ChannelDef * QSPICHx )
```

Parameters

<i>QSPICHx</i>	QSPI channel definition of type seQSPI_ChannelDef
----------------	---

Return values

<i>QSPICLK_n</i>	frequency [Hz].
----------------------------	-----------------

2.35.1.17 seQSPI_GetIntFlag()

```
seInterruptStatus seQSPI_GetIntFlag (
    QSPI_0_Type * QSPIdx,
    seQSPI_IntFlag flag )
```

Parameters

<i>QSPIdx</i>	QSPI to select the QSPI peripheral.
<i>flag</i>	Interrupt to check, see seQSPI_IntFlag .

Return values

<i>seInterruptStatus</i>	see seInterruptStatus .
--------------------------	---

2.35.1.18 seQSPI_Init()

```
seStatus seQSPI_Init (
    seQSPI_ChannelDef * QSPICHx,
    seQSPI_InitTypeDef * QSPI_InitStruct )
```

Parameters

<i>QSPICHx</i>	QSPI channel definition of type seQSPI_ChannelDef
<i>QSPI_InitStruct</i>	pointer to a seQSPI_InitTypeDef structure that contains the configuration information for the specified QSPI peripheral.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.19 seQSPI_InitStructForMaster()

```
void seQSPI_InitStructForMaster (
    seQSPI\_InitTypeDef * QSPI_InitStruct )
```

Parameters

<i>QSPI_InitStruct</i>	pointer to a seQSPI_InitTypeDef structure that contains the configuration information for the specified QSPI peripheral.
------------------------	--

Return values

<i>None</i>	
-------------	--

2.35.1.20 seQSPI_InitStructForSlave()

```
void seQSPI_InitStructForSlave (
    seQSPI\_InitTypeDef * QSPI_InitStruct )
```

Parameters

<i>QSPI_InitStruct</i>	pointer to a seQSPI_InitTypeDef structure that contains the configuration information for the specified QSPI peripheral.
------------------------	--

Return values

<i>None</i>	
-------------	--

2.35.1.21 seQSPI_NEGATE_MST_CS0()

```
void seQSPI_NEGATE_MST_CS0 (
    void )
```

Return values

<i>None</i>	
-------------	--

2.35.1.22 seQSPI_NEGATE_MST_CS1()

```
void seQSPI_NEGATE_MST_CS1 (
```

```
void )
```

Return values

<i>None</i>	
-------------	--

2.35.1.23 seQSPI_NEGATE_MST_CS2()

```
void seQSPI_NEGATE_MST_CS2 (
    void )
```

Return values

<i>None</i>	
-------------	--

2.35.1.24 seQSPI_NEGATE_MST_CS3()

```
void seQSPI_NEGATE_MST_CS3 (
    void )
```

Return values

<i>None</i>	
-------------	--

2.35.1.25 seQSPI_Reset()

```
seStatus seQSPI_Reset (
    QSPI_0_Type * QSPIdx )
```

Parameters

<i>QSPIdx</i>	QSPI to select the QSPI peripheral.
---------------	-------------------------------------

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.26 seQSPI_RxBytes()

```
seStatus seQSPI_RxBytes (
    QSPI_0_Type * QSPIdx,
    uint8_t data[],
    uint32_t size )
```

Parameters

<i>QSPiPlx</i>	QSPI to select the QSPI peripheral.
<i>data</i>	The received data pointer.
<i>size</i>	Data size in number of bytes.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.27 seQSPI_RxHWords()

```
seStatus seQSPI_RxHWords (
    QSPI_0_Type * QSPiPlx,
    uint16_t data[],
    uint32_t size )
```

Parameters

<i>QSPiPlx</i>	QSPI to select the QSPI peripheral.
<i>data</i>	The received data pointer.
<i>size</i>	Data size in number of half words.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.28 seQSPI_SetBusSpeed()

```
seStatus seQSPI_SetBusSpeed (
    seQSPI_ChannelDef * QSPiCHx,
    uint32_t speed )
```

Parameters

<i>QSPiCHx</i>	QSPI channel definition of type seQSPI_ChannelDef
<i>speed</i>	QSPICLK frequency [Hz].

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.29 seQSPI_SetIO()

```
void seQSPI_SetIO (
    QSPI_0_Type * QSPIdx,
    seQSPI_IO direction )
```

Parameters

<i>QSPIdx</i>	QSPIdx_CH0 to select the QSPI peripheral.
---------------	---

Return values

<i>None</i>	
-------------	--

2.35.1.30 seQSPI_SetMasterRxMMA()

```
seStatus seQSPI_SetMasterRxMMA (
    QSPI_0_Type * QSPIdx,
    uint32_t raddr,
    uint16_t flash_rcmd )
```

Parameters

<i>QSPIdx</i>	QSPIdx_CH0 to select the QSPI peripheral.
<i>raddr</i>	Remapping address.
<i>rcmd</i>	Read command defined by flash type used for mapping.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.31 seQSPI_SetMode()

```
seStatus seQSPI_SetMode (
    QSPI_0_Type * QSPIdx,
    seQSPI_TransferMode mode,
    seQSPI_Clocks chln,
    seQSPI_Clocks chdl )
```

Parameters

<i>QSPIdx</i>	QSPI to select the QSPI peripheral.
<i>mode</i>	Sets single, dual or quad, see seQSPI_TransferMode .
<i>chln</i>	Number of clocks (seQSPI_01CLK is prohibited) for data transfer, see seQSPI_Clocks .
<i>chdl</i>	Number of clocks to drive the serial data lines, see seQSPI_Clocks .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.32 seQSPI_Start()

```
seStatus seQSPI_Start (
    seQSPI_ChannelDef * QSPICHx )
```

Parameters

<i>QSPICHx</i>	QSPI channel definition of type seQSPI_ChannelDef
----------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.33 seQSPI_Stop()

```
void seQSPI_Stop (
    seQSPI_ChannelDef * QSPICHx )
```

Parameters

<i>QSPICHx</i>	QSPI channel definition of type seQSPI_ChannelDef
----------------	---

Return values

<i>None</i>	
-------------	--

2.35.1.34 seQSPI_TermMasterTx()

```
seStatus seQSPI_TermMasterTx (
    seQSPI_ChannelDef * QSPICHx )
```

Parameters

<i>QSPICHx</i>	QSPI channel definition of type seQSPI_ChannelDef
----------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.35 seQSPI_TxBytes()

```
seStatus seQSPI_TxBytes (
    QSPI_0_Type * QSPIdx,
    uint8_t data[],
    uint32_t size )
```

Parameters

<i>QSPIdx</i>	QSPI to select the QSPI peripheral.
<i>data</i>	Pointer to data to be transmitted.
<i>size</i>	Data size in number of bytes.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.36 seQSPI_TxHWords()

```
seStatus seQSPI_TxHWords (
    QSPI_0_Type * QSPIdx,
    uint16_t data[],
    uint32_t size )
```

Parameters

<i>QSPIdx</i>	QSPI to select the QSPI peripheral.
<i>data</i>	Pointer to data to be transmitted.
<i>size</i>	Data size in number of half words.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.35.1.37 seQSPI_TxValue()

```
seStatus seQSPI_TxValue (
    QSPI_0_Type * QSPIdx,
    uint8_t value,
    uint32_t count )
```

Parameters

<i>QSPIdx</i>	QSPI to select the QSPI peripheral.
<i>value</i>	Data value to be transmitted.

Parameters

<i>count</i>	Data size in number of bytes.
--------------	-------------------------------

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.36 REMC2

The REMC2 circuit generates infrared remote control output signals. This circuit can also be applicable to an EL lamp drive circuit by adding a simple external circuit.

Modules

- [REMC2_Constants](#)
- [REMC2_Types](#)
- [REMC2_Functions](#)

2.37 REMC2_Constants

Typedefs

- typedef [seCLG_ClkSrc](#) **seREMC2_ClkSrc**

Enumerations

- enum [seREMC2_IOSC_ClkDiv](#) {
[seREMC2_IOSC_CLKDIV_1](#) = 0,
[seREMC2_IOSC_CLKDIV_2](#) = 1,
[seREMC2_IOSC_CLKDIV_4](#) = 2,
[seREMC2_IOSC_CLKDIV_8](#) = 3,
[seREMC2_IOSC_CLKDIV_16](#) = 4,
[seREMC2_IOSC_CLKDIV_32](#) = 5,
[seREMC2_IOSC_CLKDIV_64](#) = 6,
[seREMC2_IOSC_CLKDIV_128](#) = 7,
[seREMC2_IOSC_CLKDIV_256](#) = 8,
[seREMC2_IOSC_CLKDIV_512](#) = 9,
[seREMC2_IOSC_CLKDIV_1024](#) = 10,
[seREMC2_IOSC_CLKDIV_2048](#) = 11,
[seREMC2_IOSC_CLKDIV_4096](#) = 12,
[seREMC2_IOSC_CLKDIV_8192](#) = 13,
[seREMC2_IOSC_CLKDIV_16384](#) = 14,
[seREMC2_IOSC_CLKDIV_32768](#) = 15 }
- enum [seREMC2_OSC1_ClkDiv](#) {
[seREMC2_OSC1_CLKDIV_1](#) = 0,
[seREMC2_OSC1_CLKDIV_2](#) = 1,
[seREMC2_OSC1_CLKDIV_4](#) = 2,
[seREMC2_OSC1_CLKDIV_8](#) = 3,
[seREMC2_OSC1_CLKDIV_16](#) = 4,
[seREMC2_OSC1_CLKDIV_32](#) = 5,
[seREMC2_OSC1_CLKDIV_64](#) = 6,
[seREMC2_OSC1_CLKDIV_128](#) = 7,
[seREMC2_OSC1_CLKDIV_256](#) = 8 }
- enum [seREMC2_OSC3_ClkDiv](#) {
[seREMC2_OSC3_CLKDIV_1](#) = 0,
[seREMC2_OSC3_CLKDIV_2](#) = 1,
[seREMC2_OSC3_CLKDIV_4](#) = 2,
[seREMC2_OSC3_CLKDIV_8](#) = 3,
[seREMC2_OSC3_CLKDIV_16](#) = 4,
[seREMC2_OSC3_CLKDIV_32](#) = 5,
[seREMC2_OSC3_CLKDIV_64](#) = 6,
[seREMC2_OSC3_CLKDIV_128](#) = 7,
[seREMC2_OSC3_CLKDIV_256](#) = 8,
[seREMC2_OSC3_CLKDIV_512](#) = 9,
[seREMC2_OSC3_CLKDIV_1024](#) = 10,
[seREMC2_OSC3_CLKDIV_2048](#) = 11,
[seREMC2_OSC3_CLKDIV_4096](#) = 12,
[seREMC2_OSC3_CLKDIV_8192](#) = 13,
[seREMC2_OSC3_CLKDIV_16384](#) = 14,
[seREMC2_OSC3_CLKDIV_32768](#) = 15 }
- enum [seREMC2_EXOSC_ClkDiv](#) { [seREMC2_EXOSC_CLKDIV_1](#) = 0 }

```
enum seREMC2_Interrupt {
    seREMC2_ALL_INT = 3,
    seREMC2_DBIF = 2,
    seREMC2_APIF = 1 }
```

2.37.1 Enumeration Type Documentation

2.37.1.1 seREMC2_EXOSC_ClkDiv

```
enum seREMC2_EXOSC_ClkDiv
```

Enumerator

seREMC2_EXOSC_CLKDIV_1	EXOSC division ratio is 1/1.
------------------------	------------------------------

2.37.1.2 seREMC2_Interrupt

```
enum seREMC2_Interrupt
```

Enumerator

seREMC2_ALL_INT	Compare all interrupts.
seREMC2_DBIF	Compare DB interrupt.
seREMC2_APIF	Compare AP interrupt.

2.37.1.3 seREMC2_IOSC_ClkDiv

```
enum seREMC2_IOSC_ClkDiv
```

Enumerator

seREMC2_IOSC_CLKDIV_1	IOSC division ratio is 1/1.
seREMC2_IOSC_CLKDIV_2	IOSC division ratio is 1/2.
seREMC2_IOSC_CLKDIV_4	IOSC division ratio is 1/4.
seREMC2_IOSC_CLKDIV_8	IOSC division ratio is 1/8.
seREMC2_IOSC_CLKDIV_16	IOSC division ratio is 1/16.
seREMC2_IOSC_CLKDIV_32	IOSC division ratio is 1/32.
seREMC2_IOSC_CLKDIV_64	IOSC division ratio is 1/64.
seREMC2_IOSC_CLKDIV_128	IOSC division ratio is 1/128.
seREMC2_IOSC_CLKDIV_256	IOSC division ratio is 1/256.
seREMC2_IOSC_CLKDIV_512	IOSC division ratio is 1/512.
seREMC2_IOSC_CLKDIV_1024	IOSC division ratio is 1/1024.
seREMC2_IOSC_CLKDIV_2048	IOSC division ratio is 1/2048.
seREMC2_IOSC_CLKDIV_4096	IOSC division ratio is 1/4096.
seREMC2_IOSC_CLKDIV_8192	IOSC division ratio is 1/8192.
seREMC2_IOSC_CLKDIV_16384	IOSC division ratio is 1/16384.

Enumerator

seREMC2_IOSC_CLKDIV_32768	IOSC division ratio is 1/32768.
---------------------------	---------------------------------

2.37.1.4 seREMC2_OSC1_ClkDiv

```
enum seREMC2_OSC1_ClkDiv
```

Enumerator

seREMC2_OSC1_CLKDIV_1	OSC1 division ratio is 1/1.
seREMC2_OSC1_CLKDIV_2	OSC1 division ratio is 1/2.
seREMC2_OSC1_CLKDIV_4	OSC1 division ratio is 1/4.
seREMC2_OSC1_CLKDIV_8	OSC1 division ratio is 1/8.
seREMC2_OSC1_CLKDIV_16	OSC1 division ratio is 1/16.
seREMC2_OSC1_CLKDIV_32	OSC1 division ratio is 1/32.
seREMC2_OSC1_CLKDIV_64	OSC1 division ratio is 1/64.
seREMC2_OSC1_CLKDIV_128	OSC1 division ratio is 1/128.
seREMC2_OSC1_CLKDIV_256	OSC1 division ratio is 1/256.

2.37.1.5 seREMC2_OSC3_ClkDiv

```
enum seREMC2_OSC3_ClkDiv
```

Enumerator

seREMC2_OSC3_CLKDIV_1	OSC3 division ratio is 1/1.
seREMC2_OSC3_CLKDIV_2	OSC3 division ratio is 1/2.
seREMC2_OSC3_CLKDIV_4	OSC3 division ratio is 1/4.
seREMC2_OSC3_CLKDIV_8	OSC3 division ratio is 1/8.
seREMC2_OSC3_CLKDIV_16	OSC1 division ratio is 1/16.
seREMC2_OSC3_CLKDIV_32	OSC1 division ratio is 1/32.
seREMC2_OSC3_CLKDIV_64	OSC1 division ratio is 1/64.
seREMC2_OSC3_CLKDIV_128	OSC1 division ratio is 1/128.
seREMC2_OSC3_CLKDIV_256	OSC1 division ratio is 1/256.
seREMC2_OSC3_CLKDIV_512	OSC1 division ratio is 1/512.
seREMC2_OSC3_CLKDIV_1024	OSC1 division ratio is 1/1024.
seREMC2_OSC3_CLKDIV_2048	OSC1 division ratio is 1/2048.
seREMC2_OSC3_CLKDIV_4096	OSC1 division ratio is 1/4096.
seREMC2_OSC3_CLKDIV_8192	OSC3 division ratio is 1/8192.
seREMC2_OSC3_CLKDIV_16384	OSC3 division ratio is 1/16384.
seREMC2_OSC3_CLKDIV_32768	OSC3 division ratio is 1/32768.

2.38 REMC2_Types

Data Structures

- struct [seREMC2_InitTypeDef](#)
REMCInit structure definition.
- struct [seREMC2_ChannelDef](#)

Variables

- [seREMC2_ChannelDef](#) **REMC2_CH**

2.39 REMC2_Functions

Functions

- `seStatus seREMC2_Init (seREMC2_ChannelDef *REMC2_CH, seREMC2_InitTypeDef *InitStruct)`
Initializes the REMC peripheral according to the specified parameters in the seREMC2_InitStruct. Stop REMC before initializing.
- `void seREMC2_ConfigureClock (REMC2_Type *REMC2x, seREMC2_ClkSrc clock, uint16_t divider)`
Configures REMC timer clock source and clock divider.
- `void seREMC2_Start (REMC2_Type *REMC2x, uint16_t aplen, uint16_t dblen)`
Starts REMC by supplying the operating clock.
- `void seREMC2_Stop (REMC2_Type *REMC2x)`
Stops the REMC module.

2.39.1 Function Documentation

2.39.1.1 seREMC2_ConfigureClock()

```
void seREMC2_ConfigureClock (
    REMC2_Type * REMC2x,
    seREMC2_ClkSrc clock,
    uint16_t divider )
```

Parameters

<i>REMC2x</i>	REMC2 peripheral.
<i>clock</i>	This parameter can be a value of seREMC2_ClkSrc.
<i>divider</i>	This parameter can be a value of seREMC2_ClkDiv.

Return values

<i>None</i>	
-------------	--

2.39.1.2 seREMC2_Init()

```
seStatus seREMC2_Init (
    seREMC2_ChannelDef * REMC2_CH,
    seREMC2_InitTypeDef * InitStruct )
```

Parameters

<i>REMC2_CH</i>	REMC2 channel definition of type <code>seREMC2_ChannelDef</code>
<i>InitStruct</i>	pointer to a <code>seREMC2_InitTypeDef</code> structure that contains the configuration information for the specified REMC peripheral. <code>seREMC2_InitTypeDef</code> .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.39.1.3 seREMC2_Start()

```
void seREMC2_Start (
    REMC2_Type * REMC2x,
    uint16_t aplen,
    uint16_t dblen )
```

Parameters

<i>REMC2x</i>	REMC2 peripheral.
<i>aplen</i>	Sets data bit active pulse length.
<i>dblen</i>	Set data bit length.

Return values

<i>None</i>	
-------------	--

2.39.1.4 seREMC2_Stop()

```
void seREMC2_Stop (
    REMC2_Type * REMC2x )
```

Parameters

<i>REMC2x</i>	REMC2 peripheral.
---------------	-------------------

Return values

<i>None</i>	
-------------	--

2.40 RFC

RFC is a R/F Converter module.

Modules

- [RFC_Constants](#)
- [RFC_Types](#)
- [RFC_Functions](#)

2.41 RFC_Constants

Macros

- `#define RFC_STS_CONV_SEN_A (1)`
Status : Sensor A oscillation completion interrupt.
- `#define RFC_STS_CONV_SEN_B (2)`
Status : Sensor B oscillation completion interrupt.
- `#define RFC_STS_CONV_ERR_OVTC (-1)`
Status : Time base counter overflow error.
- `#define RFC_STS_CONV_ERR_OVMC (-2)`
Status : Measurement counter overflow error.

Typedefs

- `typedef seCLG_ClkSrc seRFC_ClkSrc`

Enumerations

- `enum seRFC__IOSC_ClkDiv {`
 `seRFC_IOSC_CLKDIV_1 = 0,`
 `seRFC_IOSC_CLKDIV_2 = 1,`
 `seRFC_IOSC_CLKDIV_4 = 2,`
 `seRFC_IOSC_CLKDIV_8 = 3 }`
- `enum seRFC_OSC1_ClkDiv { seRFC_OSC1_CLKDIV_1 = 0 }`
- `enum seRFC_OSC3_ClkDiv {`
 `seRFC_OSC3_CLKDIV_1 = 0,`
 `seRFC_OSC3_CLKDIV_2 = 1,`
 `seRFC_OSC3_CLKDIV_4 = 2,`
 `seRFC_OSC3_CLKDIV_8 = 3 }`
- `enum seRFC_EXOSC_ClkDiv { seRFC_EXOSC_CLKDIV_1 = 0 }`
- `enum seRFC_OscMode {`
 `seRFC_DcMode = 0,`
 `seRFC_AcMode = 1 }`
- `enum seRFC_Interrupt {`
 `seRFC_OVTCIF = 0x00010UL,`
 `seRFC_OVMCIF = 0x00008UL,`
 `seRFC_ESENBIF = 0x00004UL,`
 `seRFC_ESENAIF = 0x00002UL,`
 `seRFC_EREFIF = 0x00001UL,`
 `seRFC_ALL_INT = 0x0001FUL }`

2.41.1 Enumeration Type Documentation

2.41.1.1 seRFC__IOSC_ClkDiv

```
enum seRFC__IOSC_ClkDiv
```

Enumerator

seRFC_IOSC_CLKDIV↔ _1	IOSC division ratio is 1/1.
seRFC_IOSC_CLKDIV↔ _2	IOSC division ratio is 1/2.
seRFC_IOSC_CLKDIV↔ _4	IOSC division ratio is 1/4.
seRFC_IOSC_CLKDIV↔ _8	IOSC division ratio is 1/8.

2.41.1.2 seRFC_EXOSC_ClkDiv

```
enum seRFC_EXOSC_ClkDiv
```

Enumerator

seRFC_EXOSC_CLKDIV↔ _1	EXOSC division ratio is 1/1.
---------------------------	------------------------------

2.41.1.3 seRFC_Interrupt

```
enum seRFC_Interrupt
```

Enumerator

seRFC_OVTCIF	Time base counter overflow error interrupt.
seRFC_OVMCIF	Measurement counter overflow error interrupt.
seRFC_ESENBIF	Sensor B oscillation completion interrupt.
seRFC_ESENAIF	Sensor A oscillation completion interrupt.
seRFC_EREFIG	Reference oscillation completion interrupt.

2.41.1.4 seRFC_OSC1_ClkDiv

```
enum seRFC_OSC1_ClkDiv
```

Enumerator

seRFC_OSC1_CLKDIV↔ _1	OSC1 division ratio is 1/1.
--------------------------	-----------------------------

2.41.1.5 seRFC_OSC3_ClkDiv

```
enum seRFC_OSC3_ClkDiv
```

Enumerator

seRFC_OSC3_CLKDIV↔ _1	OSC3 division ratio is 1/1.
seRFC_OSC3_CLKDIV↔ _2	OSC3 division ratio is 1/2.
seRFC_OSC3_CLKDIV↔ _4	OSC3 division ratio is 1/4.
seRFC_OSC3_CLKDIV↔ _8	OSC3 division ratio is 1/8.

2.41.1.6 seRFC_OscMode

```
enum seRFC_OscMode
```

Enumerator

seRFC_DcMode	DC oscillation mode for resistive sensor measurements.
seRFC_AcMode	AC oscillation mode for resistive sensor measurements.

2.42 RFC_Types

Data Structures

- struct [seRFC_InitTypeDef](#)
RFCInit structure definition.
- struct [seRFC_ChannelDef](#)
RFC Channel definition.

Variables

- [seRFC_ChannelDef](#) **RFC_CH0**

2.43 RFC_Functions

Functions

- `seStatus seRFC_Init (seRFC_ChannelDef *RFCCHx, seRFC_InitTypeDef *InitStruct)`
Initializes the RFC peripheral according to the specified parameters in the seRFC_InitStruct. Stop RFC before initializing.
- `void seRFC_ConfigureClock (RFC_0_Type *RFCx, seRFC_ClkSrc clock, uint16_t divider)`
Configures RFC timer clock source and clock divider.
- `void seRFC_Start (RFC_0_Type *RFCx)`
Starts RFC by supplying the operating clock.
- `void seRFC_Stop (RFC_0_Type *RFCx)`
Stops the RFC module.
- `void seRFC_DisableInt (RFC_0_Type *RFCx, seRFC_Interrupt irq)`
Disables the RFC interrupts.
- `void seRFC_EnableInt (RFC_0_Type *RFCx, seRFC_Interrupt irq)`
Enables the RFC interrupts.
- `void seRFC_ClearIntFlag (RFC_0_Type *RFCx, seRFC_Interrupt irq)`
Clears Interrupt flags.
- `void seRFC_SetMeasurementCounter (RFC_0_Type *RFCx, uint32_t count)`
Set measurement counter data.
- `uint32_t seRFC_GetMeasurementCounter (RFC_0_Type *RFCx)`
Get measurement counter data.
- `void seRFC_SetTimeBaseCounter (RFC_0_Type *RFCx, uint32_t count)`
Set Time Base Counter.
- `uint32_t seRFC_GetTimeBaseCounter (RFC_0_Type *RFCx)`
Get Time Base Counter.
- `int16_t seRFC_RunConvertingOperation (RFC_0_Type *RFCx, uint32_t startCount, uint32_t *sensorCount)`
Executes converting operation.
- `seStatus ConfigurePortsForRFC (seRFC_ChannelDef *RFCCHx)`
Configures ports for this module. Override this function to configure specific ports.

2.43.1 Function Documentation

2.43.1.1 ConfigurePortsForRFC()

```
seStatus ConfigurePortsForRFC (
    seRFC_ChannelDef * RFCCHx )
```

Parameters

<code>RFCCHx</code>	RFC channel definition of type <code>seRFC_ChannelDef</code>
---------------------	--

Return values

<code>Status</code>	can be a value of <code>seStatus</code>
---------------------	---

2.43.1.2 seRFC_ClearIntFlag()

```
void seRFC_ClearIntFlag (
    RFC_0_Type * RFCx,
    seRFC_Interrupt irq )
```

Parameters

<i>RFCx</i>	RFC peripheral
<i>irq</i>	This parameter can be a value of seRFC_Interrupt .

Return values

<i>None</i>	
-------------	--

2.43.1.3 seRFC_ConfigureClock()

```
void seRFC_ConfigureClock (
    RFC_0_Type * RFCx,
    seRFC_ClkSrc clock,
    uint16_t divider )
```

Parameters

<i>RFCx</i>	RFC peripheral
<i>clock</i>	This parameter can be a value of seRFC_ClkSrc .
<i>divider</i>	This parameter can be a value of seRFC_ClkDiv .

Return values

<i>None</i>	
-------------	--

2.43.1.4 seRFC_DisableInt()

```
void seRFC_DisableInt (
    RFC_0_Type * RFCx,
    seRFC_Interrupt irq )
```

Parameters

<i>RFCx</i>	RFC peripheral
<i>irq</i>	This parameter can be a value of seRFC_Interrupt .

Return values

<i>None</i>	
-------------	--

2.43.1.5 seRFC_EnableInt()

```
void seRFC_EnableInt (
    RFC_0_Type * RFCx,
    seRFC_Interrupt irq )
```

Parameters

<i>RFCx</i>	RFC peripheral
<i>irq</i>	This parameter can be a value of seRFC_Interrupt .

Return values

<i>None</i>	
-------------	--

2.43.1.6 seRFC_GetMeasurementCounter()

```
uint32_t seRFC_GetMeasurementCounter (
    RFC_0_Type * RFCx )
```

Parameters

<i>RFCx</i>	RFC peripheral
-------------	----------------

Return values

<i>Measurement</i>	counter data.
--------------------	---------------

2.43.1.7 seRFC_GetTimeBaseCounter()

```
uint32_t seRFC_GetTimeBaseCounter (
    RFC_0_Type * RFCx )
```

Parameters

<i>RFCx</i>	RFC peripheral
-------------	----------------

Return values

<i>Measurement</i>	count.
--------------------	--------

2.43.1.8 seRFC_Init()

```
seStatus seRFC_Init (
```

```

seRFC_ChannelDef * RFCCHx,
seRFC_InitTypeDef * InitStruct )

```

Parameters

<i>RFCCHx</i>	RFC channel definition of type seRFC_ChannelDef
<i>InitStruct</i>	pointer to a seRFC_InitTypeDef structure that contains the configuration information for the specified RFC peripheral. seRFC_InitTypeDef .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.43.1.9 seRFC_RunConvertingOperation()

```

int16_t seRFC_RunConvertingOperation (
    RFC_0_Type * RFCx,
    uint32_t startCount,
    uint32_t * sensorCount )

```

Parameters

<i>startCount</i>	Start Count.
<i>sensorCount</i>	Sensor Count.

Return values

<i>Measurement</i>	count.
--------------------	--------

2.43.1.10 seRFC_SetMeasurementCounter()

```

void seRFC_SetMeasurementCounter (
    RFC_0_Type * RFCx,
    uint32_t count )

```

Parameters

<i>RFCx</i>	RFC peripheral
<i>count</i>	Measurement counter data.

2.43.1.11 seRFC_SetTimeBaseCounter()

```

void seRFC_SetTimeBaseCounter (
    RFC_0_Type * RFCx,
    uint32_t count )

```

Parameters

<i>RFCx</i>	RFC peripheral
<i>count</i>	Measurement count.

2.43.1.12 seRFC_Start()

```
void seRFC_Start (
    RFC_0_Type * RFCx )
```

Parameters

<i>RFCx</i>	RFC peripheral
-------------	----------------

Return values

<i>None</i>	
-------------	--

2.43.1.13 seRFC_Stop()

```
void seRFC_Stop (
    RFC_0_Type * RFCx )
```

Parameters

<i>RFCx</i>	RFC peripheral
-------------	----------------

Return values

<i>None</i>	
-------------	--

2.44 RTCA

RTCA is a real-time clock with a perpetual calendar function.

Modules

- [RTCA_Constants](#)
- [RTCA_Types](#)
- [RTCA_Functions](#)

2.45 RTCA_Constants

Macros

- #define `seRTCA_INTS(a)` ((`seRTCA_Interrupt`)((a)))
Combination of any of the seRTCA_Interrupt enumerations.
- #define `seRTCA_VALID_12HOURS(h)` ((h) <= 12 && (h) > 0)
- #define `seRTCA_VALID_24HOURS(h)` ((h) < 24)
- #define `seRTCA_VALID_MINUTES(m)` ((m) < 60)
- #define `seRTCA_VALID_SECONDS(s)` ((s) < 60)

Enumerations

- enum `seRTCA_Hours12_24` {
 `seRTCA_CTL_RTC24H_12` = 0,
 `seRTCA_CTL_RTC24H_24` = 1 }
- enum `seRTCA_AM_PM` {
 `seRTCA_HUR_RTCAP_AM` = 0,
 `seRTCA_HUR_RTCAP_PM` = 1 }
- enum `seRTCA_DayOfTheWeek` {
 `seRTCA_YAR_RTCWK_SUN` = 0,
 `seRTCA_YAR_RTCWK_MON` = 1,
 `seRTCA_YAR_RTCWK_TUE` = 2,
 `seRTCA_YAR_RTCWK_WED` = 3,
 `seRTCA_YAR_RTCWK_THU` = 4,
 `seRTCA_YAR_RTCWK_FRI` = 5,
 `seRTCA_YAR_RTCWK_SAT` = 6 }
- enum `seRTCA_Interrupt` {
 `seRTCA_1_32SECI` = 0x0001U,
 `seRTCA_1_8SECI` = 0x0002U,
 `seRTCA_1_4SECI` = 0x0004U,
 `seRTCA_1_2SECI` = 0x0008U,
 `seRTCA_1SECI` = 0x0010U,
 `seRTCA_1MINI` = 0x0020U,
 `seRTCA_1HURI` = 0x0040U,
 `seRTCA_1DAYI` = 0x0080U,
 `seRTCA_ALARM1` = 0x0100U,
 `seRTCA_SW100I` = 0x1000U,
 `seRTCA_SW10I` = 0x2000U,
 `seRTCA_SW1I` = 0x4000U,
 `seRTCA_RTCTRM1` = 0x8000U,
 `seRTCA_ALL_INT` = 0xF1FFU }

2.45.1 Enumeration Type Documentation

2.45.1.1 seRTCA_AM_PM

enum `seRTCA_AM_PM`

Enumerator

<code>seRTCA_HUR_RTCAP_AM</code>	Mode - A.M.
----------------------------------	-------------

Enumerator

seRTCA_HUR_RTCAP_PM	Mode - P.M.
---------------------	-------------

2.45.1.2 seRTCA_DayOfTheWeek

enum `seRTCA_DayOfTheWeek`

Enumerator

seRTCA_YAR_RTCWK_SUN	Day of the week - Sunday.
seRTCA_YAR_RTCWK_MON	Day of the week - Monday.
seRTCA_YAR_RTCWK_TUE	Day of the week - Tuesday.
seRTCA_YAR_RTCWK_WED	Day of the week - Wednesday.
seRTCA_YAR_RTCWK_THU	Day of the week - Thursday.
seRTCA_YAR_RTCWK_FRI	Day of the week - Friday.
seRTCA_YAR_RTCWK_SAT	Day of the week - Saturday.

2.45.1.3 seRTCA_Hours12_24

enum `seRTCA_Hours12_24`

Enumerator

seRTCA_CTL_RTC24H_12	Hour counter - 12H mode.
seRTCA_CTL_RTC24H_24	Hour counter - 24H mode.

2.45.1.4 seRTCA_Interrupt

enum `seRTCA_Interrupt`

Enumerator

seRTCA_1_32SECI	1/32 second interrupt.
seRTCA_1_8SECI	1/8 second interrupt.
seRTCA_1_4SECI	1/4 second interrupt.
seRTCA_1_2SECI	1/2 second interrupt.
seRTCA_1SECI	1 second interrupt.
seRTCA_1MINI	1 minute interrupt.
seRTCA_1HURI	1 hour interrupt.
seRTCA_1DAYI	1 day interrupt.
seRTCA_ALARMI	ALARM interrupt.
seRTCA_SW100I	100 Hz interrupt.
seRTCA_SW10I	10 Hz interrupt.
seRTCA_SW1I	1 Hz interrupt.

Enumerator

seRTCA_RTCTRM	Theoretical regulation interrupt.
seRTCA_ALL_INT	All.

2.46 RTCA_Types

Data Structures

- struct [swCounter](#)
RTCA Stopwatch counter structure definition.
- struct [seRTCA_InitTypeDef](#)
RTCA Init structure definition.

2.47 RTCA_Functions

Functions

- **seStatus seRTCA_Init** (seRTCA_InitTypeDef *InitStruct)
Initializes the RTCA peripheral according to the specified parameters in the seRTCA_InitStruct.
- **seStatus seRTCA_Start** (void)
Starts RTCA channel by start supplying operating clock.
- **void seRTCA_Stop** (void)
Stops RTCA channel by stop supplying operating clock.
- **void RTCA_IRQHandler** (void)
RTCA Interrupt Service Routine.
- **void seRTCA_EnableInt** (seRTCA_Interrupt irq)
Enables RTCA channel interrupt.
- **void seRTCA_DisableInt** (seRTCA_Interrupt irq)
Disables RTCA channel interrupt.
- **seInterruptStatus seRTCA_GetIntFlag** (seRTCA_Interrupt irq)
Returns RTCA interrupt flag.
- **void seRTCA_ClearIntFlag** (seRTCA_Interrupt irq)
Clears RTCA channel interrupt.
- **void seRTCA_Enable1SecTimer** (void)
Start 1 sec timer. The 1-second interrupt of RTCA is enabled. User shall enable interrupt globally and provide an interrupt service routine.
- **void seRTCA_Disable1SecTimer** (void)
Stops 1 sec timer. The 1-second interrupt of RTCA is disabled. User shall provide appropriate actions in the interrupt service routine.
- **uint8_t seRTCA_CalculateTrm** (int32_t m, int32_t n)
Calculate theoretical regulation bits.
- **seStatus seRTCA_Set12_24Mode** (seRTCA_Hours12_24 mode)
Writes 12/24 mode to hardware.
- **seRTCA_Hours12_24 seRTCA_Get12_24Mode** (void)
Reads 12/24 mode from hardware.
- **seStatus seRTCA_SetYearMonthDayWeek** (uint8_t year, uint8_t month, uint8_t day, uint8_t week)
Setting a year, month, day, day of the week in hardware. When a value out of the effective range is set to the year, day of the week, the counter will be cleared to 0 at the next count-up timing. When a such value is set to the month, day, the counter will be set to 1 at the next count-up timing.
- **seStatus seRTCA_GetYearMonthDayWeek** (uint8_t *year, uint8_t *month, uint8_t *day, uint8_t *week)
Getting a year, month, day, day of the week from hardware.
- **seStatus seRTCA_SetHourMinuteSecond** (uint8_t h, uint8_t m, uint8_t s)
Setting an hour, minute, second in hardware. When a value out of the effective range is set to hour (in 24H mode) counter, the counter will be cleared to 0 at the next count-up timing. When a such value is set to hour (in 12H mode) counter, the counter will be set to 1 at the next count-up timing.
- **seStatus seRTCA_GetHourMinuteSecond** (uint8_t *h, uint8_t *m, uint8_t *s)
Getting an hour, minute, second from hardware.
- **seStatus seRTCA_SetAM_PM** (seRTCA_AM_PM indicator)
Set AM/PM indicator.
- **seRTCA_AM_PM seRTCA_GetAM_PM** (void)
Get AM/PM indicator.

- **seStatus seRTCA_SetAlarm** (uint8_t h, uint8_t m, uint8_t s, **seRTCA_AM_PM** am)
Sets an alarm clock and enables alarm interrupt, user provides an interrupt service routine and enables interrupt globally.
- **void seRTCA_GetAlarm** (uint8_t *h, uint8_t *m, uint8_t *s, **seRTCA_AM_PM** *am)
Gets an alarm clock and enables alarm interrupt, user provides an interrupt service routine and enables interrupt globally.
- **seStatus seRTCA_SetSecondsAlarm** (uint32_t alarmsec)
*Sets an alarm in number of seconds from current time. **seRTCA_SetAlarm()** is called. User provides an interrupt service routine and enables interrupt globally.*
- **seStatus seRTCA_Set30secCorrection** (void)
Set 30-second correction. This function is provided to set the time-of-day clock by the time signal. Writing 1 to the CTL RTCADJ bit adds 1 to the minute counter if the second counter represents 30 to 59 seconds, or clears the second counter with the minute counter left unchanged if the second counter represents 0 to 29 seconds.
- **uint8_t seRTCA_CalcWeekDay** (uint8_t y, uint8_t m, uint8_t d)
Calculate weekday from year, month, day.
- **void seRTCA_StartStopWatchCount** (**seRTCA_Interrupt** irqs)
Start Stopwatch counter.
- **void seRTCA_StopStopWatchCount** (**seRTCA_Interrupt** irqs)
Stop Stopwatch counter.
- **void seRTCA_ResetStopWatchCount** (**swCounter** *StopWatchCounter)
Reset Stopwatch counter and set members of StopWatchCounter to 0.
- **seStatus seRTCA_ReadStopWatchCount** (**swCounter** *StopWatchCounter)
Read the 1/10-second and 1/100-second digits of the stopwatch counter. Note, that seconds and minutes should be assigned outside of this function.
- **seStatus seRTCA_InitTheoreticalRegulation** (uint32_t sampling_period, int16_t curr_freqerr_mHz)
*Initialize Theoretical Regulation variables and set RTC alarm for next adjustment. **seRTCA_SetAlarm()** is called. User provides an interrupt service routine and enables interrupt globally.*
- **seStatus seRTCA_TheoreticalRegulationTrim** (int16_t curr_freqerr_mHz)
*Theoretical Regulation adjustment calculation/trimming and set RTC alarm for next adjustment. **seRTCA_SetAlarm()** is called. User provides an interrupt service routine and enables interrupt globally.*

2.47.1 Function Documentation

2.47.1.1 RTCA_IRQHandler()

```
void RTCA_IRQHandler (
    void )
```

Return values

None	
------	--

2.47.1.2 seRTCA_CalculateTrm()

```
uint8_t seRTCA_CalculateTrm (
    int32_t m,
    int32_t n )
```

1. Measure the frequency tolerance "m [ppm]" of fOSC1.

2. Determine the theoretical regulation execution cycle time "n seconds." The correction value for theoretical regulation can be specified within the range from -64 to +63 and it should be written to the CTL register's RTCTRM[6:0] bits as a twos-complement number.

Parameters

<i>m</i>	This parameter is the frequency tolerance in ppm time ten (to preserve accuracy in integer math.).
<i>n</i>	This param is execution cycle time in seconds.

Return values

<i>This</i>	function returns TRM bits.
-------------	----------------------------

2.47.1.3 seRTCA_CalcWeekDay()

```
uint8_t seRTCA_CalcWeekDay (
    uint8_t y,
    uint8_t m,
    uint8_t d )
```

Parameters

<i>y</i>	Integer year value.
<i>m</i>	Integer month value.
<i>d</i>	Integer day of the month value.

Return values

<i>weekday</i>	seRTCA_DayOfTheWeek .
----------------	---------------------------------------

2.47.1.4 seRTCA_ClearIntFlag()

```
void seRTCA_ClearIntFlag (
    seRTCA\_Interrupt irq )
```

Parameters

<i>irq</i>	Selected interrupt flag seRTCA_Interrupt
------------	--

Return values

<i>None</i>	
-------------	--

2.47.1.5 seRTCA_Disable1SecTimer()

```
void seRTCA_Disable1SecTimer (  
    void )
```

Return values

None	
------	--

2.47.1.6 seRTCA_DisableInt()

```
void seRTCA_DisableInt (  
    seRTCA_Interrupt irq )
```

Parameters

irq	Selected interrupt flag seRTCA_Interrupt
-----	--

Return values

None	
------	--

2.47.1.7 seRTCA_Enable1SecTimer()

```
void seRTCA_Enable1SecTimer (  
    void )
```

Return values

None	
------	--

2.47.1.8 seRTCA_EnableInt()

```
void seRTCA_EnableInt (  
    seRTCA_Interrupt irq )
```

Parameters

irq	Selected interrupt flag seRTCA_Interrupt
-----	--

Return values

None	
------	--

2.47.1.9 seRTCA_Get12_24Mode()

```
seRTCA_Hours12_24 seRTCA_Get12_24Mode (
    void )
```

Return values

12/24	Mode: a value of
-------	------------------

2.47.1.10 seRTCA_GetAlarm()

```
void seRTCA_GetAlarm (
    uint8_t * h,
    uint8_t * m,
    uint8_t * s,
    seRTCA_AM_PM * am )
```

Parameters

<i>h</i>	pointer to Integer hour value from 0 to 24 seRTCA_VALID_24HOURS.
<i>m</i>	pointer to Integer minute value from 0 to 60 seRTCA_MINUTES.
<i>s</i>	pointer to Integer second value from 0 to 60 seRTCA_VALID_SECONDS.
<i>am</i>	pointer to AM/FM settings seRTCA_AM_PM.

Return values

None	
------	--

2.47.1.11 seRTCA_GetAM_PM()

```
seRTCA_AM_PM seRTCA_GetAM_PM (
    void )
```

Return values

indicator	seRTCA_AM_PM.
-----------	---------------

2.47.1.12 seRTCA_GetHourMinuteSecond()

```
seStatus seRTCA_GetHourMinuteSecond (
    uint8_t * h,
    uint8_t * m,
    uint8_t * s )
```

Parameters

<i>h</i>	Pointer to integer hour value.
----------	--------------------------------

Parameters

<i>m</i>	Pointer to integer minute value.
<i>s</i>	Pointer to integer second value.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.13 seRTCA_GetIntFlag()

```
seInterruptStatus seRTCA_GetIntFlag (
    seRTCA_Interrupt irq )
```

Parameters

<i>irq</i>	Selected interrupt flag seRTCA_Interrupt
------------	--

Return values

<i>InterruptStatus</i>	
------------------------	--

2.47.1.14 seRTCA_GetYearMonthDayWeek()

```
seStatus seRTCA_GetYearMonthDayWeek (
    uint8_t * year,
    uint8_t * month,
    uint8_t * day,
    uint8_t * week )
```

Parameters

<i>year</i>	Integer year value (from 0 to 99).
<i>month</i>	Integer month value (from 1 to 12).
<i>day</i>	Integer day value (from 1 to 31).
<i>week</i>	Integer day of the week value of type seRTCA_DayOfTheWeek .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.15 seRTCA_Init()

```
seStatus seRTCA_Init (
```

```
seRTCA_InitTypeDef * InitStruct )
```

Parameters

<i>InitStruct</i>	pointer to a seRTCA_InitTypeDef structure that contains the configuration information for the specified RTCA peripheral.
-------------------	--

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.16 seRTCA_InitTheoreticalRegulation()

```
seStatus seRTCA_InitTheoreticalRegulation (
    uint32_t sampling_period,
    int16_t curr_freqerr_mHz )
```

Parameters

<i>sampling_period</i>	sampling period for Theoretical Regulation adjustments.
<i>curr_freqerr_mHz</i>	current OSC1 frequency error in milliHertz.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.17 seRTCA_ReadStopWatchCount()

```
seStatus seRTCA_ReadStopWatchCount (
    swCounter * StopWatchCounter )
```

Parameters

<i>StopWatchCounter</i>	pointer to StopWatchCounter value of the stopwatch counter
-------------------------	--

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.18 seRTCA_ResetStopWatchCount()

```
void seRTCA_ResetStopWatchCount (
    swCounter * StopWatchCounter )
```

Parameters

<i>StopWatchCounter</i>	1Hz, 10Hz and 100Hz Interrupt, see swCounter
-------------------------	--

Return values

<i>None</i>	
-------------	--

2.47.1.19 seRTCA_Set12_24Mode()

```
seStatus seRTCA_Set12_24Mode (
    seRTCA_Hours12_24 mode )
```

Parameters

<i>mode</i>	12/24 mode
-------------	------------

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.20 seRTCA_Set30secCorrection()

```
seStatus seRTCA_Set30secCorrection (
    void )
```

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.21 seRTCA_SetAlarm()

```
seStatus seRTCA_SetAlarm (
    uint8_t h,
    uint8_t m,
    uint8_t s,
    seRTCA_AM_PM am )
```

Parameters

<i>Integer</i>	hour value from 0 to 24 seRTCA_VALID_24HOURS.
<i>Integer</i>	minute value from 0 to 60 seRTCA_MINUTES.
<i>Integer</i>	second value from 0 to 60 seRTCA_VALID_SECONDS.
<i>AM/FM</i>	settings seRTCA_AM_PM .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.22 seRTCA_SetAM_PM()

```
seStatus seRTCA_SetAM_PM (
    seRTCA_AM_PM indicator )
```

Parameters

<i>indicator</i>	AM/PM indicator. seRTCA_AM_PM_indicator
------------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.23 seRTCA_SetHourMinuteSecond()

```
seStatus seRTCA_SetHourMinuteSecond (
    uint8_t h,
    uint8_t m,
    uint8_t s )
```

Parameters

<i>h</i>	Integer hour value.
<i>m</i>	Integer minute value.
<i>s</i>	Integer second value.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.24 seRTCA_SetSecondsAlarm()

```
seStatus seRTCA_SetSecondsAlarm (
    uint32_t alarmsec )
```

Parameters

<i>alarmsec</i>	alarm seconds from current time.
-----------------	----------------------------------

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.25 seRTCA_SetYearMonthDayWeek()

```
seStatus seRTCA_SetYearMonthDayWeek (
    uint8_t year,
    uint8_t month,
    uint8_t day,
    uint8_t week )
```

Parameters

<i>year</i>	Integer year value (from 0 to 99).
<i>month</i>	Integer month value (from 1 to 12).
<i>day</i>	Integer day value (from 1 to 31).
<i>week</i>	Integer day of the week value of type seRTCA_DayOfTheWeek .

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.26 seRTCA_Start()

```
seStatus seRTCA_Start (
    void )
```

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.47.1.27 seRTCA_StartStopWatchCount()

```
void seRTCA_StartStopWatchCount (
    seRTCA_Interrupt irqs )
```

Parameters

<i>irqs</i>	1Hz, 10Hz and 100Hz Interrupt, see seRTCA_Interrupt
-------------	---

Return values

<i>None</i>	
-------------	--

2.47.1.28 seRTCA_Stop()

```
void seRTCA_Stop (
    void )
```

Return values

<i>None</i>	
-------------	--

2.47.1.29 seRTCA_StopStopWatchCount()

```
void seRTCA_StopStopWatchCount (
    seRTCA_Interrupt irqs )
```

Parameters

<i>irqs</i>	1Hz, 10Hz and 100Hz Interrupt, see seRTCA_Interrupt
-------------	---

Return values

<i>None</i>	
-------------	--

2.47.1.30 seRTCA_TheoreticalRegulationTrim()

```
seStatus seRTCA_TheoreticalRegulationTrim (
    int16_t curr_freqerr_mHz )
```

Parameters

<i>curr_freqerr_mHz</i>	current OSC1 frequency error in milliHertz.
-------------------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.48 SNDA

SNDA is a sound generator that generates melodies and buzzer signals.

Modules

- [SNDA_Constants](#)
- [SNDA_Types](#)
- [SNDA_Functions](#)

2.49 SNDA_Constants

Typedefs

- typedef [seCLG_ClkSrc](#) [seSNDA_ClkSrc](#)
Brief description.

Enumerations

- enum [seSNDA_IOSC_ClkDiv](#) {
 [seSNDA_IOSC_CLKDIV_16](#) = 0,
 [seSNDA_IOSC_CLKDIV_32](#) = 1,
 [seSNDA_IOSC_CLKDIV_64](#) = 2,
 [seSNDA_IOSC_CLKDIV_128](#) = 3,
 [seSNDA_IOSC_CLKDIV_256](#) = 4,
 [seSNDA_IOSC_CLKDIV_512](#) = 5 }
- enum [seSNDA_OSC1_ClkDiv](#) { [seSNDA_OSC1_CLKDIV_1](#) = 0 }
- enum [seSNDA_OSC3_ClkDiv](#) {
 [seSNDA_OSC3_CLKDIV_16](#) = 0,
 [seSNDA_OSC3_CLKDIV_32](#) = 1,
 [seSNDA_OSC3_CLKDIV_64](#) = 2,
 [seSNDA_OSC3_CLKDIV_128](#) = 3,
 [seSNDA_OSC3_CLKDIV_256](#) = 4,
 [seSNDA_OSC3_CLKDIV_512](#) = 5 }
- enum [seSNDA_EXOSC_ClkDiv](#) { [seSNDA_EXOSC_CLKDIV_1](#) = 0 }
- enum [seSNDA_DriveMode](#) {
 [seSNDA_SingleDrive](#) = 1,
 [seSNDA_DirectDrive](#) = 0 }
- enum [seSNDA_ModeSel](#) {
 [seSNDA_NormalBuzzer](#) = 0,
 [seSNDA_OneShotBuzzer](#) = 1,
 [seSNDA_Melody](#) = 2 }
- enum [seSNDA_InterruptSrc](#) {
 [seSNDA_ED_INT](#) = 1,
 [seSNDA_EM_INT](#) = 2,
 [seSNDA_ALL_INT](#) = 3 }

2.49.1 Enumeration Type Documentation

2.49.1.1 seSNDA_DriveMode

enum [seSNDA_DriveMode](#)

Enumerator

seSNDA_SingleDrive	Single Pin Drive.
seSNDA_DirectDrive	Direct Drive.

2.49.1.2 seSNDA_EXOSC_ClkDiv

```
enum seSNDA_EXOSC_ClkDiv
```

Enumerator

seSNDA_EXOSC_CLKDIV↔ _1	EXOSC division ratio is 1/1.
----------------------------	------------------------------

2.49.1.3 seSNDA_InterruptSrc

```
enum seSNDA_InterruptSrc
```

Enumerator

seSNDA_ED_INT	Sound output completion interrupt.
seSNDA_EM_INT	Sound buffer empty interrupt.
seSNDA_ALL_INT	All interrupts.

2.49.1.4 seSNDA_IOSC_ClkDiv

```
enum seSNDA_IOSC_ClkDiv
```

Enumerator

seSNDA_IOSC_CLKDIV_16	IOSC division ratio is 1/16.
seSNDA_IOSC_CLKDIV_32	IOSC division ratio is 1/32.
seSNDA_IOSC_CLKDIV_64	IOSC division ratio is 1/64.
seSNDA_IOSC_CLKDIV_128	IOSC division ratio is 1/128.
seSNDA_IOSC_CLKDIV_256	IOSC division ratio is 1/256.
seSNDA_IOSC_CLKDIV_512	IOSC division ratio is 1/512.

2.49.1.5 seSNDA_ModeSel

```
enum seSNDA_ModeSel
```

Enumerator

seSNDA_NormalBuzzer	Normal buzzer mode.
seSNDA_OneShotBuzzer	One-shot buzzer mode.
seSNDA_Melody	Melody mode.

2.49.1.6 seSNDA_OSC1_ClkDiv

```
enum seSNDA_OSC1_ClkDiv
```

Enumerator

seSNDA_OSC1_CLKDIV↔ _1	OSC1 division ratio is 1/1.
---------------------------	-----------------------------

2.49.1.7 seSNDA_OSC3_ClkDiv

```
enum seSNDA_OSC3_ClkDiv
```

Enumerator

seSNDA_OSC3_CLKDIV_16	OSC3 division ratio is 1/16.
seSNDA_OSC3_CLKDIV_32	OSC3 division ratio is 1/32.
seSNDA_OSC3_CLKDIV_64	OSC3 division ratio is 1/64.
seSNDA_OSC3_CLKDIV_128	OSC3 division ratio is 1/128.
seSNDA_OSC3_CLKDIV_256	OSC3 division ratio is 1/256.
seSNDA_OSC3_CLKDIV_512	OSC3 division ratio is 1/512.

2.50 SNDA_Types

Data Structures

- struct [seSNDA_InitTypeDef](#)
SNDA Init structure definition.
- struct [seSNDA_ChannelDef](#)

Variables

- [seSNDA_ChannelDef](#) **SNDA_CH**

2.51 SNDA_Functions

Functions

- void [seSNDA_InitStruct](#) ([seSNDA_InitTypeDef](#) *SNDA_InitStruct)
Fills each [seSNDA_InitTypeDef](#) member with its default value.
- [seStatus seSNDA_Init](#) ([seSNDA_ChannelDef](#) *SNDA_CH, [seSNDA_InitTypeDef](#) *SNDA_InitStruct)
Initializes the SNDA peripheral according to the specified parameters in the [seSNDA_InitStruct](#).
- void [seSNDA_Start](#) (SNDA_Type *SNDAx, uint16_t frequency, uint16_t duty_ratio)
Starts SNDA in normal mode.
- void [seSNDA_StartOneShot](#) (SNDA_Type *SNDAx, uint16_t frequency, uint16_t duty_ratio, uint16_t duration)
Starts SNDA in one-shot mode.
- void [seSNDA_StartMelody](#) (SNDA_Type *SNDAx, const uint16_t *data, uint32_t size, uint16_t tempo)
Starts SNDA in melody mode.
- [seStatus seSNDA_Stop](#) (SNDA_Type *SNDAx)
Stops SNDA.
- void [seSNDA_Enable](#) (SNDA_Type *SNDAx)
Enables SNDA by start supplying operating clock.
- void [seSNDA_Disable](#) (SNDA_Type *SNDAx)
Disables SNDA by stop supplying operating clock.
- void [seSNDA_ConfigureClock](#) (SNDA_Type *SNDAx, [seSNDA_ClkSrc](#) clock, uint16_t divider)
Configures clock source and clock divider.
- uint32_t [seSNDA_GetCik](#) (SNDA_Type *SNDAx)
Discovers SNDA clock from registers.
- void [seSNDA_EnableInt](#) (SNDA_Type *SNDAx, [seSNDA_InterruptSrc](#) irq)
Enables SNDA interrupt.
- void [seSNDA_DisableInt](#) (SNDA_Type *SNDAx, [seSNDA_InterruptSrc](#) irq)
Disables SNDA interrupt.
- [seInterruptStatus seSNDA_GetIntFlag](#) (SNDA_Type *SNDAx, [seSNDA_InterruptSrc](#) irq)
Returns SNDA interrupt status.
- void [seSNDA_ClearIntFlag](#) (SNDA_Type *SNDAx)
Clears SNDA interrupt.
- void [SNDA_IRQHandler](#) (void)
SNDA Interrupt Service Routine.

2.51.1 Function Documentation

2.51.1.1 [seSNDA_ClearIntFlag\(\)](#)

```
void seSNDA_ClearIntFlag (
    SNDA_Type * SNDAx )
```

Parameters

SNDAx	SNDA peripheral.
-----------------------	------------------

Return values

<i>None</i>	
-------------	--

2.51.1.2 seSNDA_ConfigureClock()

```
void seSNDA_ConfigureClock (
    SNDA_Type * SNDAX,
    seSNDA_ClkSrc clock,
    uint16_t divider )
```

Parameters

<i>SNDAX</i>	SNDA peripheral.
<i>clock</i>	This parameter can be a value of seSNDA_ClkSrc .
<i>divider</i>	This parameter can be a value of seSNDA_ClkDiv .

Return values

<i>None</i>	
-------------	--

2.51.1.3 seSNDA_Disable()

```
void seSNDA_Disable (
    SNDA_Type * SNDAX )
```

Parameters

<i>SNDAX</i>	SNDA peripheral.
--------------	------------------

Return values

<i>None</i>	
-------------	--

2.51.1.4 seSNDA_DisableInt()

```
void seSNDA_DisableInt (
    SNDA_Type * SNDAX,
    seSNDA_InterruptSrc irq )
```

Parameters

<i>SNDAX</i>	SNDA peripheral.
<i>irq</i>	This parameter can be a value of .

Return values

<i>None</i>	
-------------	--

2.51.1.5 seSNDA_Enable()

```
void seSNDA_Enable (
    SNDA_Type * SNDAx )
```

Parameters

<i>SNDAx</i>	SNDA peripheral.
--------------	------------------

Return values

<i>None</i>	
-------------	--

2.51.1.6 seSNDA_EnableInt()

```
void seSNDA_EnableInt (
    SNDA_Type * SNDAx,
    seSNDA_InterruptSrc irq )
```

Parameters

<i>SNDAx</i>	SNDA peripheral.
<i>irq</i>	This parameter can be a value of .

Return values

<i>None</i>	
-------------	--

2.51.1.7 seSNDA_GetClk()

```
uint32_t seSNDA_GetClk (
    SNDA_Type * SNDAx )
```

Parameters

<i>SNDAx</i>	SNDA peripheral.
--------------	------------------

Return values

<i>SNDA</i>	clock.
-------------	--------

2.51.1.8 `seSNDA_GetIntFlag()`

```
seInterruptStatus seSNDA_GetIntFlag (
    SNDA_Type * SNDAx,
    seSNDA_InterruptSrc irq )
```

Parameters

<i>SNDAx</i>	SNDA peripheral.
<i>irq</i>	This parameter can be a value of seSNDA_InterruptSrc .

Return values

<i>InterruptStatus</i>	see seInterruptStatus
------------------------	---------------------------------------

2.51.1.9 `seSNDA_Init()`

```
seStatus seSNDA_Init (
    seSNDA_ChannelDef * SNDA_CH,
    seSNDA_InitTypeDef * SNDA_InitStruct )
```

Note

This function configures the module, and module's interrupts. It clears module's interrupts but does not enable interrupt from the module to CPU. This function enables module by start supplying operating clock.

Parameters

<i>SNDA_CH</i>	SNDA channel definition of type seSNDA_ChannelDef
<i>InitStruct</i>	pointer to a seSNDA_InitTypeDef structure that contains the configuration information for the specified SNDA peripheral.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.51.1.10 `seSNDA_InitStruct()`

```
void seSNDA_InitStruct (
    seSNDA_InitTypeDef * SNDA_InitStruct )
```

Parameters

<i>SNDA_InitStruct</i>	pointer to an seSNDA_InitTypeDef structure which will be initialized.
------------------------	---

Return values

<i>None</i>	
-------------	--

2.51.1.11 seSNDA_Start()

```
void seSNDA_Start (
    SNDA_Type * SNDAX,
    uint16_t frequency,
    uint16_t duty_ratio )
```

Parameters

<i>SNDAX</i>	SNDA peripheral.
<i>frequency</i>	This parameter defines frequency.
<i>duty_ratio</i>	This parameter defines duty ratio.

Return values

<i>None</i>	
-------------	--

2.51.1.12 seSNDA_StartMelody()

```
void seSNDA_StartMelody (
    SNDA_Type * SNDAX,
    const uint16_t * data,
    uint32_t size,
    uint16_t tempo )
```

Parameters

<i>SNDAX</i>	SNDA peripheral.
<i>data</i>	This parameter defines data of music.
<i>size</i>	This parameter defines size of music data by number of words.
<i>tempo</i>	This parameter defines tempo of playing.

Return values

<i>None</i>	
-------------	--

2.51.1.13 seSNDA_StartOneShot()

```
void seSNDA_StartOneShot (
    SNDA_Type * SNDAX,
    uint16_t frequency,
```

```
uint16_t duty_ratio,  
uint16_t duration )
```

Parameters

<i>SNDAx</i>	SNDA peripheral.
<i>frequency</i>	This parameter defines frequency.
<i>duty_ratio</i>	This parameter defines duty ratio.
<i>duration</i>	This parameter defines duty duration.

Return values

<i>None</i>	
-------------	--

2.51.1.14 seSNDA_Stop()

```
seStatus seSNDA_Stop (  
    SNDA_Type * SNDAx )
```

Parameters

<i>SNDAx</i>	SNDA peripheral.
--------------	------------------

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.51.1.15 SNDA_IRQHandler()

```
void SNDA_IRQHandler (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.52 SPIA

The SPIA module is a subset of the SPI bus interface.

Modules

- [SPIA_Constants](#)

The SPIA module exported constants.

- [SPIA_Types](#)
- [SPIA_Functions](#)

2.53 SPIA_Constants

The SPIA module exported constants.

Macros

- #define **seSPIA_FLGS**(a) ((seSPIA_InterruptFlags)((a)))
Combination of any of seSPIA_IntFlag enumerations.
- #define **seSPIA_INTS**(a) ((seSPIA_Interrupts)((a)))
Combination of any of seSPIA_Interrupt enumerations.

Enumerations

- enum **seSPIA_DataTransferLength** {
 seDATA_TR_LENGTH_16BIT = 0xf,
 seDATA_TR_LENGTH_15BIT = 0xe,
 seDATA_TR_LENGTH_14BIT = 0xd,
 seDATA_TR_LENGTH_13BIT = 0xc,
 seDATA_TR_LENGTH_12BIT = 0xb,
 seDATA_TR_LENGTH_11BIT = 0xa,
 seDATA_TR_LENGTH_10BIT = 0x9,
 seDATA_TR_LENGTH_09BIT = 0x8,
 seDATA_TR_LENGTH_08BIT = 0x7,
 seDATA_TR_LENGTH_07BIT = 0x6,
 seDATA_TR_LENGTH_06BIT = 0x5,
 seDATA_TR_LENGTH_05BIT = 0x4,
 seDATA_TR_LENGTH_04BIT = 0x3,
 seDATA_TR_LENGTH_03BIT = 0x2,
 seDATA_TR_LENGTH_02BIT = 0x1 }
- enum **seSPIA_Format** {
 seSPIA_LSB_FST = 1,
 seSPIA_MSB_FST = 0 }
- enum **seSPIA_Polarity** {
 seSPIA_POL_LOW = 0,
 seSPIA_POL_HIGH = 1 }
- enum **seSPIA_Phase** {
 seSPIA_PH_RISE = 0,
 seSPIA_PH_FALL = 1 }
- enum **seSPIA_OperMode** {
 seSPIA_MODE_SLAVE = 0,
 seSPIA_MODE_MASTER = 1 }
- enum **seSPIA_IntFlag** {
 seSPIA_BSY = 0x0080U,
 seSPIA_OEIF = 0x0008U,
 seSPIA_TENDIF = 0x0004U,
 seSPIA_RBFIF = 0x0002U,
 seSPIA_TBEIF = 0x0001U,
 seSPIA_ALL_IF = seSPIA_OEIF | seSPIA_TENDIF | seSPIA_RBFIF | seSPIA_TBEIF }
- enum **seSPIA_Interrupt** {
 seSPIA_OEIE = 0x0008U,
 seSPIA_TENDIE = 0x0004U,


```

seSPIA_RBFIE = 0x0002U,
seSPIA_TBEIE = 0x0001U,
seSPIA_ALL_IE = seSPIA_OEIE | seSPIA_TENDIE | seSPIA_RBFIE | seSPIA_TBEIE }

```

2.53.1 Enumeration Type Documentation

2.53.1.1 seSPIA_Format

```
enum seSPIA_Format
```

Enumerator

seSPIA_LSB_FST	Specify the data format (input/output permutation) LSB first.
seSPIA_MSB_FST	Specify the data format (input/output permutation) MSB first.

2.53.1.2 seSPIA_Interrupt

```
enum seSPIA_Interrupt
```

Enumerator

seSPIA_OEIE	Overrun error interrupt.
seSPIA_TENDIE	End-of-transmission interrupt.
seSPIA_RBFIE	Receive buffer full interrupt.
seSPIA_TBEIE	Transmit buffer empty interrupt.

2.53.1.3 seSPIA_IntFlag

```
enum seSPIA_IntFlag
```

Enumerator

seSPIA_BSY	Transfer Busy/Slave Selected.
seSPIA_OEIF	Overrun error interrupt.
seSPIA_TENDIF	End-of-transmission interrupt.
seSPIA_RBFIF	Receive buffer full interrupt.
seSPIA_TBEIF	Transmit buffer empty interrupt.

2.53.1.4 seSPIA_OperMode

```
enum seSPIA_OperMode
```

Enumerator

seSPIA_MODE_SLAVE	Specifies the SPI mode slave.
-------------------	-------------------------------

Enumerator

seSPIA_MODE_MASTER	Specifies the SPI mode master.
--------------------	--------------------------------

2.53.1.5 seSPIA_Phase

enum `seSPIA_Phase`

Enumerator

seSPIA_PH_RISE	Triggers on positive edge of clock.
seSPIA_PH_FALL	Triggers on negative edge of clock.

2.53.1.6 seSPIA_Polarity

enum `seSPIA_Polarity`

Enumerator

seSPIA_POL_LOW	Output is low when clock off.
seSPIA_POL_HIGH	Output is high when clock off.

2.54 SPIA_Types

Data Structures

- struct [seSPIA_InitTypeDef](#)
SPIA Init structure definition.
- struct [seSPIA_ChannelDef](#)
SPIA Channel definition.

Variables

- [seSPIA_ChannelDef](#) **SPIA_CH0**
- [seSPIA_ChannelDef](#) **SPIA_CH1**

2.55 SPIA_Functions

Functions

- void **seSPIA_InitStructForMaster** (**seSPIA_InitTypeDef** *SPIA_InitStruct)
Initializes the SPIx peripheral as Master according to the specified parameters in the SPIA_InitStruct.
- void **seSPIA_InitStructForSlave** (**seSPIA_InitTypeDef** *SPIA_InitStruct)
Initializes the SPIx peripheral as Slave according to the specified parameters in the SPIA_InitStruct.
- **seStatus seSPIA_Init** (**seSPIA_ChannelDef** *SPICHx, **seSPIA_InitTypeDef** *SPIA_InitStruct)
Initializes the SPIx peripheral according to the specified parameters in the SPIA_InitStruct.
- **seStatus seSPIA_Start** (**seSPIA_ChannelDef** *SPICHx)
Starts or disables the specified SPI channel.
- void **seSPIA_Stop** (**seSPIA_ChannelDef** *SPICHx)
Stops or disables the specified SPI channel.
- **seStatus seSPIA_Reset** (**SPIA_0_Type** *SPIx)
Software reset of the specified SPI channel.
- **seStatus seSPIA_TxHWords** (**SPIA_0_Type** *SPIx, uint16_t data[], uint32_t size)
Sends a data through the SPIx peripheral.
- **seStatus seSPIA_DmaTxHWords** (**SPIA_0_Type** *SPIx, uint16_t data[], uint32_t size)
Sends a data through the SPIx peripheral.
- **seStatus seSPIA_TxBytes** (**SPIA_0_Type** *SPIx, uint8_t data[], uint32_t size)
Sends a data through the SPIx peripheral.
- **seStatus seSPIA_RxHWords** (**SPIA_0_Type** *SPIx, uint16_t data[], uint32_t size)
Returns the received data by the SPIx peripheral.
- **seStatus seSPIA_DmaRxHWords** (**SPIA_0_Type** *SPIx, uint16_t data[], uint32_t size)
Returns the received data by the SPIx peripheral.
- **seStatus seSPIA_RxBytes** (**SPIA_0_Type** *SPIx, uint8_t data[], uint32_t size)
Returns the received data by the SPIx peripheral.
- void **seSPIA_EnableInt** (**SPIA_0_Type** *SPIx, **seSPIA_Interrupt** irq)
Enables SPI channel interrupt.
- void **seSPIA_DisableInt** (**SPIA_0_Type** *SPIx, **seSPIA_Interrupt** irq)
Disables SPI channel interrupt.
- **seInterruptStatus seSPIA_GetIntFlag** (**SPIA_0_Type** *SPIx, **seSPIA_IntFlag** flag)
Gets SPI channel interrupt flag.
- void **seSPIA_ClearIntFlag** (**SPIA_0_Type** *SPIx, **seSPIA_IntFlag** flag)
Clears SPI channel interrupt flag.
- **seStatus seSPIA_SetBusSpeed** (**seSPIA_ChannelDef** *SPICHx, uint32_t speed)
Configures SPICLKn frequency [Hz] (= baud rate [bps])
- uint32_t **seSPIA_GetBusSpeed** (**seSPIA_ChannelDef** *SPICHx)
Discovers SPICLKn frequency [Hz] (= baud rate [bps])
- void **seSPIA_ENABLE_MST_CS0** (void)
Function allocates GPIO for Chip Slave Select 0 and enables it. This function should be defined by user.
- void **seSPIA_ENABLE_MST_CS1** (void)
Function allocates GPIO for Chip Slave Select 1 and enables it. This function should be defined by user.
- void **seSPIA_ASSERT_MST_CS0** (void)
Function asserts Chip Slave Select 0 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.

- void [seSPIA_ASSERT_MST_CS1](#) (void)
Function asserts Chip Slave Select 1 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.
- void [seSPIA_NEGATE_MST_CS0](#) (void)
Function negates Chip Slave Select 0 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.
- void [seSPIA_NEGATE_MST_CS1](#) (void)
Function negates Chip Slave Select 1 (Master mode only). This function should be defined by user. The function should be implemented by using GPIO.
- void [SPIA_0_IRQHandler](#) (void)
SPIA_CH0 Interrupt Service Routine (defined by user).
- [seStatus](#) [ConfigurePortsForSPI](#) ([seSPIA_ChannelDef](#) *SPICHx, [seSPIA_OperMode](#) OperMode)
Configures ports for this module. Override this function to configure specific ports.

2.55.1 Function Documentation

2.55.1.1 ConfigurePortsForSPI()

```
seStatus ConfigurePortsForSPI (
    seSPIA_ChannelDef * SPICHx,
    seSPIA_OperMode OperMode )
```

Return values

Status	can be a value of seStatus
--------	--

2.55.1.2 seSPIA_ASSERT_MST_CS0()

```
void seSPIA_ASSERT_MST_CS0 (
    void )
```

Return values

None	
------	--

2.55.1.3 seSPIA_ASSERT_MST_CS1()

```
void seSPIA_ASSERT_MST_CS1 (
    void )
```

Return values

None	
------	--

2.55.1.4 `seSPIA_ClearIntFlag()`

```
void seSPIA_ClearIntFlag (
    SPIA_0_Type * SPIx,
    seSPIA_IntFlag flag )
```

Parameters

<i>SPIx</i>	SPI peripheral
<i>flag</i>	interrupt to clear, see seSPIA_IntFlag .

Return values

<i>None</i>	
-------------	--

2.55.1.5 `seSPIA_DisableInt()`

```
void seSPIA_DisableInt (
    SPIA_0_Type * SPIx,
    seSPIA_Interrupt irq )
```

Parameters

<i>SPIx</i>	SPI peripheral
<i>irq</i>	interrupt to disable, see seSPIA_IntFlag .

Return values

<i>None</i>	
-------------	--

2.55.1.6 `seSPIA_DmaRxHWords()`

```
seStatus seSPIA_DmaRxHWords (
    SPIA_0_Type * SPIx,
    uint16_t data[],
    uint32_t size )
```

Parameters

<i>SPIx</i>	SPI peripheral
<i>data</i>	The received data.
<i>size</i>	Data size in number of words.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.7 seSPIA_DmaTxHWords()

```
seStatus seSPIA_DmaTxHWords (
    SPIA_0_Type * SPIx,
    uint16_t data[],
    uint32_t size )
```

Parameters

$S \leftarrow Plx$	SPI peripheral
<i>data</i>	Data to be transmitted.
<i>size</i>	Data size in number of words.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.8 seSPIA_ENABLE_MST_CS0()

```
void seSPIA_ENABLE_MST_CS0 (
    void )
```

Return values

<i>None</i>	
-------------	--

2.55.1.9 seSPIA_ENABLE_MST_CS1()

```
void seSPIA_ENABLE_MST_CS1 (
    void )
```

Return values

<i>None</i>	
-------------	--

2.55.1.10 seSPIA_EnableInt()

```
void seSPIA_EnableInt (
    SPIA_0_Type * SPIx,
    seSPIA_Interrupt irq )
```

Parameters

<i>S↔Plx</i>	SPI peripheral
<i>irq</i>	interrupt to enable, see seSPIA_IntFlag .

Return values

<i>None</i>	
-------------	--

2.55.1.11 seSPIA_GetBusSpeed()

```
uint32_t seSPIA_GetBusSpeed (
    seSPIA_ChannelDef * SPICHx )
```

Parameters

<i>SPICHx</i>	SPI channel definition of type seSPIA_ChannelDef
---------------	--

Return values

<i>SPICLK_n</i>	frequency [Hz]
---------------------------	----------------

2.55.1.12 seSPIA_GetIntFlag()

```
seInterruptStatus seSPIA_GetIntFlag (
    SPIA_0_Type * SPIx,
    seSPIA_IntFlag flag )
```

Parameters

<i>S↔Plx</i>	SPI peripheral
<i>flag</i>	interrupt to check, see seSPIA_IntFlag .

Return values

<i>seInterruptStatus</i>	see seInterruptStatus
--------------------------	---------------------------------------

2.55.1.13 seSPIA_Init()

```
seStatus seSPIA_Init (
    seSPIA_ChannelDef * SPICHx,
    seSPIA_InitTypeDef * SPIA_InitStruct )
```


Parameters

<i>SPICHx</i>	SPI channel definition of type seSPIA_ChannelDef
<i>SPIA_InitStruct</i>	pointer to a seSPIA_InitTypeDef structure that contains the configuration information for the specified SPI peripheral.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.14 seSPIA_InitStructForMaster()

```
void seSPIA_InitStructForMaster (
    seSPIA\_InitTypeDef * SPIA_InitStruct )
```

Parameters

<i>SPIx</i>	SPI peripheral
<i>SPIA_InitStruct</i>	pointer to a seSPIA_InitTypeDef structure that contains the configuration information for the specified SPI peripheral.

Return values

<i>None</i>	
-------------	--

2.55.1.15 seSPIA_InitStructForSlave()

```
void seSPIA_InitStructForSlave (
    seSPIA\_InitTypeDef * SPIA_InitStruct )
```

Parameters

<i>SPIx</i>	SPI peripheral
<i>SPIA_InitStruct</i>	pointer to a seSPIA_InitTypeDef structure that contains the configuration information for the specified SPI peripheral.

Return values

<i>None</i>	
-------------	--

2.55.1.16 seSPIA_NEGATE_MST_CS0()

```
void seSPIA_NEGATE_MST_CS0 (
    void )
```

Return values

<i>None</i>	
-------------	--

2.55.1.17 seSPIA_NEGATE_MST_CS1()

```
void seSPIA_NEGATE_MST_CS1 (
    void )
```

Return values

<i>None</i>	
-------------	--

2.55.1.18 seSPIA_Reset()

```
seStatus seSPIA_Reset (
    SPIA_0_Type * SPIx )
```

Parameters

$S \leftrightarrow$ <i>Plx</i>	SPI peripheral
-----------------------------------	----------------

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.19 seSPIA_RxBytes()

```
seStatus seSPIA_RxBytes (
    SPIA_0_Type * SPIx,
    uint8_t data[],
    uint32_t size )
```

Parameters

$S \leftrightarrow$ <i>Plx</i>	SPI peripheral
<i>data</i>	The received data.
<i>size</i>	Data size in number of bytes.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.20 seSPIA_RxHWords()

```
seStatus seSPIA_RxHWords (
    SPIA_0_Type * SPIx,
    uint16_t data[],
    uint32_t size )
```

Parameters

<i>SPIx</i>	SPI peripheral
<i>data</i>	The received data.
<i>size</i>	Data size in number of words.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.21 seSPIA_SetBusSpeed()

```
seStatus seSPIA_SetBusSpeed (
    seSPIA_ChannelDef * SPICHx,
    uint32_t speed )
```

Parameters

<i>SPICHx</i>	SPI channel definition of type seSPIA_ChannelDef
<i>speed</i>	SPICLKn frequency [Hz]

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.22 seSPIA_Start()

```
seStatus seSPIA_Start (
    seSPIA_ChannelDef * SPICHx )
```

Parameters

<i>SPICHx</i>	SPI channel definition of type seSPIA_ChannelDef
---------------	--

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.23 seSPIA_Stop()

```
void seSPIA_Stop (
    seSPIA_ChannelDef * SPICHx )
```

Parameters

<i>SPICHx</i>	SPI channel definition of type seSPIA_ChannelDef
---------------	--

Return values

<i>None</i>	
-------------	--

2.55.1.24 seSPIA_TxBytes()

```
seStatus seSPIA_TxBytes (
    SPIA_0_Type * SPIx,
    uint8_t data[],
    uint32_t size )
```

Parameters

<i>S_↔ Plx</i>	SPI peripheral
<i>data</i>	Data to be transmitted.
<i>size</i>	Data size in number of bytes.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.25 seSPIA_TxHWords()

```
seStatus seSPIA_TxHWords (
    SPIA_0_Type * SPIx,
    uint16_t data[],
    uint32_t size )
```

Parameters

<i>S_↔ Plx</i>	SPI peripheral
<i>data</i>	Data to be transmitted.
<i>size</i>	Data size in number of words.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.55.1.26 SPIA_0_IRQHandler()

```
void SPIA_0_IRQHandler (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.56 SVD2

SVD2 is a supply voltage detector to monitor the power supply voltage on the VDD pin or the voltage applied to an external pin.

Modules

- [SVD2_Constants](#)
- [SVD_Types](#)
- [SVD_Functions](#)

2.57 SVD2_Constants

Data Structures

- union [seSVD2_ClkDiv](#)

Typedefs

- typedef [seCLG_ClkSrc](#) **seSVD2_ClkSrc**

Enumerations

- enum [seSVD2_IOSC_ClkDiv](#) {
[seSVD2_IOSC_CLKDIV_16](#) = 0,
[seSVD2_IOSC_CLKDIV_32](#) = 1,
[seSVD2_IOSC_CLKDIV_64](#) = 2,
[seSVD2_IOSC_CLKDIV_128](#) = 3,
[seSVD2_IOSC_CLKDIV_256](#) = 4,
[seSVD2_IOSC_CLKDIV_512](#) = 5 }
- enum [seSVD2_OSC1_ClkDiv](#) { [seSVD2_OSC1_CLKDIV_1](#) = 0 }
- enum [seSVD2_OSC3_ClkDiv](#) {
[seSVD2_OSC3_CLKDIV_16](#) = 0,
[seSVD2_OSC3_CLKDIV_32](#) = 1,
[seSVD2_OSC3_CLKDIV_64](#) = 2,
[seSVD2_OSC3_CLKDIV_128](#) = 3,
[seSVD2_OSC3_CLKDIV_256](#) = 4,
[seSVD2_OSC3_CLKDIV_512](#) = 5 }
- enum [seSVD2_EXOSC_ClkDiv](#) { [seSVD2_EXOSC_CLKDIV_1](#) = 0 }
- enum [seSVD2_VoltageSource](#) {
[seSVD2_EXSVD](#) = 0x0001,
[seSVD2_VDD](#) = 0x0000 }
- enum [seSVD2_IntermittentMode](#) {
[seSVD2_SVDCLK_512](#) = 3,
[seSVD2_SVDCLK_256](#) = 2,
[seSVD2_SVDCLK_128](#) = 1,
[seSVD2_Continuous](#) = 0 }
- enum [seSVD2_DetectMode](#) {
[seSVD2_CTL_UPPER_DETECT](#) = 1,
[seSVD2_CTL_LOWER_DETECT](#) = 0 }
- enum [seSVD2_SamplingResCnt](#) {
[seSVD2_CTL_8_TIMES_ROW](#) = 3,
[seSVD2_CTL_4_TIMES_ROW](#) = 2,
[seSVD2_CTL_2_TIMES_ROW](#) = 1,
[seSVD2_CTL_1_TIMES_ROW](#) = 0 }
- enum [seSVD2_PowerSupply](#) {
[seSVD2_POWER_HIGH](#) = 0,
[seSVD2_POWER_LOW](#) = 1 }
- enum [seSVD2_IntFlag](#) {
[seSVD2_SVDDT](#) = 0x0100U,
[seSVD2_SVDIF](#) = 0x0001U }
- enum [seSVD2_Interrupt](#) { [seSVD2_SVDIE](#) = 0x0001U }

2.57.1 Enumeration Type Documentation

2.57.1.1 seSVD2_DetectMode

enum `seSVD2_DetectMode`

Enumerator

seSVD2_CTL_UPPER_DETECT	Upper detect.
seSVD2_CTL_LOWER_DETECT	Lower detect.

2.57.1.2 seSVD2_EXOSC_ClkDiv

enum `seSVD2_EXOSC_ClkDiv`

Enumerator

seSVD2_EXOSC_CLKDIV↔ _1	EXOSC division ratio is 1/1.
----------------------------	------------------------------

2.57.1.3 seSVD2_IntermittentMode

enum `seSVD2_IntermittentMode`

Enumerator

seSVD2_SVDCLK_512	SVDCLK/512(about 16msec)
seSVD2_SVDCLK_256	SVDCLK/256(about 8msec)
seSVD2_SVDCLK_128	SVDCLK/128(about 4msec)
seSVD2_Continuous	continuous

2.57.1.4 seSVD2_Interrupt

enum `seSVD2_Interrupt`

Enumerator

seSVD2_SVDIE	SVD interrupt enable interrupt.
--------------	---------------------------------

2.57.1.5 seSVD2_IntFlag

enum `seSVD2_IntFlag`

Enumerator

seSVD2_SVDDT	SVD detection monitor.
--------------	------------------------

Enumerator

seSVD2_SVDIF	SVD interrupt factor flag.
--------------	----------------------------

2.57.1.6 seSVD2_IOSC_ClkDiv

```
enum seSVD2_IOSC_ClkDiv
```

Enumerator

seSVD2_IOSC_CLKDIV_16	IOSC division ratio is 1/16.
seSVD2_IOSC_CLKDIV_32	IOSC division ratio is 1/32.
seSVD2_IOSC_CLKDIV_64	IOSC division ratio is 1/64.
seSVD2_IOSC_CLKDIV_128	IOSC division ratio is 1/128.
seSVD2_IOSC_CLKDIV_256	IOSC division ratio is 1/256.
seSVD2_IOSC_CLKDIV_512	IOSC division ratio is 1/512.

2.57.1.7 seSVD2_OSC1_ClkDiv

```
enum seSVD2_OSC1_ClkDiv
```

Enumerator

seSVD2_OSC1_CLKDIV↔ _1	OSC1 division ratio is 1/1.
---------------------------	-----------------------------

2.57.1.8 seSVD2_OSC3_ClkDiv

```
enum seSVD2_OSC3_ClkDiv
```

Enumerator

seSVD2_OSC3_CLKDIV_16	OSC3 division ratio is 1/16.
seSVD2_OSC3_CLKDIV_32	OSC3 division ratio is 1/32.
seSVD2_OSC3_CLKDIV_64	OSC3 division ratio is 1/64.
seSVD2_OSC3_CLKDIV_128	OSC3 division ratio is 1/128.
seSVD2_OSC3_CLKDIV_256	OSC3 division ratio is 1/256.
seSVD2_OSC3_CLKDIV_512	OSC3 division ratio is 1/512.

2.57.1.9 seSVD2_PowerSupply

```
enum seSVD2_PowerSupply
```

Enumerator

seSVD2_POWER_HIGH	Power supply voltage(VDD or EXSVD) \geq comparison voltage.
seSVD2_POWER_LOW	Power supply voltage(VDD or EXSVD) $<$ comparison voltage.

2.57.1.10 seSVD2_SamplingResCnt

enum `seSVD2_SamplingResCnt`

Enumerator

seSVD2_CTL_8_TIMES_ROW	8 times in a row
seSVD2_CTL_4_TIMES_ROW	4 times in a row
seSVD2_CTL_2_TIMES_ROW	2 times in a row
seSVD2_CTL_1_TIMES_ROW	1 times in a row

2.57.1.11 seSVD2_VoltageSource

enum `seSVD2_VoltageSource`

Enumerator

seSVD2_EXSVD	External SVD.
seSVD2_VDD	VDD.

2.58 SVD_Types

Data Structures

- struct `seSVD2_InitTypeDef`
SVD Init structure definition.
- struct `seSVD2_ChannelDef`
SVD Channel definition.

Variables

- `seSVD2_ChannelDef` **SVD2_CH0**
- `seSVD2_ChannelDef` **SVD2_CH1**

2.59 SVD_Functions

Functions

- void [seSVD2_InitStruct](#) ([seSVD2_InitTypeDef](#) *SVD_InitStruct)
Fills each seSVD_InitStruct member with its default value.
- [seStatus](#) [seSVD2_Init](#) ([seSVD2_ChannelDef](#) *SVDCHx, [seSVD2_InitTypeDef](#) *SVD_InitStruct)
Initializes the SVDx peripheral according to the specified parameters in the SVD_InitStruct.
- void [seSVD2_Start](#) (SVD2_0_Type *SVDx)
Starts the specified SVD peripheral.
- void [seSVD2_Stop](#) (SVD2_0_Type *SVDx)
Stops the SVDx specified peripheral.
- [seStatus](#) [seSVD2_SetComparisonVoltage](#) (SVD2_0_Type *SVDx, uint8_t volt)
Set Comparison voltage for detecting low voltage.
- [seStatus](#) [seSVD2_SetVoltageSource](#) (SVD2_0_Type *SVDx, [seSVD2_VoltageSource](#) voltagesource)
Set voltage source for detecting low voltage.
- [seSVD2_PowerSupply](#) [seSVD2_GetVoltageDetection](#) (SVD2_0_Type *SVDx)
Get power supply voltage detection results.
- [seInterruptStatus](#) [seSVD2_IsIntLowVoltage](#) (SVD2_0_Type *SVDx)
Check SVD low voltage interrupt.
- void [seSVD2_ClearIntLowVoltage](#) (SVD2_0_Type *SVDx)
Clear SVD low voltage interrupt.
- void [SVD2_0_IRQHandler](#) (void)
SVD Interrupt Service Routine.
- void [SVD2_1_IRQHandler](#) (void)
SVD_USB Interrupt Service Routine.
- [seStatus](#) [ConfigurePortsForSVD2](#) ([seSVD2_ChannelDef](#) *SVDCHx)
Configures ports for this module. Override this function to configure specific ports.

2.59.1 Function Documentation

2.59.1.1 ConfigurePortsForSVD2()

```
seStatus ConfigurePortsForSVD2 (
    seSVD2\_ChannelDef * SVDCHx )
```

Parameters

SVDCHx	SVD channel definition of type seSVD2_ChannelDef
------------------------	--

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.59.1.2 seSVD2_ClearIntLowVoltage()

```
void seSVD2_ClearIntLowVoltage (
    SVD2_0_Type * SVDx )
```

Parameters

<i>SVDx</i>	SVD peripherhal.
-------------	------------------

Return values

<i>None</i>	
-------------	--

2.59.1.3 seSVD2_GetVoltageDetection()

```
seSVD2_PowerSupply seSVD2_GetVoltageDetection (
    SVD2_0_Type * SVDx )
```

Parameters

<i>SVDx</i>	SVD peripherhal.
-------------	------------------

Return values

<i>powerstatus</i>	seSVD2_PowerSupply
--------------------	------------------------------------

Note

powerstatus of seSVD2_POWER_OK Power supply voltage(VDD or EXSVD) >= comparison voltage. or seSVD2_POWER_LOW Power supply voltage(VDD or EXSVD) < comparison voltage.

2.59.1.4 seSVD2_Init()

```
seStatus seSVD2_Init (
    seSVD2_ChannelDef * SVDCHx,
    seSVD2_InitTypeDef * SVD_InitStruct )
```

Parameters

<i>SVDCHx</i>	SVD channel definition of type seSVD2_ChannelDef
<i>SVD_InitStruct</i>	pointer to a seSVD2_InitTypeDef structure that contains the configuration information for the specified SVD peripheral.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.59.1.5 `seSVD2_InitStruct()`

```
void seSVD2_InitStruct (
    seSVD2_InitTypeDef * SVD_InitStruct )
```

Parameters

<i>SVD_InitStruct</i>	pointer to an seSVD2_InitTypeDef structure which will be initialized.
-----------------------	---

Return values

<i>None</i>	
-------------	--

2.59.1.6 `seSVD2_IsIntLowVoltage()`

```
seInterruptStatus seSVD2_IsIntLowVoltage (
    SVD2_0_Type * SVDx )
```

Parameters

<i>SVDx</i>	SVD peripherhal.
-------------	------------------

Return values

<i>InterruptStatus</i>	see seInterruptStatus
------------------------	---------------------------------------

2.59.1.7 `seSVD2_SetComparisonVoltage()`

```
seStatus seSVD2_SetComparisonVoltage (
    SVD2_0_Type * SVDx,
    uint8_t volt )
```

Parameters

<i>SVDx</i>	SVD peripherhal.
<i>volt</i>	Comparison voltage for detecting low voltage.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.59.1.8 `seSVD2_SetVoltageSource()`

```
seStatus seSVD2_SetVoltageSource (
```

```
SVD2_0_Type * SVDx,
seSVD2_VoltageSource voltagesource )
```

Parameters

<i>SVDx</i>	SVD periperal.
<i>voltagesource</i>	Voltage source of type seSVD2_VoltageSource

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.59.1.9 seSVD2_Start()

```
void seSVD2_Start (
    SVD2_0_Type * SVDx )
```

Parameters

<i>SVDx</i>	SVD periperal.
-------------	----------------

Return values

<i>None</i>	
-------------	--

2.59.1.10 seSVD2_Stop()

```
void seSVD2_Stop (
    SVD2_0_Type * SVDx )
```

Parameters

<i>SVDx</i>	SVD periperal.
-------------	----------------

Return values

<i>None</i>	
-------------	--

2.59.1.11 SVD2_0_IRQHandler()

```
void SVD2_0_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.59.1.12 SVD2_1_IRQHandler()

```
void SVD2_1_IRQHandler (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.60 T16

T16 is a 16 bit timer.

Modules

- [T16_Constants](#)
- [T16_Types](#)
- [T16_Functions](#)

2.61 T16_Constants

Typedefs

- typedef [seCLG_ClkSrc](#) **seT16_ClkSrc**

Enumerations

- enum [seT16_IOSC_ClkDiv](#) {
[seT16_IOSC_CLKDIV_1](#) = 0,
[seT16_IOSC_CLKDIV_2](#) = 1,
[seT16_IOSC_CLKDIV_4](#) = 2,
[seT16_IOSC_CLKDIV_8](#) = 3,
[seT16_IOSC_CLKDIV_16](#) = 4,
[seT16_IOSC_CLKDIV_32](#) = 5,
[seT16_IOSC_CLKDIV_64](#) = 6,
[seT16_IOSC_CLKDIV_128](#) = 7,
[seT16_IOSC_CLKDIV_256](#) = 8,
[seT16_IOSC_CLKDIV_512](#) = 9,
[seT16_IOSC_CLKDIV_1024](#) = 10,
[seT16_IOSC_CLKDIV_2048](#) = 11,
[seT16_IOSC_CLKDIV_4096](#) = 12,
[seT16_IOSC_CLKDIV_8192](#) = 13,
[seT16_IOSC_CLKDIV_16384](#) = 14,
[seT16_IOSC_CLKDIV_32768](#) = 15 }
- enum [seT16_OSC1_ClkDiv](#) {
[seT16_OSC1_CLKDIV_1](#) = 0,
[seT16_OSC1_CLKDIV_2](#) = 1,
[seT16_OSC1_CLKDIV_4](#) = 2,
[seT16_OSC1_CLKDIV_8](#) = 3,
[seT16_OSC1_CLKDIV_16](#) = 4,
[seT16_OSC1_CLKDIV_32](#) = 5,
[seT16_OSC1_CLKDIV_64](#) = 6,
[seT16_OSC1_CLKDIV_128](#) = 7,
[seT16_OSC1_CLKDIV_256](#) = 8 }
- enum [seT16_OSC3_ClkDiv](#) {
[seT16_OSC3_CLKDIV_1](#) = 0,
[seT16_OSC3_CLKDIV_2](#) = 1,
[seT16_OSC3_CLKDIV_4](#) = 2,
[seT16_OSC3_CLKDIV_8](#) = 3,
[seT16_OSC3_CLKDIV_16](#) = 4,
[seT16_OSC3_CLKDIV_32](#) = 5,
[seT16_OSC3_CLKDIV_64](#) = 6,
[seT16_OSC3_CLKDIV_128](#) = 7,
[seT16_OSC3_CLKDIV_256](#) = 8,
[seT16_OSC3_CLKDIV_512](#) = 9,
[seT16_OSC3_CLKDIV_1024](#) = 10,
[seT16_OSC3_CLKDIV_2048](#) = 11,
[seT16_OSC3_CLKDIV_4096](#) = 12,
[seT16_OSC3_CLKDIV_8192](#) = 13,
[seT16_OSC3_CLKDIV_16384](#) = 14,
[seT16_OSC3_CLKDIV_32768](#) = 15 }
- enum [seT16_EXOSC_ClkDiv](#) { [seT16_EXOSC_CLKDIV_1](#) = 0 }

```
enum seT16_CounterMode {
    seT16_RepeatMode,
    seT16_OneShotMode }

```

2.61.1 Enumeration Type Documentation

2.61.1.1 seT16_CounterMode

```
enum seT16_CounterMode

```

Enumerator

seT16_RepeatMode	Timer operates in Repeat Mode. Select this mode to generate periodic underflow interrupts or when using the timer to output a clock to a peripheral circuit.
seT16_OneShotMode	Timer operates in One-shot Mode. Select this mode to stop the counter after an interrupt has occurred once.

2.61.1.2 seT16_EXOSC_ClkDiv

```
enum seT16_EXOSC_ClkDiv

```

Enumerator

seT16_EXOSC_CLKDIV↔ _1	EXOSC division ratio is 1/1.
---------------------------	------------------------------

2.61.1.3 seT16_IOSC_ClkDiv

```
enum seT16_IOSC_ClkDiv

```

Enumerator

seT16_IOSC_CLKDIV_1	IOSC division ratio is 1/1.
seT16_IOSC_CLKDIV_2	IOSC division ratio is 1/2.
seT16_IOSC_CLKDIV_4	IOSC division ratio is 1/4.
seT16_IOSC_CLKDIV_8	IOSC division ratio is 1/8.
seT16_IOSC_CLKDIV_16	IOSC division ratio is 1/16.
seT16_IOSC_CLKDIV_32	IOSC division ratio is 1/32.
seT16_IOSC_CLKDIV_64	IOSC division ratio is 1/64.
seT16_IOSC_CLKDIV_128	IOSC division ratio is 1/128.
seT16_IOSC_CLKDIV_256	IOSC division ratio is 1/256.
seT16_IOSC_CLKDIV_512	IOSC division ratio is 1/512.
seT16_IOSC_CLKDIV_1024	IOSC division ratio is 1/1024.
seT16_IOSC_CLKDIV_2048	IOSC division ratio is 1/2048.
seT16_IOSC_CLKDIV_4096	IOSC division ratio is 1/4096.
seT16_IOSC_CLKDIV_8192	IOSC division ratio is 1/8192.
seT16_IOSC_CLKDIV_16384	IOSC division ratio is 1/16384.
seT16_IOSC_CLKDIV_32768	IOSC division ratio is 1/32768.

2.61.1.4 seT16_OSC1_ClkDiv

```
enum seT16_OSC1_ClkDiv
```

Enumerator

seT16_OSC1_CLKDIV_1	OSC1 division ratio is 1/1.
seT16_OSC1_CLKDIV_2	OSC1 division ratio is 1/2.
seT16_OSC1_CLKDIV_4	OSC1 division ratio is 1/4.
seT16_OSC1_CLKDIV_8	OSC1 division ratio is 1/8.
seT16_OSC1_CLKDIV_16	OSC1 division ratio is 1/16.
seT16_OSC1_CLKDIV_32	OSC1 division ratio is 1/32.
seT16_OSC1_CLKDIV_64	OSC1 division ratio is 1/64.
seT16_OSC1_CLKDIV_128	OSC1 division ratio is 1/128.
seT16_OSC1_CLKDIV_256	OSC1 division ratio is 1/256.

2.61.1.5 seT16_OSC3_ClkDiv

```
enum seT16_OSC3_ClkDiv
```

Enumerator

seT16_OSC3_CLKDIV_1	OSC3 division ratio is 1/1.
seT16_OSC3_CLKDIV_2	OSC3 division ratio is 1/2.
seT16_OSC3_CLKDIV_4	OSC3 division ratio is 1/4.
seT16_OSC3_CLKDIV_8	OSC3 division ratio is 1/8.
seT16_OSC3_CLKDIV_16	OSC3 division ratio is 1/16.
seT16_OSC3_CLKDIV_32	OSC3 division ratio is 1/32.
seT16_OSC3_CLKDIV_64	OSC3 division ratio is 1/64.
seT16_OSC3_CLKDIV_128	OSC3 division ratio is 1/128.
seT16_OSC3_CLKDIV_256	OSC3 division ratio is 1/256.
seT16_OSC3_CLKDIV_512	OSC3 division ratio is 1/512.
seT16_OSC3_CLKDIV_1024	OSC3 division ratio is 1/1024.
seT16_OSC3_CLKDIV_2048	OSC3 division ratio is 1/2048.
seT16_OSC3_CLKDIV_4096	OSC3 division ratio is 1/4096.
seT16_OSC3_CLKDIV_8192	OSC3 division ratio is 1/8192.
seT16_OSC3_CLKDIV_16384	OSC3 division ratio is 1/16384.
seT16_OSC3_CLKDIV_32768	OSC3 division ratio is 1/32768.

2.62 T16_Types

Data Structures

- struct [seT16_InitTypeDef](#)
T16 Init structure definition.

2.63 T16_Functions

Functions

- void [seT16_InitStruct](#) ([seT16_InitTypeDef](#) *T16_InitStruct)
Fills each [seT16_InitTypeDef](#) member with its default value.
- [seStatus seT16_Init](#) (T16_0_Type *T16x, [seT16_InitTypeDef](#) *T16_InitStruct)
Initializes the T16 peripheral according to the specified parameters in the [seT16_InitStruct](#).
- void [seT16_Start](#) (T16_0_Type *T16x)
Starts Timer channel.
- void [seT16_Stop](#) (T16_0_Type *T16x)
Stops Timer channel.
- void [seT16_Enable](#) (T16_0_Type *T16x)
Enables Timer channel by start supplying operating clock.
- void [seT16_Disable](#) (T16_0_Type *T16x)
Disables Timer channel by stop supplying operating clock.
- void [seT16_ConfigureClock](#) (T16_0_Type *T16x, [seT16_ClkSrc](#) clock, uint16_t divider)
Configures timer clock source and clock divider.
- uint32_t [seT16_GetClk](#) (T16_0_Type *T16x)
Discovers T16 clock from registers.
- void [seT16_ConfigureCounterMode](#) (T16_0_Type *T16x, [seT16_CounterMode](#) mode)
Configures timer counter mode.
- [seStatus seT16_SetCounter](#) (T16_0_Type *T16x, uint16_t counter)
Sets Timer counter.
- uint16_t [seT16_GetCounter](#) (T16_0_Type *T16x)
Gets Timer counter value.
- void [seT16_EnableInt](#) (T16_0_Type *T16x)
Enables Timer channel interrupt.
- void [seT16_DisableInt](#) (T16_0_Type *T16x)
Disables Timer channel interrupt.
- [seInterruptStatus seT16_GetIntFlag](#) (T16_0_Type *T16x)
Returns Timer interrupt flag.
- void [seT16_ClearIntFlag](#) (T16_0_Type *T16x)
Clears Timer channel interrupt.
- void [T16_0_IRQHandler](#) (void)
Timer0 Interrupt Service Routine.
- void [T16_1_IRQHandler](#) (void)
Timer1 Interrupt Service Routine.
- void [T16_2_IRQHandler](#) (void)
Timer2 Interrupt Service Routine.
- void [T16_3_IRQHandler](#) (void)
Timer3 Interrupt Service Routine.

2.63.1 Function Documentation

2.63.1.1 [seT16_ClearIntFlag\(\)](#)

```
void seT16_ClearIntFlag (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	pointer to T16 peripheral.
-------------	----------------------------

Return values

<i>None</i>	
-------------	--

2.63.1.2 seT16_ConfigureClock()

```
void seT16_ConfigureClock (
    T16_0_Type * T16x,
    seT16_ClkSrc clock,
    uint16_t divider )
```

Parameters

<i>T16x</i>	This parameter defines a timer channel and can be a value of T16_0_Type.
<i>clock</i>	This parameter can be a value of seT16_ClkSrc.
<i>divider</i>	This parameter can be a value of seT16_ClkDiv.

Return values

<i>None</i>	
-------------	--

2.63.1.3 seT16_ConfigureCounterMode()

```
void seT16_ConfigureCounterMode (
    T16_0_Type * T16x,
    seT16_CounterMode mode )
```

Parameters

<i>T16x</i>	This parameter defines a timer channel and can be a value of T16_0_Type.
<i>mode</i>	This parameter can be a value of seT16_CounterMode.

Return values

<i>None</i>	
-------------	--

2.63.1.4 seT16_Disable()

```
void seT16_Disable (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	pointer to T16 peripheral.
-------------	----------------------------

Return values

<i>None</i>	
-------------	--

2.63.1.5 seT16_DisableInt()

```
void seT16_DisableInt (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	pointer to T16 peripheral.
-------------	----------------------------

Return values

<i>None</i>	
-------------	--

2.63.1.6 seT16_Enable()

```
void seT16_Enable (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	pointer to T16 peripheral.
-------------	----------------------------

Return values

<i>None</i>	
-------------	--

2.63.1.7 seT16_EnableInt()

```
void seT16_EnableInt (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	pointer to T16 peripheral.
-------------	----------------------------

Return values

<i>None</i>	
-------------	--

2.63.1.8 seT16_GetClk()

```
uint32_t seT16_GetClk (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	pointer to T16 peripheral.
-------------	----------------------------

Return values

<i>T16</i>	clock.
------------	--------

2.63.1.9 seT16_GetCounter()

```
uint16_t seT16_GetCounter (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	This parameter defines a timer channel and can be a value of T16_0_Type.
-------------	--

Return values

<i>16-bit</i>	counter value.
---------------	----------------

2.63.1.10 seT16_GetIntFlag()

```
seInterruptStatus seT16_GetIntFlag (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	pointer to T16 peripheral.
-------------	----------------------------

Return values

<i>InterruptStatus</i>	seInterruptStatus
------------------------	-----------------------------------

2.63.1.11 seT16_Init()

```
seStatus seT16_Init (
    T16_0_Type * T16x,
    seT16_InitTypeDef * T16_InitStruct )
```

Note

This function configures the module, and module's interrupts. It clears module's interrupts but does not enable interrupt from the module to CPU. This function enables module by start supplying operating clock.

Parameters

<i>T16x</i>	pointer to T16 peripheral.
<i>InitStruct</i>	pointer to a seT16_InitTypeDef structure that contains the configuration information for the specified T16 peripheral.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.63.1.12 seT16_InitStruct()

```
void seT16_InitStruct (
    seT16_InitTypeDef * T16_InitStruct )
```

Parameters

<i>T16_InitStruct</i>	pointer to an seT16_InitTypeDef structure which will be initialized.
-----------------------	--

Return values

<i>None</i>	
-------------	--

2.63.1.13 seT16_SetCounter()

```
seStatus seT16_SetCounter (
    T16_0_Type * T16x,
    uint16_t counter )
```

Parameters

<i>T16x</i>	This parameter defines a timer channel and can be a value of T16_0_Type.
<i>counter</i>	This parameter can be a 16-bit value.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.63.1.14 seT16_Start()

```
void seT16_Start (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	pointer to T16 peripheral.
-------------	----------------------------

Return values

<i>None</i>	
-------------	--

2.63.1.15 seT16_Stop()

```
void seT16_Stop (
    T16_0_Type * T16x )
```

Parameters

<i>T16x</i>	pointer to T16 peripheral.
-------------	----------------------------

Return values

<i>None</i>	
-------------	--

2.63.1.16 T16_0_IRQHandler()

```
void T16_0_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.63.1.17 T16_1_IRQHandler()

```
void T16_1_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.63.1.18 T16_2_IRQHandler()

```
void T16_2_IRQHandler (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.63.1.19 T16_3_IRQHandler()

```
void T16_3_IRQHandler (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.64 T16B

T16B is a 16-bit PWM timer with comparator or capture functions.

Modules

- [T16B_Constants](#)
- [T16B_Types](#)
- [T16B_Functions](#)

2.65 T16B_Constants

Macros

- #define `seT16B_FLGS(a)` ((seT16B_IntFlag)((a)))
Combination of any of seT16B_IntFlag enumerations.
- #define `seT16B_INTS(a)` ((seT16B_Interrupt)((a)))
Combination of any of seT16B_Interrupt enumerations.

Enumerations

- enum `seT16B_ClkSrc` {
`seT16B_IOSC = 0,`
`seT16B_OSC1 = 1,`
`seT16B_OSC3 = 2,`
`seT16B_EXOSC = 3,`
`seT16B_EXCLN0 = 4,`
`seT16B_EXCLN1 = 5,`
`seT16B_EXCLN0_INV = 6,`
`seT16B_EXCLN1_INV = 7` }
Brief description.
- enum `seT16B_IOSC_ClkDiv` {
`seT16B_IOSC_CLKDIV_1 = 0,`
`seT16B_IOSC_CLKDIV_2 = 1,`
`seT16B_IOSC_CLKDIV_4 = 2,`
`seT16B_IOSC_CLKDIV_8 = 3,`
`seT16B_IOSC_CLKDIV_16 = 4,`
`seT16B_IOSC_CLKDIV_32 = 5,`
`seT16B_IOSC_CLKDIV_64 = 6,`
`seT16B_IOSC_CLKDIV_128 = 7,`
`seT16B_IOSC_CLKDIV_256 = 8,`
`seT16B_IOSC_CLKDIV_512 = 9,`
`seT16B_IOSC_CLKDIV_1024 = 10,`
`seT16B_IOSC_CLKDIV_2048 = 11,`
`seT16B_IOSC_CLKDIV_4096 = 12,`
`seT16B_IOSC_CLKDIV_8192 = 13,`
`seT16B_IOSC_CLKDIV_16384 = 14,`
`seT16B_IOSC_CLKDIV_32768 = 15` }
- enum `seT16B_OSC1_ClkDiv` {
`seT16B_OSC1_CLKDIV_1 = 0,`
`seT16B_OSC1_CLKDIV_2 = 1,`
`seT16B_OSC1_CLKDIV_4 = 2,`
`seT16B_OSC1_CLKDIV_8 = 3,`
`seT16B_OSC1_CLKDIV_16 = 4,`
`seT16B_OSC1_CLKDIV_32 = 5,`
`seT16B_OSC1_CLKDIV_64 = 6,`
`seT16B_OSC1_CLKDIV_128 = 7,`
`seT16B_OSC1_CLKDIV_256 = 8` }
- enum `seT16B_OSC3_ClkDiv` {
`seT16B_OSC3_CLKDIV_1 = 0,`
`seT16B_OSC3_CLKDIV_2 = 1,`
`seT16B_OSC3_CLKDIV_4 = 2,`

```

seT16B_OSC3_CLKDIV_8 = 3,
seT16B_OSC3_CLKDIV_16 = 4,
seT16B_OSC3_CLKDIV_32 = 5,
seT16B_OSC3_CLKDIV_64 = 6,
seT16B_OSC3_CLKDIV_128 = 7,
seT16B_OSC3_CLKDIV_256 = 8,
seT16B_OSC3_CLKDIV_512 = 9,
seT16B_OSC3_CLKDIV_1024 = 10,
seT16B_OSC3_CLKDIV_2048 = 11,
seT16B_OSC3_CLKDIV_4096 = 12,
seT16B_OSC3_CLKDIV_8192 = 13,
seT16B_OSC3_CLKDIV_16384 = 14,
seT16B_OSC3_CLKDIV_32768 = 15 }
• enum seT16B_EX_ClkDiv { seT16B_EX_CLKDIV_1 = 0 }
• enum seT16B_ONEST {
    seT16B_RepeatMode,
    seT16B_OneShotMode }
• enum seT16B_CNTMD {
    seT16B_CountUp,
    seT16B_CountDown,
    seT16B_CountUpDown }
• enum seT16B_SCS {
    seT16B_SyncCapture = 0x1,
    seT16B_AsyncCapture = 0x0 }
• enum seT16B_CBUFMD {
    seT16B_Clear = 0x4,
    seT16B_Excep = 0x3,
    seT16B_Compare = 0x2,
    seT16B_Period = 0x1,
    seT16B_None = 0x0 }
• enum seT16B_CAPIS {
    seT16B_Input_0 = 0x3,
    seT16B_Input_1 = 0x2,
    seT16B_CCI = 0x0 }
• enum seT16B_CAPTRG {
    seT16B_UpDown = 0x3,
    seT16B_Down = 0x2,
    seT16B_Up = 0x1,
    seT16B_DisCapt = 0x0 }
• enum seT16B_TOUTMT {
    seT16B_UseBothComparator = 0x1,
    seT16B_UseOneComparator = 0x0 }
• enum seT16B_TOUTO {
    seT16B_HighLevelOutput = 0x1,
    seT16B_LowLevelOutput = 0x0 }
• enum seT16B_TOUTMD {
    seT16B_ResetSet = 0x7,
    seT16B_ToggleSet = 0x6,
    seT16B_Reset = 0x5,
    seT16B_Toggle = 0x4,
    seT16B_SetReset = 0x3,
    seT16B_ToggleReset = 0x2,
    seT16B_Set = 0x1,
    seT16B_SoftwareControl = 0x0 }

```

- enum `seT16B_TOUTINV` {
`seT16B_Inverted` = 0x1,
`seT16B_Normal` = 0x0 }
- enum `seT16B_CCMD` {
`seT16B_ComparatorMode` = 0,
`seT16B_CaptureMode` = 1 }
- enum `seT16B_IntFlag` {
`seT16B_CAPOW5IF` = 0x2000U,
`seT16B_CMPCAP5IF` = 0x1000U,
`seT16B_CAPOW4IF` = 0x0800U,
`seT16B_CMPCAP4IF` = 0x0400U,
`seT16B_CAPOW3IF` = 0x0200U,
`seT16B_CMPCAP3IF` = 0x0100U,
`seT16B_CAPOW2IF` = 0x0080U,
`seT16B_CMPCAP2IF` = 0x0040U,
`seT16B_CAPOW1IF` = 0x0020U,
`seT16B_CMPCAP1IF` = 0x0010U,
`seT16B_CAPOW0IF` = 0x0008U,
`seT16B_CMPCAP0IF` = 0x0004U,
`seT16B_CNTMAXIF` = 0x0002U,
`seT16B_CNTZEROIF` = 0x0001U,
`seT16B_ALLIF` }
- enum `seT16B_Interrupt` {
`seT16B_CAPOW5IE` = 0x2000U,
`seT16B_CMPCAP5IE` = 0x1000U,
`seT16B_CAPOW4IE` = 0x0800U,
`seT16B_CMPCAP4IE` = 0x0400U,
`seT16B_CAPOW3IE` = 0x0200U,
`seT16B_CMPCAP3IE` = 0x0100U,
`seT16B_CAPOW2IE` = 0x0080U,
`seT16B_CMPCAP2IE` = 0x0040U,
`seT16B_CAPOW1IE` = 0x0020U,
`seT16B_CMPCAP1IE` = 0x0010U,
`seT16B_CAPOW0IE` = 0x0008U,
`seT16B_CMPCAP0IE` = 0x0004U,
`seT16B_CNTMAXIE` = 0x0002U,
`seT16B_CNTZEROIE` = 0x0001U,
`seT16B_ALLIE` }

2.65.1 Enumeration Type Documentation

2.65.1.1 seT16B_CAPIS

enum `seT16B_CAPIS`

Enumerator

<code>seT16B_Input↔ _0</code>	Capture input signal select.
-----------------------------------	------------------------------

2.65.1.2 seT16B_CAPTRG

```
enum seT16B_CAPTRG
```

Enumerator

seT16B_UpDown	Capture trigger select.
---------------	-------------------------

2.65.1.3 seT16B_CBUFMD

```
enum seT16B_CBUFMD
```

Enumerator

seT16B_Clear	Select the timing to load the comparison value written in the T16BnCCRM register to the compare buffer.
--------------	---

2.65.1.4 seT16B_CCMD

```
enum seT16B_CCMD
```

Enumerator

seT16B_ComparatorMode	Selects the operating mode of the comparator/capture circuit.
-----------------------	---

2.65.1.5 seT16B_ClkSrc

```
enum seT16B_ClkSrc
```

T16B Detailed description

Enumerator

seT16B_IOSC	IOSC is a SYSCLK source.
seT16B_OSC1	OSC1 is a SYSCLK source.
seT16B_OSC3	OSC3 is a SYSCLK source.
seT16B_EXOSC	EXOSC is a SYSCLK source.
seT16B_EXCLN0	EXCLN0 is a SYSCLK source.
seT16B_EXCLN1	EXCLN1 is a SYSCLK source.
seT16B_EXCLN0_INV	EXCLN0_INV inverted input.
seT16B_EXCLN1_INV	EXCLN1_INV inverted input.

2.65.1.6 seT16B_CNTMD

```
enum seT16B_CNTMD
```

Enumerator

seT16B_CountUp	Timer operates in Up Count mode (one shot or repeat mode)
seT16B_CountDown	Timer operates in Down Count mode (one shot or repeat mode)
seT16B_CountUpDown	Timer operates in Up-Down Count mode (one shot or repeat mode)

2.65.1.7 seT16B_EX_ClkDiv

```
enum seT16B_EX_ClkDiv
```

Enumerator

seT16B_EX_CLKDIV↔ _1	EXOSC, EXCLN0, EXCLN1 division ratio is 1/1.
-------------------------	--

2.65.1.8 seT16B_IOSC_ClkDiv

```
enum seT16B_IOSC_ClkDiv
```

Enumerator

seT16B_IOSC_CLKDIV_1	IOSC division ratio is 1/1.
seT16B_IOSC_CLKDIV_2	IOSC division ratio is 1/2.
seT16B_IOSC_CLKDIV_4	IOSC division ratio is 1/4.
seT16B_IOSC_CLKDIV_8	IOSC division ratio is 1/8.
seT16B_IOSC_CLKDIV_16	IOSC division ratio is 1/16.
seT16B_IOSC_CLKDIV_32	IOSC division ratio is 1/32.
seT16B_IOSC_CLKDIV_64	IOSC division ratio is 1/64.
seT16B_IOSC_CLKDIV_128	IOSC division ratio is 1/128.
seT16B_IOSC_CLKDIV_256	IOSC division ratio is 1/256.
seT16B_IOSC_CLKDIV_512	IOSC division ratio is 1/512.
seT16B_IOSC_CLKDIV_1024	IOSC division ratio is 1/1024.
seT16B_IOSC_CLKDIV_2048	IOSC division ratio is 1/2048.
seT16B_IOSC_CLKDIV_4096	IOSC division ratio is 1/4096.
seT16B_IOSC_CLKDIV_8192	IOSC division ratio is 1/8192.
seT16B_IOSC_CLKDIV_16384	IOSC division ratio is 1/16384.
seT16B_IOSC_CLKDIV_32768	IOSC division ratio is 1/32768.

2.65.1.9 seT16B_ONEST

```
enum seT16B_ONEST
```

Enumerator

seT16B_RepeatMode	Timer operates in Repeat Mode.
seT16B_OneShotMode	Timer operates in One-shot Mode.

2.65.1.10 seT16B_OSC1_ClkDiv

```
enum seT16B_OSC1_ClkDiv
```

Enumerator

seT16B_OSC1_CLKDIV_1	OSC1 division ratio is 1/1.
seT16B_OSC1_CLKDIV_2	OSC1 division ratio is 1/2.
seT16B_OSC1_CLKDIV_4	OSC1 division ratio is 1/4.
seT16B_OSC1_CLKDIV_8	OSC1 division ratio is 1/8.
seT16B_OSC1_CLKDIV_16	OSC1 division ratio is 1/16.
seT16B_OSC1_CLKDIV_32	OSC1 division ratio is 1/32.
seT16B_OSC1_CLKDIV_64	OSC1 division ratio is 1/64.
seT16B_OSC1_CLKDIV_128	OSC1 division ratio is 1/128.
seT16B_OSC1_CLKDIV_256	OSC1 division ratio is 1/256.

2.65.1.11 seT16B_OSC3_ClkDiv

```
enum seT16B_OSC3_ClkDiv
```

Enumerator

seT16B_OSC3_CLKDIV_1	OSC3 division ratio is 1/1.
seT16B_OSC3_CLKDIV_2	OSC3 division ratio is 1/2.
seT16B_OSC3_CLKDIV_4	OSC3 division ratio is 1/4.
seT16B_OSC3_CLKDIV_8	OSC3 division ratio is 1/8.
seT16B_OSC3_CLKDIV_16	OSC3 division ratio is 1/16.
seT16B_OSC3_CLKDIV_32	OSC3 division ratio is 1/32.
seT16B_OSC3_CLKDIV_64	OSC3 division ratio is 1/64.
seT16B_OSC3_CLKDIV_128	OSC3 division ratio is 1/128.
seT16B_OSC3_CLKDIV_256	OSC3 division ratio is 1/256.
seT16B_OSC3_CLKDIV_512	OSC3 division ratio is 1/512.
seT16B_OSC3_CLKDIV_1024	OSC3 division ratio is 1/1024.
seT16B_OSC3_CLKDIV_2048	OSC3 division ratio is 1/2048.
seT16B_OSC3_CLKDIV_4096	OSC3 division ratio is 1/4096.
seT16B_OSC3_CLKDIV_8192	OSC3 division ratio is 1/8192.
seT16B_OSC3_CLKDIV_16384	OSC3 division ratio is 1/16384.
seT16B_OSC3_CLKDIV_32768	OSC3 division ratio is 1/32768.

2.65.1.12 seT16B_SCS

```
enum seT16B_SCS
```

Enumerator

seT16B_SyncCapture	Selects either synchronous capture mode or asynchronous capture mode.
--------------------	---

2.65.1.13 seT16B_TOUTINV

enum `seT16B_TOUTINV`

Enumerator

<code>seT16B_Inverted</code>	This bit selects the TOUTnm signal polarity.
------------------------------	--

2.65.1.14 seT16B_TOUTMD

enum `seT16B_TOUTMD`

Enumerator

<code>seT16B_ResetSet</code>	Select TOUT signal generation mode.
------------------------------	-------------------------------------

2.65.1.15 seT16B_TOUTMT

enum `seT16B_TOUTMT`

Enumerator

<code>seT16B_UseBothComparator</code>	Selects whether the comparator MATCH signal of another system is used.
---------------------------------------	--

2.65.1.16 seT16B_TOUTO

enum `seT16B_TOUTO`

Enumerator

<code>seT16B_HighLevelOutput</code>	Selects signal output level when software control mode used.
-------------------------------------	--

2.66 T16B_Types

Data Structures

- struct [seT16B_CCCTL](#)
- struct [seT16B_InitTypeDef](#)
T16B Init structure definition.
- struct [seT16B_ChannelDef](#)
T16B Channel definition.
- struct [seT16B_CCRegsDef](#)
T16B Capture/Compare registers definition.

Variables

- [seT16B_ChannelDef](#) **T16B_CH0**
- [seT16B_ChannelDef](#) **T16B_CH1**

2.67 T16B_Functions

Functions

- void [seT16B_InitStruct](#) ([seT16B_InitTypeDef](#) *T16B_InitStruct)
Fills each [seT16B_InitTypeDef](#) member with its default value.
- [seStatus seT16B_Init](#) ([seT16B_ChannelDef](#) *T16BCHx, [seT16B_InitTypeDef](#) *T16B_InitStruct)
Initializes the T16B peripheral according to the specified parameters in the [seT16B_InitStruct](#).
- void [seT16B_Start](#) (T16B_0_Type *T16Bx)
Starts Timer channel.
- void [seT16B_Stop](#) (T16B_0_Type *T16Bx)
Stops Timer channel.
- void [seT16B_Enable](#) (T16B_0_Type *T16Bx)
Enables Timer channel by start supplying operating clock.
- void [seT16B_Disable](#) (T16B_0_Type *T16Bx)
Disables Timer channel by stop supplying operating clock.
- void [seT16B_ConfigureClock](#) (T16B_0_Type *T16Bx, [seT16B_ClkSrc](#) clock, uint16_t divider)
Configures timer clock source and clock divider.
- uint32_t [seT16B_GetClk](#) (T16B_0_Type *T16Bx)
Discovers T16B clock from registers.
- void [seT16B_ConfigureCounterMode](#) (T16B_0_Type *T16Bx, [seT16B_ONEST](#) mode)
Configures timer counter mode.
- uint16_t [seT16B_GetCmpCapCnt](#) (T16B_0_Type *T16Bx, uint8_t ccsubchan)
Get comparator count 0.
- void [seT16B_SetCmpCapCnt](#) (T16B_0_Type *T16Bx, uint8_t ccsubchan, uint16_t count)
Set comparator count 0.
- uint16_t [seT16B_GetTimerCount](#) (T16B_0_Type *T16Bx)
Get timer count.
- void [seT16B_SetTriggerSignal](#) (T16B_0_Type *T16Bx, uint8_t ccsubchan, [seT16B_CAPIS](#) Level)
Set Trigger Signal 0.
- void [seT16B_SetMaxCounter](#) (T16B_0_Type *T16Bx, uint16_t counter)
Sets Timer counter.
- uint16_t [seT16B_GetMaxCounter](#) (T16B_0_Type *T16Bx)
Gets Timer counter value.
- void [seT16B_EnableInt](#) (T16B_0_Type *T16Bx, [seT16B_Interrupt](#) irq)
Enables Timer channel interrupt.
- void [seT16B_DisableInt](#) (T16B_0_Type *T16Bx, [seT16B_Interrupt](#) irq)
Disables Timer channel interrupt.
- [seInterruptStatus seT16B_GetIntFlag](#) (T16B_0_Type *T16Bx, [seT16B_IntFlag](#) flag)
Returns Timer interrupt flag.
- void [seT16B_ClearIntFlag](#) (T16B_0_Type *T16Bx, [seT16B_IntFlag](#) flag)
Clears Timer channel interrupt.
- void [T16B_0_IRQHandler](#) (void)
Timer0 Interrupt Service Routine.
- void [T16B_1_IRQHandler](#) (void)
Timer1 Interrupt Service Routine.
- [seStatus ConfigurePortsForT16B](#) ([seT16B_ChannelDef](#) *T16BCHx)
Configures ports for this module. Override this function to configure specific ports.

2.67.1 Function Documentation

2.67.1.1 ConfigurePortsForT16B()

```
seStatus ConfigurePortsForT16B (
    seT16B_ChannelDef * T16BCHx )
```

Parameters

<i>T16BCHx</i>	T16B channel definition of type seT16B_ChannelDef
----------------	---

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.67.1.2 seT16B_ClearIntFlag()

```
void seT16B_ClearIntFlag (
    T16B_0_Type * T16Bx,
    seT16B_IntFlag flag )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>flag</i>	Refers to an interrupt seT16B_IntFlag .

Return values

<i>None</i>	
-------------	--

2.67.1.3 seT16B_ConfigureClock()

```
void seT16B_ConfigureClock (
    T16B_0_Type * T16Bx,
    seT16B_ClkSrc clock,
    uint16_t divider )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>clock</i>	This parameter can be a value of seT16B_ClkSrc .
<i>divider</i>	This parameter can be a value of seT16B_ClkDiv .

Return values

<i>None</i>	
-------------	--

2.67.1.4 seT16B_ConfigureCounterMode()

```
void seT16B_ConfigureCounterMode (
    T16B_0_Type * T16Bx,
    seT16B_ONEST mode )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>mode</i>	Set repeat or one shot mode, see seT16B_ONEST .

Return values

<i>None</i>	
-------------	--

2.67.1.5 seT16B_Disable()

```
void seT16B_Disable (
    T16B_0_Type * T16Bx )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
--------------	--

Return values

<i>None</i>	
-------------	--

2.67.1.6 seT16B_DisableInt()

```
void seT16B_DisableInt (
    T16B_0_Type * T16Bx,
    seT16B_Interrupt irq )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>irq</i>	Interrupt to disable seT16B_Interrupt.

Return values

<i>None</i>	
-------------	--

2.67.1.7 seT16B_Enable()

```
void seT16B_Enable (
    T16B_0_Type * T16Bx )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
--------------	--

Return values

<i>None</i>	
-------------	--

2.67.1.8 seT16B_EnableInt()

```
void seT16B_EnableInt (
    T16B_0_Type * T16Bx,
    seT16B_interrupt irq )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>irq</i>	Interrupt to enable.

Return values

<i>None</i>	
-------------	--

2.67.1.9 seT16B_GetClk()

```
uint32_t seT16B_GetClk (
    T16B_0_Type * T16Bx )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
--------------	--

Return values

<i>T16B</i>	clock.
-------------	--------

2.67.1.10 seT16B_GetCmpCapCnt()

```
uint16_t seT16B_GetCmpCapCnt (
    T16B_0_Type * T16Bx,
```

```
uint8_t ccsubchan )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>ccsubchan</i>	Capture/Compare subchannel

Return values

<i>Comparator</i>	count 0 value.
-------------------	----------------

2.67.1.11 seT16B_GetIntFlag()

```
seInterruptStatus seT16B_GetIntFlag (
    T16B_0_Type * T16Bx,
    seT16B_IntFlag flag )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>flag</i>	Refers to an interrupt seT16B_IntFlag.

Return values

<i>InterruptStatus</i>	see seInterruptStatus
------------------------	---------------------------------------

2.67.1.12 seT16B_GetMaxCounter()

```
uint16_t seT16B_GetMaxCounter (
    T16B_0_Type * T16Bx )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
--------------	--

Return values

<i>16-bit</i>	counter value.
---------------	----------------

2.67.1.13 seT16B_GetTimerCount()

```
uint16_t seT16B_GetTimerCount (
    T16B_0_Type * T16Bx )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
--------------	--

Return values

<i>Timer</i>	count value.
--------------	--------------

2.67.1.14 seT16B_Init()

```
seStatus seT16B_Init (
    seT16B_ChannelDef * T16BCHx,
    seT16B_InitTypeDef * T16B_InitStruct )
```

Note

This function configures the module, and module's interrupts. It clears module's interrupts but does not enable interrupt from the module to CPU. This function enables module by start supplying operating clock.

Parameters

<i>T16BCHx</i>	T16B channel definition of type seT16B_ChannelDef
<i>InitStruct</i>	pointer to a seT16B_InitTypeDef structure that contains the configuration information for the specified T16B peripheral.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.67.1.15 seT16B_InitStruct()

```
void seT16B_InitStruct (
    seT16B_InitTypeDef * T16B_InitStruct )
```

Parameters

<i>T16B_InitStruct</i>	pointer to an seT16B_InitTypeDef structure which will be initialized.
------------------------	---

Return values

<i>None</i>	
-------------	--

2.67.1.16 seT16B_SetCmpCapCnt()

```
void seT16B_SetCmpCapCnt (
    T16B_0_Type * T16Bx,
    uint8_t ccsubchan,
    uint16_t count )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>ccsubchan</i>	Capcture/Compare subchannel
<i>count</i>	Value to be set.

Return values

<i>None</i>	
-------------	--

2.67.1.17 seT16B_SetMaxCounter()

```
void seT16B_SetMaxCounter (
    T16B_0_Type * T16Bx,
    uint16_t counter )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>counter</i>	This parameter can be a 16-bit value.

Return values

<i>None</i>	
-------------	--

2.67.1.18 seT16B_SetTriggerSignal()

```
void seT16B_SetTriggerSignal (
    T16B_0_Type * T16Bx,
    uint8_t ccsubchan,
    seT16B\_CAPIS Level )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
<i>ccsubchan</i>	Capcture/Compare subchannel
<i>Level</i>	See seT16B_CAPIS .

Return values

<i>None</i>	
-------------	--

2.67.1.19 seT16B_Start()

```
void seT16B_Start (
    T16B_0_Type * T16Bx )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
--------------	--

Return values

<i>None</i>	
-------------	--

2.67.1.20 seT16B_Stop()

```
void seT16B_Stop (
    T16B_0_Type * T16Bx )
```

Parameters

<i>T16Bx</i>	Pointer to T16B peripheral, a timer channel.
--------------	--

Return values

<i>None</i>	
-------------	--

2.67.1.21 T16B_0_IRQHandler()

```
void T16B_0_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.67.1.22 T16B_1_IRQHandler()

```
void T16B_1_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.68 UART2

The UART is an asynchronous serial interface.

Modules

- [UART2_Constants](#)
- [UART_Types](#)
- [UART_Functions](#)

2.69 UART2_Constants

Macros

- #define `seUART2_INTS(a)` ((`seUART2_Interrupt`)(a))
Combination of any of the `seUART2_Interrupt` enumerations.

Typedefs

- typedef `seCLG_ClkSrc` `seUART2_ClkSrc`

Enumerations

- enum `seUART2_IOSC_ClkDiv` {
 `seUART2_IOSC_CLKDIV_1` = 0,
 `seUART2_IOSC_CLKDIV_2` = 1,
 `seUART2_IOSC_CLKDIV_4` = 2,
 `seUART2_IOSC_CLKDIV_8` = 3 }
- enum `seUART2_OSC1_ClkDiv` { `seUART2_OSC1_CLKDIV_1` = 0 }
- enum `seUART2_OSC3_ClkDiv` {
 `seUART2_OSC3_CLKDIV_1` = 0,
 `seUART2_OSC3_CLKDIV_2` = 1,
 `seUART2_OSC3_CLKDIV_4` = 2,
 `seUART2_OSC3_CLKDIV_8` = 3 }
- enum `seUART2_EXOSC_ClkDiv` { `seUART2_EXOSC_CLKDIV_1` = 0 }
- enum `seUART2_MOD_Stpb` {
 `seUART2_MOD_STPB_1BIT` = 0,
 `seUART2_MOD_STPB_2BIT` = 1 }
- enum `seUART2_MOD_Prmd` {
 `seUART2_MOD_PRMD_EVEN` = 0,
 `seUART2_MOD_PRMD_ODD` = 1 }
- enum `seUART2_MOD_Pren` {
 `seUART2_MOD_PREN_NO_PARITY` = 0,
 `seUART2_MOD_PREN_WITH_PARITY` = 1 }
- enum `seUART2_MOD_Chln` {
 `seUART2_MOD_CHLN_7BIT` = 0,
 `seUART2_MOD_CHLN_8BIT` = 1 }
- enum `seUART2_MOD_Irmd` {
 `seUART2_MOD_IRMD_NO_IRDA` = 0,
 `seUART2_MOD_IRMD_WITH_IRDA` = 1 }
- enum `seUART2_MOD_Outmd` {
 `seUART2_MOD_OUTMD_PUSH_PULL` = 0,
 `seUART2_MOD_OUTMD_OPEN_DRAIN` = 1 }
- enum `seUART2_MOD_Puen` {
 `seUART2_MOD_PUEN_DISABLE` = 0,
 `seUART2_MOD_PUEN_ENABLE` = 1 }
- enum `seUART2_MOD_Invirtx` {
 `seUART2_MOD_INVIRTX_NORMAL` = 0,
 `seUART2_MOD_INVIRTX_INVERT` = 1 }
- enum `seUART2_MOD_Invirrx` {
 `seUART2_MOD_INVIRRX_NORMAL` = 0,
 `seUART2_MOD_INVIRRX_INVERT` = 1 }

- enum **seUART2_BaudRate** {
seUART2_BAUD_RATE_4800 = 4800,
seUART2_BAUD_RATE_9600 = 9600,
seUART2_BAUD_RATE_14400 = 14400,
seUART2_BAUD_RATE_19200 = 19200,
seUART2_BAUD_RATE_38400 = 38400,
seUART2_BAUD_RATE_56000 = 56000,
seUART2_BAUD_RATE_115200 = 115200,
seUART2_BAUD_RATE_128000 = 128000,
seUART2_BAUD_RATE_256000 = 256000 }
- enum **seUART2_Interrupt** {
seUART2_TEDIE_INT = 0x40U,
seUART2_FEIE_INT = 0x20U,
seUART2_PEIE_INT = 0x10U,
seUART2_OEIE_INT = 0x08U,
seUART2_RB2FIE_INT = 0x04U,
seUART2_RB1FIE_INT = 0x02U,
seUART2_TBEIE_INT = 0x01U,
seUART2_ALL_INT }

2.69.1 Enumeration Type Documentation

2.69.1.1 seUART2_EXOSC_ClkDiv

```
enum seUART2_EXOSC_ClkDiv
```

Enumerator

seUART2_EXOSC_CLKDIV _1	EXOSC division ratio is 1/1.
-----------------------------------	------------------------------

2.69.1.2 seUART2_Interrupt

```
enum seUART2_Interrupt
```

Enumerator

seUART2_TEDIE_INT	End of transmisson int. enable.
seUART2_FEIE_INT	Framing error int. enable.
seUART2_PEIE_INT	Parity error int. enable.
seUART2_OEIE_INT	Overrun error int. enable.
seUART2_RB2FIE_INT	Receive buffer 2 byte full int. enable.
seUART2_RB1FIE_INT	Receive buffer 1 byte full int. enable.
seUART2_TBEIE_INT	Transmit buffer empty int. enable.

2.69.1.3 seUART2_IOSC_ClkDiv

```
enum seUART2_IOSC_ClkDiv
```

Enumerator

seUART2_IOSC_CLKDIV↔ _1	IOSC division ratio is 1/1.
seUART2_IOSC_CLKDIV↔ _2	IOSC division ratio is 1/2.
seUART2_IOSC_CLKDIV↔ _4	IOSC division ratio is 1/4.
seUART2_IOSC_CLKDIV↔ _8	IOSC division ratio is 1/8.

2.69.1.4 seUART2_MOD_Chln

```
enum seUART2_MOD_Chln
```

Enumerator

seUART2_MOD_CHLN_7BIT	Character length 7 bit.
seUART2_MOD_CHLN_8BIT	Character length 8 bit.

2.69.1.5 seUART2_MOD_Invirrx

```
enum seUART2_MOD_Invirrx
```

Enumerator

seUART2_MOD_INVIRRX_NORMAL	Invert receive IrDA signal normal.
seUART2_MOD_INVIRRX_INVERT	Invert receive IrDA signal invert.

2.69.1.6 seUART2_MOD_Invirtx

```
enum seUART2_MOD_Invirtx
```

Enumerator

seUART2_MOD_INVIRTX_NORMAL	Invert transmit IrDA signal normal.
seUART2_MOD_INVIRTX_INVERT	Invert transmit IrDA signalinvert.

2.69.1.7 seUART2_MOD_Irmd

```
enum seUART2_MOD_Irmd
```

Enumerator

seUART2_MOD_IRMD_NO_IRDA	No Irda.
seUART2_MOD_IRMD_WITH_IRDA	Irda.

2.69.1.8 seUART2_MOD_Outmd

enum [seUART2_MOD_Outmd](#)

Enumerator

seUART2_MOD_OUTMD_PUSH_PULL	Push pull.
seUART2_MOD_OUTMD_OPEN_DRAIN	Open drain.

2.69.1.9 seUART2_MOD_Pren

enum [seUART2_MOD_Pren](#)

Enumerator

seUART2_MOD_PREN_NO_PARITY	No parity.
seUART2_MOD_PREN_WITH_PARITY	Parity.

2.69.1.10 seUART2_MOD_Prmd

enum [seUART2_MOD_Prmd](#)

Enumerator

seUART2_MOD_PRMD_EVEN	Parity is even.
seUART2_MOD_PRMD_ODD	Parity is odd.

2.69.1.11 seUART2_MOD_Puen

enum [seUART2_MOD_Puen](#)

Enumerator

seUART2_MOD_PUEN_DISABLE	USIN pullup disable.
seUART2_MOD_PUEN_ENABLE	USIN pullup enable.

2.69.1.12 seUART2_MOD_Stpb

```
enum seUART2_MOD_Stpb
```

Enumerator

seUART2_MOD_STPB_1BIT	Stop is 1 bit.
seUART2_MOD_STPB_2BIT	Stop is 2 bit.

2.69.1.13 seUART2_OSC1_ClkDiv

```
enum seUART2_OSC1_ClkDiv
```

Enumerator

seUART2_OSC1_CLKDIV↔ _1	OSC1 division ratio is 1/1.
----------------------------	-----------------------------

2.69.1.14 seUART2_OSC3_ClkDiv

```
enum seUART2_OSC3_ClkDiv
```

Enumerator

seUART2_OSC3_CLKDIV↔ _1	OSC3 division ratio is 1/1.
seUART2_OSC3_CLKDIV↔ _2	OSC3 division ratio is 1/2.
seUART2_OSC3_CLKDIV↔ _4	OSC3 division ratio is 1/4.
seUART2_OSC3_CLKDIV↔ _8	OSC3 division ratio is 1/8.

2.70 UART_Types

Data Structures

- union [seUART2_ClkDiv](#)
UART Init structure definition.
- union [seUART2_Mode](#)
UART Mode structure definition.
- struct [seUART2_InitTypeDef](#)
UART Init structure definition.
- struct [seUART2_ChannelDef](#)
UART Channel definition.

Variables

- [seUART2_ChannelDef](#) **UART2_CH0**
- [seUART2_ChannelDef](#) **UART2_CH1**

2.71 UART_Functions

Functions

- [seStatus seUART2_Init](#) ([seUART2_ChannelDef](#) *UARTCHx, [seUART2_InitTypeDef](#) *InitStruct)
Initializes the UART peripheral according to the specified parameters in the seUART2_InitStruct.
- void [seUART2_InitStruct](#) ([seUART2_InitTypeDef](#) *UART2_InitStruct)
Fills each seUART2_InitTypeDef member with its default value.
- void [seUART2_Enable](#) (UART2_0_Type *UARTx)
Enables UART channel by start supplying operating clock.
- void [seUART2_Disable](#) (UART2_0_Type *UARTx)
Disables UART channel by stop supplying operating clock.
- void [seUART2_ConfigureClock](#) (UART2_0_Type *UARTx, [seUART2_ClkSrc](#) clock, uint16_t divider)
Configures UART clock source and clock divider.
- void [seUART2_ConfigureMode](#) (UART2_0_Type *UARTx, [seUART2_Mode](#) mode)
Configures UART mode register.
- uint32_t [seUART2_GetUartClk](#) (UART2_0_Type *UARTx)
Discovers UART clock from registers.
- [seStatus seUART2_SetBaudRate](#) (UART2_0_Type *UARTx, uint32_t bps)
Sets UART baud rate.
- void [seUART2_SetBaudRateReg](#) (UART2_0_Type *UARTx, uint16_t BRT, uint16_t FMD)
Sets UART baud rate by setting of the BRT and FMD registers.
- void [UART2_0_IRQHandler](#) (void)
Uart0 Interrupt Service Routine.
- void [UART2_1_IRQHandler](#) (void)
Uart1 Interrupt Service Routine.
- void [seUART2_EnableInt](#) (UART2_0_Type *UARTx, [seUART2_Interrupt](#) irq)
Enables UART channel interrupt.
- void [seUART2_DisableInt](#) (UART2_0_Type *UARTx, [seUART2_Interrupt](#) irq)
Disables UART channel interrupt.
- [seInterruptStatus seUART2_GetIntFlag](#) (UART2_0_Type *UARTx, [seUART2_Interrupt](#) irq)
Returns UART interrupt flag status.
- void [seUART2_ClearIntFlag](#) (UART2_0_Type *UARTx, [seUART2_Interrupt](#) irq)
Clears UART channel interrupt.
- uint32_t [seUART2_Send](#) (UART2_0_Type *UARTx, const uint8_t data[], uint32_t size)
Starts sending data to UART transmitter.
- uint32_t [seUART2_Receive](#) (UART2_0_Type *UARTx, uint8_t data[], uint32_t size)
Starts receiving data from UART receiver.
- uint16_t [seUART2_GetData](#) (UART2_0_Type *UARTx)
Receives a byte or two bytes of data from UART receiver.
- void [seUART2_SetData](#) (UART2_0_Type *UARTx, uint8_t byte)
Sends a byte to UART transmitter if the transmit buffer is empty.
- void [seUART2_EnableRxDMAReq](#) (UART2_0_Type *UARTx, [seDMAC_CHANNEL](#) chan)
Enables Receive Buffer Full DMA Request.
- void [seUART2_EnableTxDMAReq](#) (UART2_0_Type *UARTx, [seDMAC_CHANNEL](#) chan)
Enables Transmit Buffer Empty DMA Request Enable.
- [seStatus ConfigurePortsForUart](#) ([seUART2_ChannelDef](#) *UARTCHx)
Configures ports for this module. Override this function to configure specific ports.

2.71.1 Function Documentation

2.71.1.1 ConfigurePortsForUart()

```
seStatus ConfigurePortsForUart (
    seUART2_ChannelDef * UARTCHx )
```

Parameters

<i>UARTCHx</i>	UART channel definition of type seUART2_ChannelDef
----------------	--

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.71.1.2 seUART2_ClearIntFlag()

```
void seUART2_ClearIntFlag (
    UART2_0_Type * UARTx,
    seUART2_Interrupt irq )
```

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
<i>irq</i>	This parameter can be a value or combination of seUART2_Interrupt except of seUART2_RB2FIE_INT, seUART2_RB1FIE_INT, seUART2_TBEIE_INT flags that must be cleared by reading / writing TXD, RXD registers.

Return values

<i>None</i>	
-------------	--

2.71.1.3 seUART2_ConfigureClock()

```
void seUART2_ConfigureClock (
    UART2_0_Type * UARTx,
    seUART2_ClkSrc clock,
    uint16_t divider )
```

Parameters

<i>UARTx</i>	This parameter defines a uart channel and can be a value of UART2_0_Type.
<i>clock</i>	This parameter can be a value of seUART2_ClkSrc.
<i>divider</i>	This parameter can be a value of seUART2_ClkDiv .

Return values

<i>None</i>	
-------------	--

2.71.1.4 `seUART2_ConfigureMode()`

```
void seUART2_ConfigureMode (
    UART2_0_Type * UARTx,
    seUART2_Mode mode )
```

Parameters

<i>UARTx</i>	This parameter defines a uart channel and can be a value of UART2_0_Type.
<i>mode</i>	This parameter can be a value of seUART2_Mode .

Return values

<i>None</i>	
-------------	--

2.71.1.5 `seUART2_Disable()`

```
void seUART2_Disable (
    UART2_0_Type * UARTx )
```

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
--------------	-----------------------------

Return values

<i>None</i>	
-------------	--

2.71.1.6 `seUART2_DisableInt()`

```
void seUART2_DisableInt (
    UART2_0_Type * UARTx,
    seUART2_Interrupt irq )
```

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
<i>irq</i>	Interrupt to disable, see seUART2_Interrupt .

Return values

<i>None</i>	
-------------	--

2.71.1.7 seUART2_Enable()

```
void seUART2_Enable (
    UART2_0_Type * UARTx )
```

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
--------------	-----------------------------

Return values

<i>None</i>	
-------------	--

2.71.1.8 seUART2_EnableInt()

```
void seUART2_EnableInt (
    UART2_0_Type * UARTx,
    seUART2\_Interrupt irq )
```

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
<i>irq</i>	Interrupt to enable, see seUART2_Interrupt .

Return values

<i>Status</i>	
---------------	--

2.71.1.9 seUART2_EnableRxDMAReq()

```
void seUART2_EnableRxDMAReq (
    UART2_0_Type * UARTx,
    seDMAC\_CHANNEL chan )
```

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
<i>chan</i>	The DMA channel, seDMAC_CHANNEL .

Return values

<i>None</i>	
-------------	--

2.71.1.10 seUART2_EnableTxDMAReq()

```
void seUART2_EnableTxDMAReq (
    UART2_0_Type * UARTx,
    seDMAC_CHANNEL chan )
```

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
<i>chan</i>	The DMA channel, seDMAC_CHANNEL .

Return values

<i>None</i>	
-------------	--

2.71.1.11 seUART2_GetData()

```
uint16_t seUART2_GetData (
    UART2_0_Type * UARTx )
```

Note

This function can be called from Interrupt Service routine since its implementation is deterministic.

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
--------------	-----------------------------

Return values

<i>data</i>	one or two bytes packed as 16 bit word.
-------------	---

2.71.1.12 seUART2_GetIntFlag()

```
seInterruptStatus seUART2_GetIntFlag (
    UART2_0_Type * UARTx,
    seUART2_Interrupt irq )
```

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
--------------	-----------------------------

Parameters

<i>irq</i>	This parameter can be a value or combination of seUART2_Interrupt . In case of combination of flags - all of them must be set in order to return "INTERRUPT_OCCURED" status.
------------	--

Return values

<i>InterruptStatus</i>	see seInterruptStatus
------------------------	---------------------------------------

2.71.1.13 seUART2_GetUartClk()

```
uint32_t seUART2_GetUartClk (
    UART2_0_Type * UARTx )
```

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
--------------	-----------------------------

Return values

<i>UART</i>	clock frequency to use in baud rate calculation.
-------------	--

2.71.1.14 seUART2_Init()

```
seStatus seUART2_Init (
    seUART2_ChannelDef * UARTCHx,
    seUART2_InitTypeDef * InitStruct )
```

Parameters

<i>UARTCHx</i>	UART channel definition of type seUART2_ChannelDef
<i>InitStruct</i>	pointer to a seUART2_InitTypeDef structure that contains the configuration information for the specified UART peripheral.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.71.1.15 seUART2_InitStruct()

```
void seUART2_InitStruct (
    seUART2_InitTypeDef * UART2_InitStruct )
```

Parameters

<i>UART2_InitStruct</i>	Pointer to an seUART2_InitTypeDef structure which will be initialized.
-------------------------	--

Return values

<i>None</i>	
-------------	--

2.71.1.16 seUART2_Receive()

```
uint32_t seUART2_Receive (
    UART2_0_Type * UARTx,
    uint8_t data[],
    uint32_t size )
```

Note

This function can not be called from an interrupt Service routine since it can wait for a timeout. Timeout value is equal to seSHORT_WAIT_TIMEOUT_MS.

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
<i>data</i>	This parameter is a pointer to a destination byte array.
<i>size</i>	This parameter is a destination array size.

Return values

<i>byterecieved</i>	Returns a number of received bytes.
---------------------	-------------------------------------

2.71.1.17 seUART2_Send()

```
uint32_t seUART2_Send (
    UART2_0_Type * UARTx,
    const uint8_t data[],
    uint32_t size )
```

Note

This function can not be called from Interrupt Service routine since it can wait for a timeout. Timeout value is equal to seSHORT_WAIT_TIMEOUT_MS.

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
<i>data</i>	This parameter is a pointer to a source byte array.
<i>size</i>	This parameter is a number of source bytes to send. User is responsible for validation of the array size and the number of bytes to send.

Return values

<i>bytesent</i>	Returns a number of sent bytes.
-----------------	---------------------------------

2.71.1.18 seUART2_SetBaudRate()

```
seStatus seUART2_SetBaudRate (
    UART2_0_Type * UARTx,
    uint32_t bps )
```

Note

For the transfer rate range configurable in the UART, refer to "UART Characteristics, Transfer baud rates" in the "Electrical Characteristics" chapter.

Parameters

<i>UARTx</i>	This parameter defines a uart channel and can be a value of UART2_0_Type.
<i>bps</i>	This parameter can be a value of seUART2_BaudRate.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.71.1.19 seUART2_SetBaudRateReg()

```
void seUART2_SetBaudRateReg (
    UART2_0_Type * UARTx,
    uint16_t BRT,
    uint16_t FMD )
```

Note

Values of these registers shall be calculated in advance based on the formula.

$$\text{bps} = \text{CLK_UART} / \{(\text{BRT} + 1) \times 16 + \text{FMD}\}$$

Parameters

<i>UARTx</i>	This parameter defines a uart channel and can be a value of UART2_0_Type.
<i>BRT</i>	This parameter can be a value of (0 to 255).
<i>FMD</i>	This parameter can be a value of (0 to 15).

Return values

<i>None</i>	
-------------	--

2.71.1.20 seUART2_SetData()

```
void seUART2_SetData (
    UART2_0_Type * UARTx,
    uint8_t byte )
```

Note

This function can be called from Interrupt Service routine since its implementation is deterministic.

Parameters

<i>UARTx</i>	Pointer to UART peripheral.
<i>byte</i>	This parameter is a byte of data.

Return values

<i>None</i>	
-------------	--

2.71.1.21 UART2_0_IRQHandler()

```
void UART2_0_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.71.1.22 UART2_1_IRQHandler()

```
void UART2_1_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.72 USB_Constants_and_Macros

Macros

- #define `seUSB_MAIN_INTS(a)` `((seUSB_MainInterrupt)((a)))`
Combination of any of the seUSB_MainInterrupt enumerations.
- #define `seUSB_SIE_INTS(a)` `((seUSB_SieInterrupt)((a)))`
Combination of any of the seUSB_SieInterrupt enumerations.
- #define `seUSB_GPE_INTS(a)` `((seUSB_GpeInterrupt)((a)))`
Combination of any of the seUSB_GpeInterrupt enumerations.
- #define `seUSB_EP0_INTS(a)` `((seUSB_Ep0Interrupt)((a)))`
Combination of any of the seUSB_Ep0Interrupt enumerations.
- #define `seUSB_EPM_INTS(a)` `((seUSB_EpmInterrupt)((a)))`
Combination of any of the seUSB_EpmInterrupt enumerations.

Enumerations

- enum `seUSB_ClkSrc` {
 `seUSB_USBOSC` = 1,
 `seUSB_PLL` = 0 }
- enum `seUSB_PSEL` {
 `seUSB_GPIO` = 0,
 `seUSB_PERIPH` = 1 }
- enum `seUSB_TRCTL_OPMODE` {
 `seUSB_TRCTL_OPMODE_NORMAL` = 0,
 `seUSB_TRCTL_OPMODE_NONDRIVING` = 1,
 `seUSB_TRCTL_OPMODE_DISBITSTUFF` = 2 }
- These bits set the opmode.*
- enum `seUSB_EPCFG_MAXSIZE` {
 `seUSB_EPCFG_MAXSIZE_8` = 1,
 `seUSB_EPCFG_MAXSIZE_16` = 2,
 `seUSB_EPCFG_MAXSIZE_32` = 4,
 `seUSB_EPCFG_MAXSIZE_64` = 8 }
- USB EP0 Maximum Packet Size.*
- enum `seUSB_EPCFG_DIR` {
 `seUSB_EPCFG_IN` = 1,
 `seUSB_EPCFG_OUT` = 0 }
- This bit sets the transfer direction of Endpoint.*
- enum `seUSB_EPCFG_TGLMOD` {
 `seUSB_EPCFG_ALWAYS` = 1,
 `seUSB_EPCFG_NORMAL` = 0 }
- This bit sets the toggle mode of Endpoint note: Always performs the toggle for every transaction. (Performs the toggle only when the transaction ends normally.*
- enum `seUSB_MainInterrupt` {
 `seUSB_MAIN_SIEIF_INT` = 0x80U,
 `seUSB_MAIN_GPEPIF_INT` = 0x40U,
 `seUSB_MAIN_EP0IF_INT` = 0x02U,
 `seUSB_MAIN_EP0SETIF_INT` = 0x01U,
 `seUSB_MAIN_ALL_INT` = `seUSB_MAIN_SIEIF_INT` | `seUSB_MAIN_GPEPIF_INT` | `seUSB_MAIN_EP0IF_INT` | `seUSB_MAIN_EP0SETIF_INT` }

SIEIF, GPEPIF, EP0IF - these *USBMAININTF* register's bits indicate the interrupt cause occurrence status in each USB interrupt group. *EP0SETIF* - this bit indicates the *EP0* setup completion interrupt cause occurrence status.

- enum `seUSB_SieInterrupt` {
`seUSB_SIE_NONJIF_INT` = 0x40U,
`seUSB_SIE_RESETIF_INT` = 0x20U,
`seUSB_SIE_SUSPENDIF_INT` = 0x10U,
`seUSB_SIE_SOFIF_INT` = 0x08U,
`seUSB_SIE_JIF_INT` = 0x04U,
`seUSB_SIE_ATADDRI_INT` = 0x01U,
`seUSB_SIE_ALL_INT` = 0x7d }

These USBSIEINTF register's bits indicate the SIE interrupt cause occurrence status.

- enum `seUSB_GpeInterrupt` {
`seUSB_GPE_EPCIF_INT` = 0x04U,
`seUSB_GPE_EPBIF_INT` = 0x02U,
`seUSB_GPE_EPAIF_INT` = 0x01U }

These USBGPEPINT register's bits indicate the General-Purpose endpoint interrupt occurrence status.

- enum `seUSB_Ep0Interrupt` {
`seUSB_EP0_INACKIF_INT` = 0x20U,
`seUSB_EP0_OUTACKIF_INT` = 0x10U,
`seUSB_EP0_INNAKIF_INT` = 0x08U,
`seUSB_EP0_OUTNAKIF_INT` = 0x04U,
`seUSB_EP0_INERRIF_INT` = 0x02U,
`seUSB_EP0_OUTERRIF_INT` = 0x01U,
`seUSB_EP0_ALL_INT` = 0x3F }

These USBEP0INTF register's bits indicate the EP0 interrupt occurrence status.

- enum `seUSB_EpmInterrupt` {
`seUSB_EPM_OUTSHACKIF_INT` = 0x40U,
`seUSB_EPM_INACKIF_INT` = 0x20U,
`seUSB_EPM_OUTACKIF_INT` = 0x10U,
`seUSB_EPM_INNAKIF_INT` = 0x08U,
`seUSB_EPM_OUTNAKIF_INT` = 0x04U,
`seUSB_EPM_INERRIF_INT` = 0x02U,
`seUSB_EPM_OUTERRIF_INT` = 0x01U }

These USBEPmINTF register's bits indicate the EPm interrupt occurrence status. USBEPmINTF.OUTSHACKIF bit: E↔Pm short packet reception interrupt USBEPmINTF.INACKIF bit: EPm ACK reception interrupt USBEPmINTF.OUTACKIF bit: EPm ACK transmission interrupt USBEPmINTF.INNAKIF bit: EPm NAK reception interrupt USBEPmINTF.OUTNAKIF bit: EPm NAK transmission interrupt USBEPmINTF.INERRIF bit: EPm STALL reception interrupt USBEPmINTF.OUTERRIF bit: EPm STALL transmission interrupt.

2.72.1 Enumeration Type Documentation

2.72.1.1 seUSB_ClkSrc

```
enum seUSB_ClkSrc
```

Enumerator

<code>seUSB_USBOSC</code>	48 MHz clock input
<code>seUSB_PLL</code>	PLL clock input.

2.72.1.2 seUSB_Ep0Interrupt

```
enum seUSB_Ep0Interrupt
```

Enumerator

seUSB_EP0_INACKIF_INT	Cleared by writing 1.
seUSB_EP0_OUTACKIF_INT	Cleared by writing 1.
seUSB_EP0_INNAKIF_INT	Cleared by writing 1.
seUSB_EP0_OUTNAKIF_INT	Cleared by writing 1.
seUSB_EP0_INERRIF_INT	Cleared by writing 1.
seUSB_EP0_OUTERRIF_INT	Cleared by writing 1.

2.72.1.3 seUSB_EPCFG_DIR

```
enum seUSB_EPCFG_DIR
```

Enumerator

seUSB_EPCFG_IN	IN.
seUSB_EPCFG_OUT	OUT.

2.72.1.4 seUSB_EPCFG_MAXSIZE

```
enum seUSB_EPCFG_MAXSIZE
```

Note

It should be set to the same size as the bMaxPacketSize0 in the Device Descriptor.

Enumerator

seUSB_EPCFG_MAXSIZE_8	8 bytes
seUSB_EPCFG_MAXSIZE_16	16 bytes
seUSB_EPCFG_MAXSIZE_32	32 bytes
seUSB_EPCFG_MAXSIZE_64	64 bytes

2.72.1.5 seUSB_EPCFG_TGLMOD

```
enum seUSB_EPCFG_TGLMOD
```

Enumerator

seUSB_EPCFG_ALWAYS	always
seUSB_EPCFG_NORMAL	normal

2.72.1.6 seUSB_EpmInterrupt

enum `seUSB_EpmInterrupt`

Enumerator

<code>seUSB_EPM_OUTSHACKIF_INT</code>	Cleared by writing 1.
<code>seUSB_EPM_INACKIF_INT</code>	Cleared by writing 1.
<code>seUSB_EPM_OUTACKIF_INT</code>	Cleared by writing 1.
<code>seUSB_EPM_INNAKIF_INT</code>	Cleared by writing 1.
<code>seUSB_EPM_OUTNAKIF_INT</code>	Cleared by writing 1.
<code>seUSB_EPM_INERRIF_INT</code>	Cleared by writing 1.
<code>seUSB_EPM_OUTERRIF_INT</code>	Cleared by writing 1.

2.72.1.7 seUSB_GpeInterrupt

enum `seUSB_GpeInterrupt`

Enumerator

<code>seUSB_GPE_EPCIF_INT</code>	Cleared by writing 1 to the interrupt flag in the USBEPCINTF register.
<code>seUSB_GPE_EPBIF_INT</code>	Cleared by writing 1 to the interrupt flag in the USBEPBINTF register.
<code>seUSB_GPE_EPAIF_INT</code>	Cleared by writing 1 to the interrupt flag in the USBEPAINTF register.

2.72.1.8 seUSB_MainInterrupt

enum `seUSB_MainInterrupt`

Note

When the interrupt is enabled using the corresponding interrupt enable bit (USBMAININTE register), setting the interrupt flag in this register outputs an interrupt request to the CPU core.

Enumerator

<code>seUSB_MAIN_SIEIF_INT</code>	Cleared by writing 1 to the interrupt flag in the USBSIEINTF register.
<code>seUSB_MAIN_GPEPIF_INT</code>	Cleared by writing 1 to the interrupt flag in the USBEPmINTF register.
<code>seUSB_MAIN_EP0IF_INT</code>	Cleared by writing 1 to the interrupt flag in the USBEP0INTF register.
<code>seUSB_MAIN_EP0SETIF_INT</code>	Cleared by writing 1.

2.72.1.9 seUSB_PSEL

enum `seUSB_PSEL`

Enumerator

seUSB_GPIO	Port Group Function Assignment.
seUSB_PERIPH	Port Group Function Assignment as peripheral (SVD2 Ch.1)

2.72.1.10 seUSB_SieInterrupt

```
enum seUSB_SieInterrupt
```

USBSIEINTF.NONJIF bit: NonJ detection interrupt USBSIEINTF.RESETIF bit: Reset detection interrupt USBSIEINTF.SUSPENDIF bit: Suspend detection interrupt USBSIEINTF.SOFIF bit: SOF reception interrupt USBSIEINTF.JIF bit: J detection interrupt USBSIEINTF.ATADDRIF bit: Automatic address setting completion interrupt

Enumerator

seUSB_SIE_NONJIF_INT	Cleared by writing 1.
seUSB_SIE_RESETIF_INT	Cleared by writing 1.
seUSB_SIE_SUSPENDIF_INT	Cleared by writing 1.
seUSB_SIE_SOFIF_INT	Cleared by writing 1.
seUSB_SIE_JIF_INT	Cleared by writing 1.
seUSB_SIE_ATADDRI_INT	Cleared by writing 1.

2.72.1.11 seUSB_TRCTL_OPMODE

```
enum seUSB_TRCTL_OPMODE
```

Enumerator

seUSB_TRCTL_OPMODE_NORMAL	NORMAL.
seUSB_TRCTL_OPMODE_NONDRIVING	NONDRIVING.
seUSB_TRCTL_OPMODE_DISBITSTUFF	DISBITSTUFF.

2.73 USB_Types

Data Structures

- struct [seUSB_InitTypeDef](#)
USB Init structure definition.

2.74 USB_Functions

Modules

- [USB_EP0_Functions](#)

This group of functions initializes and manages Control Endpoint.

- [USB_EPM_Functions](#)

This group of functions initializes and manages general purpose endpoints (EPa, EPb, and EPc).

- [USB_EPn_Functions](#)

This group of functions and manages all endpoints (EP0, EPa, EPb, and EPc), and it is also used for their registers.

- [USB_Bus_Management](#)

The auto-negotiation function set in [seUSB_InitUsbModule\(\)](#) automatically performs Suspend detection, Reset detection, and Resume detection, with checking the state of the USB bus for each operation. Following functions are called from interrupt service routines upon bus state changes.

- [USB_FIFO_Management](#)

- [USB_Interrupt_Management](#)

This group of functions initializes and manages USB module interrupts. For details refer to [seUSB_MainInterrupt](#), [seUSB_SieInterrupt](#), [seUSB_GpeInterrupt](#), [seUSB_Ep0Interrupt](#), [seUSB_EpmInterrupt](#).

Functions

- [seStatus seUSB_Init](#) ([seUSB_InitTypeDef](#) *InitStruct)

Initializes the USB peripheral according to the specified parameters in the [seUSB_InitStruct](#).

- void [seUSB_Enable](#) (void)

Enables USB channel by start supplying operating clock.

- void [seUSB_Disable](#) (void)

Disables USB channel by stop supplying operating clock.

- [seStatus seUSB_Attach](#) (void)

Perform necessary actions when VBus is connected.

- void [seUSB_Detach](#) (void)

Perform necessary actions when VBus is disconnected.

- void [seUSB_InitUsbModule](#) (void)

The basic configurations for the USB controller. This function could be called on Attach event after clocks are activated.

- void [seUSB_ConfigureDefaultEndpoints](#) (void)

The basic configurations for the USB endpoints.

- void [seUSB_ConfigureEPM](#) (uint32_t EPNum, uint32_t val, uint32_t dir)

Configures a general purpose endpoint.

- void [seUSB_EnableEPM](#) (uint32_t EPNum)

Enables a general purpose endpoint.

- void [seUSB_DisableEPM](#) (uint32_t EPNum)

Disables a general purpose endpoint.

- void [seUSB_SetStall](#) (uint32_t EPNum)

Sets Stall.

- void [seUSB_ClrStall](#) (uint32_t EPNum)

Clears Stall.

- void [SVD2_1_IRQHandler](#) (void)

USB Interrupt Service Routine.

- void [PORT_IRQHandler](#) (void)

Port Interrupt Service Routine.

- void [seUSB_EnableInt](#) (IRQn_Type irq)

Enable USB Interrupt in NVIC.

- void [seUSB_DisableInt](#) (IRQn_Type irq)

Disables USB Interrupt in NVIC.

- uint16_t [seUSB_IsVbusConnected](#) (void)

Shows VBUS state. It should not change over last 5 times.

- void [seUSB_ConfigurePortsForUsb](#) ([seUSB_PSEL](#) selection)

Assign the USB output functions to the ports.

- void [seUSB_ConfSvdDetectDisconnect](#) (void)

Configure and activate the SVD for detecting VBUS disconnection.

- void [seUSB_ActivateUSBCLK](#) (void)

Turn on pll and start supplying USB clock. This function is used When using PLL for USB circuit.

- void [seUSB_DeactivateUSBCLK](#) (void)

Turn off pll and stop supplying USB clock. This function is used When using PLL for USB circuit.

2.74.1 Function Documentation

2.74.1.1 PORT_IRQHandler()

```
void PORT_IRQHandler (
    void )
```

Return values

None	
------	--

2.74.1.2 seUSB_ActivateUSBCLK()

```
void seUSB_ActivateUSBCLK (
    void )
```

Return values

None	
------	--

2.74.1.3 seUSB_Attach()

```
seStatus seUSB_Attach (
    void )
```

Return values

Status	can be a value of seStatus
--------	--

2.74.1.4 seUSB_ClrStall()

```
void seUSB_ClrStall (
    uint32_t EPNum )
```

Parameters

<i>EPNum</i>	Endpoint number
--------------	-----------------

Return values

<i>None</i>	
-------------	--

2.74.1.5 seUSB_ConfigureDefaultEndpoints()

```
void seUSB_ConfigureDefaultEndpoints (
    void )
```

Return values

<i>None</i>	
-------------	--

2.74.1.6 seUSB_ConfigureEPm()

```
void seUSB_ConfigureEPm (
    uint32_t EPNum,
    uint32_t val,
    uint32_t dir )
```

Parameters

<i>EPNum</i>	Endpoint number
<i>val</i>	Endpoint max size
<i>dir</i>	Endpoint direction

Return values

<i>None</i>	
-------------	--

2.74.1.7 seUSB_ConfigurePortsForUsb()

```
void seUSB_ConfigurePortsForUsb (
    seUSB_PSEL selection )
```

Parameters

<i>selection</i>	This parameter is of seUSB_PSEL .
------------------	---

Return values

<i>None</i>	
-------------	--

2.74.1.8 seUSB_ConfSvdDetectDisconnect()

```
void seUSB_ConfSvdDetectDisconnect (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.74.1.9 seUSB_DeactivateUSBCLK()

```
void seUSB_DeactivateUSBCLK (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.74.1.10 seUSB_Detach()

```
void seUSB_Detach (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.74.1.11 seUSB_Disable()

```
void seUSB_Disable (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.74.1.12 seUSB_DisableEPm()

```
void seUSB_DisableEPm (
    uint32_t EPNum )
```

Parameters

<i>EPNum</i>	Endpoint number
--------------	-----------------

Return values

<i>None</i>	
-------------	--

2.74.1.13 seUSB_DisableInt()

```
void seUSB_DisableInt (
    IRQn_Type irq )
```

Parameters

<i>irq</i>	This parameter is of IRQn_Type.
------------	---------------------------------

Return values

<i>None</i>	
-------------	--

2.74.1.14 seUSB_Enable()

```
void seUSB_Enable (
    void )
```

Return values

<i>None</i>	
-------------	--

2.74.1.15 seUSB_EnableEPm()

```
void seUSB_EnableEPm (
    uint32_t EPNum )
```

Parameters

<i>EPNum</i>	Endpoint number
--------------	-----------------

Return values

<i>None</i>	
-------------	--

2.74.1.16 seUSB_EnableInt()

```
void seUSB_EnableInt (
    IRQn_Type irq )
```

Parameters

<i>irq</i>	This parameter is of IRQn_Type.
------------	---------------------------------

Return values

<i>None</i>	
-------------	--

2.74.1.17 seUSB_Init()

```
seStatus seUSB_Init (
    seUSB_InitTypeDef * InitStruct )
```

Parameters

<i>InitStruct</i>	pointer to a seUSB_InitTypeDef structure that contains the configuration information for the specified USB peripheral.
-------------------	--

Note

If USB_DMA is defined then library software initializes DMAC and reserves seDMAC_CH0 as DMA read fifo channel and seDMAC_CH1 as DMA write fifo channel. If USB_DMA is not defined, default behavior, then library does not use DMA for copying data to or from fifo.

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.74.1.18 seUSB_InitUsbModule()

```
void seUSB_InitUsbModule (
    void )
```

Return values

<i>None</i>	
-------------	--

2.74.1.19 seUSB_IsVbusConnected()

```
uint16_t seUSB_IsVbusConnected (
    void )
```

Return values

<i>connected</i>	1 if Vbus connected and 0 if not.
------------------	-----------------------------------

2.74.1.20 seUSB_SetStall()

```
void seUSB_SetStall (
    uint32_t EPNum )
```

Parameters

<i>EPNum</i>	Endpoint number
--------------	-----------------

Return values

<i>None</i>	
-------------	--

2.74.1.21 SVD2_1_IRQHandler()

```
void SVD2_1_IRQHandler (
    void )
```

Return values

<i>None</i>	
-------------	--

2.75 USB_EP0_Functions

This group of functions initializes and manages Control Endpoint.

Functions

- void [seUSB_InitEp0](#) (void)
This function initializes EP0.
- void [seUSB_ClearEP0Fifo](#) (void)
This function clears EP0 FIFO.
- void [seUSB_SetEp0Dir](#) (uint32_t dir)
This function sets EP0 direction.
- uint32_t [seUSB_GetEp0Dir](#) (void)
This function gets EP0 direction.
- void [seUSB_SetupStage](#) (void)
This function is a place holder for a standard stack call. It is not needed with this HW.
- void [seUSB_DataInStage](#) (void)
This function configures EP0 to In.
- void [seUSB_DataOutStage](#) (void)
This function configures EP0 to Out.
- void [seUSB_StatusOutStage](#) (void)
This function performs actions conforming with the Status Out stage.
- void [seUSB_StatusInStage](#) (void)
This function performs actions conforming with the Status In stage.
- void [seUSB_GetSetupPacket](#) (USB_SETUP_PACKET *packet)
This function reads and parses control packet.

2.75.1 Function Documentation

2.75.1.1 seseUSB_StatusOutStage()

```
void seseUSB_StatusOutStage (
    void )
```

Return values

None	
------	--

2.75.1.2 seUSB_ClearEP0Fifo()

```
void seUSB_ClearEP0Fifo (
    void )
```

Return values

None	
------	--

2.75.1.3 seUSB_DataInStage()

```
void seUSB_DataInStage (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.75.1.4 seUSB_DataOutStage()

```
void seUSB_DataOutStage (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.75.1.5 seUSB_GetEp0Dir()

```
uint32_t seUSB_GetEp0Dir (  
    void )
```

Return values

<i>Returns</i>	1 (IN) or 0 (OUT).
----------------	--------------------

2.75.1.6 seUSB_GetSetupPacket()

```
void seUSB_GetSetupPacket (  
    USB_SETUP_PACKET * packet )
```

Parameters

<i>packet</i>	This parameter is of USB_SETUP_PACKET
---------------	---------------------------------------

Return values

<i>None</i>	
-------------	--

2.75.1.7 seUSB_InitEp0()

```
void seUSB_InitEp0 (  
    void )
```

Note

: Auto force NAK for EP0

Return values

<i>None</i>	
-------------	--

2.75.1.8 seUSB_SetEp0Dir()

```
void seUSB_SetEp0Dir (
    uint32_t dir )
```

Parameters

<i>dir</i>	1 (IN) or 0 (OUT).
------------	--------------------

Return values

<i>None</i>	
-------------	--

2.75.1.9 seUSB_SetupStage()

```
void seUSB_SetupStage (
    void )
```

Return values

<i>None</i>	
-------------	--

2.75.1.10 seUSB_StatusInStage()

```
void seUSB_StatusInStage (
    void )
```

Return values

<i>None</i>	
-------------	--

2.76 USB_EPm_Functions

This group of functions initializes and manages general purpose endpoints (EPa, EPb, and EPc).

Functions

- void [seUSB_SetEPmDir](#) (uint32_t EPNum, uint32_t dir)
This function sets general purpose endpoints direction.
- uint32_t [seUSB_GetEPmDir](#) (uint32_t EPNum)
This function gets general purpose endpoints direction.
- void [seUSB_ClearEPmFifo](#) (uint16_t EPNum)
This function clears general purpose endpoints FIFO.
- void [seUSB_ResetEPm](#) (uint16_t EPNum)
This function resets general purpose endpoints.

2.76.1 Function Documentation

2.76.1.1 seUSB_ClearEPmFifo()

```
void seUSB_ClearEPmFifo (
    uint16_t EPNum )
```

Parameters

<i>EPNum</i>	Endpoint number
--------------	-----------------

Return values

<i>None</i>	
-------------	--

2.76.1.2 seUSB_GetEPmDir()

```
uint32_t seUSB_GetEPmDir (
    uint32_t EPNum )
```

Parameters

<i>EPNum</i>	Endpoint number
--------------	-----------------

Return values

<i>Returns</i>	1 (IN) or 0 (OUT).
----------------	--------------------

2.76.1.3 seUSB_ResetEPm()

```
void seUSB_ResetEPm (
    uint16_t EPNum )
```

Parameters

<i>EPNum</i>	Endpoint number
--------------	-----------------

Return values

<i>None</i>	
-------------	--

2.76.1.4 seUSB_SetEPmDir()

```
void seUSB_SetEPmDir (
    uint32_t EPNum,
    uint32_t dir )
```

Parameters

<i>EPNum</i>	Endpoint number
<i>dir</i>	1 (IN) or 0 (OUT).

Return values

<i>None</i>	
-------------	--

2.77 USB_EPn_Functions

This group of functions manages all endpoints (EP0, EPa, EPb, and EPc), and it is also used for their registers.

Functions

- void [seUSB_ClearEPnFifos](#) (void)

This function clears all FIFOs at once.

2.77.1 Function Documentation

2.77.1.1 seUSB_ClearEPnFifos()

```
void seUSB_ClearEPnFifos (  
    void )
```

Return values

None	
------	--

2.78 USB_Bus_Management

The auto-negotiation function set in `seUSB_InitUsbModule()` automatically performs Suspend detection, Reset detection, and Resume detection, with checking the state of the USB bus for each operation. Following functions are called from interrupt service routines upon bus state changes.

Functions

- void `seUSB_Reset` (void)
This Function sets Device in Reset State. According to 2.0 spec:
- void `seUSB_Suspend` (void)
In suspend mode make sure the NONJIF interrupt is enabled.
- void `seUSB_Resume` (void)
The host requests to return from Suspend. In this case, the USBCTL.NONJDETEN bit should be cleared.
- void `seUSB_Snooze` (`seState` en)
The USB controller has a Snooze function to reduce current consumption when it is not active or in Suspend state. When the SNOOZE signal is negated, the USB controller resumes operating after 5 clocks have elapsed. (The oscillation must be stabilized at this time.) Notes: Be sure to avoid accessing to the USB controller for five 48-MHz clock cycles from the SNOOZE signal being asserted/negated. Be sure to avoid accessing to the FIFO while the 48 MHz clock is stopped after the SNOOZE signal is asserted.
- `seStatus` `seUSB_IsSnoozing` (void)
Reports USB controller snooze status.
- void `seUSB_WakeUp` (void)
To resume from Suspend with a remote wake-up set controls the remote wakeup signal (K) output. The bit is set to 1 and in 15 msec set to 0.
- void `seUSB_Connect` (void)
This function enables USB controller operations and enables the USB_DP pin (D+ line) pull-up resistor.
- void `seUSB_Disconnect` (void)
this function disables USB controller operations and disables the USB_DP pin (D+ line) pull-up resistor
- void `seUSB_SetAddress` (uint32_t adr, uint32_t setup)
Auto address setup, adr argument ignored.
- uint32_t `seUSB_GetAddress` (void)
This function reads back the device address.

2.78.1 Function Documentation

2.78.1.1 seUSB_Connect()

```
void seUSB_Connect (
    void )
```

Return values

None	
------	--

2.78.1.2 seUSB_Disconnect()

```
void seUSB_Disconnect (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.78.1.3 seUSB_GetAddress()

```
uint32_t seUSB_GetAddress (  
    void )
```

Return values

<i>addr</i>	Device address.
-------------	-----------------

2.78.1.4 seUSB_IsSnoozing()

```
seStatus seUSB_IsSnoozing (  
    void )
```

Return values

<i>Status</i>	can be a value of seStatus
---------------	--

2.78.1.5 seUSB_Reset()

```
void seUSB_Reset (  
    void )
```

1. After receiving a reset, the device is then addressable at the default address.
2. When the reset process is complete, the USB device is operating at the correct speed.
3. The device must also respond successfully to device and configuration descriptor requests and return appropriate information.

Return values

<i>None</i>	
-------------	--

2.78.1.6 seUSB_Resume()

```
void seUSB_Resume (  
    void )
```

```
void )
```

Return values

<i>None</i>	
-------------	--

2.78.1.7 seUSB_SetAddress()

```
void seUSB_SetAddress (
    uint32_t adr,
    uint32_t setup )
```

Parameters

<i>adr</i>	ignored
<i>setup</i>	1 set auto address mode

Return values

<i>None</i>	
-------------	--

2.78.1.8 seUSB_Snooze()

```
void seUSB_Snooze (
    seState en )
```

Parameters

<i>en</i>	see seState
-----------	-----------------------------

Return values

<i>None</i>	
-------------	--

2.78.1.9 seUSB_Suspend()

```
void seUSB_Suspend (
    void )
```

Return values

<i>None</i>	
-------------	--

2.78.1.10 seUSB_WakeUp()

```
void seUSB_WakeUp (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.79 USB_FIFO_Management

Functions

- uint16_t [seUSB_ReadFifo](#) (uint16_t EPNum, uint8_t *buf, uint16_t size)
Reads available data from FIFO.
- uint16_t [seUSB_WriteFifo](#) (uint16_t EPNum, const uint8_t *buf, uint16_t size)
Writes data to available space in FIFO.
- void [seUSB_DmaCopyFromFifo](#) (uint32_t daddress, uint32_t transfcoun)
DMA Copies data from selected FIFO. Channel 0 is used in this function.
- void [seUSB_DmaCopyToFifo](#) (uint32_t saddress, uint32_t transfcoun)
DMA Copies data to selected FIFO. Channel 1 is used in this function.

2.79.1 Function Documentation

2.79.1.1 seUSB_DmaCopyFromFifo()

```
void seUSB_DmaCopyFromFifo (
    uint32_t daddress,
    uint32_t transfcoun )
```

Parameters

<i>daddress</i>	Destination address.
<i>transfcoun</i>	Transfer count

Return values

<i>Returns</i>	none.
----------------	-------

2.79.1.2 seUSB_DmaCopyToFifo()

```
void seUSB_DmaCopyToFifo (
    uint32_t saddress,
    uint32_t transfcoun )
```

Parameters

<i>saddress</i>	Source address.
<i>transfcoun</i>	Transfer count

Return values

<i>None</i>	
-------------	--

2.79.1.3 seUSB_ReadFifo()

```
uint16_t seUSB_ReadFifo (
    uint16_t EPNum,
    uint8_t * buf,
    uint16_t size )
```

Parameters

<i>EPNum</i>	Endpoint number.
<i>buf</i>	Pointer to a destination byte array.
<i>size</i>	Destination array size.

Return values

<i>Returns</i>	a number of bytes read.
----------------	-------------------------

2.79.1.4 seUSB_WriteFifo()

```
uint16_t seUSB_WriteFifo (
    uint16_t EPNum,
    const uint8_t * buf,
    uint16_t size )
```

Parameters

<i>EPNum</i>	Endpoint number.
<i>buf</i>	Pointer to a source byte array.
<i>size</i>	Source array size.

Return values

<i>Returns</i>	a number of bytes written.
----------------	----------------------------

2.80 USB_Interrupt_Management

This group of functions initializes and manages USB module interrupts. For details refer to [seUSB_MainInterrupt](#), [seUSB_SieInterrupt](#), [seUSB_GpeInterrupt](#), [seUSB_Ep0Interrupt](#), [seUSB_EpmlInterrupt](#).

Macros

- **#define UsbMainIntEnable(x)** (USB->MAININTE |= (x))
- **#define UsbMainIntDisable(x)** (USB->MAININTE &= ~(x))
- **#define UsbGetMainInt()** (USB->MAININTF)
- **#define UsbGetMainIntEn()** (USB->MAININTE)
- **#define UsbClearMainInt(x)** (USB->MAININTF = (x))
- **#define UsbSIEIntEnable(x)** (USB->SIEINTE |= (x))
- **#define UsbSIEIntDisable(x)** (USB->SIEINTE &= ~(x))
- **#define UsbGetSIEInt()** (USB->SIEINTF)
- **#define UsbGetSIEIntEn()** (USB->SIEINTE)
- **#define UsbClearSIEInt(x)** (USB->SIEINTF = (x))
- **#define UsbGEPIntEnable(x)** (USB->GPEPINTE |= (x))
- **#define UsbGEPIntDisable(x)** (USB->GPEPINTE &= ~(x))
- **#define UsbGetGEPInt()** (USB->GPEPINTF)
- **#define UsbGetGEPIntEn()** (USB->GPEPINTE)
- **#define UsbClearGEPInt(x)** (USB->GPEPINTF = (x))
- **#define UsbEP0IntEnable(x)** (USB->EP0INTE |= (x))
- **#define UsbEP0IntDisable(x)** (USB->EP0INTE &= ~(x))
- **#define UsbGetEP0Int()** (USB->EP0INTF)
- **#define UsbGetEP0IntEn()** (USB->EP0INTE)
- **#define UsbClearEP0Int(x)** (USB->EP0INTF = (x))
- **#define UsbEPaIntEnable(x)** (USB->EPAINTE |= (x))
- **#define UsbEPaIntDisable(x)** (USB->EPAINTE &= ~(x))
- **#define UsbGetEPaInt()** (USB->EPAINTF)
- **#define UsbGetEPaIntEn()** (USB->EPAINTE)
- **#define UsbClearEPaInt(x)** (USB->EPAINTF = (x))
- **#define UsbEPbIntEnable(x)** (USB->EPBINTE |= (x))
- **#define UsbEPbIntDisable(x)** (USB->EPBINTE &= ~(x))
- **#define UsbGetEPbInt()** (USB->EPBINTF)
- **#define UsbGetEPbIntEn()** (USB->EPBINTE)
- **#define UsbClearEPbInt(x)** (USB->EPBINTF = (x))
- **#define UsbEPCIntEnable(x)** (USB->EPCINTE |= (x))
- **#define UsbEPCIntDisable(x)** (USB->EPCINTE &= ~(x))
- **#define UsbGetEPCInt()** (USB->EPCINTF)
- **#define UsbGetEPCIntEn()** (USB->EPCINTE)
- **#define UsbClearEPCInt(x)** (USB->EPCINTF = (x))

Functions

- void [seUSB_ClearAllIntFlags](#) (void)
This function clears all interrupt flags.
- void [seUSB_EnableAllInt](#) (void)
This function enables all interrupts.
- void [seUSB_DisableAllInt](#) (void)
This function disables all interrupts.

2.80.1 Function Documentation

2.80.1.1 seUSB_ClearAllIntFlags()

```
void seUSB_ClearAllIntFlags (
    void )
```

Note

- : 1. Main interrupt are cleared by writing 1 to corresponding interrupt sources in Sie interrupt Flag Register, Sie interrupt Flag Register, General-Purpose endpoint interrupt Flag Register, andEP0 interrupt Flag Register.
- : 2. Writing 1 to reserved bits has no effect.

Return values

None	
------	--

2.80.1.2 seUSB_DisableAllInt()

```
void seUSB_DisableAllInt (
    void )
```

Return values

None	
------	--

2.80.1.3 seUSB_EnableAllInt()

```
void seUSB_EnableAllInt (
    void )
```

Return values

None	
------	--

2.81 WDT2

WDT2 restarts the system if a problem occurs, such as when the program cannot be executed normally.

Modules

- [WDT2_Constants](#)
- [WDT2_Types](#)
- [WDT2_Functions](#)

2.82 WDT2_Constants

Typedefs

- typedef [seCLG_ClkSrc](#) **seWDT2_ClkSrc**

Enumerations

- enum [seWDT2_IOSC_ClkDiv](#) {
[seWDT2_IOSC_CLKDIV_8192](#) = 0,
[seWDT2_IOSC_CLKDIV_16384](#) = 1,
[seWDT2_IOSC_CLKDIV_32768](#) = 2,
[seWDT2_IOSC_CLKDIV_65536](#) = 3 }
- enum [seWDT2_OSC1_ClkDiv](#) { [seWDT2_OSC1_CLKDIV_128](#) = 0 }
- enum [seWDT2_OSC3_ClkDiv](#) {
[seWDT2_OSC3_CLKDIV_8192](#) = 0,
[seWDT2_OSC3_CLKDIV_16384](#) = 1,
[seWDT2_OSC3_CLKDIV_32768](#) = 2,
[seWDT2_OSC3_CLKDIV_65536](#) = 3 }
- enum [seWDT2_EXOSC_ClkDiv](#) { [seWDT2_EXOSC_CLKDIV_1](#) = 0 }
- enum [seWDT2_Mode](#) {
[seWDT2_NMIMode](#) = 1,
[seWDT2_ResetMode](#) = 0,
[seWDT2_ResetAndNMIMode](#) = 2 }

2.82.1 Enumeration Type Documentation

2.82.1.1 seWDT2_EXOSC_ClkDiv

```
enum seWDT2\_EXOSC\_ClkDiv
```

Enumerator

seWDT2_EXOSC_CLKDIV_1	EXOSC division ratio is 1/1.
---------------------------------------	------------------------------

2.82.1.2 seWDT2_IOSC_ClkDiv

```
enum seWDT2\_IOSC\_ClkDiv
```

Enumerator

seWDT2_IOSC_CLKDIV_8192	IOSC division ratio is 1/8192.
seWDT2_IOSC_CLKDIV_16384	IOSC division ratio is 1/16384.
seWDT2_IOSC_CLKDIV_32768	IOSC division ratio is 1/32768.
seWDT2_IOSC_CLKDIV_65536	IOSC division ratio is 1/65536.

2.82.1.3 seWDT2_Mode

```
enum seWDT2_Mode
```

Enumerator

seWDT2_NMIMode	Watchdog Timer operates in NMI mode or Reset mode. Select this operating mode to generate overflow NMI interrupt.
seWDT2_ResetMode	Watchdog Timer operates in Reset Mode. Select this operating mode to generate reset on timer overflow event.
seWDT2_ResetAndNMIMode	Watchdog Timer operates in Reset and NMI Mode. If the STATNMI bit is not cleared to 0 after an NMI interrupt has occurred due to a counter compare match, WDT2 issues a reset when the next compare match occurs.

2.82.1.4 seWDT2_OSC1_ClkDiv

```
enum seWDT2_OSC1_ClkDiv
```

Enumerator

seWDT2_OSC1_CLKDIV_128	OSC1 division ratio is 1/128.
------------------------	-------------------------------

2.82.1.5 seWDT2_OSC3_ClkDiv

```
enum seWDT2_OSC3_ClkDiv
```

Enumerator

seWDT2_OSC3_CLKDIV_8192	OSC3 division ratio is 1/8192.
seWDT2_OSC3_CLKDIV_16384	OSC3 division ratio is 1/16384.
seWDT2_OSC3_CLKDIV_32768	OSC3 division ratio is 1/32768.
seWDT2_OSC3_CLKDIV_65536	OSC3 division ratio is 1/65536.

2.83 WDT2_Types

Data Structures

- struct [seWDT2_InitTypeDef](#)
WDT Init structure definition.

2.84 WDT2_Functions

Functions

- void [seWDT2_InitStruct](#) ([seWDT2_InitTypeDef](#) *WDT_InitStruct)
Fills each [seWDT2_InitTypeDef](#) member with its default value.
- [seStatus seWDT2_Init](#) ([seWDT2_InitTypeDef](#) *WDT_InitStruct)
Initializes the WDT peripheral according to the specified parameters in the [seWDT2_InitStruct](#).
- [seStatus seWDT2_Start](#) (void)
Starts Watchdog Timer overflow counter.
- void [seWDT2_Stop](#) (void)
Stops Watchdog Timer overflow counter.
- void [seWDT2_ResetCounter](#) (void)
Resets Watchdog Timer overflow counter.
- void [seWDT2_ConfigureClock](#) ([seWDT2_ClkSrc](#) clock, uint16_t divider)
Configures Watchdog timer clock source and clock divider.
- uint32_t [seWDT2_GetClk](#) (void)
Discovers WDT clock from registers.
- [seStatus seWDT2_Set_tWDTSecs](#) (uint16_t tWDT)
Sets Watchdog Timer overflow counter cycle to provide tWDT in seconds. The function finds the selected clock dividers matching requested cycle value. No change is done to the clock dividers if desired timing is not possible.
- uint16_t [seWDT2_Get_tWDTSecs](#) (void)
Gets Watchdog Timer overflow counter cycle value.
- void [seWDT2_SetMode](#) ([seWDT2_Mode](#) mode)
Sets Watchdog Timer mode.
- void [seWDT2_SetCMP](#) (uint16_t value)
Sets Comparator value.
- uint16_t [seWDT2_GetCMP](#) (void)
Gets Comparator value.
- void [seWDT2_ChipReset](#) ([seWDT2_Mode](#) mode)
Resets Chip and Chip hardware. The reset actions depend on the selected WDT mode.
- void [NMI_Handler](#) (void)
Non-Maskable Interrupt Handler Routine.

2.84.1 Function Documentation

2.84.1.1 NMI_Handler()

```
void NMI_Handler (
    void )
```

Return values

None	
------	--

2.84.1.2 seWDT2_ChipReset()

```
void seWDT2_ChipReset (
    seWDT2_Mode mode )
```

Parameters

<i>mode</i>	could be a value of seWDT2_Mode
-------------	---

Return values

<i>None</i>	
-------------	--

2.84.1.3 seWDT2_ConfigureClock()

```
void seWDT2_ConfigureClock (
    seWDT2_ClkSrc clock,
    uint16_t divider )
```

Parameters

<i>clock</i>	This parameter can be a value of seWDT2_ClkSrc.
<i>divider</i>	This parameter can be a value of seWDT2_ClkDiv.

Return values

<i>None</i>	
-------------	--

2.84.1.4 seWDT2_Get_tWDTSecs()

```
uint16_t seWDT2_Get_tWDTSecs (
    void )
```

Return values

<i>vale</i>	10-bit counter cycle value in seconds.
-------------	--

2.84.1.5 seWDT2_GetClk()

```
uint32_t seWDT2_GetClk (
    void )
```

Return values

<i>Hz</i>	WDT clock in Hz.
-----------	------------------

2.84.1.6 seWDT2_GetCMP()

```
uint16_t seWDT2_GetCMP (
    void )
```

Return values

<i>value</i>	10-bit comparator value.
<i>None</i>	

2.84.1.7 seWDT2_Init()

```
seStatus seWDT2_Init (
    seWDT2_InitTypeDef * WDT_InitStruct )
```

Note

This function configures the module, and module's interrupts. It clears module's interrupts but does not enable interrupt from the module to CPU.

Parameters

<i>WDT_InitStruct</i>	pointer to a seWDT2_InitTypeDef structure that contains the configuration information for the specified WDT peripheral.
-----------------------	---

Return values

<i>Status</i>	see seStatus
---------------	------------------------------

2.84.1.8 seWDT2_InitStruct()

```
void seWDT2_InitStruct (
    seWDT2_InitTypeDef * WDT_InitStruct )
```

Parameters

<i>WDT_InitStruct</i>	pointer to an seWDT2_InitTypeDef structure which will be initialized.
-----------------------	---

Return values

<i>None</i>	
-------------	--

2.84.1.9 seWDT2_ResetCounter()

```
void seWDT2_ResetCounter (
```



```
void )
```

Return values

<i>None</i>	
-------------	--

2.84.1.10 seWDT2_Set_tWDTSecs()

```
seStatus seWDT2_Set_tWDTSecs (  
    uint16_t tWDT )
```

Parameters

<i>tWDT</i>	This parameter can be a value less or equal to 4.
-------------	---

Return values

<i>Status</i>	see seStatus
---------------	------------------------------

2.84.1.11 seWDT2_SetCMP()

```
void seWDT2_SetCMP (  
    uint16_t value )
```

Parameters

<i>value</i>	This parameter defines 10-bit comparator value.
--------------	---

Return values

<i>None</i>	
-------------	--

2.84.1.12 seWDT2_SetMode()

```
void seWDT2_SetMode (  
    seWDT2_Mode mode )
```

Parameters

<i>mode</i>	This parameter defines Watchdog timer operating mode and can be a value of seWDT2_Mode .
-------------	--

Return values

<i>None</i>	
-------------	--

2.84.1.13 seWDT2_Start()

```
seStatus seWDT2_Start (  
    void )
```

Return values

<i>Status</i>	see seStatus
---------------	------------------------------

2.84.1.14 seWDT2_Stop()

```
void seWDT2_Stop (  
    void )
```

Return values

<i>None</i>	
-------------	--

2.85 USB

The USB 2.0 FS Device Controller is a USB target device controller that supports FS mode based on the USB 2.0 standard.

Modules

- [USB_Constants_and_Macros](#)
- [USB_Types](#)
- [USB_Functions](#)

Chapter 3

Data Structure Documentation

3.1 seCLG_ClkDiv Union Reference

Data Fields

- [seCLG_IOSC_ClkDiv](#) **IOSC_ClkDiv**
- [seCLG_OSC1_ClkDiv](#) **OSC1_ClkDiv**
- [seCLG_OSC3_ClkDiv](#) **OSC3_ClkDiv**
- [seCLG_EXOSC_ClkDiv](#) **EXOSC_ClkDiv**
- [seCLG_CLG_ClkDiv](#) **CLG_ClkDiv**

3.2 seCLG_InitTypeDef Struct Reference

CLG Init structure definition.

```
#include <se_clg.h>
```

Data Fields

- [seCLG_ClkSrc](#) **SysClkSrc**
Specifies the System clock source.
- [seCLG_ClkDiv](#) **SysClkDiv**
Specifies the System clock source divide.
- [seCLG_ClkSrc](#) **WkUpSysClkSrc**
Specifies the System clock source on Wakeup from Sleep.
- [seCLG_ClkDiv](#) **WkUpClkDiv**
Specifies the System clock source divide on Wakeup from Sleep.
- [seState](#) **SysClkSwitchOnWkUpEn**
Specifies if System clock switch on wakeup should be enabled.
- [seState](#) **OperationInSlpEn**
Specifies the Operation in Sleep mode should be enabled.

3.3 seDMAC_CtrlData Union Reference

DMAC Control Data definition.

```
#include <se_dmac.h>
```

Data Fields

- uint32_t [ctrldata](#)
Configuration details for the transfer.
 - struct {
 - uint32_t [cycle_ctrl](#): 3
Transfer Mode.
 - uint32_t [RES_3](#): 1
Must write 0 to this bit.
 - uint32_t [n_minus_1](#): 10
Transfer Total.
 - uint32_t [R_power](#): 4
Transfer Size Select.
 - uint32_t [RES_23_18](#): 6
Must write 0 to this field.
 - uint32_t [src_size](#): 2
Source Data Size Select.
 - uint32_t [src_inc](#): 2
Source Address Increment Mode.
 - uint32_t [dest_size](#): 2
Destination Data Size Select.
 - uint32_t [dest_inc](#): 2
Destination Address Increment Mode.
- } [ctrldata_b](#)
- BitSize.*

3.4 seDMAC_DataStruct Struct Reference

DMAC Descriptors structure definition.

```
#include <se_dmac.h>
```

Data Fields

- uint32_t [transfer_source_end_pointer](#)
< DMAC Descriptors structure definition
- uint32_t [transfer_destination_end_pointer](#)
The last destination address of the transfer.
- uint32_t [control_data](#)
Configuration details for the transfer [seDMAC_CtrlData](#).
- uint32_t [request_num](#)
Firmware can use this byte for number of requests tracking.

3.4.1 Field Documentation

3.4.1.1 transfer_source_end_pointer

```
uint32_t seDMAC_DataStruct::transfer_source_end_pointer
```

The address of the last source data in the transfer

3.5 sel2C_ChannelDef Struct Reference

I2C Channel definition.

```
#include <se_i2c.h>
```

Data Fields

- [I2C_0_Type * I2Cx](#)
Pointer to I2C peripheral channel.
- [seUPMUX_Channel_Sel channelNo](#)
I2C channel number.
- [sePPORT_PeriphPortDef SCL](#)
SCL pin port definition.
- [sePPORT_PeriphPortDef SDA](#)
SDA pin port definition.

3.6 sel2C_ClkDiv Union Reference

Data Fields

- [sel2C_EXOSC_ClkDiv exosc](#)
- [sel2C_IOSC_ClkDiv iosc](#)
- [sel2C_OSC3_ClkDiv osc3](#)
- [sel2C_OSC1_ClkDiv osc1](#)

3.7 sel2C_InitTypeDef Struct Reference

I2C Init structure definition.

```
#include <se_i2c.h>
```

Data Fields

- [sel2C_mode I2C_mode](#)
Specifies Master/Slave mode, a value of [I2C_mode](#).
- [sel2C_ClkSrc ClkSrc](#)
Specifies the i2c clock source selection.
- [uint16_t ClkDivider](#)

Specifies the prescaler value used to divide the i2c clock.

- uint16_t [BRT](#)
- [seI2C_AddrMode](#) AddrMode

Specifies if 7-bit or 10-bit address, a [seI2C_AddrMode](#) value.

- uint16_t [SlaveAddr](#)
- [seState](#) RespGenCalls

Specifies if slave is to response to master's general calls.

3.7.1 Field Documentation

3.7.1.1 BRT

```
uint16_t seI2C_InitTypeDef::BRT
```

Specifies the baud rate($\text{bps} = \text{CLK_I2C} / (\text{BRT} + 3) \times 2$).

- 8.00MHz / 20 = 400000 bps. Note: The I2C bus transfer rate is limited to 100 kbit/s in standard mode or 400 kbit/s in fast mode. Do not set a transfer rate exceeding the limit.

3.7.1.2 SlaveAddr

```
uint16_t seI2C_InitTypeDef::SlaveAddr
```

Specifies the slave address. This parameter can be a 7-bit or 10-bit address.

3.8 seLCD32B_ClkDiv Union Reference

Data Fields

- [seLCD32B_EXOSC_ClkDiv](#) **exosc**
- [seLCD32B_IOSC_ClkDiv](#) **iosc**
- [seLCD32B_OSC3_ClkDiv](#) **osc3**
- [seLCD32B_OSC1_ClkDiv](#) **osc1**

3.9 seLCD32B_InitTypeDef Struct Reference

LCD32B Init structure definition.

```
#include <se_lcd32b.h>
```

Data Fields

- [seLCD32B_ClkSrc](#) ClkSrc
Specifies the System clock source.
- uint16_t [ClkDivider](#)
Configures the LCD32B clock source divider.
- [seState](#) CtlLcdIoDischargeEn

- uint16_t [Tim1FrameCount](#)
Configures the LCD32B Frame Count used in frame frequency calculation.
- uint16_t [Tim1Duty](#)
Configures the LCD32B Duty used in frame frequency calculation.
- [seLCD32B_BoostClk](#) [Tim2BoostClk](#)
Configures the LCD32B Booster clock. It could be a value [seLCD32B_BoostClk](#).
- uint16_t [Tim2Nline](#)
Configures the LCD32B SEG n line reverse drive selection.
- [seState](#) [PwrVoltageRegulatorEn](#)
*Configures voltage regulator enable/disable. */.*
- [seState](#) [PwrHeavyLoadProtectionModeEn](#)
*Configures heavy load protection mode enable/disable. */.*
- [seState](#) [PwrBoosterEn](#)
Configures voltage booster enable/disable.
- [seState](#) [PwrExternalVcSel](#)
Configures the LCD drive voltage supply mode.
- [seLCD32B_PwrBias](#) [PwrBiasSel](#)
Configures the LCD drive bias. It could be a value [seLCD32B_PwrBias](#).
- [seLCD32B_Area](#) [DspSelectArea](#)
Select one of the display areas.
- uint16_t [DspSelComPinDir](#)
- uint16_t [DspSelSegPinDir](#)

3.9.1 Field Documentation

3.9.1.1 CtlLcdIoDischargeEn

[seState](#) [seLCD32B_InitTypeDef::CtlLcdIoDischargeEn](#)

Specifies if IO discharge should be enabled. When the panel is connected, must be set to seENABLE even if display is turned off.

3.9.1.2 DspSelComPinDir

uint16_t [seLCD32B_InitTypeDef::DspSelComPinDir](#)

Select COM pins assignment direction. If 1, memory bits are assigned to common pins in ascending order. If 0, memory bits are assigned to common pins in descending order.

3.9.1.3 DspSelSegPinDir

uint16_t [seLCD32B_InitTypeDef::DspSelSegPinDir](#)

Select SEG ins assignment direction. If 1, memory addresses are assigned to segment pins in ascending order. If 0, memory addresses are assigned to segment pins in descending order.

3.10 sePPORT_Group Struct Reference

Data Fields

- `__IO uint16_t DAT`
< PPORT Structure
- `__IO uint16_t IOEN`
PPort Enable Register.
- `__IO uint16_t RCTL`
PPort Pull-up/down Control Register.
- `__IO uint16_t INTF`
PPort Interrupt Flag Register.
- `__IO uint16_t INTCTL`
PPort Interrupt Control Register.
- `__IO uint16_t CHATEN`
PPort Chattering Filter Enable Register.
- `__IO uint16_t MODSEL`
PPort Mode Select Register.
- `__IO uint16_t FNCSEL`
PPort Function Select Register.

3.10.1 Field Documentation

3.10.1.1 DAT

```
__IO uint16_t sePPORT_Group::DAT
```

PPort Data Register

3.11 sePPORT_InitTypeDef Struct Reference

PPORTInit structure definition.

```
#include <se_pport.h>
```

Data Fields

- `sePPORT_ClkSrc ClkSrc`
Specifies the timer clock source selection.
- `uint16_t ClkDivider`
Specifies the prescaler value used to divide the PPORT clock.
- `seState StopInSleep`
Specifies if operation stops or continues in sleep mode.

3.12 sePPORT_PeriphPortDef Struct Reference

Data Fields

- [sePPORT_Id portID](#)
Port identifier.
- [sePPORT_PeriphPortInit portinit](#)
Peripheral port initialization type.
- [sePPORT_AltFunc AltFunc](#)
Specifies the alternate function number.

3.13 seQSPI_ChannelDef Struct Reference

QSPI Channel definition.

```
#include <se_qspi.h>
```

Data Fields

- [QSPI_0_Type * QSP1x](#)
Pointer to QSPI peripheral channel.
- [T16_0_Type * T16x](#)
Pointer to T16 timer used for QSPI peripheral channel.
- [sePPORT_PeriphPortDef QSPID0](#)
QSPISD0 pin port definition.
- [sePPORT_PeriphPortDef QSPID1](#)
QSPISD1 pin port definition.
- [sePPORT_PeriphPortDef QSPID2](#)
QSPISD2 pin port definition.
- [sePPORT_PeriphPortDef QSPID3](#)
QSPISD3 pin port definition.
- [sePPORT_PeriphPortDef QSPICLK](#)
QSPICLK pin port definition.
- [sePPORT_PeriphPortDef QSPISS](#)
QSPISS# pin port definition.

3.14 seQSPI_InitTypeDef Struct Reference

QSPI Init structure definition.

```
#include <se_qspi.h>
```

Data Fields

- `seQSPI_Clocks CHDL`
Set a number of clocks to drive the serial data lines.
- `seQSPI_Clocks CHLN`
Set a number of clocks for data transfer.
- `seState PUEN`
Set input pin pull-up/down.
- `seState NOCLKDIV`
Set master mode operating clock.
- `seQSPI_Format LSBFST`
Set MSB first/LSB first.
- `seQSPI_Phase CPHA`
Set clock phase.
- `seQSPI_Polarity CPOL`
Set clock polarity.
- `seQSPI_OperMode MST`
Set master/slave mode.
- `seQSPI_TransferMode TMOD`
Set transfer mode (single/dual/quad).
- `seQSPI_Clocks TCSH`
Set slave select signal negation period.
- `uint16_t RMADR`
Set an offset, remapping start address high reg.
- `seQSPI_Clocks DUMDL`
Set dummy cycle drive length.
- `seQSPI_Clocks DUMLN`
Set dummy cycle length.
- `seQSPI_TransferMode DATTMOD`
Set data cycle transfer mode.
- `seQSPI_TransferMode DUMTMOD`
Set dummy cycle transfer mode.
- `seQSPI_TransferMode ADRTMOD`
Set address cycle transfer mode.
- `seQSPI_AddrMode ADRCYC`
Set 24- or 32-bit address cycle.
- `uint8_t XIPACT`
Set XIP activation mode byte.
- `uint8_t XIPEXT`
Set XIP termination mode byte.
- `seQSPI_Interrupt INTE`

3.15 seREMC2_ChannelDef Struct Reference

Data Fields

- REMC2_Type * [REMC2x](#)
Pointer to REMC2 peripheral channel.
- [sePPORT_PeriphPortDef REMO](#)
REMO pin port definition.
- [sePPORT_PeriphPortDef CLPLS](#)
CLPLS pin port definition.

3.16 seREMC2_InitTypeDef Struct Reference

REMCInit structure definition.

```
#include <se_remc2.h>
```

Data Fields

- [seREMC2_ClkSrc ClkSrc](#)
Specifies the timer clock source selection.
- uint16_t [ClkDivider](#)
Specifies the prescaler value used to divide the REMC clock.
- [seState StopInSleep](#)
Specifies if operation stops or continues in sleep mode.
- uint16_t [carr](#)
Carrier Waveform Register.

3.17 seRFC_ChannelDef Struct Reference

RFC Channel definition.

```
#include <se_rfc.h>
```

Data Fields

- RFC_0_Type * [RFCx](#)
Pointer to RFC peripheral channel.
- [sePPORT_PeriphPortDef SENA](#)
SENA pin port definition.
- [sePPORT_PeriphPortDef SENB](#)
SENB pin port definition.
- [sePPORT_PeriphPortDef REF](#)
REF pin port definition.
- [sePPORT_PeriphPortDef RFIN](#)
RFIN pin port definition.
- [sePPORT_PeriphPortDef RFCLKO](#)
RFCLKO pin port definition.

3.18 seRFC_InitTypeDef Struct Reference

RFCInit structure definition.

```
#include <se_rfc.h>
```

Data Fields

- [seRFC_ClkSrc ClkSrc](#)
Specifies the timer clock source selection.
- [uint16_t ClkDivider](#)
Specifies the prescaler value used to divide the RFC clock.
- [seState StopInSleep](#)
Specifies if operation stops or continues in sleep mode.
- [seState EnableExternClockInput](#)
Enables external clock inputmode.
- [seRFC_OscMode OscMode](#)
Configures an oscillation mode for resistive measurements.

3.19 seRTCA_InitTypeDef Struct Reference

RTCA Init structure definition.

```
#include <se_rtca.h>
```

Data Fields

- [seState ClkSupldInDebugMode](#)
Specifies if System clock supplied in CPU Sleep Mode.
- [seRTCA_Hours12_24 H12_24Format](#)
Specifies the RTC 12/24 Hour Format.

Note

3.20 seSNDA_ChannelDef Struct Reference

Data Fields

- [SNDA_Type * SNDAX](#)
Pointer to SNDA peripheral channel.
- [sePPORT_PeriphPortDef BZOUT](#)
BZOUT pin port definition.
- [sePPORT_PeriphPortDef NBZOUT](#)
BZOUT pin port definition

3.21 seSNDA_InitTypeDef Struct Reference

SNDA Init structure definition.

```
#include <se_snda.h>
```

Data Fields

- [seSNDA_ClkSrc ClkSrc](#)
Specifies the timer clock source selection.
- [uint16_t ClkDivider](#)
Specifies the prescaler value used to divide the SNDA clock.
- [seSNDA_DriveMode DriveMode](#)
Specifies Pin Drive Mode.
- [seDMAC_CHANNEL DMACHannel](#)
Specifies DMA channel.
- [seSNDA_InterruptSrc EnableInt](#)
Specifies Interrupts to be enabled.

3.22 seSPIA_ChannelDef Struct Reference

SPIA Channel definition.

```
#include <se_spia.h>
```

Data Fields

- [SPIA_0_Type * SPIx](#)
Pointer to SPIA peripheral channel.
- [T16_0_Type * T16x](#)
Pointer to T16 timer used for SPIA peripheral channel.
- [seUPMUX_Channel_Sel channelNo](#)
SPIA channel number.
- [sePPORT_PeriphPortDef SPISS](#)
SPISS# pin port definition.
- [sePPORT_PeriphPortDef SDI](#)
SDI pin port definition.
- [sePPORT_PeriphPortDef SDO](#)
SDO pin port definition.
- [sePPORT_PeriphPortDef SPICLK](#)
SPICLK pin port definition.

3.23 seSPIA_InitTypeDef Struct Reference

SPIA Init structure definition.

```
#include <se_spia.h>
```

Data Fields

- `seSPIA_DataTransferLength` [CHLN](#)
Specifies the data transfer bit length.
- `seState` [PUEN](#)
Enables/Disables input pin pull-up/down.
- `seState` [NOCLKDIV](#)
Select master mode operating clock.
- `seSPIA_Format` [LSBFST](#)
Select MSB first/LSB first.
- `seSPIA_Phase` [CPHA](#)
Select clock phase.
- `seSPIA_Polarity` [CPOL](#)
Select clock polarity.
- `seSPIA_OperMode` [MST](#)
Select master/slave mode.

3.24 seSVD2_ChannelDef Struct Reference

SVD Channel definition.

```
#include <se_svd2.h>
```

Data Fields

- `SVD2_0_Type` * [SVDx](#)
Pointer to SVD peripheral channel.
- `sePPORT_PeriphPortDef` [EXSVD](#)
EXSVD pin port definition.

3.25 seSVD2_ClkDiv Union Reference

Data Fields

- `seSVD2_EXOSC_ClkDiv` [exosc](#)
- `seSVD2_IOSC_ClkDiv` [iosc](#)
- `seSVD2_OSC3_ClkDiv` [osc3](#)
- `seSVD2_OSC1_ClkDiv` [osc1](#)

3.26 seSVD2_InitTypeDef Struct Reference

SVD Init structure definition.

```
#include <se_svd2.h>
```


Data Fields

- [seSVD2_ClkSrc ClkSrc](#)
Specifies the i2c clock source selection.
- [uint16_t ClkDivider](#)
Specifies the prescaler value used to divide the i2c clock.
- [seSVD2_VoltageSource VDSEL](#)
Voltage source select.
- [seSVD2_Interrupt SVDIE](#)
SVD interrupt enable.
- [seState ResetEnable](#)
SVD reset enable.
- [seSVD2_IntermittentMode IntermittentMode](#)
SVD intermittent mode.
- [uint32_t CompareVoltage](#)
SVD compare voltage.
- [seSVD2_SamplingResCnt SamplingResCnt](#)
SVD sampling result count.
- [seSVD2_DetectMode DetectMode](#)
SVD detect mode.

3.27 seT16_InitTypeDef Struct Reference

T16 Init structure definition.

```
#include <se_t16.h>
```

Data Fields

- [seT16_ClkSrc ClkSrc](#)
Specifies the timer clock source selection.
- [uint16_t ClkDivider](#)
Specifies the prescaler value used to divide the T16 clock.
- [seT16_CounterMode CounterMode](#)
Specifies the counter mode.
- [uint16_t Period](#)
Specifies the period value to be loaded into the counter.

Note

This structure is used with all T16 but not T16B.

3.28 seT16B_CCCTL Struct Reference

Data Fields

- [seT16B_SCS](#) **SCS**
- [seT16B_CBUFMD](#) **CBUFMD**
Compare buffer mode.
- [seT16B_CAPIS](#) **CAPIS**
- [seT16B_CAPTRG](#) **CAPTRG**
- [seT16B_TOUTMT](#) **TOUTMT**
- [seT16B_TOUTO](#) **TOUTO**
- [seT16B_TOUTMD](#) **TOUTMD**
- [seT16B_TOUTINV](#) **TOUTINV**
- [seT16B_CCMD](#) **CCMD**
Specifies comparator or capture mode.
- `uint16_t` **CCR**

3.29 seT16B_CCRegsDef Struct Reference

T16B Capture/Compare registers definition.

```
#include <se_t16b.h>
```

Data Fields

- union {
 __IO uint16_t [CCCTL](#)
 struct {
 __IO uint16_t [CCMD](#): 1
 __IO uint16_t [TOUTINV](#): 1
 __IO uint16_t [TOUTMD](#): 3
 __IO uint16_t [TOUTO](#): 1
 __IO uint16_t [TOUTMT](#): 1
 uint16_t [__pad0__](#): 1
 __IO uint16_t [CAPTRG](#): 2
 __IO uint16_t [CAPIS](#): 2
 __IO uint16_t [CBUFMD](#): 3
 __IO uint16_t [SCS](#): 1
 } [CCCTL_b](#)
};
- union {
 __IO uint16_t [CCR](#)
 struct {
 __IO uint16_t [CC](#): 16
 } [CCR_b](#)
};

- union {
 - __IO uint16_t **CCDMAEN**
 - struct {
 - __IO uint16_t **CC0DMAEN**: 4
 - } **CCDMAEN_b**
- };
- __IO uint16_t **RESERVED4**

3.29.1 Field Documentation

3.29.1.1 CAPIS

`__IO uint16_t seT16B_CCRegsDef::CAPIS`

Capture input signal select

3.29.1.2 CAPTRG

`__IO uint16_t seT16B_CCRegsDef::CAPTRG`

Capture trigger select

3.29.1.3 CBUFMD

`__IO uint16_t seT16B_CCRegsDef::CBUFMD`

Compare buffer mode select

3.29.1.4 CC

`__IO uint16_t seT16B_CCRegsDef::CC`

Compare/Capture Data Register

3.29.1.5 CC0DMAEN

`__IO uint16_t seT16B_CCRegsDef::CC0DMAEN`

Compare/Capture request enable

3.29.1.6 CCCTL

`__IO uint16_t seT16B_CCRegsDef::CCCTL`

Compare/Capture Control Register

3.29.1.7 CCCTL_b

`struct { ... } seT16B_CCRegsDef::CCCTL_b`

BitSize

3.29.1.8 CCDMAEN

```
__IO uint16_t seT16B_CCRegsDef::CCDMAEN
```

CC DMA Enable Register

3.29.1.9 CCDMAEN_b

```
struct { ... } seT16B_CCRegsDef::CCDMAEN_b
```

BitSize

3.29.1.10 CCMD

```
__IO uint16_t seT16B_CCRegsDef::CCMD
```

T16B CCA register mode select

3.29.1.11 CCR

```
__IO uint16_t seT16B_CCRegsDef::CCR
```

Compare/Capture Data Register

3.29.1.12 CCR_b

```
struct { ... } seT16B_CCRegsDef::CCR_b
```

BitSize

3.29.1.13 SCS

```
__IO uint16_t seT16B_CCRegsDef::SCS
```

SCS register mode select

3.29.1.14 TOUTINV

```
__IO uint16_t seT16B_CCRegsDef::TOUTINV
```

Tout invert

3.29.1.15 TOUTMD

```
__IO uint16_t seT16B_CCRegsDef::TOUTMD
```

Tout Output mode select

3.29.1.16 TOUTMT

```
__IO uint16_t seT16B_CCRegsDef::TOUTMT
```

Tout Motor mode select

3.29.1.17 TOUTO

```
__IO uint16_t seT16B_CCRegsDef::TOUTO
```

Tout Output select

3.30 seT16B_ChannelDef Struct Reference

T16B Channel definition.

```
#include <se_t16b.h>
```

Data Fields

- [T16B_0_Type * T16Bx](#)
Pointer to T16B peripheral channel.
- [seUPMUX_Channel_Sel](#) channelNo
T16B channel number.
- [sePPORT_PeriphPortDef](#) TOUTCAP [6]
TOUT/CAP pin definitions array.

3.31 seT16B_InitTypeDef Struct Reference

T16B Init structure definition.

```
#include <se_t16b.h>
```

Data Fields

- [seT16B_ClkSrc](#) ClkSrc
Specifies the timer clock source selection.
- [uint16_t](#) ClkDivider
Specifies the prescaler value used to divide the T16B clock.
- [seT16B_CCCTL](#) CTL [6]
- [seT16B_ONEST](#) ONEST
Specifies the counter mode.
- [seT16B_CNTMD](#) CNTMD
- [uint16_t](#) MaxCounter
- [uint16_t](#) Period
Specifies the period value to be loaded into the counter.

Note

This structure is used with all T16B.

3.32 seUART2_ChannelDef Struct Reference

UART Channel definition.

```
#include <se_uart2.h>
```

Data Fields

- [UART2_0_Type * UARTx](#)
Pointer to I2C peripheral channel.
- [seUPMUX_Channel_Sel channelNo](#)
I2C channel number.
- [sePPORT_PeriphPortDef USIN](#)
USIN pin port definition.
- [sePPORT_PeriphPortDef USOUT](#)
USOUT pin port definition.

3.33 seUART2_ClkDiv Union Reference

UART Init structure definition.

```
#include <se_uart2.h>
```

Data Fields

- [seUART2_EXOSC_ClkDiv exosc](#)
- [seUART2_IOSC_ClkDiv iosc](#)
- [seUART2_OSC3_ClkDiv osc3](#)
- [seUART2_OSC1_ClkDiv osc1](#)

Note

This structure is used with all UART but not UARTB.

3.34 seUART2_InitTypeDef Struct Reference

UART Init structure definition.

```
#include <se_uart2.h>
```

Data Fields

- [seUART2_ClkSrc ClkSrc](#)
Specifies the uart clock source selection.
- [uint16_t ClkDivider](#)
Specifies the prescaler value used to divide the UART clock.
- [seUART2_Mode Mode](#)
Specifies the mode of operation [seUART2_Mode](#).

Note

3.35 seUART2_Mode Union Reference

UART Mode structure definition.

```
#include <se_uart2.h>
```

Data Fields

- uint16_t [reg](#)
UART Mode.
 - struct {
 - uint16_t [stpb](#): 1
UART select 1/2-bit stop bit length.
 - uint16_t [prmd](#): 1
UART select even/odd parity function.
 - uint16_t [pren](#): 1
UART enable parity function.
 - uint16_t [chln](#): 1
UART set 7/8-bit data rate.
 - uint16_t [irmd](#): 1
UART enable Irda interface.
 - uint16_t [outmd](#): 1
UART enable USOUTn pn open-drain output.
 - uint16_t [puen](#): 1
UART enable USINn pin pull-up.
 - uint16_t [reserved_7](#): 1
Reserved bit.
 - uint16_t [invirtx](#): 1
Invert transmit IrDA signal.
 - uint16_t [invirrx](#): 1
Invert receive IrDA signal.
 - uint16_t [reserved_15_10](#): 6
Reserved bits.
- } [reg_b](#)
- BitSize.*

Note

3.36 seUSB_InitTypeDef Struct Reference

USB Init structure definition.

```
#include <se_usb.h>
```

Data Fields

- [seUSB_ClkSrc ClkSrc](#)
Specifies the usb clock source selection.

Note

3.37 seWDT2_InitTypeDef Struct Reference

WDT Init structure definition.

```
#include <se_wdt2.h>
```

Data Fields

- [seWDT2_ClkSrc ClkSrc](#)
Specifies the timer clock source selection.
- [uint16_t ClkDivider](#)
- [uint16_t CMP](#)
Specifies Comparator cycle.
- [seWDT2_Mode mode](#)
Choose between NMI or Reset mode.

Note

This structure is used with all WDT but not T16B.

3.37.1 Field Documentation

3.37.1.1 ClkDivider

```
uint16_t seWDT2_InitTypeDef::ClkDivider
```

Specifies the prescaler value used to divide the WDT clock. The clock frequency should be set to around 256 Hz.

3.38 swCounter Struct Reference

RTCA Stopwatch counter structure definition.

```
#include <se_rtca.h>
```

Data Fields

- [uint16_t swInt1HzCount](#)
Counts of 1Hz interrupts(0-65535).
- [uint8_t swChar10HzDigit](#)

- 10Hz-digit stopwatch count(0-9).*
 - uint8_t [swChar100HzDigit](#)
100Hz-digit stopwatch count(0-9).

Note

Index

BRT
 sel2C_InitTypeDef, [272](#)

CAPIS
 seT16B_CCRegsDef, [283](#)

CAPTRG
 seT16B_CCRegsDef, [283](#)

CBUFMD
 seT16B_CCRegsDef, [283](#)

CC0DMAEN
 seT16B_CCRegsDef, [283](#)

CCCTL_b
 seT16B_CCRegsDef, [283](#)

CCCTL
 seT16B_CCRegsDef, [283](#)

CCDMAEN_b
 seT16B_CCRegsDef, [284](#)

CCDMAEN
 seT16B_CCRegsDef, [283](#)

CCMD
 seT16B_CCRegsDef, [284](#)

CCR_b
 seT16B_CCRegsDef, [284](#)

CCR
 seT16B_CCRegsDef, [284](#)

CLG_Constants, [12](#)
 seCLG_CLG_ClkDiv, [13](#)
 seCLG_ClkSrc, [13](#)
 seCLG_EXOSC_ClkDiv, [13](#)
 seCLG_IOSC_ClkDiv, [14](#)
 seCLG_IOSC_IoscFq, [14](#)
 seCLG_IntFlag, [14](#)
 seCLG_Interrupt, [13](#)
 seCLG_OSC1_ClkDiv, [15](#)
 seCLG_OSC3_ClkDiv, [15](#)

CLG_Functions, [17](#)
 CLG_IRQHandler, [18](#)
 ConfigurePortsForOSC3, [18](#)
 seCLG_ClearIntFlag, [18](#)
 seCLG_DisableInt, [18](#)
 seCLG_EnableInt, [19](#)
 seCLG_GetIntFlag, [19](#)
 seCLG_GetIoscFreqSel, [19](#)
 seCLG_GetOperInSlp, [19](#)
 seCLG_GetSysClk, [20](#)
 seCLG_GetSysClkDiv, [20](#)
 seCLG_GetSysClkSrc, [20](#)
 seCLG_Init, [20](#)
 seCLG_RunAutoTrimming, [21](#)
 seCLG_SetIoscFreqSel, [21](#)
 seCLG_SetOperInSlp, [21](#)
 seCLG_SetStopDetection, [22](#)
 seCLG_SetWkUpSysClk, [22](#)
 seCLG_Start, [23](#)
 seCLG_Stop, [23](#)
 seCLG_SwitchSysClkSrc, [23](#)

CLG_IRQHandler
 CLG_Functions, [18](#)

CLG_Types, [16](#)

CLG, [11](#)

CC
 seT16B_CCRegsDef, [283](#)

ClkDivider
 seWDT2_InitTypeDef, [288](#)

Common, [5](#)

Common_Constants, [6](#)
 seInterruptStatus, [6](#)
 seState, [6](#)
 seStatus, [6](#)
 seTimeoutMs, [7](#)
 seWriteProtect, [7](#)

Common_Functions, [9](#)
 seAssert, [9](#)
 seClamp16, [9](#)
 seClamp32, [10](#)
 seProtectSys, [10](#)

Common_Macros, [8](#)
 WAIT, [8](#)

ConfigurePortsForI2C
 I2C_Functions, [45](#)

ConfigurePortsForLcd32B
 LCD32B_Functions, [66](#)

ConfigurePortsForOSC3
 CLG_Functions, [18](#)

ConfigurePortsForQSPI
 QSPI_Functions, [101](#)

ConfigurePortsForRFC
 RFC_Functions, [127](#)

ConfigurePortsForSPI
 SPIA_Functions, [165](#)

ConfigurePortsForSVD2

- SVD_Functions, 180
- ConfigurePortsForT16B
 - T16B_Functions, 207
- ConfigurePortsForUart
 - UART_Functions, 223
- CtlLcdIoDischargeEn
 - seLCD32B_InitTypeDef, 273
- DAT
 - sePPORT_Group, 274
- DMAC_Constants, 25
 - seDMAC_CHANNEL, 25
 - seDMAC_Inc, 26
 - seDMAC_InterruptSrc, 26
 - seDMAC_Mode, 26
 - seDMAC_Size, 27
- DMAC_Functions, 29
 - DMAC_IRQHandler, 30
 - seDMAC_AlternateDisable, 30
 - seDMAC_AlternateEnable, 30
 - seDMAC_ClearIntFlag, 31
 - seDMAC_ConfigMemToPeriph, 31
 - seDMAC_ConfigPeriphToMem, 31
 - seDMAC_Disable, 32
 - seDMAC_DisableInt, 32
 - seDMAC_DisableRequestMask, 33
 - seDMAC_Enable, 33
 - seDMAC_EnableInt, 33
 - seDMAC_EnableRequestMask, 33
 - seDMAC_GetAltDataStrucPtr, 34
 - seDMAC_GetDataStrucPtr, 34
 - seDMAC_GetIntFlag, 34
 - seDMAC_GetMode, 34
 - seDMAC_GetNMinus1, 35
 - seDMAC_Init, 35
 - seDMAC_NonBlockTransfMemToPeriph, 35
 - seDMAC_NonBlockTransfPeriphToMem, 36
 - seDMAC_PriorityDecrease, 36
 - seDMAC_PriorityIncrease, 37
 - seDMAC_SetChannel, 37
 - seDMAC_SetDataStrucPtr, 37
 - seDMAC_Start, 38
- DMAC_IRQHandler
 - DMAC_Functions, 30
- DMAC_Types, 28
- DMAC, 24
- DspSelCompPinDir
 - seLCD32B_InitTypeDef, 273
- DspSelSegPinDir
 - seLCD32B_InitTypeDef, 273
- I2C_0_IRQHandler
 - I2C_Functions, 46
- I2C_1_IRQHandler
 - I2C_Functions, 46
- I2C_Constants, 40
 - sel2C_AddrMode, 41
 - sel2C_EXOSC_ClkDiv, 41
 - sel2C_IOSC_ClkDiv, 42
 - sel2C_IntFlag, 41
 - sel2C_Interrupt, 41
 - sel2C_OSC1_ClkDiv, 42
 - sel2C_OSC3_ClkDiv, 43
 - sel2C_mode, 42
- I2C_Functions, 45
 - ConfigurePortsForI2C, 45
 - I2C_0_IRQHandler, 46
 - I2C_1_IRQHandler, 46
 - sel2C_ClearIntFlag, 46
 - sel2C_Disable, 46
 - sel2C_DisableInt, 47
 - sel2C_Enable, 47
 - sel2C_EnableInt, 47
 - sel2C_GetIntFlag, 48
 - sel2C_Init, 48
 - sel2C_InitStructForMaster, 48
 - sel2C_InitStructForSlave, 49
 - sel2C_MstReceiveData, 49
 - sel2C_MstSendData, 49
 - sel2C_Reset, 50
 - sel2C_SlvReceiveData, 50
 - sel2C_SlvSendData, 51
- I2C_Types, 44
- I2C, 39
- LCD32B_Clock, 54
 - seLCD32B_BoostClk, 54
 - seLCD32B_EXOSC_ClkDiv, 55
 - seLCD32B_IOSC_ClkDiv, 55
 - seLCD32B_OSC1_ClkDiv, 55
 - seLCD32B_OSC3_ClkDiv, 55
- LCD32B_Constants, 53
- LCD32B_Contrast, 62
 - seLCD32B_Contrast, 62
- LCD32B_DisplayState, 57
 - seLCD32B_DisplInvState, 57
 - seLCD32B_DispState, 57
- LCD32B_Frame_Frequency_Control, 58
 - seLCD32B_Duty, 58
- LCD32B_Functions, 65
 - ConfigurePortsForLcd32B, 66
 - LCD32B_IRQHandler, 66
 - seLCD32B_ClearDisplayMemory, 66
 - seLCD32B_ClearIntFlag, 66
 - seLCD32B_ConfigureClock, 67
 - seLCD32B_ConfigureDisplay, 67
 - seLCD32B_ConfigurePower, 67
 - seLCD32B_CopyToDisplayArea, 68
 - seLCD32B_Disable, 68

- seLCD32B_DisableInt, 68
- seLCD32B_Enable, 68
- seLCD32B_EnableInt, 69
- seLCD32B_GetDisplayArea, 69
- seLCD32B_GetDisplayInvState, 69
- seLCD32B_GetDisplayState, 69
- seLCD32B_GetIntFlag, 69
- seLCD32B_GetPanelContrast, 70
- seLCD32B_Init, 70
- seLCD32B_SetBoostClk, 70
- seLCD32B_SetDisplayArea, 71
- seLCD32B_SetDisplayInvState, 71
- seLCD32B_SetDisplayMemory, 71
- seLCD32B_SetDisplayState, 72
- seLCD32B_SetFrameFreq, 72
- seLCD32B_SetPanelContrast, 72
- LCD32B_IRQHandler
 - LCD32B_Functions, 66
- LCD32B_Power_Settings, 61
 - seLCD32B_PwrBias, 61
- LCD32B_RAM_Area, 63
 - seLCD32B_Area, 63
 - seLCD32B_SegRamAddress, 63
- LCD32B_Types, 64
- LCD32B, 52
- NMI_Handler
 - WDT2_Functions, 262
- PORT_IRQHandler
 - PSPORT_Public_Functions, 87
 - USB_Functions, 238
- PSPORT_Exported_Types, 85
- PSPORT_Exported_constants, 74
 - sePSPORT_AltFunc, 77
 - sePSPORT_Data, 78
 - sePSPORT_EXOSC_ClkDiv, 78
 - sePSPORT_Edge, 78
 - sePSPORT_IOSC_ClkDiv, 80
 - sePSPORT_Id, 78
 - sePSPORT_OSC1_ClkDiv, 80
 - sePSPORT_OSC3_ClkDiv, 81
 - sePSPORT_PeriphPortInit, 81
 - sePSPORT_PortGroup, 82
 - sePSPORT_PortNumber, 82
 - seUPMUX_Channel_Sel, 82
 - seUPMUX_I2C_Fnc, 83
 - seUPMUX_Peripheral_Sel, 83
 - seUPMUX_SPIA_Fnc, 83
 - seUPMUX_T16B_Fnc, 83
 - seUPMUX_UART_Fnc, 84
- PSPORT_Public_Functions, 86
 - PORT_IRQHandler, 87
 - sePSPORT_ClearIntFlag, 87
 - sePSPORT_ConfigureClock, 87
 - sePSPORT_DisableBuiltInResistor, 87
 - sePSPORT_DisableChatteringFilter, 88
 - sePSPORT_DisableInt, 88
 - sePSPORT_EnableChatteringFilter, 88
 - sePSPORT_EnableInt, 89
 - sePSPORT_EnablePullDownResistor, 89
 - sePSPORT_EnablePullUpResistor, 90
 - sePSPORT_GetChatteringFilter, 90
 - sePSPORT_GetInput, 90
 - sePSPORT_GetIntFlag, 90
 - sePSPORT_GetOutput, 91
 - sePSPORT_Init, 91
 - sePSPORT_InitAsAltFunction, 91
 - sePSPORT_InitAsHiZ, 92
 - sePSPORT_InitAsInput, 92
 - sePSPORT_InitAsOutput, 92
 - sePSPORT_SetOutput, 93
 - sePSPORT_UpMuxFunction, 93
- PSPORT, 73
- QSPI_Constants, 95
 - seQSPI_AddrMode, 96
 - seQSPI_Format, 96
 - seQSPI_IntFlag, 96
 - seQSPI_Interrupt, 96
 - seQSPI_IO, 97
 - seQSPI_OperMode, 97
 - seQSPI_Phase, 97
 - seQSPI_Polarity, 97
 - seQSPI_TransferMode, 98
- QSPI_Functions, 100
 - ConfigurePortsForQSPI, 101
 - QSPI_IRQHandler, 102
 - seQSPI_ASSERT_MST_CS0, 102
 - seQSPI_ASSERT_MST_CS1, 102
 - seQSPI_ASSERT_MST_CS2, 102
 - seQSPI_ASSERT_MST_CS3, 102
 - seQSPI_ClearIntFlag, 103
 - seQSPI_ClearMasterRxMMA, 103
 - seQSPI_DisableInt, 103
 - seQSPI_DmaRxBytes, 104
 - seQSPI_DmaRxHWords, 104
 - seQSPI_DmaRxMmaWords, 105
 - seQSPI_DmaTxBytes, 105
 - seQSPI_DmaTxHWords, 106
 - seQSPI_EnableInt, 106
 - seQSPI_GetBusSpeed, 107
 - seQSPI_GetIntFlag, 107
 - seQSPI_Init, 107
 - seQSPI_InitStructForMaster, 108
 - seQSPI_InitStructForSlave, 108
 - seQSPI_NEGATE_MST_CS0, 108
 - seQSPI_NEGATE_MST_CS1, 108
 - seQSPI_NEGATE_MST_CS2, 109

- seQSPI_NEGATE_MST_CS3, 109
- seQSPI_Reset, 109
- seQSPI_RxBytes, 109
- seQSPI_RxHWords, 110
- seQSPI_SetBusSpeed, 110
- seQSPI_SetIO, 110
- seQSPI_SetMasterRxMMA, 111
- seQSPI_SetMode, 111
- seQSPI_Start, 112
- seQSPI_Stop, 112
- seQSPI_TermMasterTx, 112
- seQSPI_TxBytes, 112
- seQSPI_TxHWords, 113
- seQSPI_TxValue, 113
- QSPI_IRQHandler
 - QSPI_Functions, 102
- QSPI_Types, 99
- QSPI, 94
- REMC2, 115
- REMC2_Constants, 116
 - seREMC2_EXOSC_ClkDiv, 117
 - seREMC2_IOSC_ClkDiv, 117
 - seREMC2_Interrupt, 117
 - seREMC2_OSC1_ClkDiv, 118
 - seREMC2_OSC3_ClkDiv, 118
- REMC2_Functions, 120
 - seREMC2_ConfigureClock, 120
 - seREMC2_Init, 120
 - seREMC2_Start, 121
 - seREMC2_Stop, 121
- REMC2_Types, 119
- RFC_Constants, 123
 - seRFC_IOSC_ClkDiv, 123
 - seRFC_EXOSC_ClkDiv, 124
 - seRFC_Interrupt, 124
 - seRFC_OSC1_ClkDiv, 124
 - seRFC_OSC3_ClkDiv, 124
 - seRFC_OscMode, 125
- RFC_Functions, 127
 - ConfigurePortsForRFC, 127
 - seRFC_ClearIntFlag, 127
 - seRFC_ConfigureClock, 128
 - seRFC_DisableInt, 128
 - seRFC_EnableInt, 129
 - seRFC_GetMeasurementCounter, 129
 - seRFC_GetTimeBaseCounter, 129
 - seRFC_Init, 129
 - seRFC_RunConvertingOperation, 130
 - seRFC_SetMeasurementCounter, 130
 - seRFC_SetTimeBaseCounter, 130
 - seRFC_Start, 131
 - seRFC_Stop, 131
- RFC_Types, 126
- RFC, 122
- RTCA_Constants, 133
 - seRTCA_AM_PM, 133
 - seRTCA_DayOfTheWeek, 134
 - seRTCA_Hours12_24, 134
 - seRTCA_Interrupt, 134
- RTCA_Functions, 137
 - RTCA_IRQHandler, 138
 - seRTCA_CalcWeekDay, 139
 - seRTCA_CalculateTrm, 138
 - seRTCA_ClearIntFlag, 139
 - seRTCA_Disable1SecTimer, 139
 - seRTCA_DisableInt, 140
 - seRTCA_Enable1SecTimer, 140
 - seRTCA_EnableInt, 140
 - seRTCA_Get12_24Mode, 140
 - seRTCA_GetAM_PM, 141
 - seRTCA_GetAlarm, 141
 - seRTCA_GetHourMinuteSecond, 141
 - seRTCA_GetIntFlag, 142
 - seRTCA_GetYearMonthDayWeek, 142
 - seRTCA_Init, 142
 - seRTCA_InitTheoreticalRegulation, 143
 - seRTCA_ReadStopWatchCount, 143
 - seRTCA_ResetStopWatchCount, 143
 - seRTCA_Set12_24Mode, 144
 - seRTCA_Set30secCorrection, 144
 - seRTCA_SetAM_PM, 145
 - seRTCA_SetAlarm, 144
 - seRTCA_SetHourMinuteSecond, 145
 - seRTCA_SetSecondsAlarm, 145
 - seRTCA_SetYearMonthDayWeek, 146
 - seRTCA_Start, 146
 - seRTCA_StartStopWatchCount, 146
 - seRTCA_Stop, 146
 - seRTCA_StopStopWatchCount, 147
 - seRTCA_TheoreticalRegulationTrim, 147
- RTCA_IRQHandler
 - RTCA_Functions, 138
- RTCA_Types, 136
- RTCA, 132
- SCS
 - seT16B_CCRregsDef, 284
- SNDA_Constants, 149
 - seSNDA_DriveMode, 149
 - seSNDA_EXOSC_ClkDiv, 149
 - seSNDA_IOSC_ClkDiv, 150
 - seSNDA_InterruptSrc, 150
 - seSNDA_ModeSel, 150
 - seSNDA_OSC1_ClkDiv, 150
 - seSNDA_OSC3_ClkDiv, 151
- SNDA_Functions, 153
 - SNDA_IRQHandler, 158

- seSNDA_ClearIntFlag, 153
- seSNDA_ConfigureClock, 154
- seSNDA_Disable, 154
- seSNDA_DisableInt, 154
- seSNDA_Enable, 155
- seSNDA_EnableInt, 155
- seSNDA_GetClk, 155
- seSNDA_GetIntFlag, 156
- seSNDA_Init, 156
- seSNDA_InitStruct, 156
- seSNDA_Start, 157
- seSNDA_StartMelody, 157
- seSNDA_StartOneShot, 157
- seSNDA_Stop, 158
- SNDA_IRQHandler
 - SNDA_Functions, 158
- SNDA_Types, 152
- SNDA, 148
- SPIA_0_IRQHandler
 - SPIA_Functions, 173
- SPIA_Constants, 160
 - seSPIA_Format, 161
 - seSPIA_IntFlag, 161
 - seSPIA_Interrupt, 161
 - seSPIA_OperMode, 161
 - seSPIA_Phase, 162
 - seSPIA_Polarity, 162
- SPIA_Functions, 164
 - ConfigurePortsForSPI, 165
 - SPIA_0_IRQHandler, 173
 - seSPIA_ASSERT_MST_CS0, 165
 - seSPIA_ASSERT_MST_CS1, 165
 - seSPIA_ClearIntFlag, 165
 - seSPIA_DisableInt, 166
 - seSPIA_DmaRxHWords, 166
 - seSPIA_DmaTxHWords, 167
 - seSPIA_ENABLE_MST_CS0, 167
 - seSPIA_ENABLE_MST_CS1, 167
 - seSPIA_EnableInt, 167
 - seSPIA_GetBusSpeed, 168
 - seSPIA_GetIntFlag, 168
 - seSPIA_Init, 168
 - seSPIA_InitStructForMaster, 169
 - seSPIA_InitStructForSlave, 169
 - seSPIA_NEGATE_MST_CS0, 169
 - seSPIA_NEGATE_MST_CS1, 170
 - seSPIA_Reset, 170
 - seSPIA_RxBytes, 170
 - seSPIA_RxHWords, 170
 - seSPIA_SetBusSpeed, 171
 - seSPIA_Start, 171
 - seSPIA_Stop, 171
 - seSPIA_TxBytes, 172
 - seSPIA_TxHWords, 172
- SPIA_Types, 163
- SPIA, 159
- SVD2, 174
 - SVD2_0_IRQHandler
 - SVD_Functions, 183
 - SVD2_1_IRQHandler
 - SVD_Functions, 184
 - USB_Functions, 243
 - SVD2_Constants, 175
 - seSVD2_DetectMode, 176
 - seSVD2_EXOSC_ClkDiv, 176
 - seSVD2_IOSC_ClkDiv, 177
 - seSVD2_IntFlag, 176
 - seSVD2_IntermittentMode, 176
 - seSVD2_Interrupt, 176
 - seSVD2_OSC1_ClkDiv, 177
 - seSVD2_OSC3_ClkDiv, 177
 - seSVD2_PowerSupply, 177
 - seSVD2_SamplingResCnt, 178
 - seSVD2_VoltageSource, 178
 - SVD_Functions, 180
 - ConfigurePortsForSVD2, 180
 - SVD2_0_IRQHandler, 183
 - SVD2_1_IRQHandler, 184
 - seSVD2_ClearIntLowVoltage, 180
 - seSVD2_GetVoltageDetection, 181
 - seSVD2_Init, 181
 - seSVD2_InitStruct, 182
 - seSVD2_IsIntLowVoltage, 182
 - seSVD2_SetComparisonVoltage, 182
 - seSVD2_SetVoltageSource, 182
 - seSVD2_Start, 183
 - seSVD2_Stop, 183
 - SVD_Types, 179
- seAssert
 - Common_Functions, 9
- seCLG_CLG_ClkDiv
 - CLG_Constants, 13
- seCLG_ClearIntFlag
 - CLG_Functions, 18
- seCLG_ClkDiv, 269
- seCLG_ClkSrc
 - CLG_Constants, 13
- seCLG_DisableInt
 - CLG_Functions, 18
- seCLG_EXOSC_ClkDiv
 - CLG_Constants, 13
- seCLG_EnableInt
 - CLG_Functions, 19
- seCLG_GetIntFlag
 - CLG_Functions, 19
- seCLG_GetIoscFreqSel
 - CLG_Functions, 19
- seCLG_GetOperInSlp

- CLG_Functions, 19
- seCLG_GetSysClk
 - CLG_Functions, 20
- seCLG_GetSysClkDiv
 - CLG_Functions, 20
- seCLG_GetSysClkSrc
 - CLG_Functions, 20
- seCLG_IOSC_ClkDiv
 - CLG_Constants, 14
- seCLG_IOSC_IoscFq
 - CLG_Constants, 14
- seCLG_Init
 - CLG_Functions, 20
- seCLG_InitTypeDef, 269
- seCLG_IntFlag
 - CLG_Constants, 14
- seCLG_Interrupt
 - CLG_Constants, 13
- seCLG_OSC1_ClkDiv
 - CLG_Constants, 15
- seCLG_OSC3_ClkDiv
 - CLG_Constants, 15
- seCLG_RunAutoTrimming
 - CLG_Functions, 21
- seCLG_SetIoscFreqSel
 - CLG_Functions, 21
- seCLG_SetOperInSlp
 - CLG_Functions, 21
- seCLG_SetStopDetection
 - CLG_Functions, 22
- seCLG_SetWkUpSysClk
 - CLG_Functions, 22
- seCLG_Start
 - CLG_Functions, 23
- seCLG_Stop
 - CLG_Functions, 23
- seCLG_SwitchSysClkSrc
 - CLG_Functions, 23
- seClamp16
 - Common_Functions, 9
- seClamp32
 - Common_Functions, 10
- seDMAC_AlternateDisable
 - DMAC_Functions, 30
- seDMAC_AlternateEnable
 - DMAC_Functions, 30
- seDMAC_CHANNEL
 - DMAC_Constants, 25
- seDMAC_ClearIntFlag
 - DMAC_Functions, 31
- seDMAC_ConfigMemToPeriph
 - DMAC_Functions, 31
- seDMAC_ConfigPeriphToMem
 - DMAC_Functions, 31
- seDMAC_CtrlData, 270
- seDMAC_DataStruct, 270
 - transfer_source_end_pointer, 271
- seDMAC_Disable
 - DMAC_Functions, 32
- seDMAC_DisableInt
 - DMAC_Functions, 32
- seDMAC_DisableRequestMask
 - DMAC_Functions, 33
- seDMAC_Enable
 - DMAC_Functions, 33
- seDMAC_EnableInt
 - DMAC_Functions, 33
- seDMAC_EnableRequestMask
 - DMAC_Functions, 33
- seDMAC_GetAltDataStrucPtr
 - DMAC_Functions, 34
- seDMAC_GetDataStrucPtr
 - DMAC_Functions, 34
- seDMAC_GetIntFlag
 - DMAC_Functions, 34
- seDMAC_GetMode
 - DMAC_Functions, 34
- seDMAC_GetNMinus1
 - DMAC_Functions, 35
- seDMAC_Inc
 - DMAC_Constants, 26
- seDMAC_Init
 - DMAC_Functions, 35
- seDMAC_InterruptSrc
 - DMAC_Constants, 26
- seDMAC_Mode
 - DMAC_Constants, 26
- seDMAC_NonBlockTransfMemToPeriph
 - DMAC_Functions, 35
- seDMAC_NonBlockTransfPeriphToMem
 - DMAC_Functions, 36
- seDMAC_PriorityDecrease
 - DMAC_Functions, 36
- seDMAC_PriorityIncrease
 - DMAC_Functions, 37
- seDMAC_SetChannel
 - DMAC_Functions, 37
- seDMAC_SetDataStrucPtr
 - DMAC_Functions, 37
- seDMAC_Size
 - DMAC_Constants, 27
- seDMAC_Start
 - DMAC_Functions, 38
- seI2C_AddrMode
 - I2C_Constants, 41
- seI2C_ChannelDef, 271
- seI2C_ClearIntFlag
 - I2C_Functions, 46

- sel2C_ClkDiv, [271](#)
- sel2C_Disable
 - I2C_Functions, [46](#)
- sel2C_DisableInt
 - I2C_Functions, [47](#)
- sel2C_EXOSC_ClkDiv
 - I2C_Constants, [41](#)
- sel2C_Enable
 - I2C_Functions, [47](#)
- sel2C_EnableInt
 - I2C_Functions, [47](#)
- sel2C_GetIntFlag
 - I2C_Functions, [48](#)
- sel2C_IOSC_ClkDiv
 - I2C_Constants, [42](#)
- sel2C_Init
 - I2C_Functions, [48](#)
- sel2C_InitStructForMaster
 - I2C_Functions, [48](#)
- sel2C_InitStructForSlave
 - I2C_Functions, [49](#)
- sel2C_InitTypeDef, [271](#)
 - BRT, [272](#)
 - SlaveAddr, [272](#)
- sel2C_IntFlag
 - I2C_Constants, [41](#)
- sel2C_Interrupt
 - I2C_Constants, [41](#)
- sel2C_MstReceiveData
 - I2C_Functions, [49](#)
- sel2C_MstSendData
 - I2C_Functions, [49](#)
- sel2C_OSC1_ClkDiv
 - I2C_Constants, [42](#)
- sel2C_OSC3_ClkDiv
 - I2C_Constants, [43](#)
- sel2C_Reset
 - I2C_Functions, [50](#)
- sel2C_SlvReceiveData
 - I2C_Functions, [50](#)
- sel2C_SlvSendData
 - I2C_Functions, [51](#)
- sel2C_mode
 - I2C_Constants, [42](#)
- selInterruptStatus
 - Common_Constants, [6](#)
- seLCD32B_Area
 - LCD32B_RAM_Area, [63](#)
- seLCD32B_BoostClk
 - LCD32B_Clock, [54](#)
- seLCD32B_ClearDisplayMemory
 - LCD32B_Functions, [66](#)
- seLCD32B_ClearIntFlag
 - LCD32B_Functions, [66](#)
- seLCD32B_ClkDiv, [272](#)
- seLCD32B_ConfigureClock
 - LCD32B_Functions, [67](#)
- seLCD32B_ConfigureDisplay
 - LCD32B_Functions, [67](#)
- seLCD32B_ConfigurePower
 - LCD32B_Functions, [67](#)
- seLCD32B_Contrast
 - LCD32B_Contrast, [62](#)
- seLCD32B_CopyToDisplayArea
 - LCD32B_Functions, [68](#)
- seLCD32B_Disable
 - LCD32B_Functions, [68](#)
- seLCD32B_DisableInt
 - LCD32B_Functions, [68](#)
- seLCD32B_DisplnVState
 - LCD32B_DisplayState, [57](#)
- seLCD32B_DispState
 - LCD32B_DisplayState, [57](#)
- seLCD32B_Duty
 - LCD32B_Frame_Frequency_Control, [58](#)
- seLCD32B_EXOSC_ClkDiv
 - LCD32B_Clock, [55](#)
- seLCD32B_Enable
 - LCD32B_Functions, [68](#)
- seLCD32B_EnableInt
 - LCD32B_Functions, [69](#)
- seLCD32B_GetDisplayArea
 - LCD32B_Functions, [69](#)
- seLCD32B_GetDisplayInvState
 - LCD32B_Functions, [69](#)
- seLCD32B_GetDisplayState
 - LCD32B_Functions, [69](#)
- seLCD32B_GetIntFlag
 - LCD32B_Functions, [69](#)
- seLCD32B_GetPanelContrast
 - LCD32B_Functions, [70](#)
- seLCD32B_IOSC_ClkDiv
 - LCD32B_Clock, [55](#)
- seLCD32B_Init
 - LCD32B_Functions, [70](#)
- seLCD32B_InitTypeDef, [272](#)
 - CtlLcdIoDischargeEn, [273](#)
 - DspSelComPinDir, [273](#)
 - DspSelSegPinDir, [273](#)
- seLCD32B_OSC1_ClkDiv
 - LCD32B_Clock, [55](#)
- seLCD32B_OSC3_ClkDiv
 - LCD32B_Clock, [55](#)
- seLCD32B_PwrBias
 - LCD32B_Power_Settings, [61](#)
- seLCD32B_SegRamAddress
 - LCD32B_RAM_Area, [63](#)
- seLCD32B_SetBoostClk

- LCD32B_Funcions, 70
- seLCD32B_SetDisplayArea
 - LCD32B_Funcions, 71
- seLCD32B_SetDisplayInvState
 - LCD32B_Funcions, 71
- seLCD32B_SetDisplayMemory
 - LCD32B_Funcions, 71
- seLCD32B_SetDisplayState
 - LCD32B_Funcions, 72
- seLCD32B_SetFrameFreq
 - LCD32B_Funcions, 72
- seLCD32B_SetPanelContrast
 - LCD32B_Funcions, 72
- sePPORT_AltFunc
 - PPORT_Exported_constants, 77
- sePPORT_ClearIntFlag
 - PPORT_Public_Funcions, 87
- sePPORT_ConfigureClock
 - PPORT_Public_Funcions, 87
- sePPORT_Data
 - PPORT_Exported_constants, 78
- sePPORT_DisableBuiltInResistor
 - PPORT_Public_Funcions, 87
- sePPORT_DisableChatteringFilter
 - PPORT_Public_Funcions, 88
- sePPORT_DisableInt
 - PPORT_Public_Funcions, 88
- sePPORT_EXOSC_ClkDiv
 - PPORT_Exported_constants, 78
- sePPORT_Edge
 - PPORT_Exported_constants, 78
- sePPORT_EnableChatteringFilter
 - PPORT_Public_Funcions, 88
- sePPORT_EnableInt
 - PPORT_Public_Funcions, 89
- sePPORT_EnablePullDownResistor
 - PPORT_Public_Funcions, 89
- sePPORT_EnablePullUpResistor
 - PPORT_Public_Funcions, 90
- sePPORT_GetChatteringFilter
 - PPORT_Public_Funcions, 90
- sePPORT_GetInput
 - PPORT_Public_Funcions, 90
- sePPORT_GetIntFlag
 - PPORT_Public_Funcions, 90
- sePPORT_GetOutput
 - PPORT_Public_Funcions, 91
- sePPORT_Group, 274
 - DAT, 274
- sePPORT_IOSOC_ClkDiv
 - PPORT_Exported_constants, 80
- sePPORT_Id
 - PPORT_Exported_constants, 78
- sePPORT_Init
 - PPORT_Public_Funcions, 91
- sePPORT_InitAsAltFunction
 - PPORT_Public_Funcions, 91
- sePPORT_InitAsHiZ
 - PPORT_Public_Funcions, 92
- sePPORT_InitAsInput
 - PPORT_Public_Funcions, 92
- sePPORT_InitAsOutput
 - PPORT_Public_Funcions, 92
- sePPORT_InitTypeDef, 274
- sePPORT_OSC1_ClkDiv
 - PPORT_Exported_constants, 80
- sePPORT_OSC3_ClkDiv
 - PPORT_Exported_constants, 81
- sePPORT_PeriphPortDef, 275
- sePPORT_PeriphPortInit
 - PPORT_Exported_constants, 81
- sePPORT_PortGroup
 - PPORT_Exported_constants, 82
- sePPORT_PortNumber
 - PPORT_Exported_constants, 82
- sePPORT_SetOutput
 - PPORT_Public_Funcions, 93
- sePPORT_UpMuxFunction
 - PPORT_Public_Funcions, 93
- sePeriphLibrary, 3
- seProtectSys
 - Common_Funcions, 10
- seQSPI_ASSERT_MST_CS0
 - QSPI_Funcions, 102
- seQSPI_ASSERT_MST_CS1
 - QSPI_Funcions, 102
- seQSPI_ASSERT_MST_CS2
 - QSPI_Funcions, 102
- seQSPI_ASSERT_MST_CS3
 - QSPI_Funcions, 102
- seQSPI_AddrMode
 - QSPI_Constants, 96
- seQSPI_ChannelDef, 275
- seQSPI_ClearIntFlag
 - QSPI_Funcions, 103
- seQSPI_ClearMasterRxMMA
 - QSPI_Funcions, 103
- seQSPI_DisableInt
 - QSPI_Funcions, 103
- seQSPI_DmaRxBytes
 - QSPI_Funcions, 104
- seQSPI_DmaRxHWords
 - QSPI_Funcions, 104
- seQSPI_DmaRxMmaWords
 - QSPI_Funcions, 105
- seQSPI_DmaTxBytes
 - QSPI_Funcions, 105
- seQSPI_DmaTxHWords

- QSPI_Funcions, [106](#)
- seQSPI_EnableInt
 - QSPI_Funcions, [106](#)
- seQSPI_Format
 - QSPI_Constants, [96](#)
- seQSPI_GetBusSpeed
 - QSPI_Funcions, [107](#)
- seQSPI_GetIntFlag
 - QSPI_Funcions, [107](#)
- seQSPI_Init
 - QSPI_Funcions, [107](#)
- seQSPI_InitStructForMaster
 - QSPI_Funcions, [108](#)
- seQSPI_InitStructForSlave
 - QSPI_Funcions, [108](#)
- seQSPI_InitTypeDef, [275](#)
- seQSPI_IntFlag
 - QSPI_Constants, [96](#)
- seQSPI_Interrupt
 - QSPI_Constants, [96](#)
- seQSPI_IO
 - QSPI_Constants, [97](#)
- seQSPI_NEGATE_MST_CS0
 - QSPI_Funcions, [108](#)
- seQSPI_NEGATE_MST_CS1
 - QSPI_Funcions, [108](#)
- seQSPI_NEGATE_MST_CS2
 - QSPI_Funcions, [109](#)
- seQSPI_NEGATE_MST_CS3
 - QSPI_Funcions, [109](#)
- seQSPI_OperMode
 - QSPI_Constants, [97](#)
- seQSPI_Phase
 - QSPI_Constants, [97](#)
- seQSPI_Polarity
 - QSPI_Constants, [97](#)
- seQSPI_Reset
 - QSPI_Funcions, [109](#)
- seQSPI_RxBytes
 - QSPI_Funcions, [109](#)
- seQSPI_RxHWords
 - QSPI_Funcions, [110](#)
- seQSPI_SetBusSpeed
 - QSPI_Funcions, [110](#)
- seQSPI_SetIO
 - QSPI_Funcions, [110](#)
- seQSPI_SetMasterRxMMA
 - QSPI_Funcions, [111](#)
- seQSPI_SetMode
 - QSPI_Funcions, [111](#)
- seQSPI_Start
 - QSPI_Funcions, [112](#)
- seQSPI_Stop
 - QSPI_Funcions, [112](#)
- seQSPI_TermMasterTx
 - QSPI_Funcions, [112](#)
- seQSPI_TransferMode
 - QSPI_Constants, [98](#)
- seQSPI_TxBytes
 - QSPI_Funcions, [112](#)
- seQSPI_TxHWords
 - QSPI_Funcions, [113](#)
- seQSPI_TxValue
 - QSPI_Funcions, [113](#)
- seREMC2_ChannelDef, [277](#)
- seREMC2_ConfigureClock
 - REMC2_Funcions, [120](#)
- seREMC2_EXOSC_ClkDiv
 - REMC2_Constants, [117](#)
- seREMC2_IOSC_ClkDiv
 - REMC2_Constants, [117](#)
- seREMC2_Init
 - REMC2_Funcions, [120](#)
- seREMC2_InitTypeDef, [277](#)
- seREMC2_Interrupt
 - REMC2_Constants, [117](#)
- seREMC2_OSC1_ClkDiv
 - REMC2_Constants, [118](#)
- seREMC2_OSC3_ClkDiv
 - REMC2_Constants, [118](#)
- seREMC2_Start
 - REMC2_Funcions, [121](#)
- seREMC2_Stop
 - REMC2_Funcions, [121](#)
- seRFC_IOSC_ClkDiv
 - RFC_Constants, [123](#)
- seRFC_ChannelDef, [277](#)
- seRFC_ClearIntFlag
 - RFC_Funcions, [127](#)
- seRFC_ConfigureClock
 - RFC_Funcions, [128](#)
- seRFC_DisableInt
 - RFC_Funcions, [128](#)
- seRFC_EXOSC_ClkDiv
 - RFC_Constants, [124](#)
- seRFC_EnableInt
 - RFC_Funcions, [129](#)
- seRFC_GetMeasurementCounter
 - RFC_Funcions, [129](#)
- seRFC_GetTimeBaseCounter
 - RFC_Funcions, [129](#)
- seRFC_Init
 - RFC_Funcions, [129](#)
- seRFC_InitTypeDef, [278](#)
- seRFC_Interrupt
 - RFC_Constants, [124](#)
- seRFC_OSC1_ClkDiv
 - RFC_Constants, [124](#)

- seRFC_OSC3_ClkDiv
 - RFC_Constants, [124](#)
- seRFC_OscMode
 - RFC_Constants, [125](#)
- seRFC_RunConvertingOperation
 - RFC_Functions, [130](#)
- seRFC_SetMeasurementCounter
 - RFC_Functions, [130](#)
- seRFC_SetTimeBaseCounter
 - RFC_Functions, [130](#)
- seRFC_Start
 - RFC_Functions, [131](#)
- seRFC_Stop
 - RFC_Functions, [131](#)
- seRTCA_AM_PM
 - RTCA_Constants, [133](#)
- seRTCA_CalcWeekDay
 - RTCA_Functions, [139](#)
- seRTCA_CalculateTrm
 - RTCA_Functions, [138](#)
- seRTCA_ClearIntFlag
 - RTCA_Functions, [139](#)
- seRTCA_DayOfTheWeek
 - RTCA_Constants, [134](#)
- seRTCA_Disable1SecTimer
 - RTCA_Functions, [139](#)
- seRTCA_DisableInt
 - RTCA_Functions, [140](#)
- seRTCA_Enable1SecTimer
 - RTCA_Functions, [140](#)
- seRTCA_EnableInt
 - RTCA_Functions, [140](#)
- seRTCA_Get12_24Mode
 - RTCA_Functions, [140](#)
- seRTCA_GetAM_PM
 - RTCA_Functions, [141](#)
- seRTCA_GetAlarm
 - RTCA_Functions, [141](#)
- seRTCA_GetHourMinuteSecond
 - RTCA_Functions, [141](#)
- seRTCA_GetIntFlag
 - RTCA_Functions, [142](#)
- seRTCA_GetYearMonthDayWeek
 - RTCA_Functions, [142](#)
- seRTCA_Hours12_24
 - RTCA_Constants, [134](#)
- seRTCA_Init
 - RTCA_Functions, [142](#)
- seRTCA_InitTheoreticalRegulation
 - RTCA_Functions, [143](#)
- seRTCA_InitTypeDef, [278](#)
- seRTCA_Interrupt
 - RTCA_Constants, [134](#)
- seRTCA_ReadStopWatchCount
 - RTCA_Functions, [143](#)
- seRTCA_ResetStopWatchCount
 - RTCA_Functions, [143](#)
- seRTCA_Set12_24Mode
 - RTCA_Functions, [144](#)
- seRTCA_Set30secCorrection
 - RTCA_Functions, [144](#)
- seRTCA_SetAM_PM
 - RTCA_Functions, [145](#)
- seRTCA_SetAlarm
 - RTCA_Functions, [144](#)
- seRTCA_SetHourMinuteSecond
 - RTCA_Functions, [145](#)
- seRTCA_SetSecondsAlarm
 - RTCA_Functions, [145](#)
- seRTCA_SetYearMonthDayWeek
 - RTCA_Functions, [146](#)
- seRTCA_Start
 - RTCA_Functions, [146](#)
- seRTCA_StartStopWatchCount
 - RTCA_Functions, [146](#)
- seRTCA_Stop
 - RTCA_Functions, [146](#)
- seRTCA_StopStopWatchCount
 - RTCA_Functions, [147](#)
- seRTCA_TheoreticalRegulationTrim
 - RTCA_Functions, [147](#)
- seSNDA_ChannelDef, [278](#)
- seSNDA_ClearIntFlag
 - SNDA_Functions, [153](#)
- seSNDA_ConfigureClock
 - SNDA_Functions, [154](#)
- seSNDA_Disable
 - SNDA_Functions, [154](#)
- seSNDA_DisableInt
 - SNDA_Functions, [154](#)
- seSNDA_DriveMode
 - SNDA_Constants, [149](#)
- seSNDA_EXOSC_ClkDiv
 - SNDA_Constants, [149](#)
- seSNDA_Enable
 - SNDA_Functions, [155](#)
- seSNDA_EnableInt
 - SNDA_Functions, [155](#)
- seSNDA_GetCk
 - SNDA_Functions, [155](#)
- seSNDA_GetIntFlag
 - SNDA_Functions, [156](#)
- seSNDA_IOSC_ClkDiv
 - SNDA_Constants, [150](#)
- seSNDA_Init
 - SNDA_Functions, [156](#)
- seSNDA_InitStruct
 - SNDA_Functions, [156](#)

- seSNDA_InitTypeDef, 279
- seSNDA_InterruptSrc
 - SNDA_Constants, 150
- seSNDA_ModeSel
 - SNDA_Constants, 150
- seSNDA_OSC1_ClkDiv
 - SNDA_Constants, 150
- seSNDA_OSC3_ClkDiv
 - SNDA_Constants, 151
- seSNDA_Start
 - SNDA_Functions, 157
- seSNDA_StartMelody
 - SNDA_Functions, 157
- seSNDA_StartOneShot
 - SNDA_Functions, 157
- seSNDA_Stop
 - SNDA_Functions, 158
- seSPIA_ASSERT_MST_CS0
 - SPIA_Functions, 165
- seSPIA_ASSERT_MST_CS1
 - SPIA_Functions, 165
- seSPIA_ChannelDef, 279
- seSPIA_ClearIntFlag
 - SPIA_Functions, 165
- seSPIA_DisableInt
 - SPIA_Functions, 166
- seSPIA_DmaRxDWords
 - SPIA_Functions, 166
- seSPIA_DmaTxHWords
 - SPIA_Functions, 167
- seSPIA_ENABLE_MST_CS0
 - SPIA_Functions, 167
- seSPIA_ENABLE_MST_CS1
 - SPIA_Functions, 167
- seSPIA_EnableInt
 - SPIA_Functions, 167
- seSPIA_Format
 - SPIA_Constants, 161
- seSPIA_GetBusSpeed
 - SPIA_Functions, 168
- seSPIA_GetIntFlag
 - SPIA_Functions, 168
- seSPIA_Init
 - SPIA_Functions, 168
- seSPIA_InitStructForMaster
 - SPIA_Functions, 169
- seSPIA_InitStructForSlave
 - SPIA_Functions, 169
- seSPIA_InitTypeDef, 279
- seSPIA_IntFlag
 - SPIA_Constants, 161
- seSPIA_Interrupt
 - SPIA_Constants, 161
- seSPIA_NEGATE_MST_CS0
 - SPIA_Functions, 169
- seSPIA_NEGATE_MST_CS1
 - SPIA_Functions, 170
- seSPIA_OperMode
 - SPIA_Constants, 161
- seSPIA_Phase
 - SPIA_Constants, 162
- seSPIA_Polarity
 - SPIA_Constants, 162
- seSPIA_Reset
 - SPIA_Functions, 170
- seSPIA_RxBytes
 - SPIA_Functions, 170
- seSPIA_RxHWords
 - SPIA_Functions, 170
- seSPIA_SetBusSpeed
 - SPIA_Functions, 171
- seSPIA_Start
 - SPIA_Functions, 171
- seSPIA_Stop
 - SPIA_Functions, 171
- seSPIA_TxBytes
 - SPIA_Functions, 172
- seSPIA_TxHWords
 - SPIA_Functions, 172
- seSVD2_ChannelDef, 280
- seSVD2_ClearIntLowVoltage
 - SVD_Functions, 180
- seSVD2_ClkDiv, 280
- seSVD2_DetectMode
 - SVD2_Constants, 176
- seSVD2_EXOSC_ClkDiv
 - SVD2_Constants, 176
- seSVD2_GetVoltageDetection
 - SVD_Functions, 181
- seSVD2_IOSC_ClkDiv
 - SVD2_Constants, 177
- seSVD2_Init
 - SVD_Functions, 181
- seSVD2_InitStruct
 - SVD_Functions, 182
- seSVD2_InitTypeDef, 280
- seSVD2_IntFlag
 - SVD2_Constants, 176
- seSVD2_IntermittentMode
 - SVD2_Constants, 176
- seSVD2_Interrupt
 - SVD2_Constants, 176
- seSVD2_IsIntLowVoltage
 - SVD_Functions, 182
- seSVD2_OSC1_ClkDiv
 - SVD2_Constants, 177
- seSVD2_OSC3_ClkDiv
 - SVD2_Constants, 177

- seSVD2_PowerSupply
 - SVD2_Constants, [177](#)
- seSVD2_SamplingResCnt
 - SVD2_Constants, [178](#)
- seSVD2_SetComparisonVoltage
 - SVD_Functions, [182](#)
- seSVD2_SetVoltageSource
 - SVD_Functions, [182](#)
- seSVD2_Start
 - SVD_Functions, [183](#)
- seSVD2_Stop
 - SVD_Functions, [183](#)
- seSVD2_VoltageSource
 - SVD2_Constants, [178](#)
- seState
 - Common_Constants, [6](#)
- seStatus
 - Common_Constants, [6](#)
- seT16_ClearIntFlag
 - T16_Functions, [190](#)
- seT16_ConfigureClock
 - T16_Functions, [191](#)
- seT16_ConfigureCounterMode
 - T16_Functions, [191](#)
- seT16_CounterMode
 - T16_Constants, [187](#)
- seT16_Disable
 - T16_Functions, [191](#)
- seT16_DisableInt
 - T16_Functions, [192](#)
- seT16_EXOSC_ClkDiv
 - T16_Constants, [187](#)
- seT16_Enable
 - T16_Functions, [192](#)
- seT16_EnableInt
 - T16_Functions, [192](#)
- seT16_GetClk
 - T16_Functions, [193](#)
- seT16_GetCounter
 - T16_Functions, [193](#)
- seT16_GetIntFlag
 - T16_Functions, [193](#)
- seT16_IOSC_ClkDiv
 - T16_Constants, [187](#)
- seT16_Init
 - T16_Functions, [193](#)
- seT16_InitStruct
 - T16_Functions, [194](#)
- seT16_InitTypeDef, [281](#)
- seT16_OSC1_ClkDiv
 - T16_Constants, [188](#)
- seT16_OSC3_ClkDiv
 - T16_Constants, [188](#)
- seT16_SetCounter
 - T16_Functions, [194](#)
- seT16_Start
 - T16_Functions, [195](#)
- seT16_Stop
 - T16_Functions, [195](#)
- seT16B_CAPIS
 - T16B_Constants, [200](#)
- seT16B_CAPTRG
 - T16B_Constants, [200](#)
- seT16B_CBUFMD
 - T16B_Constants, [201](#)
- seT16B_CCCTL, [282](#)
- seT16B_CCMD
 - T16B_Constants, [201](#)
- seT16B_CCRegsDef, [282](#)
 - CAPIS, [283](#)
 - CAPTRG, [283](#)
 - CBUFMD, [283](#)
 - CC0DMAEN, [283](#)
 - CCCTL_b, [283](#)
 - CCCTL, [283](#)
 - CCDMAEN_b, [284](#)
 - CCDMAEN, [283](#)
 - CCMD, [284](#)
 - CCR_b, [284](#)
 - CCR, [284](#)
 - CC, [283](#)
 - SCS, [284](#)
 - TOUTINV, [284](#)
 - TOUTMD, [284](#)
 - TOUTMT, [284](#)
 - TOUTO, [284](#)
- seT16B_CNTMD
 - T16B_Constants, [201](#)
- seT16B_ChannelDef, [285](#)
- seT16B_ClearIntFlag
 - T16B_Functions, [207](#)
- seT16B_ClkSrc
 - T16B_Constants, [201](#)
- seT16B_ConfigureClock
 - T16B_Functions, [207](#)
- seT16B_ConfigureCounterMode
 - T16B_Functions, [208](#)
- seT16B_Disable
 - T16B_Functions, [208](#)
- seT16B_DisableInt
 - T16B_Functions, [208](#)
- seT16B_EX_ClkDiv
 - T16B_Constants, [202](#)
- seT16B_Enable
 - T16B_Functions, [208](#)
- seT16B_EnableInt
 - T16B_Functions, [209](#)
- seT16B_GetClk

- T16B_Funcions, [209](#)
- seT16B_GetCmpCapCnt
 - T16B_Funcions, [209](#)
- seT16B_GetIntFlag
 - T16B_Funcions, [210](#)
- seT16B_GetMaxCounter
 - T16B_Funcions, [210](#)
- seT16B_GetTimerCount
 - T16B_Funcions, [210](#)
- seT16B_IOSC_ClkDiv
 - T16B_Constants, [202](#)
- seT16B_Init
 - T16B_Funcions, [211](#)
- seT16B_InitStruct
 - T16B_Funcions, [211](#)
- seT16B_InitTypeDef, [285](#)
- seT16B_ONEST
 - T16B_Constants, [202](#)
- seT16B_OSC1_ClkDiv
 - T16B_Constants, [203](#)
- seT16B_OSC3_ClkDiv
 - T16B_Constants, [203](#)
- seT16B_SCS
 - T16B_Constants, [203](#)
- seT16B_SetCmpCapCnt
 - T16B_Funcions, [211](#)
- seT16B_SetMaxCounter
 - T16B_Funcions, [212](#)
- seT16B_SetTriggerSignal
 - T16B_Funcions, [212](#)
- seT16B_Start
 - T16B_Funcions, [213](#)
- seT16B_Stop
 - T16B_Funcions, [213](#)
- seT16B_TOUTINV
 - T16B_Constants, [203](#)
- seT16B_TOUTMD
 - T16B_Constants, [204](#)
- seT16B_TOUTMT
 - T16B_Constants, [204](#)
- seT16B_TOUTO
 - T16B_Constants, [204](#)
- seTimeoutMs
 - Common_Constants, [7](#)
- seUART2_ChannelDef, [286](#)
- seUART2_ClearIntFlag
 - UART_Funcions, [223](#)
- seUART2_ClkDiv, [286](#)
- seUART2_ConfigureClock
 - UART_Funcions, [223](#)
- seUART2_ConfigureMode
 - UART_Funcions, [224](#)
- seUART2_Disable
 - UART_Funcions, [224](#)
- seUART2_DisableInt
 - UART_Funcions, [224](#)
- seUART2_EXOSC_ClkDiv
 - UART2_Constants, [217](#)
- seUART2_Enable
 - UART_Funcions, [225](#)
- seUART2_EnableInt
 - UART_Funcions, [225](#)
- seUART2_EnableRxDMAReq
 - UART_Funcions, [225](#)
- seUART2_EnableTxDMAReq
 - UART_Funcions, [226](#)
- seUART2_GetData
 - UART_Funcions, [226](#)
- seUART2_GetIntFlag
 - UART_Funcions, [226](#)
- seUART2_GetUartClk
 - UART_Funcions, [227](#)
- seUART2_IOSC_ClkDiv
 - UART2_Constants, [217](#)
- seUART2_Init
 - UART_Funcions, [227](#)
- seUART2_InitStruct
 - UART_Funcions, [227](#)
- seUART2_InitTypeDef, [286](#)
- seUART2_Interrupt
 - UART2_Constants, [217](#)
- seUART2_MOD_ChIn
 - UART2_Constants, [218](#)
- seUART2_MOD_Invirrx
 - UART2_Constants, [218](#)
- seUART2_MOD_Invirtx
 - UART2_Constants, [218](#)
- seUART2_MOD_Irmd
 - UART2_Constants, [218](#)
- seUART2_MOD_Outmd
 - UART2_Constants, [219](#)
- seUART2_MOD_Pren
 - UART2_Constants, [219](#)
- seUART2_MOD_Prmd
 - UART2_Constants, [219](#)
- seUART2_MOD_Puen
 - UART2_Constants, [219](#)
- seUART2_MOD_Stpb
 - UART2_Constants, [219](#)
- seUART2_Mode, [287](#)
- seUART2_OSC1_ClkDiv
 - UART2_Constants, [220](#)
- seUART2_OSC3_ClkDiv
 - UART2_Constants, [220](#)
- seUART2_Receive
 - UART_Funcions, [228](#)
- seUART2_Send
 - UART_Funcions, [228](#)

- seUART2_SetBaudRate
 - UART_Functions, [229](#)
- seUART2_SetBaudRateReg
 - UART_Functions, [229](#)
- seUART2_SetData
 - UART_Functions, [229](#)
- seUPMUX_Channel_Sel
 - PPORT_Exported_constants, [82](#)
- seUPMUX_I2C_Fnc
 - PPORT_Exported_constants, [83](#)
- seUPMUX_Peripheral_Sel
 - PPORT_Exported_constants, [83](#)
- seUPMUX_SPIA_Fnc
 - PPORT_Exported_constants, [83](#)
- seUPMUX_T16B_Fnc
 - PPORT_Exported_constants, [83](#)
- seUPMUX_UART_Fnc
 - PPORT_Exported_constants, [84](#)
- seUSB_ActivateUSBCLK
 - USB_Functions, [238](#)
- seUSB_Attach
 - USB_Functions, [238](#)
- seUSB_ClearAllIntFlags
 - USB_Interrupt_Management, [257](#)
- seUSB_ClearEP0Fifo
 - USB_EP0_Functions, [244](#)
- seUSB_ClearEPmFifo
 - USB_EPm_Functions, [247](#)
- seUSB_ClearEPnFifos
 - USB_EPn_Functions, [249](#)
- seUSB_ClkSrc
 - USB_Constants_and_Macros, [232](#)
- seUSB_ClrStall
 - USB_Functions, [238](#)
- seUSB_ConfSvdDetectDisconnect
 - USB_Functions, [240](#)
- seUSB_ConfigureDefaultEndpoints
 - USB_Functions, [239](#)
- seUSB_ConfigureEPm
 - USB_Functions, [239](#)
- seUSB_ConfigurePortsForUsb
 - USB_Functions, [239](#)
- seUSB_Connect
 - USB_Bus_Management, [250](#)
- seUSB_DataInStage
 - USB_EP0_Functions, [244](#)
- seUSB_DataOutStage
 - USB_EP0_Functions, [245](#)
- seUSB_DeactivateUSBCLK
 - USB_Functions, [240](#)
- seUSB_Detach
 - USB_Functions, [240](#)
- seUSB_Disable
 - USB_Functions, [240](#)
- seUSB_DisableAllInt
 - USB_Interrupt_Management, [257](#)
- seUSB_DisableEPm
 - USB_Functions, [240](#)
- seUSB_DisableInt
 - USB_Functions, [241](#)
- seUSB_Disconnect
 - USB_Bus_Management, [250](#)
- seUSB_DmaCopyFromFifo
 - USB_FIFO_Management, [254](#)
- seUSB_DmaCopyToFifo
 - USB_FIFO_Management, [254](#)
- seUSB_EPCFG_DIR
 - USB_Constants_and_Macros, [233](#)
- seUSB_EPCFG_MAXSIZE
 - USB_Constants_and_Macros, [233](#)
- seUSB_EPCFG_TGLMOD
 - USB_Constants_and_Macros, [233](#)
- seUSB_Enable
 - USB_Functions, [241](#)
- seUSB_EnableAllInt
 - USB_Interrupt_Management, [257](#)
- seUSB_EnableEPm
 - USB_Functions, [241](#)
- seUSB_EnableInt
 - USB_Functions, [242](#)
- seUSB_Ep0Interrupt
 - USB_Constants_and_Macros, [232](#)
- seUSB_EpmInterrupt
 - USB_Constants_and_Macros, [234](#)
- seUSB_GetAddress
 - USB_Bus_Management, [251](#)
- seUSB_GetEPmDir
 - USB_EPm_Functions, [247](#)
- seUSB_GetEp0Dir
 - USB_EP0_Functions, [245](#)
- seUSB_GetSetupPacket
 - USB_EP0_Functions, [245](#)
- seUSB_GpeInterrupt
 - USB_Constants_and_Macros, [234](#)
- seUSB_Init
 - USB_Functions, [242](#)
- seUSB_InitEp0
 - USB_EP0_Functions, [245](#)
- seUSB_InitTypeDef, [287](#)
- seUSB_InitUsbModule
 - USB_Functions, [242](#)
- seUSB_IsSnoozing
 - USB_Bus_Management, [251](#)
- seUSB_IsVbusConnected
 - USB_Functions, [243](#)
- seUSB_MainInterrupt
 - USB_Constants_and_Macros, [234](#)
- seUSB_PSEL

- USB_Constants_and_Macros, 234
- seUSB_ReadFifo
 - USB_FIFO_Management, 254
- seUSB_Reset
 - USB_Bus_Management, 251
- seUSB_ResetEPm
 - USB_EPm_Functions, 247
- seUSB_Resume
 - USB_Bus_Management, 251
- seUSB_SetAddress
 - USB_Bus_Management, 252
- seUSB_SetEPmDir
 - USB_EPm_Functions, 248
- seUSB_SetEp0Dir
 - USB_EP0_Functions, 246
- seUSB_SetStall
 - USB_Functions, 243
- seUSB_SetupStage
 - USB_EP0_Functions, 246
- seUSB_SieInterrupt
 - USB_Constants_and_Macros, 235
- seUSB_Snooze
 - USB_Bus_Management, 252
- seUSB_StatusInStage
 - USB_EP0_Functions, 246
- seUSB_Suspend
 - USB_Bus_Management, 252
- seUSB_TRCTL_OPMODE
 - USB_Constants_and_Macros, 235
- seUSB_WakeUp
 - USB_Bus_Management, 252
- seUSB_WriteFifo
 - USB_FIFO_Management, 255
- seWDT2_ChipReset
 - WDT2_Functions, 262
- seWDT2_ConfigureClock
 - WDT2_Functions, 263
- seWDT2_EXOSC_ClkDiv
 - WDT2_Constants, 259
- seWDT2_Get_tWDTSecs
 - WDT2_Functions, 263
- seWDT2_GetCMP
 - WDT2_Functions, 263
- seWDT2_GetClk
 - WDT2_Functions, 263
- seWDT2_IOSC_ClkDiv
 - WDT2_Constants, 259
- seWDT2_Init
 - WDT2_Functions, 264
- seWDT2_InitStruct
 - WDT2_Functions, 264
- seWDT2_InitTypeDef, 288
 - ClkDivider, 288
- seWDT2_Mode
 - WDT2_Constants, 259
- seWDT2_OSC1_ClkDiv
 - WDT2_Constants, 260
- seWDT2_OSC3_ClkDiv
 - WDT2_Constants, 260
- seWDT2_ResetCounter
 - WDT2_Functions, 264
- seWDT2_Set_tWDTSecs
 - WDT2_Functions, 265
- seWDT2_SetCMP
 - WDT2_Functions, 265
- seWDT2_SetMode
 - WDT2_Functions, 265
- seWDT2_Start
 - WDT2_Functions, 265
- seWDT2_Stop
 - WDT2_Functions, 266
- seWriteProtect
 - Common_Constants, 7
- seseUSB_StatusOutStage
 - USB_EP0_Functions, 244
- SlaveAddr
 - seI2C_InitTypeDef, 272
- swCounter, 288
- T16, 185
- T16_0_IRQHandler
 - T16_Functions, 195
- T16_1_IRQHandler
 - T16_Functions, 195
- T16_2_IRQHandler
 - T16_Functions, 196
- T16_3_IRQHandler
 - T16_Functions, 196
- T16_Constants, 186
 - seT16_CounterMode, 187
 - seT16_EXOSC_ClkDiv, 187
 - seT16_IOSC_ClkDiv, 187
 - seT16_OSC1_ClkDiv, 188
 - seT16_OSC3_ClkDiv, 188
- T16_Functions, 190
 - seT16_ClearIntFlag, 190
 - seT16_ConfigureClock, 191
 - seT16_ConfigureCounterMode, 191
 - seT16_Disable, 191
 - seT16_DisableInt, 192
 - seT16_Enable, 192
 - seT16_EnableInt, 192
 - seT16_GetClk, 193
 - seT16_GetCounter, 193
 - seT16_GetIntFlag, 193
 - seT16_Init, 193
 - seT16_InitStruct, 194
 - seT16_SetCounter, 194

- seT16_Start, [195](#)
- seT16_Stop, [195](#)
- T16_0_IRQHandler, [195](#)
- T16_1_IRQHandler, [195](#)
- T16_2_IRQHandler, [196](#)
- T16_3_IRQHandler, [196](#)
- T16_Types, [189](#)
- T16B_0_IRQHandler
 - T16B_Functions, [213](#)
- T16B_1_IRQHandler
 - T16B_Functions, [213](#)
- T16B_Constants, [198](#)
 - seT16B_CAPIS, [200](#)
 - seT16B_CAPTRG, [200](#)
 - seT16B_CBUFMD, [201](#)
 - seT16B_CCMD, [201](#)
 - seT16B_CNTMD, [201](#)
 - seT16B_ClkSrc, [201](#)
 - seT16B_EX_ClkDiv, [202](#)
 - seT16B_IOSC_ClkDiv, [202](#)
 - seT16B_ONEST, [202](#)
 - seT16B_OSC1_ClkDiv, [203](#)
 - seT16B_OSC3_ClkDiv, [203](#)
 - seT16B_SCS, [203](#)
 - seT16B_TOUTINV, [203](#)
 - seT16B_TOUTMD, [204](#)
 - seT16B_TOUTMT, [204](#)
 - seT16B_TOUTO, [204](#)
- T16B_Functions, [206](#)
 - ConfigurePortsForT16B, [207](#)
 - seT16B_ClearIntFlag, [207](#)
 - seT16B_ConfigureClock, [207](#)
 - seT16B_ConfigureCounterMode, [208](#)
 - seT16B_Disable, [208](#)
 - seT16B_DisableInt, [208](#)
 - seT16B_Enable, [208](#)
 - seT16B_EnableInt, [209](#)
 - seT16B_GetClk, [209](#)
 - seT16B_GetCmpCapCnt, [209](#)
 - seT16B_GetIntFlag, [210](#)
 - seT16B_GetMaxCounter, [210](#)
 - seT16B_GetTimerCount, [210](#)
 - seT16B_Init, [211](#)
 - seT16B_InitStruct, [211](#)
 - seT16B_SetCmpCapCnt, [211](#)
 - seT16B_SetMaxCounter, [212](#)
 - seT16B_SetTriggerSignal, [212](#)
 - seT16B_Start, [213](#)
 - seT16B_Stop, [213](#)
 - T16B_0_IRQHandler, [213](#)
 - T16B_1_IRQHandler, [213](#)
- T16B_Types, [205](#)
- T16B, [197](#)
- TOUTINV
 - seT16B_CCRegsDef, [284](#)
- TOUTMD
 - seT16B_CCRegsDef, [284](#)
- TOUTMT
 - seT16B_CCRegsDef, [284](#)
- TOUTO
 - seT16B_CCRegsDef, [284](#)
- transfer_source_end_pointer
 - seDMAC_DataStruct, [271](#)
- UART2, [215](#)
 - UART2_0_IRQHandler
 - UART_Functions, [230](#)
 - UART2_1_IRQHandler
 - UART_Functions, [230](#)
 - UART2_Constants, [216](#)
 - seUART2_EXOSC_ClkDiv, [217](#)
 - seUART2_IOSC_ClkDiv, [217](#)
 - seUART2_Interrupt, [217](#)
 - seUART2_MOD_ChIn, [218](#)
 - seUART2_MOD_Invirrx, [218](#)
 - seUART2_MOD_Invirtx, [218](#)
 - seUART2_MOD_Irmd, [218](#)
 - seUART2_MOD_Outmd, [219](#)
 - seUART2_MOD_Pren, [219](#)
 - seUART2_MOD_Prmd, [219](#)
 - seUART2_MOD_Puen, [219](#)
 - seUART2_MOD_Stpb, [219](#)
 - seUART2_OSC1_ClkDiv, [220](#)
 - seUART2_OSC3_ClkDiv, [220](#)
 - UART_Functions, [222](#)
 - ConfigurePortsForUart, [223](#)
 - seUART2_ClearIntFlag, [223](#)
 - seUART2_ConfigureClock, [223](#)
 - seUART2_ConfigureMode, [224](#)
 - seUART2_Disable, [224](#)
 - seUART2_DisableInt, [224](#)
 - seUART2_Enable, [225](#)
 - seUART2_EnableInt, [225](#)
 - seUART2_EnableRxDMAReq, [225](#)
 - seUART2_EnableTxDMAReq, [226](#)
 - seUART2_GetData, [226](#)
 - seUART2_GetIntFlag, [226](#)
 - seUART2_GetUartClk, [227](#)
 - seUART2_Init, [227](#)
 - seUART2_InitStruct, [227](#)
 - seUART2_Receive, [228](#)
 - seUART2_Send, [228](#)
 - seUART2_SetBaudRate, [229](#)
 - seUART2_SetBaudRateReg, [229](#)
 - seUART2_SetData, [229](#)
 - UART2_0_IRQHandler, [230](#)
 - UART2_1_IRQHandler, [230](#)
 - UART_Types, [221](#)

- USB_Bus_Management, 250
 - seUSB_Connect, 250
 - seUSB_Disconnect, 250
 - seUSB_GetAddress, 251
 - seUSB_IsSnoozing, 251
 - seUSB_Reset, 251
 - seUSB_Resume, 251
 - seUSB_SetAddress, 252
 - seUSB_Snooze, 252
 - seUSB_Suspend, 252
 - seUSB_WakeUp, 252
- USB_Constants_and_Macros, 231
 - seUSB_ClkSrc, 232
 - seUSB_EPCFG_DIR, 233
 - seUSB_EPCFG_MAXSIZE, 233
 - seUSB_EPCFG_TGLMOD, 233
 - seUSB_Ep0Interrupt, 232
 - seUSB_EpmInterrupt, 234
 - seUSB_GpeInterrupt, 234
 - seUSB_MainInterrupt, 234
 - seUSB_PSEL, 234
 - seUSB_SieInterrupt, 235
 - seUSB_TRCTL_OPMODE, 235
- USB_EP0_Functions, 244
 - seUSB_ClearEP0Fifo, 244
 - seUSB_DataInStage, 244
 - seUSB_DataOutStage, 245
 - seUSB_GetEp0Dir, 245
 - seUSB_GetSetupPacket, 245
 - seUSB_InitEp0, 245
 - seUSB_SetEp0Dir, 246
 - seUSB_SetupStage, 246
 - seUSB_StatusInStage, 246
 - seUSB_StatusOutStage, 244
- USB_EPm_Functions, 247
 - seUSB_ClearEPmFifo, 247
 - seUSB_GetEPmDir, 247
 - seUSB_ResetEPm, 247
 - seUSB_SetEPmDir, 248
- USB_EPn_Functions, 249
 - seUSB_ClearEPnFifos, 249
- USB_FIFO_Management, 254
 - seUSB_DmaCopyFromFifo, 254
 - seUSB_DmaCopyToFifo, 254
 - seUSB_ReadFifo, 254
 - seUSB_WriteFifo, 255
- USB_Functions, 237
 - PORT_IRQHandler, 238
 - SVD2_1_IRQHandler, 243
 - seUSB_ActivateUSBCLK, 238
 - seUSB_Attach, 238
 - seUSB_ClrStall, 238
 - seUSB_ConfSvdDetectDisconnect, 240
 - seUSB_ConfigureDefaultEndpoints, 239
 - seUSB_ConfigureEPm, 239
 - seUSB_ConfigurePortsForUsb, 239
 - seUSB_DeactivateUSBCLK, 240
 - seUSB_Detach, 240
 - seUSB_Disable, 240
 - seUSB_DisableEPm, 240
 - seUSB_DisableInt, 241
 - seUSB_Enable, 241
 - seUSB_EnableEPm, 241
 - seUSB_EnableInt, 242
 - seUSB_Init, 242
 - seUSB_InitUsbModule, 242
 - seUSB_IsVbusConnected, 243
 - seUSB_SetStall, 243
- USB_Interrupt_Management, 256
 - seUSB_ClearAllIntFlags, 257
 - seUSB_DisableAllInt, 257
 - seUSB_EnableAllInt, 257
- USB_Types, 236
- USB, 267
- WAIT
 - Common_Macros, 8
- WDT2, 258
- WDT2_Constants, 259
 - seWDT2_EXOSC_ClkDiv, 259
 - seWDT2_IOSC_ClkDiv, 259
 - seWDT2_Mode, 259
 - seWDT2_OSC1_ClkDiv, 260
 - seWDT2_OSC3_ClkDiv, 260
- WDT2_Functions, 262
 - NMI_Handler, 262
 - seWDT2_ChipReset, 262
 - seWDT2_ConfigureClock, 263
 - seWDT2_Get_tWDTSecs, 263
 - seWDT2_GetCMP, 263
 - seWDT2_GetClk, 263
 - seWDT2_Init, 264
 - seWDT2_InitStruct, 264
 - seWDT2_ResetCounter, 264
 - seWDT2_Set_tWDTSecs, 265
 - seWDT2_SetCMP, 265
 - seWDT2_SetMode, 265
 - seWDT2_Start, 265
 - seWDT2_Stop, 266
- WDT2_Types, 261